نتایجی جدید در
گونه‌های مختلف مسائل قابلیت دید

توسط

مصطفی نوری بایگی

رساله ارائه شده به عنوان بخشی از ملزومات برای دریافت درجه
دکتری نرم‌افزار کامپیوتر

زیر نظر

دکتر محمد قدسی

دی ۱۳۹۱

دانشکده مهندسی کامپیوتر
دانشگاه صنعتی شریف
تهران


بسم الله الرحمن الرحيم
الحمد لله رب العالمين الرحمن الرحيم مالك يوم الدين
اهدنا الصراط المستقيم صراط الذين
انعمت عليهم غير المغضوب عليهم
ولا الضالين

به نام خدا

دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

رساله دکتری

نتایجی جدید در گونه‌های مختلف مسائل قابلیت دید

مصطفی نوری بایگی

کمیته ممتحنین

استاد راهنما: دکتر محمد قدسی	امضاء .....
داور داخلی: دکتر منصور جم‌زاد	امضاء .....
داور داخلی: دکتر حمید بیگی	امضاء .....
داور داخلی: دکتر محمد آبام	امضاء .....
داور خارجی: دکتر محمدرضا رزازی	امضاء .....
داور خارجی: دکتر رامتین خسروی	امضاء .....
داور خارجی: دکتر محمد فرشی	امضاء .....

تاریخ: ۲۸ بهمن ۱۳۹۱

تقدیم به همسرم
که در تمام این سال‌ها، در کنارم بود،
و بیش از من برای موفقیتم تلاش کرد.

بر خود لازم می‌دانم، از زحمات بی‌شائبه دکتر قدسی که با راهنمایی‌های همیشگی خود، نه تنها در راه کسب علم، که در طی مسیر زندگی همچون معلمی مهربان دستم را گرفت و هدایت کرد، تشکر کنم. و بی‌شک هرگز نخواهم توانست این زحمات را جبران نمایم که «من علّمنی حرفاً فقد صیّرنی عبداً». امید بتوانم با سرمشق قرار دادن منش ایشان و دنبال کردن مسیر رشد علمی، رضایت پروردگار و همه معلمانم را کسب نمایم.

همچنین از دوستان و همراهانم در طول این مدت و به ویژه برادرم مجتبی نوری بایگی، دوستم و یاورم شروین دانش‌پژوه و همه کسانی که بدون کمک و یاری آنان نمی‌توانستم این بار سنگین را بر دوش بکشم، تشکر می‌کنم.

و در انتها از همسرم، که همپای من مرارت‌ها و سختی‌ها را تحمل کرد و دم برنیاورد، و از پدر و مادرم که با از خودگذشتگی‌های فراوانشان این امکان را برای من فراهم آوردند تا ذره‌ای کوچک باشم برای حرکت به سوی نور و روشنی‌ها، تشکر می‌کنم و امیدوارم بتوانم بخشی از فداکاری‌هایشان را پاسخ‌گو باشم.

چکیده

مطالعه مسائل قابلیت دید، عمری ۱۰۰ ساله دارد، زمانی که Brunn در سال ۱۹۱۳ قضیه‌ای را در مورد هسته یک مجموعه اثبات کرد. امروزه قابلیت دید یکی از زمینه‌های هندسه محاسباتی است که کار فراوانی روی آن انجام شده است. این امر به دو دلیل رخ داده است. دلیل اول اینکه مسائل قابلیت دید به شکل طبیعی در زمینه‌هایی که ابزارها و الگوریتم‌های هندسه محاسباتی استفاده می‌شوند، دیده می‌شود. به عنوان مثال در گرافیک کامپیوتری، رباتیک، مسیریابی، سیستم‌های اطلاعاتی جغرافیایی، بازی‌های کامپیوتری، معماری کامپیوتری و شناسایی الگو، مسائل قابلیت دید مطرح می‌شوند. دلیل دوم اینکه راه‌حل مسائل قابلیت دید، به عنوان عناصر سازنده در راه‌حل سایر مسائل، مثل یافتن کوتاه‌ترین مسیر یا مسائل مسیریابی، استفاده می‌شود.

بسیاری از مسائل قابلیت دید پیش از این مورد مطالعه عمیق و جدی در فضای دو بعدی و بالاتر قرار گرفته‌اند. در این رساله ما برخی از این مسائل را مورد بررسی مجدد قرار داده‌ایم و نتایج جدیدی را به‌دست آورده‌ایم. تلاش ما بر این بوده است که نتایج جدید، تفاوت قابل توجهی نسبت به نتایج قبلی داشته باشند. برخی دیگر از مسائل نیز به‌شکلی که ما مورد بررسی قرار داده‌ایم، جدید هستند و پیش از این مطالعه نشده بودند. در این رساله، چهار مسأله اصلی به همراه برخی از مسائل مشابه آن‌ها مطالعه شده‌اند. این مسائل عبارتند از مسأله آزمون قابلیت دید ضعیف دو شیء، مسأله محاسبه چندضلعی قابل دید نقطه پرس‌وجو، مسأله محاسبه چندضلعی قابل دید جزئی نقطه و مسائل مربوط به آلفا–رؤیت‌پذیری.

کلیدواژه‌ها: مسائل قابلیت دید، چندضلعی قابل دید، آزمون قابلیت دید، قابلیت دید جزئی، آلفا–رؤیت‌پذیری.

فهرست مندرجات

فهرست شکل‌ها

فصل ۱

مقدمه

یکی از مسائلی که به صورت گسترده در زمینه‌های مختلف مرتبط با کامپیوتر، مثل گرافیک کامپیوتری و رباتیک مطرح می‌شود، مسأله تعیین مرئی بودن اشیاء نسبت به یکدیگر است. این مفهوم در زندگی روزمره نیز به طور طبیعی بسیار مورد استفاده قرار می‌گیرد. ما براساس نحوه قرارگیری اشیائی که در پیرامون خود می‌بینیم، در مورد جهت حرکت تصمیم‌گیری می‌کنیم. یک شیء ممکن است که به‌طور کامل قابل رؤیت نباشد، در این صورت تعیین بخش‌هایی از هر شیء که از مکان ناظر قابل رؤیت است، اهمیت پیدا می‌کند. همچنین معمولاً با حرکت ناظر، بخش‌های قابل رؤیت اشیاء تغییر می‌کند. بنابراین در صورت متحرک بودن ناظر، به‌روز رسانی این بخش‌ها همراه با حرکت ناظر نیز اهمیت پیدا می‌کند. این مسائل که در هندسه محاسباتی در دسته مسائل قابلیت دید رده‌بندی می‌شوند، به دلیل کاربرد فراوان، مورد مطالعه جدی قرار گرفته‌اند و منجر به نتایج جالب توجهی شده‌اند. در این رساله، نتایجی را که در زمینه قابلیت دید در مدت تحقیقات کسب نموده‌ایم، ارائه می‌نماییم.

۱.۱ انگیزه پژوهش

قابلیت دید در شاخه‌های مختلفی مثل گرافیک کامپیوتری، هندسه محاسباتی، بینایی ماشین، رباتیک، ارتباطات راه دور و ... کاربرد دارد. بنابراین طبیعی است که در هر کدام از این شاخه‌ها بسته به نیاز، مسائل قابلیت دید مختلفی طرح و متناسب با آن نیازها راه‌حل‌هایی برای این مسائل ارائه شود.

به عنوان مثال، در گرافیک کامپیوتری، روش‌هایی مثل z-بافر و خط پویش[1] برای تعیین فضای قابل دید مورد استفاده قرار می‌گیرند. این روش‌های کاربردی، به‌دلیل خواص ذاتی ابزارهای نمایشی

[1] Scan line

متداول، از روش‌های دقیق و نظری مناسب‌تر به‌نظر می‌رسند. در مقابل، روش‌های نظری قادر به حل مسائل به‌طور دقیق هستند. برخلاف روش‌های کاربردی، عموماً در این روش‌ها زمان و حافظه مصرف شده بسیار کمتر است. ولی به دلیل پیچیدگی نسبی پیاده‌سازی آن‌ها، برای استفاده در مسائل گرافیک کامپیوتری، مورد اقبال کمتری واقع شده‌اند.

با این وجود، به‌دلیل ارزش بالای راه‌حل‌های نظری و دقت و سرعت آن‌ها، در شاخه‌هایی از علوم کامپیوتر و از جمله هندسه محاسباتی، مسائل قابلیت دید از جنبه نظری مورد توجه زیاد قرار گرفته است. کار بر روی مسائل مرتبط با قابلیت دید در حوزه هندسه محاسباتی از اواخر دهه هفتاد میلادی آغاز شد. در طول سه دهه گذشته محققان متعددی، حضور مؤثر داشتند و به تکامل این بخش کمک کردند.

هرچند بسیاری از مسائل مهم در زمینه قابلیت دید، بررسی شده و روش‌های حل بهینه آن‌ها ارائه گشته است، هنوز مسائل زیادی در این حوزه وجود دارند که یا اصلاً به آن‌ها پرداخته نشده و یا راه‌حل‌های بهینه آن‌ها یافت نشده‌اند. در ادامه این رساله، نمونه‌هایی از این مسائل را خواهیم دید و راه‌حل‌هایی را برای آن‌ها ارائه می‌کنیم. همچنین مسائل مشابهی که برای تحقیقات بعدی مناسب به نظر می‌رسند، عنوان خواهند شد.

۲.۱ نتیجه‌های به‌دست آمده

هدف این رساله، بررسی مسائلی در حوزه قابلیت دید است. این مسائل شامل قابلیت دید نقطه و شیء در صفحه می باشند. همچنین مدل‌های مختلف قابلیت دید به صورت دقیق و تقریبی تحلیل شده‌اند. در زیر این نتایج به همراه توضیحات مختصری عنوان می‌شوند.

۱.۲.۱ آزمون قابلیت دید ضعیف دو شیء در چندضلعی حفره‌دار

در آغاز کار، مسأله‌ای بدین صورت تعریف شد. تعدادی شیء چندضلعی در صفحه با مجموع اضلاع n وجود دارند. آیا دو شیء داده شده خاص (از میان اشیاء موجود یا اشیائی جدید) از یکدیگر قابل دید ضعیف هستند؟ وقتی دو شیء از یکدیگر قابل دید ضعیف هستند که پاره‌خطی یافت شود که دو شیء را به هم وصل کند، بدون این‌که سایر اشیاء قطع شوند. این مسأله در حالت‌های متفاوت بررسی شد و برای هر کدام راه حلی ارائه شد.

در حالت اول، مسأله بدون مرحله پیش‌پردازش و برای اشیاء با شکل‌های نقطه، پاره خط و چندضلعی در نظر گرفته شد و برای هر کدام زمان بهینه ارائه شد. در حالت بعد، مسأله با این فرض در نظر گرفته شد که صحنه (به همراه موانع) و یکی از اشیاء (ناظر) در زمان پیش‌پردازش داده شده و ما

می‌خواهیم در کمترین زمان ممکن، برای شیء داده شده در زمان پرس‌وجو تعیین کنیم که آیا از شیء اول قابل دید ضعیف است یا خیر. در این حالت نیز مسأله برای شکل‌های مختلف ناظر حل شد.

حالت سوم که پیچیده‌ترین شکل مسأله است، فقط صحنه در زمان پیش‌پردازش داده می‌شود و محل دو شیء در زمان پرس‌وجو دریافت می‌شوند و ما باید مشخص کنیم آیا از یکدیگر قابل دید ضعیف هستند یا خیر. در روش پیشنهادی، زمان لازم برای پیش پردازش مسأله $O(n^{۲+\epsilon})$ و حافظه مورد نیاز $O(n^۲)$ است. با انجام پیش‌پردازش یاد شده، می‌توان پرس‌وجوهای مسأله را در زمان $O(n)$ پاسخ داد. در مرتبه زمانی پیش‌پردازش، ϵ یک عدد دلخواه مثبت است. این نتایج در مقاله [۵۵] منتشر شد.

پس از حل مسأله محاسبه چندضلعی قابل دید یک نقطه (نتیجه بخش ۲.۲.۱ را ببینید)، نتیجه آن در شکل سوم مسأله آزمون قابلیت دید ضعیف دو شیء استفاده شد و توازنی بین حافظه و زمان پرس‌وجو برقرار شد. به طور دقیق‌تر، با پیش پردازش صحنه در زمان $O(n^{۲+\epsilon} + m\log^{\epsilon} n)$ با استفاده از مقدار $O(m)$ حافظه، هر پرس‌وجو در زمان $O(\frac{n^۲}{\sqrt{m}}\log^{۴}\frac{\sqrt{m}}{n})$ پاسخ داده می‌شود. در این رابطه‌ها، m پارامتر دلخواهی است با شرط $n^۲ \leq m \leq n^۴$ که میزان حافظه مصرفی را مشخص می‌کند و ϵ عدد دلخواه مثبت. با استفاده از روش جدید پیشنهاد شده، برحسب کاربرد می‌توان حافظه مصرفی را افزایش یا کاهش داد که متناسب با آن زمان پرس‌وجو کاهش یا افزایش می‌یابد. این نتیجه در مقاله [۵۲] ارسال شده است. توضیحات دقیق‌تر این نتایج در فصل‌های ۳ و ۵ آمده است.

۲.۲.۱ محاسبه چندضلعی قابل دید یک نقطه پرس‌وجو در چندضلعی حفره‌دار

دومین مسأله مورد پژوهش، یافتن چندضلعی قابل دید یک نقطه دلخواه در چندضلعی حفره‌دار داده شده از قبل است. این یکی از مسائلی است که کار زیادی بر روی آن شده است و نتایج قابل قبولی در حالت‌های مختلف برای آن به دست آمده است. اما انگیزه‌ای که ما را به سمت بررسی دوباره این مسأله سوق داد، رسیدن به توازن بین حافظه و زمان پیش‌پردازش در مسأله بخش ۱.۲.۱ بود. در بین الگوریتم‌های محاسبه چندضلعی قابل دید، تنها الگوریتمی که بین حافظه و زمان پیش‌پردازش توازن برقرار می‌کرد، از آن Aronov و همکارانش [۴] بود که فقط در چندضلعی ساده قابل استفاده بود. بنابراین ما این مسأله را با هدف رسیدن به توازنی بین حافظه و زمان پیش‌پردازش بررسی کردیم.

نتیجه‌ای که از بررسی این مسأله به دست آمد، بدین صورت است. با پیش‌پردازش چندضلعی در زمان $O(m\log(\sqrt{m}/n))$ و با حافظه $O(m)$، می‌توان چندضلعی قابل دید یک نقطه پرس‌وجوی q را در زمان $O(n^۲\log(۱+\sqrt{m}/n)/\sqrt{m}+|\mathrm{V}(q)|)$ محاسبه کرد. در این رابطه‌ها m یک پارامتر دلخواه با شرط $n^۲ \leq m \leq n^۴$ و $|\mathrm{V}(q)|$ اندازه چندضلعی قابل دید نقطه q است.

با مقایسه نتیجه یافت شده با نتایج موجود، ویژگی‌هایی در الگوریتم ما دیده می‌شود که در الگوریتم‌های قبلی نیست. مهمترین نکته این است که پیچیدگی زمانی الگوریتم ما برای بدترین حالت

داده شده است. برای مثال، وقتی $m = n^۴$، زمان پرس‌وجو $O(\log n + |\mathrm{V}(q)|)$ خواهد بود و با تغییر شکل صحنه، زمان الگوریتم تغییر نمی کند. اما در برخی از الگوریتم‌های قبلی، زمان پرس‌وجو بسته به شکل صحنه و محل نقطه، ممکن است به زمان بدون پیش‌پردازش یعنی $O(n \log n)$ برسد. همچنین در الگوریتم ما امکان محاسبه برخی از خصوصیات چندضلعی قابل دید (مثل اندازه آن) بدون محاسبه خود چندضلعی وجود دارد که بدین شکل زمان پرس‌وجو به طرز چشمگیری کاهش می‌یابد، در حالی که این امکان در الگوریتم‌های قبلی وجود ندارد.

با استفاده از روش ارائه شده برای محاسبه چندضلعی قابل دید، راه‌حل‌هایی نیز برای دو مسأله دیگر، یعنی محاسبه چندضلعی قابل دید ناظر نقطه‌ای متحرک و چندضلعی قابل دید ضعیف پاره‌خط ارائه شده است. ما نشان می‌دهیم می‌توان محلی را که برای اولین بار چندضلعی قابل دید یک ناظر نقطه‌ای متحرک تغییر می‌کند، در زمان لگاریتمی تعیین کرد و در همین زمان چندضلعی قابل دید را به‌روز کرد (با انجام پیش‌پردازش کافی). همچنین نشان می‌دهیم که چگونه می‌توان چندضلعی قابل دید ضعیف یک پاره‌خط را در زمانی وابسته به اندازه آن محاسبه کرد.

نتایج مربوط به چندضلعی قابل دید نقطه در [۵۳] ارائه و در مجله Computational Geometry: Theory and Applications [۵۶] چاپ شد. توضیحات کامل در مورد الگوریتم محاسبه چندضلعی قابل دید نقطه در فصل ۴ آمده است. سایر نتایج مرتبط نیز در فصل ۲ آمده است.

۳.۲.۱ محاسبه چند ضلعی قابل دید جزئی

ناظری را در نظر بگیرید که در محیطی قرار گرفته است که شامل شیشه‌هایی تیره است که فقط بخشی از نور را از خود عبور می‌دهند. در چنین محیطی، ناظر ناحیه پشت شیشه را به وضوح نمی‌بیند، بلکه فقط بخشی از نور محیط پشت شیشه به ناظر می‌رسد. در این محیط، نور یک جسم، بعد از عبور از چند شیشه، به حدی ضعیف می‌شود که ناظر در عمل نمی‌تواند چیزی ببیند. این مطلب ایده‌ای شد برای تعریف و تحلیل مسائل قابلیت دید جزئی.

مسأله به طور دقیق بدین صورت تعریف می‌شود: یک چندضلعی حفره‌دار با تعداد اضلاع n و عدد k با شرط $k < n$ داده شده‌اند. می‌خواهیم با دریافت نقطه محل ناظر و تعداد موانعی که باعث مسدود شدن کامل دید می‌شوند، چندضلعی قابل دید ناظر را محاسبه کنیم. عدد k حداکثر تعداد موانعی است که می‌تواند دید ناظر را مسدود کند. چندضلعی قابل دید به‌ازای عبور از i مانع را با $\mathrm{V}_i(q)$ نمایش می‌دهیم.

چندضلعی قابل دید جزئی، در زمینه‌هایی مثل شبکه‌های حسگر کاربرد دارد. Liu و Hung [۴۶] مدلی برای مکان‌یابی بی‌سیم معرفی کردند که در آن سیگنال ارسال شده توسط یک حسگر حداکثر از k مانع عبور می‌کند و اگر تعداد موانع بین حسگر و ایستگاه مرکزی بیش از k باشد، سیگنالی دریافت

نمی‌شود. ایده مسأله ما از کار Fulek و همکارانش [۲۹] گرفته شد. در مقاله آن‌ها، برای مجموعه S از n شیء و نقطه p، $\tau(p,S)$ بیشینه تعداد اشیائی تعریف می‌شود که توسط یک شعاع ساطع شده از p قطع می‌شود. همچنین $\tau(S)$ کمترین مقدار $\tau(p,S)$ به ازای همه نقاط صفحه برای مجموعه S تعریف می‌شود. آن‌ها نشان دادند که در زمان $O(n^{۴}\log n)$ می‌توان مقدار $\tau(S)$ را محاسبه کرد.

برای مسأله چندضلعی قابل دید جزئی نشان دادیم، می‌توان به روشی مشابه روش محاسبه چندضلعی قابل دید یک نقطه، چندضلعی قابل دید جزئی را محاسبه کرد. بنابراین می‌توان یک چندضلعی حفره‌دار را در مدت زمان $O(m\log(۱+\sqrt{m}/n))$ و با استفاده از حافظه به اندازه $O(m)$ پیش‌پردازش کرد، به گونه‌ای که برای هر نقطه q و هر مقدار i، $\mathrm{V}_i(q)$ را در مدت زمان $O(\frac{n^{۲}}{\sqrt{m}}\log(۱+\frac{\sqrt{m}}{n})+|\mathrm{V}_i(q)|)$ محاسبه کرد. نتایج حاصل در مورد قابلیت دید جزئی در مقاله [۵۴] منتشر گردید. جزئیات مربوط به این نتایج در فصل ٦ آمده است.

۴.۲.۱ قابلیت دید آلفا بین اشیاء

در مدت زمان پژوهش، کارهایی برای تقریب زدن قابلیت دید انجام شد. مهمترین نتیجه‌ای که در این زمینه حاصل شد، تعریف قابلیت دید آلفا، و مسائلی بود که براساس این تعریف طرح شدند. مهمترین ویژگی قابلیت دید آلفا، سادگی و کاربردپذیری زیاد آن است. با وجود این سادگی، پیش از این بر روی آن کار نشده بود. انتظار می‌رود که با آشنایی سایر پژوهشگران با این تعاریف، کارهای زیادی مبتنی بر این کار انجام یابند.

قابلیت دید آلفا براساس مقدار زاویه α تعریف می‌شود به‌گونه‌ای که ناظر بخشی از شیء قابل دید را با زاویه α ببیند. به صورت دقیق‌تر پاره‌خط t از نقطه p قابل دید آلفاست، اگر و فقط اگر مثلثی به رأس p و زاویه α بر روی p وجود داشته باشد که ضلع روبرو به p روی t قرار بگیرد و هیچ مانعی را قطع نکند. بر همین اساس، قابلیت دید (ضعیف و کامل) آلفا بین دو پاره‌خط نیز تعریف می‌شود.

قابلیت دید آلفا، به لحاظ مدل کردن قابلیت دید چشم و ابزارهای مکانیکی، بسیار مناسب به نظر می‌رسد. به عنوان مثال، از نظر فیزیک نور، چشم انسان قادر به مشاهده زاویه‌هایی کوچک‌تر از یک مقدار مشخص نیست. این زاویه از روی نسبت طول موج نور به قطر مردک چشم تعیین می‌شود که از ۰٫۰۰۰۱ رادیان بزرگتر نیست. این محدودیت به این معناست که ما قادر به دیدن یک مو در فاصله‌ای به اندازه دو تا سه برابر اندازه دست خود نیستیم.

نتایجی که در ارتباط با قابلیت دید آلفا حاصل شد، به قرار زیر است.

- مجموعه پاره‌خط $\mathcal{S}$و زاویه α داده شده‌اند. با پیش‌پردازش $\mathcal{S}$در زمان $O(n\log n)$ و حافظه $O(n)$، در زمان پرس‌وجوی $O(\log n)$ مشخص کنیم که آیا پاره‌خط $t\in\mathcal{S}$ از نقطه پرس‌وجوی q قابل دید آلفا است.

- مسأله بالا برای زمانی که α در زمان پرس‌وجو مشخص شود. در این حالت زمان پیش‌پردازش $O(n\log^{۳} n)$ و حافظه $O(n\log^{۲} n)$ و زمان پرس‌وجو $O(\sqrt{n}\log^{۳} n)$ خواهند بود.
- نشان دادیم که اندازه گراف قابل دید آلفا (ضعیف یا کامل) خطی است و در زمان $O(n\log n)$ قابل محاسبه هستند. با محاسبه گرف قابل دید آلفا، می‌توان پرس‌وجوهایی به شکل «آیا پاره‌خط $t \in \mathcal{S}$ از پاره‌خط $s \in \mathcal{S}$ قابل دید آلفاست؟» را در زمان $O(۱)$ پاسخ داد.
- با پیش پردازش مجموعه $\mathcal{S}$در زمان $O(n\log n)$ و با حافظه $O(n)$ می‌توان پرس‌وجوهایی به شکل «آیا پاره خط $t \in \mathcal{S}$ از پاره‌خط دلخواه s قابل دید آلفاست؟» را در زمان $O(\log n)$ پاسخ داد.

نتایج کسب شده در ارتباط با قابلیت دید آلفا، در مقاله [۳۲] منتشر شده است و در حال آماده سازی و ارسال برای مجله نیز می‌باشد. جزئیات مربوط به قابلیت دید آلفا در فصل ۷ آمده است.

۳.۱ طرح رساله

در ادامه رساله، ابتدا در فصل ۲ مفاهیم و تعاریف اصلی مورد نیاز در زمینه قابلیت دید می‌آید. همچنین مسائل پایه‌ای مطرح در زمینه قابلیت دید عنوان شده و ضمن برشمردن تاریخچه‌ای از کارهای انجام شده پیش از این، بخشی از اصلی‌ترین کارها به‌طور خلاصه عنوان می‌شوند. این کارها شامل الگوریتم‌های محاسبه چندضلعی‌های قابل دید نقطه و پاره‌خط در صفحه هستند. در فصل ۳ به سراغ مسأله آزمون قابلیت دید ضعیف دو شیء از هم در صفحه می‌رویم و روشی را برای حل مسأله ارائه می‌کنیم. مسأله در حالت‌های مختلف «بدون پیش‌پردازش»، «با پیش‌پردازش و یک شیء پرس‌وجو» و «با پیش‌پردازش و دو شیء پرس‌وجو» بررسی می‌شوند. همچنین حالت‌های مختلف اشیاء، یعنی نقطه، پاره‌خط و چندضلعی درنظر گرفته می‌شوند. فصل ۴ به الگوریتم‌های محاسبه چندضلعی قابل دید نقطه در صفحه اختصاص دارد. در این فصل ابتدا الگوریتمی برای محاسبه چندضلعی قابل دید نقطه در چندضلعی حفره‌دار با زمان پرس‌وجوی لگاریتمی عنوان می‌شود، و سپس با اعمال تغییراتی در الگوریتم، توازنی بین حافظه و زمان پرس‌وجو برقرار می‌گردد. در فصل ۵ دوباره به سراغ مسأله آزمون قابلیت دید ضعیف دو شیء از هم که در فصل ۳ معرفی شد، می‌رویم. در این فصل تلاش می‌کنیم تا با استفاده از الگوریتمی که برای محاسبه چندضلعی قابل دید نقطه در فصل ۴ بدست آمد، الگوریتمی با قابلیت برقراری توازن بین حافظه و زمان پرس‌وجو ارائه دهیم. فصل ٦ به مسأله قابلیت دید جزئی می‌پردازد. ابتدا مفهوم و کاربرد قابلیت دید جزئی ذکر شده و سپس الگوریتمی با امکان برقراری توازن بین حافظه و زمان پرس‌وجو ارائه می‌شود. در نهایت نیز ایده حل مسأله، در مسائل مشابه استفاده شده و روش‌هایی برای حل آن‌ها مطرح می‌شود. در فصل ۷، قابلیت دید آلفا تعریف شده و مسائل جدیدی که در ارتباط با این تعریف معنی پیدا می‌کنند، مطرح و حل می‌شوند. در ابتدا قابلیت دید آلفا برای نقطه

و پاره‌خط تعریف شده، و سپس الگوریتم‌هایی برای آزمون قابل دید بودن آلفای پاره‌خط از نقطه یا پاره‌خط ارائه می‌شوند. همچنین گراف قابلیت دید آلفا مورد بررسی قرار می‌گیرد و برخی از خواص آن را توضیح می‌دهیم. در ادامه به سراغ فضای سه‌بعدی رفته و نشان می‌دهیم که خاصیت خطی بودن گراف قابلیت دید آلفا، در فضای سه‌بعدی برقرار نیست. در انتها در فصل ۸، خلاصه کارهای انجام شده، به همراه مسائل باز و حوزه‌های با امکان پژوهش بیشتر عنوان می‌شوند.

فصل ۲

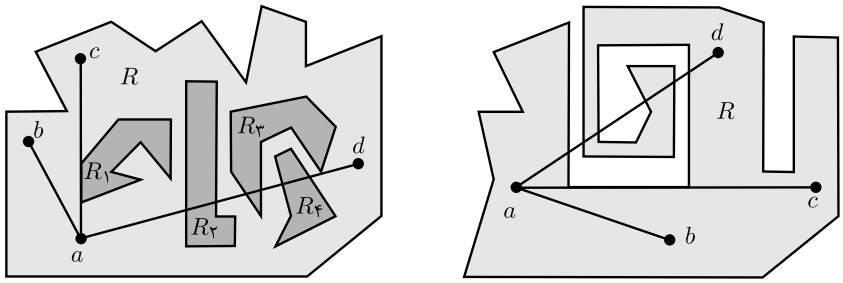
مفاهیم و تعاریف

معمولاً در مسائل مرتبط با قابلیت دید، صحنه از یک ناحیه چندضلعی – که ممکن است شامل موانعی نیز باشد – تشکیل شده است. بنابراین در آغاز تعاریف، باید صحنه‌ای را که مسأله در آن طرح می‌شود به طور دقیق مشخص کنیم. یک چندضلعی P، ناحیه بسته‌ای در صفحه با نام R است که با تعدادی پاره‌خط که به آن‌ها یال‌های چندضلعی گفته می‌شود، محدود شده است. بین هر دو نقطه در ناحیه R، مسیری وجود دارد که هیچ‌یک از یال‌های P را قطع نمی‌کند. نقاط انتهایی یال‌های P رئوس P نامیده می‌شوند. چون P یک ناحیه بسته است، مرز آن از زنجیره‌هایی تشکیل شده که یال‌های P به‌وجود آورده‌اند و در هر زنجیره، هر دو یال متوالی یک رأس مشترک دارند. در صورتی که P از یک زنجیره تشکیل شده باشد، P را یک چندضلعی ساده (شکل ۱.۲– الف) و در غیراینصورت یک چندضلعی حفره‌دار (شکل ۱.۲– ب) می‌نامیم.

ناحیه R، ناحیه داخلی یا داخل و مجموعه نواحی دیگر بجز R، نواحی خارجی یا خارج P نامیده می‌شوند. یک رأس از P محدب خوانده می‌شود اگر زاویه داخلی تشکیل شده توسط دو یال آن رأس حداکثر π باشد و در غیراینصورت رأس مقعر نام دارد.

همان‌طور که در بالا گفته شد، یک چندضلعی ساده P، یک ناحیه محدود به زنجیره‌ای از یال‌هاست که هیچ دو یال غیرمتوالی با یکدیگر برخورد ندارند. در حالت کلی P را به صورت یک لیست پیوندی دوطرفه از رئوس نگه‌داری می‌کنیم. هر رأس در این لیست، شامل مختصات x و y خود و دو اشاره‌گر به رئوس بعدی در جهت ساعت‌گرد و پادساعت‌گرد است. می‌توان دید که اگر یال‌های چندضلعی ساده P در جهت ساعت‌گرد (پادساعت‌گرد) طی شوند، ناحیه داخلی P همیشه در سمت راست (چپ) یال‌های P قرار می‌گیرد.

در مورد چندضلعی حفره‌دار نیز می‌توان روش مشابهی را به‌کار برد. فرض کنید P یک چندضلعی



ب الف

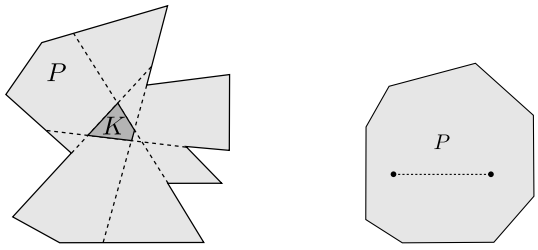
شکل ۱.۲ : الف. نمونه یک چندضلعی ساده. ب. نمونه یک چندضلعی حفره‌دار. در هر دو چندضلعی، b و c از a دیده می‌شوند، ولی d دیده نمی‌شود.

حفره‌دار با h حفره باشد. زنجیره یال‌های P را با $c_۰, c_۱, \ldots, c_h$ نمایش می‌دهیم به‌طوری‌که $c_۰$ مرز خارجی P باشد. همچنین ناحیه داخلی هر زنجیره c_j را با R_j نشان می‌دهیم (شکل ۱.۲- ب). چون P یک ناحیه بسته و محدود است، برای هر $j > ۰$، $R_j \subset R_۰$. همچنین برای هر $j \neq k > ۰$، $R_j \cap R_k = \emptyset$ و نتیجه این‌که $R = R_۰ - (R_۱ \cup \cdots \cup R_h)$. مشاهده می‌شود که اگر P یک چندضلعی ساده باشد، آنگاه مرز P یعنی $R = R_۰$ فقط از یک زنجیره $c_۰$ تشکیل شده است (شکل ۱.۲- الف). همانند حالت قبل، فرض می‌کنیم که P، با h زنجیره تعیین می‌شود که هرکدام از این زنجیره‌ها در یک لیست پیوندی دوطرفه از رئوس نگه‌داری می‌شود و برای هر زنجیره یک اشاره‌گر به یکی از رئوس آن وجود دارد تا با استفاده از آن به زنجیره دسترسی یابیم.

دو نقطه p و q از یکدیگر قابل دید هستند اگر پاره‌خطی که این دو نقطه را به هم وصل می‌کند، به‌طور کامل در داخل P قرار بگیرد و شامل هیچ نقطه‌ای از خارج P نباشد. با این تعریف، پاره‌خط pq می‌تواند از یک رأس مقعر عبور کند و یا بر یک یال چندضلعی مماس باشد. چون وقتی q p را می‌بیند، q هم p را می‌بیند، می‌توان قابل دید بودن p و q از یکدیگر را با q p را می‌بیند، عنوان کرد. در شکل‌های ۱.۲، نقطه a، نقاط b و c را می‌بیند ولی نقطه d را نمی‌بیند.

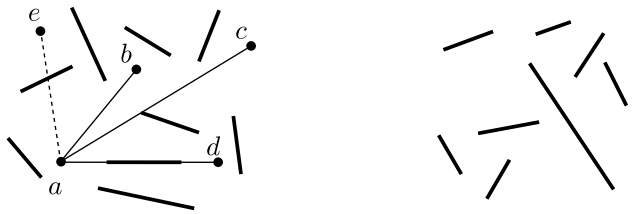
با استفاده از مفهوم قابل دید بودن، می‌توان دو دسته از چندضلعی‌ها را تعریف کرد، دسته اول چندضلعی‌های محدب و دسته دوم چندضلعی‌های ستاره‌ای هستند. یک چندضلعی ساده P، محدب نامیده می‌شود، اگر هر دو زوج نقطه در داخل P از یکدیگر قابل دید باشند (شکل ۲.۲- الف). همان‌طور که از شکل پیداست، زاویه داخلی هر رأس چندضلعی محدب حداکثر π است. می‌توان چندضلعی محدب را اشتراک کراندار تعدادی نیم‌صفحه بسته نیز تعریف کرد.

یک چندضلعی P ستاره‌ای خوانده می‌شود، اگر نقطه‌ای مانند z درون آن باشد که همه نقاط P از z دیده شوند (شکل ۲.۲- ب). مجموعه همه نقاط شبیه z در P، هسته P نامیده می‌شود. نتیجه‌ای که از تعریف چندضلعی محدب و هسته چندضلعی ستاره‌ای می‌توان گرفت این است که هسته P همواره



ب الف

شکل ۲.۲ : نمونه یک چندضلعی الف. محدب ب. ستاره‌ای



ب الف

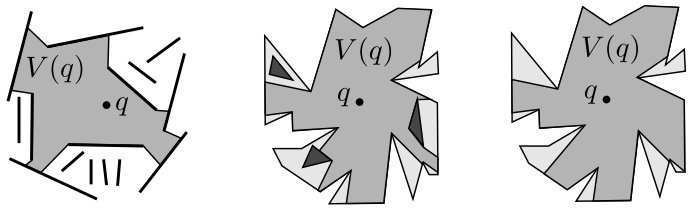
شکل ۳.۲ : الف. آرایشی از پاره‌خط‌ها ب. نقطه a، نقطه b، c و d را می‌بیند ولی e را نمی‌بیند.

محدب است، زیرا هر دو نقطه در هسته یکدیگر را می‌بینند. وقتی نقطه z عضو هسته P باشد، ترتیب رئوس روی مرز P با ترتیب زاویه‌ای آن‌ها حول z یکسان است، یا به بیان دیگر نقاط با ترتیب زاویه‌ای حول z مرتب هستند. نتیجه دیگر این‌که یک چندضلعی محدب، ستاره‌ای هم هست و تمام نقاط آن به هسته تعلق دارند.

آرایشی از پاره‌خط‌ها[۱]، مجموعه‌ای از پاره‌خط‌ها در صفحه است که نمونه‌ای از آن در شکل ۳.۲- الف دیده می‌شود. در حالت کلی، مرز یک چندضلعی ساده یا چندضلعی حفره‌دار آرایشی از پاره‌خط‌هاست که پاره‌خط‌ها نقاط انتهایی مشترک دارند. در آرایش پاره‌خط‌های A، قدرت دید را می‌توان به این صورت تعریف کرد: دو نقطه p و q در A یکدیگر را می‌بینند، اگر پاره‌خط اتصال دهنده p و q، از میان هیچ یک از پاره‌خط‌های A عبور نکند. این تعریف به پاره‌خط pq اجازه مماس شدن بر پاره‌خط‌های دیگر (مثل مرئی بودن d برای a در شکل ۳.۲- ب) و یا عبور از نقاط انتهایی پاره‌خط‌های دیگر (مثل مرئی بودن c برای a در شکل ۳.۲- ب) را می‌دهد. حالت‌های مختلف قدرت دید یک نقطه را در آرایش پاره‌خط‌ها می‌توان در شکل ۳.۲- ب دید.

مجموعه همه نقاطی از چندضلعی P که از نقطه q درون P قابل دید هستند، چندضلعی قابل دید q در P یا $\mathrm{V}(q)$ نامیده می‌شود. به بیان دقیق‌تر $\mathrm{V}(q) = \{p \in P | p, q$ را می‌بیند$\}$. مسأله محاسبه

[۱] Line segment arrangement



الف ب پ

شکل ۴.۲ : چندضلعی قابل دید نقطه q در الف. چندضلعی ساده ب. چندضلعی حفره‌دار پ. مجموعه‌ای از خطوط

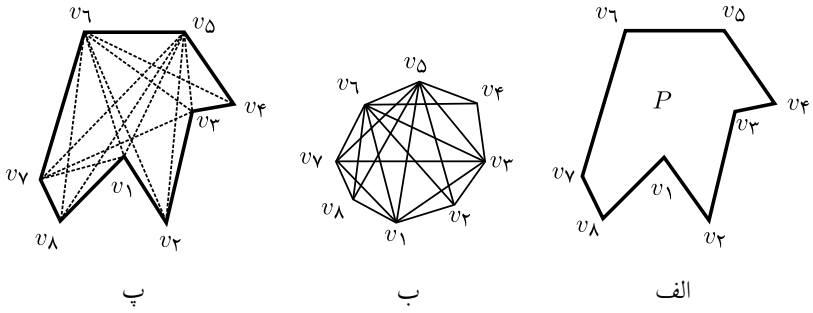
چندضلعی قابل دید نقطه q با مسأله حذف خطوط مخفی که در فرایند ترسیم در گرافیک کامپیوتری استفاده می‌شود، ارتباط نزدیکی دارد. در شکل ۴.۲ نمونه چندضلعی قابل دید یک نقطه در یک چندضلعی ساده، چندضلعی حفره‌دار و مجموعه‌ای از پاره‌خط‌ها نشان داده شده است. طبق تعریف، چندضلعی قابل دید نقطه q یک چندضلعی ستاره‌ای است که q به هسته آن تعلق دارد. همچنین چندضلعی قابل دید یک نقطه در آرایشی از پاره‌خط‌ها ممکن است کران‌دار نباشد، که البته خلاف تعریفی است که برای چندضلعی داشتیم. بنابراین بهتر است، صحنه مسأله را با استفاده از یک چندضلعی به اندازه کافی بزرگ، به یک ناحیه کران‌دار محدود کنیم.

می‌توان چندضلعی قابل دید اشیاء غیرنقطه‌ای را به روشی مشابه چندضلعی قابل دید شیء نقطه‌ای تعریف کرد. مثلاً برای پاره‌خط s، چندضلعی قابل دید به صورت مجموعه نقاطی از صحنه که از همه نقاط s قابل دید هستند تعریف می‌شود، به‌صورت دقیق‌تر $\mathrm{V}(s) = \{p \in P | \forall q \in s :$ p، q را می‌بیند$\}$ یا $\mathrm{V}(s) = \cap_{q \in s} \mathrm{V}(q)$. این نوع چندضلعی قابل دید در مورد اشیاء غیرنقطه‌ای گاهی اوقات با نام چندضلعی قابل دید کامل[۲] معرفی می‌شود. این تغییر نام به دلیل وجود نوع دیگری از چندضلعی قابل دید در مورد اشیاء غیرنقطه‌ای است.

چندضلعی دیگری که در مورد دید اشیاء غیرنقطه‌ای تعریف می‌شود، چندضلعی قابل دید ضعیف است. چندضلعی قابل دید ضعیف را در مورد اشیاء غیرنقطه‌ای در درون یک چندضلعی P به صورت مجموعه نقاطی از P که حداقل از یک نقطه آن شیء دیده می‌شود، تعریف می‌کنند. به صورت دقیق‌تر برای یک شیء O چندضلعی قابل دید به صورت $\mathrm{WV}(O) = \{p \in P | \exists q \in O :$ p، q را می‌بیند$\}$ یا $\mathrm{WV}(O) = \cup_{q \in O} \mathrm{V}(q)$ تعریف می‌شود.

این نام‌گذاری‌ها بر اساس کاری است که توسط Avis و Toussaint [۸] انجام شده است. آن‌ها ضمن مطالعه مسأله موزه هنر[۳] سه گونه مختلف دید را برای یک پاره‌خط تعریف می‌کنند. گونه اول و دوم همان دید کامل و ضعیف هستند که در بالا تعریف شد. گونه سوم، دید قوی یک پاره‌خط است.

[۲] Complete visibility polygon
[۳] Art gallery



شکل ۵.۲ : الف. چندضلعی P ب. گراف قابلیت دید P پ. گراف قابلیت دید رسم شده بر روی P

چندضلعی P از پاره‌خط s به‌طور قوی قابل دید نامیده می‌شود، اگر نقطه‌ای مانند w روی s وجود داشته باشد که همه نقاط P از w قابل دید باشند.

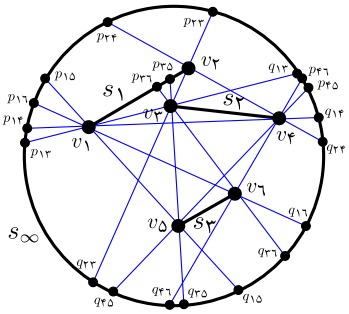
همان‌طور که از تعاریف بالا برمی‌آید، ارتباط زیادی بین چندضلعی قابل دید اشیاء غیرنقطه‌ای با چندضلعی قابل دید اشیاء نقطه‌ای وجود دارد و تعیین چندضلعی قابل دید اشیاء نقطه‌ای، اساس تعیین چندضلعی قابل دید اشیاء غیرنقطه‌ای است. به همین ترتیب چندضلعی قابل دید اشکال پیچیده‌تر با استفاده از چندضلعی قابل دید پاره‌خط‌های مرزی آن تعیین می‌شود. به همین دلیل در فصل‌های آتی به روش‌های موجود برای تعیین قدرت دید اشیاء نقطه‌ای و پاره‌خط‌ها می‌پردازیم و مسائل و راه‌حل‌های مختلف را در این دو حوزه می‌بینیم.

آخرین مفهومی که در این فصل بررسی می‌کنیم، مفهوم گراف قابلیت دید است. گراف قابلیت دید رئوس (یا برای سادگی گراف قابلیت دید[۴]) چندضلعی P که با $\mathrm{VG}(P)$ نمایش داده می‌شود، گرافی است که رابطه بین رئوس P را نشان می‌دهد. در این گراف برای هر رأس P یک گره و برای هر دو رأسی که یکدیگر را می‌بینند، یک یال در نظر گرفته می‌شود. در شکل ۵.۲، در قسمت ب گراف قابلیت دید چندضلعی نشان داده شده در قسمت الف رسم شده است. گراف قابلیت دید در اکثر اوقات بطور مستقیم بر روی چندضلعی رسم می‌شود (همانند شکل ۵.۲- پ). این روش نمایش، بیانگر این است که گراف قابلیت دید مجموعه همه یال‌ها و قطرهای چندضلعی است.

به روش مشابهی می‌توان گراف قابلیت دید را برای چندضلعی‌های حفره‌دار و آرایشی از پاره‌خط‌ها تعریف کرد. در مورد آرایش پاره‌خط‌ها، نقاط انتهایی پاره‌خط‌ها گره‌های گراف قابلیت دید هستند. همچنین اشکال دیگر گراف قابلیت دید، مثل گراف قابلیت دید یال‌ها (گرافی که رابطه دید ضعیف بین یال‌ها را نشان می‌دهد)، گراف قابلیت دید رأس-یال و گراف قابلیت دید یک نقطه، را می‌توان به راحتی تعریف کرد.

در برخی از مسائل، شکل دیگری از گراف قابلیت دید مورد استفاده قرار می‌گیرد. این گراف

[۴] Visibility graph



شکل ۶.۲ : گراف توسعه‌یافته قابلیت دید. پاره‌خط‌های خط‌چین یال‌های گراف توسعه‌یافته قابلیت دید $S = \{s_۱, s_۲, s_۳, s_\infty\}$ است.

که گراف توسعه‌یافته قابلیت دید[۵] نامیده می‌شود، از امتداد دادن هر یال e در گراف قابلیت دید از دو طرف، تا جایی که در نقاط p_e و q_e به دو مانع برخورد کند، به‌دست می‌آید. برای راحتی، یک مانع s_∞ نیز به مجموعه موانع اضافه می‌شود تا در صورتی که امتداد یک یال گراف، با هیچ کدام از موانع برخورد نداشته باشد، با این مانع برخورد بکند. در شکل ۶.۲ نمونه‌ای از گراف توسعه‌یافته قابلیت دید نشان داده شده است. واضح است که اندازه گراف توسعه‌یافته قابلیت دید همانند گراف قابلیت دید بوده و حداکثر $O(n^۲)$ است. Suri و O'Rourke [۶۲] روشی را ارائه کرده‌اند که می‌توان گراف توسعه‌یافته قابلیت دید را در زمان $O(n^۲)$ محاسبه کرد. اما با ترکیب روش Ghosh و Mount [۳۳] برای ساخت گراف قابلیت دید و روش Keil و همکارانش [۴۳] برای تعیین مانعی که توسط امتداد یک پاره‌خط قطع می‌شود، می‌توان در زمان $O(k + n \log n)$ نیز گراف توسعه‌یافته قابلیت دید را محاسبه کرد که k تعداد یال‌های گراف قابلیت دید است و حداکثر $\Theta(n^۲)$ است.

۱.۲ قابلیت دید ناظر نقطه‌ای

از آنجایی که مسأله تعیین قدرت دید ناظر نقطه‌ای از پایه‌ای‌ترین مسائل حیطه قدرت دید است، در این فصل به‌طور دقیق‌تر به این مسأله نگاه می‌کنیم و مسائل مختلف مرتبط با این موضوع و راه‌حل‌های ارائه شده در مقالات مختلف را بررسی می‌کنیم. در این فصل مسائل قدرت دید ناظر نقطه‌ای را در چهار بخش جداگانه مطالعه می‌کنیم. بخش ۱.۱.۲ به روش‌های مختلف محاسبه چندضلعی قابل دید در چندضلعی‌های ساده اختصاص یافته است. در بخش ۲.۱.۲ همین مسأله در چندضلعی‌های حفره‌دار بررسی می‌شود. در بخش ۳.۱.۲ و ۴.۱.۲ همین مسائل با افزودن یک مرحله پیش‌پردازش، برای کاستن زمان پاسخ به پرس‌وجو مورد مطالعه قرار می‌گیرند.

[۵] Extended visibility graph

۱.۱.۲ چندضلعی قابل دید درون چندضلعی ساده

در این بخش الگوریتم‌های محاسبه چندضلعی قابل دید نقطه q در درون چندضلعی ساده P را مرور می‌کنیم. در آغاز انواع یال‌های تشکیل دهنده $\mathrm{V}(q)$ را مشخص می‌کنیم. برخی از یال‌های $\mathrm{V}(q)$ همان یال‌های مرز P هستند. یال‌های دیگر $\mathrm{V}(q)$ که متعلق به مرز P نیستند یال ساختگی[۶] نامیده می‌شوند. یال ab از مرز $\mathrm{V}(q)$ را یال ساختگی می‌نامیم اگر اولاً بجز a و b هیچ نقطه‌ای از ab به مرز P تعلق نداشته باشد و ثانیاً سه نقطه q، a و b هم‌خط باشند و ثالثاً a یا b رأسی از P باشد. واضح است که اگر یال‌های ساختگی $\mathrm{V}(q)$ مشخص باشند، مرز $\mathrm{V}(q)$ با اضافه کردن یال‌های مرز P که بین دو یال ساختگی متوالی قرار می‌گیرند، مشخص می‌شود. چون هر یال ساختگی را می‌توان در زمان $O(n)$ محاسبه کرد و تعداد یال‌های $\mathrm{V}(q)$ خطی است، یک الگوریتم ابتدایی می‌توان ارائه کرد که در زمان $O(n^2)$ چندضلعی قابل دید را محاسبه کند.

الگوریتم‌های ارائه شده

مسأله محاسبه $\mathrm{V}(q)$ در چندضلعی‌های ساده به شکل نظری اولین بار توسط Davis و Benedikt [۲۰] مورد توجه قرار گرفت و آن دو الگوریتمی با زمان $O(n^2)$ ارائه کردند. بعد از آن ElGindy و Avis [۲۶] و Lee [۴۴] الگوریتم‌هایی را معرفی کردند که در زمان $O(n)$ این مسأله را حل می‌کردند.

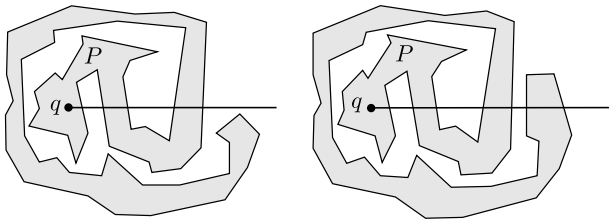
Joe [۴۱] و Joe و Simpson [۴۲] نشان دادند در صورتی که عدد پیچش[۷] چندضلعی بزرگتر از ۱ باشد، یعنی مرز چندضلعی حداقل دو دور پیچ بخورد (شکل ۷.۲)، هر دو الگوریتم ارائه شده توسط ElGindy و Avis، و Lee دچار مشکل شده و درست عمل نمی‌کنند. عدد پیچش چندضلعی P نسبت به نقطه $z \in P$، برابر تعداد دورهایی است که مرز P به دور z می‌پیچد. اگر عدد پیچش P نسبت به z یک باشد، مانند شکل ۷.۲- ب، P یک چندضلعی بدون پیچش خوانده می‌شود. آن‌ها همچنین در [۴۲] الگوریتمی ارائه کردند که ضمن حل مشکل پیچش با نگه‌داری تعداد دورها حول q، می‌تواند در زمان $O(n)$، $\mathrm{V}(q)$ را محاسبه کند.

نکته جالب در این مسأله این است که بخشی از چندضلعی ساده P که عدد پیچش P را نسبت به q بیشتر از ۱ می‌کند، از q قابل دید نیست. بنابراین بهتر است قبل از اعمال الگوریتم ElGindy و Avis یا Lee این بخش‌ها را از P حذف کنیم. با این کار اولاً عدد پیچش P نسبت به q یک خواهد شد و ثانیاً P ساده شده، شامل q و $\mathrm{V}(q)$ خواهد بود. کار ساده‌سازی P را می‌توان با الگوریتم Bhattacharya و همکارانش [۱۰] در زمان خطی انجام داد.

---

[۶] Constructed edge

[۷] Winding



شکل ۷.۲ : الف. عدد پیچش ۲ ب. عدد پیچش ۱ نسبت به q.

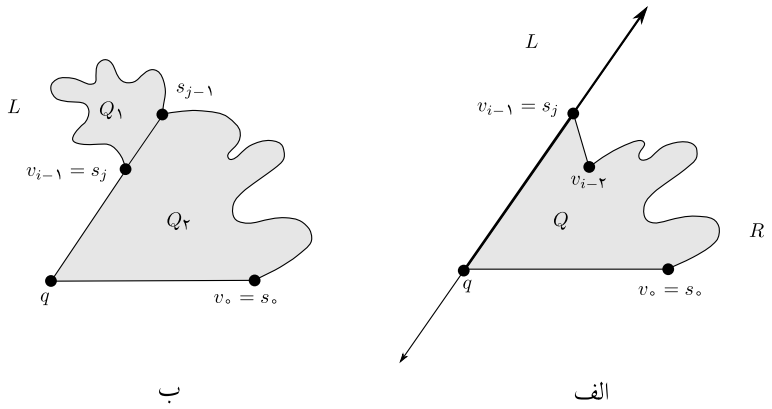
محاسبه چندضلعی قابل دید در چندضلعی ساده بدون پیچش

در این بخش، الگوریتم Lee [۴۴] برای محاسبه چندضلعی قابل دید ارائه می‌شود. همان‌طور که پیش از این ذکر شد، این الگوریتم وقتی پیچش چندضلعی برای ناظر بیشتر از ۱ باشد درست عمل نمی‌کند. بنابراین در این‌گونه حالات، باید از روش Joe و Simpson [۴۲] استفاده کرد و یا با استفاده از روش Bhattacharya و همکارانش [۱۰]، ابتدا پیچش P را نسبت به q از بین برد و سپس الگوریتم زیر را بر روی آن اجرا کرد.

اولین گام در این الگوریتم، یافتن نقطه v_0 روی مرز P است، به‌طوری که نقطه q آن را دقیقاً در سمت راست خود ببیند. برای این کار، ابتدا تک تک یال‌های P بررسی شده و یال‌هایی انتخاب می‌شوند که یک سر آن‌ها بالا و سر دیگر پایین q قرار می‌گیرد. این یال‌ها، یال‌هایی هستند که توسط خط افقی رسم شده از q قطع می‌شوند. با مقایسه محل تلاقی این یال‌ها با خط افقی مذکور، نقطه‌ای را به عنوان v_0 انتخاب می‌کنیم که در بین نقاط تلاقی سمت راست q، مؤلفه x کوچکتری داشته باشد. فرض کنید رئوس P در جهت پادساعت‌گرد v_1، v_2، . . . و v_n نام‌گذاری شده‌اند به گونه‌ای که v_1 اولین رأس مشاهده شده بعد از v_0 در جهت پادساعت‌گرد باشد. با توجه به این‌که P فاقد پیچش است، جهت‌های ساعت‌گرد و پادساعت‌گرد به‌راحتی قابل تشخیص هستند.

الگوریتم Lee هر یک از رئوس P را به ترتیب پردازش کرده و پشته‌ای را که در آن رئوس قابل دید از q نگه‌داری می‌شود، به روز می‌رساند. در پایان الگوریتم، رئوس نهایی $\mathrm{V}(q)$ در پشته قرار دارند. در ابتدا نقطه v_0 در انتهای پشته و v_1 بالای آن قرار دارد. حالت الگوریتم را در هر لحظه با زوج مرتب $(x;S)$ نمایش می‌دهیم که x رأسی است که در حال حاضر پردازش می‌شود و S محتوای پشته است که از پایین به بالا فهرست می‌شود. بنابراین حالت اولیه الگوریتم $(v_2;v_0,v_1)$ است.

در حالت کلی که الگوریتم در حالت $(v_i;v_0=s_0,s_1,\ldots,s_j=v_{i-1})$ قرار دارد، رأس v_{i-2} یکی از دو سر پاره‌خطی است که اخیراً ملاحظه شده است. بر اساس جهت قرارگیری رأس v_{i-2} نسبت به پرتو $q\vec{s}_j$، دو وضعیت ممکن است رخ دهد.



شکل ۸.۲ : نواحی تشکیل شده براساس محل قرارگیری v_{i-2} نسبت به $q\vec{s}_j$.

وضعیت ۱. رأس v_{i-2} سمت راست $q\vec{s}_j$ قرار می‌گیرد.

فرض کنید Q چندضلعی ساخته شده با رئوس q، s_0، s_1، . . . و s_j باشد. همچنین فرض کنید خارج Q به دو بخش L و R تقسیم شده است به‌طوری‌که L نقاط روی $q\vec{s}_j$ و سمت چپ آن و R نقاط سمت راست $q\vec{s}_j$ باشد (شکل ۸.۲- الف). برحسب محل قرارگیری v_i در R، L و یا Q سه وضعیت زیر بررسی می‌شود.

الف. $v_i \in R$ (شکل ۹.۲- الف).

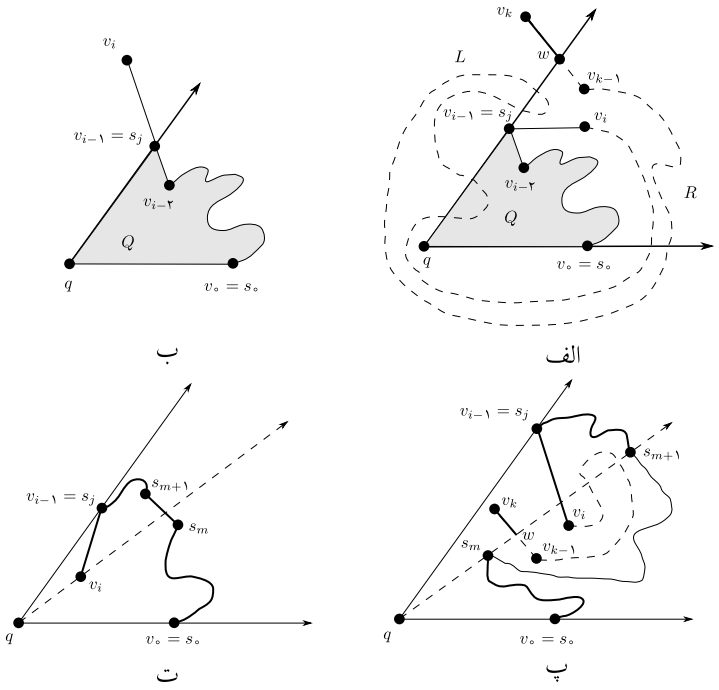
در این وضعیت رأس v_i از q دیده نمی‌شود. بنابراین رأس‌های P را به ترتیب از v_i پردازش می‌کنیم تا زمانی که با رأس v_k مواجه شویم به‌طوری‌که یال $v_{k-1}v_k$ با پرتو ساطع شده از s_j هم‌جهت با $q\vec{s}_j$ برخورد داشته باشد. چنین رأسی همیشه وجود خواهد داشت. اگر محل تلاقی پرتو با یال $v_{k-1}v_k$ را w بنامیم، w و v_k را در پشته قرار می‌دهیم و به حالت $(v_{k+1}; s_0, \ldots, s_j, w, v_k)$ می‌رسیم. در گام بعدی، الگوریتم در وضعیت ۱ قرار خواهد گرفت.

نکته قابل توجه در این وضعیت این است که اگر پیش از رسیدن به v_k، یالی از مرز P پرتو $q\vec{s}_0$ را قطع کند، باید زمانی به دنبال v_k باشیم که مرز P به تعدادی زوج، پرتو $q\vec{s}_0$ را قطع کرده باشد. در غیراینصورت وضعیت‌هایی مثل شکل ۹.۲- الف به صورت نادرستی بررسی می‌شوند.

ب. $v_i \in L$ (شکل ۹.۲- ب).

در این وضعیت، فقط v_i را در پشته قرار می‌دهیم. حالت بعدی الگوریتم $(v_{i+1}; s_0, \ldots, s_j, v_i)$ و در گام بعدی در وضعیت ۱ قرار می‌گیرد.

پ. $v_i \in Q$ (شکل ۹.۲- پ).



شکل ۹.۲ : وضعیت‌هایی که در آن‌ها v_{i-2} سمت راست $q\vec{s}_j$ قرار می‌گیرد.

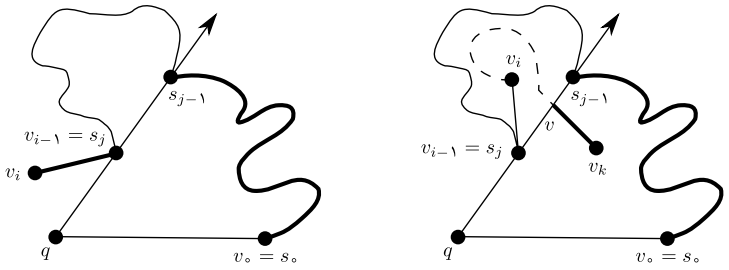
در این وضعیت، رأس $s_j = v_{i-1}$ که در بالای پشته قرار دارد، از q دیده نمی‌شود و ممکن است سایر عناصر روی پشته نیز قابل دید نباشند. بنابراین، عناصر روی پشته، یکی پس از دیگری از روی آن برداشته می‌شوند تا زمانی که به نقطه‌ای مثل s_m بر روی پشته برسیم که یا $v_{i-1}v_i$، s_ms_{m+1} را در نقطه‌ای مثل v قطع کند و یا v_i سمت چپ $q\vec{s}_m$ قرار بگیرد.

در مورد اول (شکل ۹.۲- پ)، v_i از q قابل دید نیست. بنابراین پردازش رئوس P را از v_i ادامه می‌دهیم تا به رأسی مثل v_k برسیم که $v_{k-1}v_k$، s_mv را نسبت به q در جهت پادساعت‌گرد قطع کند. چنین رأس v_kی حتماً وجود خواهد داشت. اگر w محل تلاقی $v_{k-1}v_k$ با s_mv باشد، w و v_k را بر روی پشته قرار می‌دهیم. حالت بعدی $(v_{k+1}; s_0, \ldots, s_m, w, v_k)$ خواهد بود و الگوریتم به وضعیت ۱ خواهد رفت.

در مورد دوم (شکل ۹.۲- ت)، اگر w محل تلاقی s_ms_{m+1} با $q\vec{v}_i$ باشد، w و v_i را در پشته قرار می‌دهیم که به حالت $(v_{i+1}; s_0, \ldots, s_m, w, v_i)$ منجر شده و الگوریتم به وضعیت ۲ خواهد رفت.

وضعیت ۲. رأس v_{i-2} سمت چپ $q\vec{s}_j$ قرار می‌گیرد.

در این وضعیت فرض کنید Q_1 چندضلعی به‌وجود آمده توسط s_js_{j-1} و مرز P از s_{j-1} به s_j،



شکل ۲.۱۰ : وضعیت‌هایی که در آن‌ها v_{i-2} سمت چپ $q\vec{s}_j$ قرار می‌گیرد.

و Q_2 چندضلعی به‌وجود آمده توسط q، s_0، . . . ، s_{j-1} و L، و L سایر بخش‌های صفحه باشد (شکل ۸.۲- ب). مانند قبل، بر اساس محل قرارگیری v_i سه وضعیت را بررسی می‌کنیم.

الف. $v_i \in Q_1$ (شکل ۲.۱۰- الف).

در این وضعیت v_i از q قابل دید نیست، بنابراین رئوس بعدی P را یکی پس از دیگری پردازش می‌کنیم تا به رأسی مثل v_k برسیم که یال $v_{k-1}v_k$، یال $s_{j-1}s_j$ را در نقطه‌ای مثل v قطع کند. در ادامه روش وضعیت ۱- پ را ادامه می‌دهیم، فقط در این وضعیت v نقش s_j را ایفا می‌کند.

ب. $v_i \in L$ (شکل ۲.۱۰- ب).

در این وضعیت، v_i در پشته قرار می‌گیرد و حالت جدید $(v_{i+1}; s_0, \ldots, s_j, v_i)$ خواهد بود و الگوریتم به وضعیت ۱ منتقل می‌شود.

پ. $v_i \in Q_2$.

در این وضعیت مشابه وضعیت ۱- پ عمل می‌شود.

در ضمن تکرار مراحل بالا، نکته‌ای که باید مد نظر قرار بگیرد این است که زاویه طی شده توسط رئوس پشته نباید بیشتر از 2π باشد. اگر چنین اتفاقی بیفتد، زمانی می‌رسد که یال $s_{m-1}s_m$ بین دو رأس بالای پشته، پرتو $q\vec{s}_0$ را قطع می‌کند. اگر محل تلاقی این دو v باشد، ما پردازش رئوس P را ادامه می‌دهیم تا وقتی که یا به s_0 برسیم و یا به رأسی مثل v_k که $v_{k-1}v_k$ پاره‌خط s_0v را قطع می‌کند. در حالت اول، به‌جای s_m، v را بر روی پشته قرار می‌دهیم و کار به اتمام می‌رسد. ولی در حالت دوم، همانند وضعیت ۱- پ در بالا عمل می‌کنیم.

مراحل عنوان شده در بالا تا زمانی ادامه پیدا می‌کند که دوباره v_0 بر روی پشته قرار بگیرد که در این حالت چندضلعی قابل دید q محاسبه شده است. درستی این روش در مقاله Lee [۴۴] اثبات شده است. قضیه زیر نتیجه این الگوریتم را (بدون در نظر گرفتن پیچش چندضلعی) خلاصه می‌کند.

قضیه ۱.۲ چندضلعی قابل دید نقطه q را در داخل چندضلعی ساده P با تعداد رأس n، می‌توان در زمان $O(n)$ محاسبه کرد.

۲.۱.۲ چندضلعی قابل دید درون چندضلعی حفره‌دار

مسأله محاسبه چندضلعی قابل دید یک نقطه در درون چندضلعی حفره‌دار کمی پیچیده‌تر از حالت قبل است و دیگر نمی‌توان آن را در زمان خطی حساب کرد. ثابت می‌شود [۶۲] که مسأله مرتب‌سازی در این مسأله قرار دارد، بنابراین مسأله محاسبه چندضلعی قابل دید در درون چندضلعی حفره‌دار با پیچیدگی $O(n)$ در بدترین حالت، حداقل به زمان $O(n \log n)$ نیاز دارد.

اولین الگوریتم برای محاسبه چندضلعی قابل دید $\mathrm{V}(q)$ درون چندضلعی حفره‌دار P با تعداد رأس n و تعداد حفره h، توسط Asano [۶] ارائه شد که در زمان $O(n \log h)$، $\mathrm{V}(q)$ را محاسبه می‌کند. Suri و O'Rourke [۶۲] نیز الگوریتمی را ارائه کردند که در زمان $O(n \log n)$ این کار را انجام می‌داد. بعدها Heffernan و Mitchell [۳۸] نیز الگوریتمی با زمان اجرای $O(n + h \log h)$ معرفی کردند.

در ادامه الگوریتم Asano [۶] را ارائه می‌کنیم که چندضلعی قابل دید نقطه q را در آرایشی از چندضلعی‌ها در زمان $O(n \log h)$ محاسبه می‌کند. هرچند الگوریتم برای چندضلعی حفره‌دار ارائه نشده است، می‌توان یک چندضلعی حفره‌دار را به آرایشی از چندضلعی‌ها تبدیل کرد. برای این کار، هر حفره را یک چندضلعی در نظر می‌گیریم و نیز مرز بیرونی چندضلعی حفره‌دار را به دو قسمت تقسیم کرده و با اضافه کردن یال‌هایی به دو چندضلعی مجزا تبدیل می‌کنیم.

ورودی الگوریتم، تعداد h چندضلعی P_1، P_2، . . . و P_h است که یکدیگر را قطع نمی‌کنند و تعداد کل رئوس آن‌ها n است. علاوه بر این نقطه q در بیرون همه این چندضلعی‌ها داده شده و الگوریتم باید چندضلعی قابل دید q را محاسبه کند. این کار با پویش زاویه‌ای[۸] حول q انجام می‌شود. پویش زاویه‌ای حول q در مدت زمان $O(n \log n)$ به‌راحتی با مرتب کردن همه پاره‌خط‌ها حول q قابل انجام است. اما این کار در زمان $O(n \log h)$ با پردازش مناسب هر چندضلعی پیش از ساخت لیست مرتب شده یال‌ها نیز میسر است.

اگر $\mathrm{V}_{\overline{P}_i}(q)$ چندضلعی قابل دید q در بیرون چندضلعی P_i باشد، می‌توان با تغییر اندکی در الگوریتم Lee [۴۴]، $\mathrm{V}_{\overline{P}_i}(q)$ را برای همه چندضلعی‌های P_i در زمان کلی $O(n)$ محاسبه کرد. واضح است که یال‌های $\mathrm{V}_{\overline{P}_i}(q)$ دنباله‌ای مرتب بر اساس زاویه حول q است.

در مرحله بعدی، هر $\mathrm{V}_{\overline{P}_i}(q)$ را به زیردنباله‌های بیشینه‌ای از یال‌های به هم پیوسته P_i تقسیم می‌کنیم. این کار با شکستن $\mathrm{V}_{\overline{P}_i}(q)$ و با حذف یال‌هایی صورت می‌گیرد که روی مرز P_i نیستند و امتداد آن‌ها از q می‌گذرد. اگر m تعداد کل زیردنباله‌های بیشینه باشد، پویش زاویه‌ای این زیردنباله‌ها در

[۸] Angular sweep

زمان $O(n + m \log h)$ قابل انجام است.

در ضمن پویش، در هر لحظه ما لیستی از دنباله‌های بیشینه‌ای که توسط شعاع پویش قطع می‌شود، نگه‌داری می‌کنیم. این لیست به ترتیب فاصله تا q مرتب شده است. در طی پویش، دو نوع واقعه اتفاق می‌افتد. این دو واقعه، اضافه شدن یک دنباله جدید به لیست و حذف یک دنباله از لیست است. این وقایع فقط در دو سر انتهایی دنباله‌ها اتفاق می‌افتد. چون چندضلعی‌های آرایش یکدیگر را قطع نمی‌کنند، این دنباله‌ها نیز متقاطع نیستند. بنابراین در بین دو سر انتهایی دنباله‌ها، کافی است فقط به دنباله یال‌هایی توجه کنیم که به q نزدیک‌ترند. لیست مرتب شده دنباله‌ها هیچ‌گاه بیشتر از h دنباله را در خود جای نمی‌دهد، بنابراین اضافه و حذف کردن دنباله جدید در زمان $O(\log h)$ قابل انجام است.

با پویش زاویه‌ای حول q بدین روش و نگه‌داری نزدیک‌ترین دنباله به q در هر زمان، چندضلعی قابل دید q را می‌توان در زمان $O(n + m \log h)$ محاسبه کرد. چون m حداکثر $O(n)$ است، در بدترین حالت، الگوریتم در زمان $O(n \log h)$ به اتمام می‌رسد. البته در برخی از کاربردها، مثلاً وقتی چندضلعی‌های P_i محدب باشند، هر چندضلعی تعداد محدودی دنباله بیشینه تولید می‌کند. در این حالات، زمان اجرای الگوریتم به $O(n + h \log h)$ بهبود می‌یابد.

از الگوریتم بالا، و روش تبدیل چندضلعی حفره‌دار به آرایش چندضلعی‌ها می‌توان قضیه زیر را نتیجه گرفت.

قضیه ۲.۲ چندضلعی قابل دید نقطه q را در داخل چندضلعی حفره‌دار P که از h حفره و n رأس تشکیل شده است، می‌توان در زمان $O(n \log h)$ محاسبه کرد.

۳.۱.۲ چندضلعی قابل دید درون چندضلعی ساده برای نقاط پرس‌وجو

فرض کنید q_1، q_2، . . . ، q_m، m نقطه در درون چندضلعی ساده P با تعداد کل n رأس باشد و ما باید چندضلعی قابل دید هر یک از این نقاط را در P محاسبه کنیم. چون محاسبه $\mathrm{V}(q_i)$ برای هر q_i با استفاده از روش‌های ذکر شده در بخش ۱.۱.۲ به زمان $O(n)$ احتیاج دارد، مسأله در مجموع در مدت $O(mn)$ قابل حل است. نکته مهم دیگری که در مورد این الگوریتم‌ها وجود دارد این است که این الگوریتم‌ها به اندازه $\mathrm{V}(q_i)$ توجهی ندارند. بنابراین اگر m در مقایسه با n عدد بزرگی باشد و اندازه متوسط $\mathrm{V}(q_i)$ کمتر از n باشد، راه‌حل مناسب‌تر این است که ابتدا P را پیش‌پردازش کنیم تا زمان محاسبه $\mathrm{V}(q_i)$ برای هر q_i کاهش یابد. مثلاً اگر بتوان با استفاده از پیش‌پردازش، زمان محاسبه $\mathrm{V}(q_i)$ را به $O(\log n + |\mathrm{V}(q)|)$ رساند، زمان محاسبه همه چندضلعی‌های قابل دید به $O(m \log n + \sum_{i=1}^{m} |\mathrm{V}(q)|)$ کاهش می‌یابد که اگر m در مقایسه با n خیلی بزرگتر باشد و اندازه متوسط $\mathrm{V}(q)$ کمتر از n باشد، این کاهش زمان اجرا بسیار مطلوب است.

این نوع مسائل که در آن‌ها محاسبات بر روی یک دامنه ثابت انجام می‌شود، و با دریافت هر

ورودی، خروجی محاسبه می‌شود، در هندسه محاسباتی به مسائل پرس‌وجو معروف هستند. کارایی یک الگوریتم پرس‌وجو، بر اساس زمان پرس‌وجو، حافظه مصرفی الگوریتم و زمان لازم برای پیش‌پردازش آن ارزیابی می‌شود.

اولین الگوریتم محاسبه $\mathrm{V}(q)$ در چندضلعی‌های ساده، با انجام یک مرحله پیش‌پردازش اولیه، توسط Guibas و همکارانش [۳۵] ارائه شد. آن‌ها نشان دادند می‌توان $\mathrm{V}(q)$ را در زمان $O(\log n + k)$ گزارش داد. الگوریتم آن‌ها در مرحله پیش‌پردازش، در زمان $O(n^3)$، P را به بخش‌هایی تقسیم کرده و این بخش‌ها را با استفاده از $O(n^3)$ حافظه ذخیره می‌کرد و در زمان پرس‌وجو با تعیین بخشی که نقطه پرس‌وجو در آن قرار گرفته است، $\mathrm{V}(q)$ را محاسبه می‌کرد.

Bose و همکارانش [۱۱] نیز با استفاده از روش مشابهی، الگوریتمی را ساختند که در همان زمان پرس‌وجو، $\mathrm{V}(q)$ را محاسبه می‌کند. زمان پیش‌پردازش در این روش $O(n^3 \log n)$ و حافظه مصرفی $O(n^3)$ است. در صورتی که نقطه q در بیرون چندضلعی P قرار بگیرد، این الگوریتم می‌تواند چندضلعی قابل دید بیرونی P را از نقطه q در همان زمان محاسبه کند. Aronov و همکارانش [۵] با تقسیم P به قطعه‌های کانونی، توانستند زمان پیش‌پردازش و حافظه مصرف‌شده را به ترتیب به $O(n^2 \log n)$ و $O(n^2)$ کاهش دهند، ولی روش آن‌ها باعث افزایش زمان پرس‌وجو به $O(\log^2 n + k)$ شد.

۴.۱.۲ چندضلعی قابل دید درون چندضلعی حفره‌دار برای نقاط پرس‌وجو

در این بخش روش‌هایی را برای محاسبه $\mathrm{V}(q)$ در چندضلعی حفره‌دار مرور می‌کنیم که همانند بخش قبلی، با اضافه کردن یک مرحله پیش‌پردازش به دنبال کاهش زمان پرس‌وجو هستند. چندضلعی P با h حفره و تعداد کل رأس n را در نظر بگیرید که باید برای نقطه q، $\mathrm{V}(q)$ را محاسبه کنیم. می‌دانیم طبق آن‌چه در بخش ۲.۱.۲ گفته شد، می‌توان $\mathrm{V}(q)$ را با استفاده از الگوریتم Asano [۶] در زمان $O(n \log h)$ محاسبه کرد.

الگوریتم‌های ارائه شده

اولین الگوریتم برای محاسبه $\mathrm{V}(q)$ با انجام پیش‌پردازش توسط Asano و همکارانش [۷] معرفی شد. این الگوریتم با انجام پیش‌پردازشی در مدت زمان $O(n^2)$ با مصرف $O(n^2)$ حافظه، $\mathrm{V}(q)$ را در زمان پرس‌وجوی $O(n)$ محاسبه می‌کند. در این روش با استفاده از تبدیل دوگان نقطه و خط، رئوس P به ترتیب زاویه قرارگیری حول q مرتب می‌شوند. سپس با استفاده از مثلث‌بندی و عملیات اجتماع مجموعه‌ها، بخش‌هایی از یال‌های P که از q قابل دید هستند، مشخص می‌شوند. قابل توجه است که هرچند ممکن است اندازه $\mathrm{V}(q)$ بسیار کمتر از n باشد، زمان پرس‌وجو در این الگوریتم بدون توجه به اندازه $\mathrm{V}(q)$، همواره $O(n)$ است.

الگوریتم دوم برای این مسأله، توسط Vegter [۶۳] ارائه شد، که نشان داد می‌توان چندضلعی قابل دید را در مدت زمانی که وابسته به اندازه $\mathrm{V}(q)$ است، محاسبه کرد. در روش Vegter، با پیش‌پردازشی در مدت $O(n^2 \log n)$ با صرف حافظه $O(n^2)$، می‌توان زمان پرس‌وجو را به $O(k \log(n/k))$ رساند که در آن k اندازه $\mathrm{V}(q)$ است.

در صورتی که P یک چندضلعی محدب باشد، با استفاده از روشی که Pocchiola و Vegter [۵۷] ارائه دادند می‌توان $\mathrm{V}(q)$ را در زمان $O(k \log n)$ محاسبه کرد. در طی پیش‌پردازش، الگوریتم آن‌ها P را در زمان $O(n \log n + E)$ با استفاده از حافظه $O(E)$ به یک مجتمع دیداری[۹] تبدیل می‌کند تا $\mathrm{V}(q)$ را در زمانی وابسته به اندازه خروجی محاسبه کند. در روابط زمان و حافظه پیش‌پردازش، E برابر تعداد یال‌های گراف قابلیت دید P است، که در آغاز فصل معرفی شد.

الگوریتم ارائه شده بعدی برای این مسأله، توسط زارعی و قدسی [۶۵] معرفی شد. در این روش چندضلعی حفره‌دار P با برش از روی برخی قطرها، به تعدادی چندضلعی ساده شکسته می‌شود و سپس با استفاده از روش [۱۱]، در هرکدام از چندضلعی‌های ساده، اشتراک $\mathrm{V}(q)$ با آن چندضلعی محاسبه می‌شود. در این روش، با صرف زمان $O(n^3 \log n)$ و حافظه $O(n^3)$ برای پیش‌پردازش، چندضلعی قابل دید $\mathrm{V}(q)$ در زمان پرس‌وجوی $O((1 + \min(h, k)) \log n + k)$ محاسبه می‌شود.

الگوریتم ارائه شده آخر در این زمینه از آن Inkulu و Kapoor [۴۰] در سال ۲۰۰۹ است. آن‌ها ابتدا چندضلعی P را به چندضلعی‌های ساده‌ای با نام دالان تقسیم کردند و نشان دادند که برای نقطه q، $\mathrm{V}(q)$ را می‌توان در هر دالان به‌طور جداگانه محاسبه کرد. سپس با استفاده از الگوریتم شلیک پرتو[۱۰] و یا محاسبه چندضلعی قابل دید در چندضلعی ساده، مسأله محاسبه چندضلعی قابل دید در چندضلعی حفره‌دار را حل کردند. بسته به الگوریتم استفاده شده، زمان پیش‌پردازش، حافظه و زمان پرس‌وجو به ترتیب $O(n^2 \log n)$، $O(n^2)$، $O((1 + \min(h, k)) \log^2 n + h + k)$ و یا $O(T + E + n \log n)$، $O(\min(E, hn) + n)$ و $O(k \log n + h)$ خواهد بود. در این رابطه‌ها T زمان لازم برای مثلث‌بندی P است که برابر $O(n + h \log^{1+\epsilon} m)$ برای عدد ثابت مثبت ϵ است.

همان‌طور که در فصل ۴ خواهیم دید، ما دو نتیجه جدید در مورد این مسأله به‌دست آورده‌ایم. اولین نتیجه الگوریتمی با زمان پرس‌وجوی لگاریتمی برای محاسبه چندضلعی قابل دید و دومین نتیجه، ایجاد یک توازن بین حافظه مورد استفاده برای پیش‌پردازش و زمان پرس‌وجو است.

محاسبه چندضلعی قابل دید در چندضلعی حفره‌دار در زمان $O(n)$

در این بخش الگوریتم ارائه شده توسط Asano و همکارانش [۷] را توضیح می‌دهیم که با استفاده از آن می‌توان چندضلعی قابل دید یک نقطه در آرایشی از پاره‌خط‌ها را در زمان $O(n)$ محاسبه کرد. زمان

[۹] Visibility complex
[۱۰] Ray shooting

و حافظه لازم برای پیش‌پردازش در این الگوریتم، $O(n^۲)$ است. هر چند این الگوریتم برای آرایش پاره‌خط‌ها ارائه شده است، می‌توان آن را برای چندضلعی‌های حفره‌دار نیز به‌کار برد، زیرا با جدا کردن یال‌های مرز یک چندضلعی از یکدیگر، آرایشی از پاره‌خط‌ها به‌دست می‌آید.

ورودی الگوریتم، مجموعه $S = \{s_۱, \ldots, s_n\}$ از پاره‌خط‌های موجود در آرایش است. این پاره‌خط‌ها فقط در نقاط انتهایی ممکن است با یکدیگر تلاقی داشته باشند. در زمان پرس‌وجو نقطه q نیز برای محاسبه چندضلعی قابل دید در اختیار الگوریتم قرار می‌گیرد.

همانند الگوریتم بخش ۲.۱.۲، قبل از محاسبه چندضلعی قابل دید، باید پاره‌خط‌های S به ترتیب زاویه‌ای حول q مرتب شوند. اما نمی‌توان از روش پویش زاویه‌ای استفاده کرد، زیرا در این‌صورت به زمان $O(n \log h)$ نیاز خواهیم داشت. همان‌گونه که در ادامه می‌بینیم، یکی از تکنیک‌های اصلی استفاده شده در این الگوریتم، تبدیل دوگان است، که باعث کاهش زمان مرتب کردن پاره‌خط‌ها می‌شود. ما الگوریتم را در سه بخش ارائه می‌کنیم. در بخش اول، مرتب کردن زاویه‌ای را توضیح می‌دهیم. در بخش دوم روش مرتب کردن پاره‌خط‌ها بر اساس فاصله نسبت به q و در بخش آخر چگونگی ساخت دنباله یال‌های قابل دید از q بیان می‌شود.

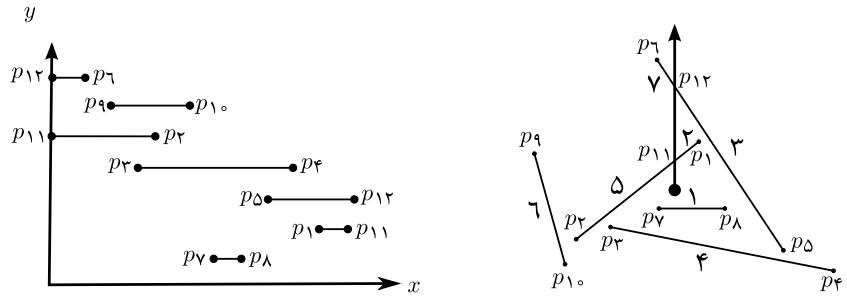
مرتب کردن نقاط به ترتیب زاویه‌ای و تبدیل دوگان. برای شروع، ابتدا باید با مفهوم تبدیل دوگان آشنا شویم. تبدیلی را تصور کنید که نقطه $p = (p_x, p_y)$ را به خط $p^* : y = p_x x - p_y$ و خط $l : y = mx + b$ را به نقطه $l^* = (m, -b)$ نگاشت می‌کند. به‌راحتی ثابت می‌شود که این تبدیل، رابطه تعلق و ترتیب را حفظ می‌کند [۲۱]، یعنی $p \in l$ اگگر $d^* \in p^*$ و p بالای l قرار می‌گیرد اگگر l^* بالای p^* قرار بگیرد.

فرض کنید $\mathcal{A}(P^*)$ آرایش مجموعه خطوط $P^* = \{p_۱^*, \ldots, p_n^*\}$ را نشان دهد. خط q^* را که دوگان نقطه پرس‌وجو است می‌توان در زمان $O(n)$ به این آرایش اضافه کرد [۱۸]. همچنین فرض کنید r_q پرتوی را نشان دهد که از q به سمت بالا ساطع می‌شود و l_q خط حامل[۱۱] آن باشد. نیم‌صفحه h_l را نیم‌صفحه سمت چپ خط عمودی گذرنده از q در نظر بگیرید. وقتی r_q حول q به اندازه π دوران می‌کند، r_q تمام نیم‌صفحه h_l را جاروب می‌کند. همچنین نقطه l_q^* در صفحه دوگان روی خط q^* از $-\infty$ تا $+\infty$ حرکت می‌کند. وقتی r_q از روی نقطه p_i عبور می‌کند، نقطه l_q^* روی خط p_i^* قرار می‌گیرد. بنابراین ترتیب نقاط p_i حول q به‌ترتیب افزایش زاویه با محور $+y$ در خلاف جهت عقربه‌های ساعت، برابر است با ترتیب تلاقی خطوط p_i^* با خط q^*. این مطلب برای نیم‌صفحه سمت راست q نیز صادق است. بنابراین می‌توان لم زیر را که توسط Asano و همکارانش [۷] اثبات شده است، نتیجه گرفت.

لم ۱.۲ با پیش‌پردازش n نقطه در صفحه با صرف حافظه و زمان $O(n^۲)$، می‌توان نقاط را حول هر نقطه پرس‌وجو در زمان $O(n)$ مرتب کرد.

مرتب کردن پاره‌خط‌ها به ترتیب فاصله نسبت به q. برای ایجاد یک ترتیب بین پاره‌خط‌ها بر

[۱۱] Supporting line



ب الف

شکل ۱۱.۲ : الف. ترتیب زاویه‌ای رئوس حول q. ب. ترتیب پاره‌خط‌ها بر اساس فاصله و زاویه نسبت به q.

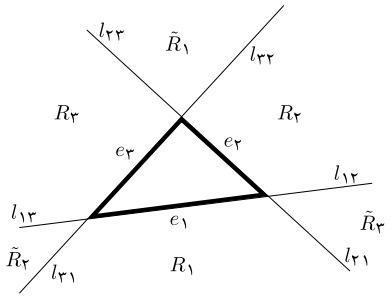
اساس فاصله تا q، به هر پاره‌خط s_i، مرتبه‌ای به‌صورت $\text{rank}(s_i)$ نسبت می‌دهیم، به‌گونه‌ای که اگر s_i از s_j به q نزدیک‌تر باشد، آنگاه $\text{rank}(s_i) < \text{rank}(s_j)$. در این بخش نحوه تعیین مرتبه پاره‌خط‌ها را در زمان $O(n)$ نسبت به هر نقطه پرس‌وجوی q توضیح می‌دهیم. زمان و حافظه مورد نیاز برای پیش‌پردازش به ترتیب $O(n \log n)$ و $O(n)$ است.

برای محاسبه مرتبه پاره‌خط‌ها، گراف جهت‌دار ترتیب جزئی بین پاره‌خط‌های S را، که میزان نزدیکی به q را نشان می‌دهد، ایجاد نموده و سپس با مرتب‌سازی توپولوژیکی گره‌های آن این گراف را تبدیل به یک رابطه ترتیب کامل می‌کنیم. در نهایت مرتبه یک پاره‌خط، مرتبه آن در ترتیب کامل خواهد بود.

رابطه $\prec_q$ را روی S در نظر بگیرید. می‌نویسیم $s_i \prec_q s_j$، اگر پرتوی از q ساطع شود به‌گونه‌ای که هر دوی s_i و s_j را قطع کند و تلاقی آن با s_i به q نزدیک‌تر از تلاقی آن با s_j باشد. گراف رابطه $\prec_q$ ممکن است دور داشته باشد، بنابراین $\prec_q$ همیشه یک ترتیب جزئی نیست. برای تبدیل آن به یک ترتیب جزئی، پاره‌خط‌هایی را که توسط پرتو رو به بالای ساطع شده از q قطع می‌شوند، به دو بخش تقسیم می‌کنیم (شکل ۱۱.۲– الف). اگر مجموعه جدید پاره‌خط‌ها را S_q بنامیم، رابطه $\prec_q$ روی S_q یک ترتیب جزئی خواهد بود.

برای محاسبه ترتیب جزئی $\prec_q$ روی S_q، یک ترتیب کامل را که $\prec_q$ در آن تعبیه شده است، پیدا می‌کنیم. این ترتیب کامل، با مثلث‌بندی آرایش پاره‌خط‌های S به‌دست می‌آید. چون مثلث بندی T تنها به S وابسته است و مستقل از q است، می‌توان آن را در مرحله پیش‌پردازش با صرف زمان $O(n \log n)$ و با حافظه $O(n)$ محاسبه کرد.

وقتی یک نقطه پرس‌وجوی q داده می‌شود، مثلثی از T که q در آن قرار گرفته، می‌یابیم و سه یال از q به رئوس این مثلث رسم می‌کنیم. همچنین یک پرتو از q رو به بالا رسم می‌کنیم و مثلث‌بندی



شکل ۱۲.۲ : الف. نحوه ساخت گراف جهت‌دار $G(T_q)$

T را بر این اساس به‌روز می‌رسانیم. اگر مثلث‌بندی جدید را T_q بنامیم، T_q در زمان $O(n)$ با تغییر T به‌دست می‌آید.

حال یک گراف جهت‌دار $G(T_q)$ را روی یال‌های T_q ایجاد می‌کنیم. اگر F مثلثی از T_q باشد که از یال‌های $e_۱$، $e_۲$ و $e_۳$ تشکیل شده است، همانند شکل ۱۲.۲، پرتوها و نواحیی که از امتداد دادن یال‌های مثلث به‌وجود می‌آید نام‌گذاری می‌کنیم. در هر مثلث F از مثلث‌بندی، برحسب محل قرارگیری q در این نواحی، یال‌هایی بین $e_۱$، $e_۲$ و $e_۳$ در گراف $G(T_q)$ قرار می‌دهیم.

- اگر q در داخل $R_۱$ قرار دارد، یال‌هایی از $e_۲$ و $e_۳$ به $e_۱$ در $G(T_q)$ رسم می‌کنیم. وقتی q در داخل $R_۲$ و $R_۳$ است، به روشی مشابه عمل می‌کنیم.
- اگر q در داخل $\tilde{R}_۱$ قرار دارد، یال‌هایی از $e_۱$ به $e_۲$ و $e_۳$ در $G(T_q)$ رسم می‌کنیم. وقتی q در داخل $\tilde{R}_۲$ و $\tilde{R}_۳$ است، به روشی مشابه عمل می‌کنیم.
- اگر q روی پرتو $l_{۱۲}$ که $R_۲$ را از $\tilde{R}_۳$ جدا می‌کند، باشد، یالی از $e_۳$ به $e_۲$ در $G(T_q)$ رسم می‌کنیم. وقتی q روی $l_{۲۱}$، $l_{۳۱}$، $l_{۱۳}$، $l_{۲۳}$ و یا $l_{۳۲}$ باشد، به روشی مشابه عمل می‌کنیم.
- اگر q روی یال e_i یا در داخل مثلث F قرار داشته باشد، هیچ یالی اضافه نمی‌شود.

گراف $G(T_q)$ که بدین طریق حاصل می‌شود، گرافی بدون دور است و بستار ترایایی[۱۲] آن شامل $\prec_q$ هم می‌شود. بنابراین با مرتب‌سازی توپولوژیکی گراف $G(T_q)$، ترتیب کاملی به‌دست می‌آید که $\prec_q$ در آن تعبیه شده است و درنتیجه مرتبه اعضای S_q را به ما می‌دهد. چون تعداد رأس‌ها و یال‌های گراف $G(T_q)$، $O(n)$ است، مرتب‌سازی توپولوژیکی آن در زمان $O(n)$ قابل انجام است.

بعد از پاسخ‌گویی به پرس‌وجو، باید T دست‌نخورده باقی بماند. بدین منظور، می‌توان برای ساخت T_q از یک نسخه کپی T استفاده کرد و یا تغییرات انجام شده برای تعیین مرتبه اعضای S_q را خنثی[۱۳] کنیم.

[۱۲] Transitive closure
[۱۳] Undo

محاسبه چندضلعی قابل دید q. طبق آن‌چه تاکنون دیدیم، با داشتن نقطه پرس‌وجوی q، می‌توان نقاط انتهایی پاره‌خط‌های S_q را به ترتیب زاویه‌ای مرتب کرد و نیز ترتیبی را بین پاره‌خط‌ها برقرار کرد که میزان نزدیکی به q را نشان دهد. نقاط انتهایی مرتب شده را می‌توان با اعداد صحیح ۱ تا $2n+1$ نشان داد. مرتبه پاره‌خط‌ها در ترتیب فاصله تا q را نیز می‌توان با اعداد ۱ تا $|S_q|$ که از $2n$ کمتر است نشان داد. می‌توان شماره ترتیب رئوس و مرتبه پاره‌خط‌ها را به عنوان مؤلفه‌های x و y در نظر گرفت و پاره‌خط‌ها را همانند شکل ۱۱.۲– ب نمایش داد که در آن پاره‌خط s_i از (l_i, i) تا (r_i, i) امتداد دارد.

اکنون مسأله محاسبه چندضلعی قابل دید تبدیل به مسأله تعیین پایین‌ترین پاره‌خط در هر x در این نمودار می‌شود. این مسأله با استفاده از الگوریتم خطی اجتماع مجموعه‌ها[۱۴] که توسط Gabow و Tarjan [۳۰] ارائه شده است قابل حل است. این الگوریتم انجام دنباله‌ای از عملیات $\text{find}(x)$ و $\text{link}(x)$ را در زمان خطی ممکن می‌سازد. تابع $\text{find}(x)$ بزرگترین عنصر مجموعه‌ای که x را شامل می‌شود به ما داده و تابع $\text{link}(x)$ مجموعه‌ای که شامل x است را با مجموعه‌ای که شامل $x+1$ است، ترکیب می‌کند.

برای استفاده از الگوریتم اجتماع مجموعه‌ها، کار را با N_x+2 مجموعه مجزای $\{0\}$، $\{1\}$، ...، $\{N_x\}$ و $\{N_x+1\}$ آغاز می‌کنیم، که N_x بزرگترین مقدار مؤلفه x دو سر پاره‌خط‌هاست ($N_x \leq 2n+1$). به‌تدریج که محاسبه پایین‌ترین پاره‌خط به‌پیش می‌رود، مجموعه‌هایی که توسط الگوریتم اجتماع پاره‌خط‌ها نگه‌داری می‌شود، بازه‌هایی را نشان می‌دهد که ما پایین‌ترین پاره‌خط را در هر بخشی از آن بازه‌ها می‌دانیم. الگوریتم در حین اجرا، آرایه vis[] را پر می‌کند و در انتها هر عنصر vis[x]، شماره i پایین ترین پاره‌خط s_i را در بازه بین x و $x+1$ نگه‌داری می‌کند.

طبق استدلال‌های صورت گرفته توسط Asano و همکارانش در [۷]، و امکان تبدیل یک چندضلعی حفره‌دار به آرایشی از پاره‌خط‌ها، می‌توان قضیه زیر را در مورد چندضلعی قابل دید یک نقطه پرس‌وجو بیان کرد.

قضیه ۳.۲ یک چندضلعی حفره‌دار P با تعداد رأس n را می‌توان در مدت زمان $O(n^2)$ و با صرف همین مقدار حافظه پیش‌پردازش کرد، به‌گونه‌ای که بتوان چندضلعی قابل دید هر نقطه پرس‌وجوی q را در زمان $O(n)$ محاسبه نمود.

۲.۲ قابلیت دید پاره‌خط

در این فصل مسأله تعیین قدرت دید پاره‌خط را مورد بررسی قرار می‌دهیم و مسائل مختلف مرتبط با این بخش و راه‌حل‌های ارائه شده در مقالات مختلف را بررسی می‌کنیم. این مسأله، بعد از قدرت دید نقطه، پایه‌ای‌ترین مسأله برای حالت‌های کلی‌تر اشیاء است. در بخش ۱.۲.۲ نتایج موجود برای محاسبه

[۱۴] Set union

چندضلعی قابل دید کامل پاره‌خط و در بخش ۲.۲.۲ نتایج موجود برای چندضلعی قابل دید ضعیف پاره‌خط عنوان می‌شوند.

۱.۲.۲ چندضلعی قابل دید کامل پاره‌خط

در ابتدا روش‌های مربوط به محاسبه چندضلعی قابل دید کامل یک پاره‌خط را مطالعه می‌کنیم. همان‌طور که در آغاز فصل ذکر شد، چندضلعی قابل دید کامل یک پاره‌خط در درون یک چندضلعی، مجموعه همه نقاطی از چندضلعی است که از همه نقاط پاره‌خط قابل دید هستند. فرض کنید P یک چندضلعی ساده و پاره‌خط pq در درون آن قرار داشته باشد (شکل ۱۳.۲– الف). z را نقطه‌ای فرض کنید که هم از p و هم از q قابل دید است. چون P حفره ندارد، مثلث $\triangle zpq$ به‌طور کامل در درون P قرار می‌گیرد. بنابراین z در چندضلعی قابل دید کامل pq قرار دارد. واضح است که هر نقطه در چندضلعی قابل دید کامل pq، هم از p و هم از q قابل دید است. بنابراین می‌توان لم زیر را نتیجه گرفت که روشی را برای محاسبه چندضلعی قابل دید کامل pq در P در زمان $O(n)$ ارائه می‌دهد.

لم ۲.۲ چندضلعی قابل دید کامل پاره‌خط pq در داخل چندضلعی ساده P از اشتراک $\mathrm{V}(p)$ و $\mathrm{V}(q)$ به‌دست می‌آید.

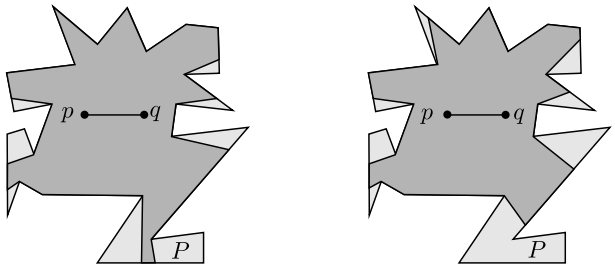
در مورد چندضلعی حفره‌دار P وضع کمی پیچیده‌تر است، زیرا در این صورت ممکن است مثلث $\triangle zpq$ شامل حفره باشد و در نتیجه z همه pq را نبیند. اما در این حالت می‌توان لم را به صورت زیر تغییر داد تا چندضلعی قابل دید کامل pq در چندضلعی حفره‌دار نیز قابل محاسبه باشد.

لم ۳.۲ چندضلعی قابل دید کامل پاره‌خط pq در داخل چندضلعی حفره‌دار P، برابر چندضلعی قابل دید q در $\mathrm{V}(p)$ است.

لم بالا که در [۸] اثبات شده است، نشان می‌دهد که برای محاسبه چندضلعی قابل دید کامل پاره‌خط pq در P، می‌توان $\mathrm{V}(p)$ را یافته و در درون آن چندضلعی قابل دید q را حساب کرد. چون محاسبه چندضلعی قابل دید $\mathrm{V}(p)$ در زمان $O(n\log n)$ و چندضلعی قابل دید q در چندضلعی ساده $\mathrm{V}(p)$ در زمان $O(n)$ انجام می‌شود، چندضلعی قابل دید کامل پاره‌خط pq در P در زمان $O(n\log n)$ قابل محاسبه است.

۲.۲.۲ چندضلعی قابل دید ضعیف پاره‌خط

در آغاز فصل، چندضلعی قابل دید ضعیف یک پاره‌خط در درون یک چندضلعی، به‌صورت مجموعه همه نقاطی که حداقل از یک نقطه پاره‌خط قابل دید باشند، تعریف شد. برخلاف چندضلعی قابل



ب الف

شکل ۱۳.۲ : الف. چندضلعی قابل دید قوی ب. چندضلعی قابل دید ضعیف پاره‌خط pq.

دید کامل، محاسبه چندضلعی قابل دید ضعیف، چه در چندضلعی‌های ساده و چه در چندضلعی‌های حفره‌دار، به راحتی قابل انجام نیست. فرض کنید P یک چندضلعی ساده و pq پاره‌خطی در P باشد. اولین الگوریتم‌های غیر بدیهی برای این مسأله توسط ElGindy [۲۵]، Lee و Lin [۴۵]، و Chazelle و Guibas [۱۷] با زمان اجرای $O(n \log n)$ ارائه شد. Guibas و همکارانش [۳۴] نشان دادند که مسأله در صورتی که مثلث‌بندی P از قبل موجود باشد، در زمان $O(n)$ قابل حل است. چون چندضلعی ساده P را می‌توان با استفاده از الگوریتم Chazelle [۱۴] در زمان خطی مثلث‌بندی کرد، الگوریتم Guibas و همکارانش در زمان خطی قابل استفاده است.

در مورد چندضلعی قابل دید ضعیف در داخل چندضلعی حفره‌دار، نتیجه به‌دست آمده چندان دلچسب نیست، زیرا Suri و O'Rourke [۶۲] ثابت کرده‌اند که در بدترین حالت، اندازه چندضلعی قابل دید ضعیف یک پاره‌خط در داخل یک چندضلعی حفره‌دار $O(n^۴)$ است. آن‌ها همچنین الگوریتمی را ارائه کرده‌اند که می‌تواند در مدت $O(n^۴)$ این چندضلعی را محاسبه کند. اما با این وجود، نکته جالب توجه این است که این چندضلعی، از اجتماع حداکثر $O(n^۲)$ مثلث تشکیل شده است و می‌توان در زمان $O(n^۲)$ این مثلث‌ها را تعیین کرد. ولی برای تعیین دقیق رئوس چندضلعی و ترتیب بین آن‌ها، زمان $O(n^۴)$ مورد نیاز است.

فصل ۳

آزمون قابلیت دید دو شیء از هم (۱)

در فصل ۲ قدرت دید ضعیف اشیاء غیرنقطه‌ای تعریف شد. این شکل قدرت دید کاربردهای زیادی دارد. مثلاً یک چشمه نور غیرنقطه‌ای، مثل لامپ‌های فلورسنت، فضایی را در محیط روشن می‌کند، که طبق قدرت دید ضعیف مشخص می‌شود. در این بخش، مسأله‌ای که مورد بررسی قرار می‌گیرد، تعیین قابلیت دید ضعیف یک شیء از یک شیء دیگر است. تعیین روشن بودن بخشی از یک شیء در محیطی که توسط یک لامپ فلورسنت روشن می‌شود، مثالی شهودی از کاربرد این مسأله است.

در این مسأله به‌صورت دقیق‌تر، دو شیء s و t که در داخل یک چندضلعی P قرار گرفته‌اند، داده شده‌اند. سؤال مطرح شده وجود رابطه دید ضعیف بین s و t است. Gajentaan و Overmars [۳۱] نشان دادند که اگر s و t دو پاره‌خط باشند و P یک چندضلعی حفره‌دار، مسأله 3SUM–سخت است. مسائل 3SUM–سخت، مسائلی هستند که قابل کاهش به مسأله 3SUM هستند. تاکنون الگوریتمی با زمان کمتر از $\Theta(n^2)$ برای مسأله 3SUM و مسائل 3SUM–سخت ارائه نشده است، بنابراین حدس زده می‌شود که $\Theta(n^2)$ حد پایین این مسائل است. Wismath [۶۴] راه‌حلی ارائه کرده است که مسأله وجود رابطه دید ضعیف بین s و t را در زمان $O(n^2)$ حل می‌کند.

در این بخش ما مسأله را وقتی s و t، یک نقطه، پاره‌خط و یا چندضلعی محدب باشند بررسی می‌کنیم. همچنین هر حالت را در سه شکل مورد مطالعه قرار می‌دهیم. در شکل اول، صحنه به همراه s و t مشخص می‌شوند و ما بدون هیچ پیش‌پردازشی مسأله را حل می‌کنیم. در شکل دوم، صحنه به همراه s پیش‌پردازش شده و t در زمان پرس‌وجو، داده می‌شود. در شکل سوم مسأله، هم s و هم t در زمان پرس‌وجو مشخص می‌شوند و ما فقط صحنه را می‌توانیم پیش‌پردازش کنیم.

نتایج اصلی به‌دست آمده در این قسمت به شرح زیر هستند.

- با پیش‌پردازش چندضلعی حفره‌دار P به همراه نقطه s داخل آن در زمان $O(n \log n)$ و با استفاده

از حافظه $O(n)$، می‌توان وجود رابطه دید ضعیف بین s و پاره‌خط دلخواه t را در زمان $O(\log n)$ مشخص کرد.

- وجود رابطه دید بین دو نقطه پرس‌وجوی s و t را در چندضلعی حفره‌دار P می‌توان در زمان $O(\sqrt{n}\log n)$ تشخیص داد، به‌شرطی‌که P را در زمان $O(n\sqrt{n}\log^{۴/۳} n)$ با حافظه $O(n\log^{۲} n)$ پیش‌پردازش کنیم.
- در چندضلعی حفره‌دار P، وجود رابطه دید ضعیف بین دو پاره‌خط یا چندضلعی محدب s و t را می‌توان با پیش‌پردازش P در زمان $O(n^{۲+\epsilon})$ و با استفاده از حافظه $O(n^{۲})$، در زمان پرس‌وجوی $O(n^{۱+\epsilon})$ تشخیص داد.

علاوه بر نتایج بالا، روشی را برای تعیین پاره‌خط‌های قطع شده در صفحه توسط دو پاره‌خط، با انجام پیش‌پردازش ارائه می‌کنیم. این روش در سایر مسائل جستجوی بازه[۱] نیز قابل استفاده است.

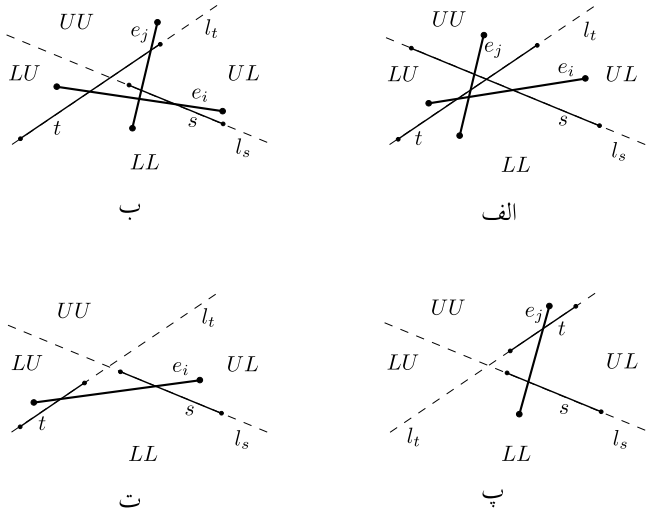
در ادامه این فصل، در بخش ۱.۳ پیش‌نیازهای لازم برای حل مسأله اصلی ارائه می‌شوند. ابتدا مسأله یافتن پاره‌خط متقاطع با دو پاره‌خط داده شده را حل می‌کنیم و سپس شرایط لازم برای قابل دید بودن دو شیء پیچیده را مطرح می‌کنیم. در بخش ۲.۳ روش تعیین قابل دید بودن یک شیء از یک ناظر نقطه‌ای در شرایط مختلف و در بخش ۳.۳ همین مسأله برای ناظر به شکل پاره‌خط عنوان می‌شود. در انتها در بخش ۴.۳ مهمترین نتیجه برای این مسائل یعنی تعیین قابل دید بودن دو شیء پرس‌وجو در داخل یک چندضلعی حفره‌دار ارائه می‌شود.

۱.۳ الگوریتم‌های مقدماتی

پیش از آن‌که به مسأله اصلی، یعنی تشخیص قابل دید بودن اشیاء بپردازیم، مسائل پیش‌نیاز آن را که در ضمن حل مسائل آتی به آن‌ها برخورد خواهیم کرد، بررسی می‌کنیم. اولین مسأله که در این بخش مطالعه می‌شود، نوعی از مسأله یافتن تلاقی بین پاره‌خط‌هاست. در این مسأله تعدادی پاره‌خط در صفحه داده شده‌اند که می‌توانیم آن‌ها را پیش‌پردازش کنیم. در زمان پرس‌وجو، دو پاره‌خط داده می‌شوند و ما باید تعیین کنیم که آیا پاره‌خطی در بین پاره‌خط‌های موجود یافت می‌شود که هر دو پاره‌خط داده شده را قطع کند.

به صورت دقیق‌تر مجموعه $E = \{e_۱, \ldots, e_n\}$ از n پاره‌خط مفروض است. پاره‌خط‌های پرس‌وجوی s و t داده می‌شوند. آیا پاره‌خطی مثل e_i وجود دارد که هم پاره‌خط s و هم t را قطع کند. این مسأله را CISD[۲] نام‌گذاری می‌کنیم.

[۱] Range searching
[۲] Common intersecting segment detection



شکل ۱.۳ : حالت‌های مختلفی که یک پاره‌خط، s و t را قطع می‌کند.

برای حل مسأله CISD، از روش حل مسائل جستجوی بازه نیم‌صفحه[۳] و جستجوی بازه مثلث[۴] استفاده می‌کنیم. فرض کنید پاره‌خط e_i با دو سر x_i و x'_i، s را قطع کند. بنابراین x_i و x'_i در دو طرف خط حامل s، یا l_s قرار می‌گیرند. همین مطلب برای دو سر s، یعنی v_s و v'_s و خط حامل e_i، یا l_{e_i} نیز برقرار می‌باشد. همان‌طور که در شکل ۱.۳ دیده می‌شود، خطوط حامل l_s و l_t صفحه را حداکثر به چهار ناحیه تقسیم می‌کنند که براساس موقعیت آن‌ها نسبت به l_s و l_t، نام‌گذاری شده‌اند. حال اگر e_i هم s و هم t را قطع کند، طبق آن‌چه در شکل دیده می‌شود، x_i و x'_i فقط می‌توانند در بخش‌های خاصی قرار بگیرند. اگر s و t متقاطع باشند، یکی از x_i و x'_i می‌تواند در UU و دیگری در LL و یا یکی در LU و دیگری در UL قرار بگیرد. اما اگر s و t متقاطع نباشند، فقط یکی از این دو حالت متصور است. بنابراین برای تشخیص این‌که آیا e_i هر دوی s و t را قطع می‌کند، می‌توان محل قرار گیری x_i و x'_i در چهار ناحیه مذکور و نیز محل قرار گیری دو سر s و t در دو ناحیه بالا و پایین l_{e_i} را بررسی کرد.

بنابراین CISD بدین طریق قابل حل است: فرض کنید s و t یکدیگر را قطع کنند. در ابتدا ما همه پاره‌خط‌های E را که یک سر آن‌ها در UU و سر دیگرشان در LL است پیدا می‌کنیم. همین کار برای دو ناحیه LU و UL نیز انجام می‌شود. نتیجه این جستجو پاره‌خط‌هایی است که l_s و l_t را قطع می‌کنند. برای تکمیل کار باید در نتیجه جستجوی قبل دو جستجوی دیگر انجام شود و پاره‌خط‌های انتخاب شوند که دو سر s و t در دو طرف خط حامل آن‌ها قرار گیرد. برای این‌کار ما s، t و همه پاره‌خط‌های نتیجه را به صفحه دوگان منتقل می‌کنیم. با این تبدیل، هر خط حامل l_{e_i} به یک نقطه و دو

[۳] Half-plane range search
[۴] Triangle range search

پاره‌خط s و t به دو گوه دوگانه[۵] نگاشت می‌شوند و ما باید به دنبال نقاطی مانند $l^*_{e_i}$ بگردیم که در فصل مشترک دو گوه دوگانه s^* و t^* قرار بگیرد.

روند کار برای حالتی که s و t متقاطع نباشند، به همین روش قابل انجام است. فقط در این حالت، ممکن است که جستجو فقط در یکی از زوج ناحیه‌های (UU, LL) و (LU, UL) لازم باشد. اگر s و t در یک طرف خط عمودیی که از محل تلاقی l_s و l_t رسم می‌شود قرار گرفته باشند، جستجو در زوج ناحیه (UU, LL) انجام می‌شود. اگر s و t در دو طرف این خط قرار بگیرند، جستجو در زوج ناحیه (LU, UL) انجام می‌شود. در صورتی که هیچ‌کدام از این حالت‌ها برقرار نبود (یکی از پاره‌خط‌ها خط عمودی را قطع کند)، جستجو در هر دو زوج انجام می‌شود.

طبق آن‌چه در بالا آمد، مسأله را می‌توان با ترکیبی از جستجوهای بازه نیم‌صفحه و مثلث حل کرد. برای این کار ما سه مسأله $\mathcal{P}_i$ و سه رابطه $\Diamond^i$ برای $1 \leq i \leq 3$ به صورت زیر تعریف می‌کنیم.

- $e\Diamond^1\mathcal{H}$: انتهای سمت چپ پاره‌خط e در نیم‌صفحه $\mathcal{H}$ قرار می‌گیرد.
- $e\Diamond^2\mathcal{H}$: انتهای سمت راست پاره‌خط e در نیم‌صفحه $\mathcal{H}$ قرار می‌گیرد.
- $e\Diamond^3\gamma$: خط حامل پاره‌خط e، پاره‌خط γ را قطع می‌کند. به بیان دیگر در صفحه دوگان، نقطه l^*_e در گوه دوگانه γ^* قرار می‌گیرد.

مسأله $\mathcal{P}_i$ با استفاده از رابطه $\Diamond^i$ تعریف می‌شود. با استفاده از این زیرمسأله‌ها، چهار مسأله لازم برای حل CISD به صورت زیر خلاصه می‌شوند.

- ۱(۲): پاره‌خط‌هایی را پیدا می‌کنیم که انتهای سمت چپ آن‌ها بالای(پایین) هر دو خط l_s و l_t قرار گرفته و انتهای سمت راست آن‌ها پایین(بالای) هر دو خط l_s و l_t قرار می‌گیرد و نیز خطوط حامل آن پاره‌خط‌ها هم s و هم t را قطع می‌کنند.
- ۳(٤): پاره‌خط‌هایی را پیدا می‌کنیم که انتهای سمت چپ آن‌ها بالای(پایین) خط l_s و پایین(بالای) l_t قرار گرفته و انتهای سمت راست آن‌ها پایین(بالای) خط l_s و بالای(پایین) l_t قرار می‌گیرد و نیز خطوط حامل آن پاره‌خط‌ها هم s و هم t را قطع می‌کنند.

در مسأله اول، به دنبال پاره‌خط‌هایی هستیم که سر چپ آن‌ها بالای l_s و l_t قرار می‌گیرد. این کار با دو بار اعمال $\mathcal{P}_1$، بار اول با نیم‌صفحه بالای l_s و بار دوم با نیم‌صفحه بالای l_t انجام می‌شود. در نتیجه حاصل، باید پاره‌خط‌هایی را انتخاب کنیم که سر راست آن‌ها پایین l_s و l_t قرار می‌گیرد. این کار نیز با دو بار اعمال $\mathcal{P}_1$ با نیم‌صفحه‌های پایین l_s و l_t قابل انجام است. در نهایت در نتیجه به‌دست آمده باید پاره‌خط‌هایی را انتخاب کنیم که هم l_s و هم l_t را قطع می‌کند. انجام این کار با دو بار اعمال $\mathcal{P}_3$، یک بار با s و بار دیگر با t میسر است.

[۵] Double wedge

بنابراین با ترکیب مسائل $\mathcal{P}_1$، $\mathcal{P}_2$ و $\mathcal{P}_3$ می‌توان مسأله اول را پاسخ گفت. همان‌طور که در [۱] توسط Agarwal و Erickson عنوان شده است، داده‌ساختارهای جستجوی بازه، قابلیت ترکیب با هم و ایجاد جستجوهای چندلایه‌ای را دارند، ضمن این‌که پیچیدگی زمانی آن‌ها حداکثر به صورت ضریبی چندلگاریتمی[۶] تغییر می‌کند.

نتیجه این‌که می‌توان مسأله اول را در زمان $O(\frac{n}{\sqrt{m}}\log(1+\frac{m}{n}))$، با انجام پیش‌پردازشی در زمان $O(n^{1+\epsilon}+m\log^{\epsilon}n)$ و با استفاده از $O(m)$ حافظه، پاسخ داد. در این رابطه‌ها m پارامتری است که به صورت $n \le m \le n^2$ انتخاب می‌شود و ϵ یک عدد ثابت مثبت دلخواه است. این مرتبه زمانی از کار Matoušek [۴۹] نتیجه می‌شود. به‌تازگی Chan [۱۲] الگوریتم‌هایی ارائه کرده‌است که با ترکیب آن‌ها با نتایج Matoušek [۴۹] می‌توان ثابت ϵ را از مرتبه زمانی حذف کرده و به‌جای آن یک ضریب چندلگاریتمی قرار داد.

به روشی مشابه، می‌توان مسائل ۲ تا ٤ را نیز به همین روش و با همان پیچیدگی زمانی و حافظه‌ای حل کرد. با حل مسائل ۱ تا ٤، می‌توان مسأله CISD را نیز حل نمود. بنابراین می‌توان نتیجه را در لم زیر خلاصه کرد.

لم ۱.۳ صحنه‌ای شامل n پاره‌خط را می‌توان در زمان $O(n^{1+\epsilon}+m\log^{\epsilon}n)$ و با صرف حافظه $O(m)$ به‌گونه‌ای پیش‌پردازش کرد که برای هر زوج پاره‌خط s و t بتوان در زمان $O(\frac{n}{\sqrt{m}}\log^{4}(1+\frac{m}{n}))$ وجود پاره‌خطی که هر دوی s و t را قطع کند، مشخص کرد. در این رابطه‌ها m پارامتری است با شرط $n \le m \le n^2$ و ϵ عدد ثابت مثبت دلخواهی است.

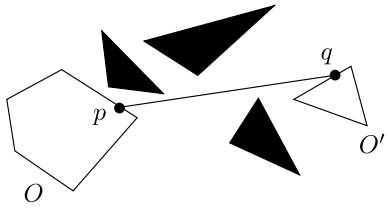
اکنون به مسأله اصلی خود در این بخش که تعیین قابل دید بودن دو شیء پیچیده از یکدیگر است می‌پردازیم. تلاش ما در این‌جا یافتن شرایطی است که در صورت برقرار بودن آن شرایط، دو شیء از یکدیگر قابل دید باشند. دو شیء از یکدیگر قابل دید ضعیف هستند، اگر نقطه‌ای روی شیء اول نقطه‌ای روی شیء دوم را ببینند. لم زیر شرط لازم و کافی را برای تعمیم روش‌های تشخیص قابلیت دید بین دو پاره‌خط به اشیاء پیچیده‌تر فراهم می‌کند.

لم ۲.۳ در یک چندضلعی P (ساده یا حفره‌دار)، دو چندضلعی O و O' یکدیگر را به صورت ضعیف می‌بینند اگگر یالی از O، یالی از O' را به صورت ضعیف ببینند.

اثبات. $\Rightarrow$: واضح است که اگر یال s از O، یال t از O' را به صورت ضعیف ببینند، دو چندضلعی O و O' نیز یکدیگر را به صورت ضعیف خواهند دید.

$\Leftarrow$: وقتی O و O' به صورت ضعیف یکدیگر را ببینند، نقطه‌ای مانند p از O، نقطه‌ای مانند q از O' را می‌بینند. پاره‌خط pq، یالی از O و یالی از O' را قطع می‌کند که این دو یال یکدیگر را به‌صورت

[۶] Polylogarithmic



شکل ۲.۳ : دید ضعیف بین دو چندضلعی.

ضعیف می‌بینند. □

با استفاده از لم بالا، کافی است مسأله تشخیص دید ضعیف را فقط برای نقاط و پاره‌خط‌ها حل کنیم و برای حل مسأله در حالت‌های پیچیده‌تر، برای هر ترکیب یک یال از یک چندضلعی و یالی از چندضلعی دیگر، مسأله را بررسی می‌کنیم. در بخش‌های ۳.۳ و ۴.۳ همین مسأله برای پاره‌خط حل شده است که به ما امکان حل مسأله تشخیص قابلیت دید بین دو شیء پیچیده‌تر را با استفاده از لم بالا می‌دهد.

۲.۳ آزمون قابلیت دید یک شیء از ناظر نقطه‌ای

در مسائل این بخش، یک صحنه چندضلعی P، یک ناظر s و یک شیء t داده می‌شوند و قابل دید بودن t از s مورد سؤال قرار می‌گیرد. در این بخش، قابل دید بودن را برای اشیاء غیرنقطه‌ای، دید ضعیف در نظر می‌گیریم. این نوع مسائل را به سه دسته تقسیم می‌کنیم و فقط حالت‌هایی را در نظر می‌گیریم که ناظر s نقطه‌ای باشد. در دسته اول یا بدون پیش‌پردازش، P، s و t با هم داده می‌شوند و ما باید قابل دید بودن t را مشخص کنیم. در دسته دوم یا SOQ[۷]، P و s از قبل مشخص هستند وما می‌توانیم با انجام پیش‌پردازش، در زمان پرس‌وجو قابل دید بودن t را مشخص کنیم. دسته سوم یا TOQ[۸]، مسائلی هستند که در ابتدا فقط P مشخص است و s و t در زمان پرس‌وجو تعیین می‌شوند.

مسائل بدون پیش‌پردازش و SOQ، با محاسبه چندضلعی قابل دید s و تصمیم‌گیری بر اساس آن قابل پاسخ‌گویی هستند. اگر بعد از محاسبه چندضلعی قابل دید، آن را برای جایابی نقطه[۹] و شلیک پرتو[۱۰] پیش‌پردازش کنیم، می‌توانیم در زمان لگاریتمی وجود اشتراک بین t و $\mathrm{V}(s)$ را مشخص کنیم. چون $\mathrm{V}(s)$، یک چندضلعی ستاره‌ای است که s در هسته آن قرار گرفته، می‌توان پیش‌پردازش جایابی

[۷] Single object query
[۸] Two objects query
[۹] Point location
[۱۰] Ray shooting

نقطه را در زمان $O(n)$ انجام داد. همچنین با روش معرفی شده توسط Chazelle و Guibas [۱۷]، می‌توان $\mathrm{V}(s)$ را در زمان $O(n)$ برای شلیک پرتو پیش‌پردازش کرد.

درصورتی‌که P یک چندضلعی ساده باشد، محاسبه چندضلعی قابل دید و پیش‌پردازش‌های لازم دیگر در زمان $O(n)$ قابل انجام است. بنابراین قابل دید بودن t در مسائل بدون پیش‌پردازش برای چندضلعی‌های ساده، در زمان $O(n)$ تعیین می‌شود. در مسائل SOQ، زمان پیش‌پردازش این صحنه‌ها $O(n)$ و زمان پاسخ‌گویی به پرس‌وجو $O(\log n)$ است. درصورتی‌که P یک چندضلعی حفره‌دار باشد، زمان محاسبه چندضلعی قابل دید به $O(n \log n)$ افزایش می‌یابد. بنابراین در این صحنه‌ها، مسائل بدون پیش‌پردازش به زمان $O(n \log n)$ و مسائل SOQ به زمان $O(n \log n)$ برای پیش‌پردازش و زمان $O(\log n)$ برای پرس‌وجو نیاز دارند.

مسائل دسته سوم یا TOQ کمی پیچیده‌تر هستند. در صورتی که t یک نقطه باشد، ما باید تعیین کنیم که آیا پاره‌خط st توسط یالی از مرز P قطع می‌شود یا خیر. این سؤال با شلیک یک پرتو از s به سمت t و تعیین محل قطع آن توسط مرز P پاسخ داده می‌شود. در صورتی که P یک چندضلعی ساده باشد، شلیک پرتو نیازمند زمان $O(n)$ برای پیش‌پردازش و زمان $O(\log n)$ برای پرس‌وجو است. اگر P یک چندضلعی حفره‌دار باشد، می‌توان از الگوریتم تصادفی Chan [۱۲] با زمان پیش‌پردازش $O(n \log^{۳} n)$ و حافظه $O(n \log^{۲} n)$ و زمان پرس‌وجوی متوسط $O(\sqrt{n} \log^{۲} n)$ برای شلیک پرتو در میان n پاره‌خط، استفاده کرد. این روش قابلیت برقراری توازن بین حافظه و زمان‌پرس‌وجو را نیز دارد و می‌توان با افزایش حافظه تا $O(n^{۲})$، زمان پرس‌وجو را به $O(\mathrm{polylog}\, n)$ رساند.

در حالتی که t یک پاره‌خط باشد، الگوریتم‌های موجود در بدترین حالت به زمان پرس‌وجوی $O(n)$ نیازمندند. در حالتی که P چندضلعی ساده باشد، راه‌حل ساده است، ابتدا $\mathrm{V}(s)$ محاسبه می‌شود و سپس تلاقی t با $\mathrm{V}(s)$ بررسی می‌گردد که هر دوی این‌ها بدون هیچ پیش‌پردازشی در زمان $O(n)$ قابل انجام است. در حالتی که P حفره‌دار باشد، می‌توان از همین روش استفاده کرد و به زمان $O(n \log n)$ رسید. همچنین می‌توان از روش Asano و همکارانش [۷] استفاده کرد و با پیش‌پردازشی در زمان $O(n^{۲})$، چندضلعی قابل دید s را در زمان $O(n)$ محاسبه کرد. با این کار زمان پرس‌وجو به $O(n)$ کاسته می‌شود.

در مورد حالت‌هایی که t یک چندضلعی محدب باشد، می‌توان از نتیجه به‌دست آمده در مورد پاره‌خط استفاده کرد. برای این کار، t را به پاره‌خط‌های تشکیل دهنده‌اش تجزیه می‌کنیم و برای هرکدام از این پاره‌خط‌ها مسأله را حل می‌کنیم. در صورتی که یکی از این پاره‌خط‌ها از s قابل دید باشد، s، t را هم می‌بینند.

۳.۳ آزمون قابلیت دید یک شیء از پاره‌خط

همانند بخش ۲.۳، در این بخش به دنبال تعیین قابل دید بودن یک شیء خاص در صحنه هستیم. فرض کنید P یک چندضلعی، s یک ناظر به شکل پاره‌خط درون P و t یک شیء خاص در درون P باشند. سؤال مطرح شده در این بخش، قابل دید بودن t از s است. منظور از قابلیت دید در این بخش همانند بخش قبلی، دید ضعیف است. مانند قبل، این مسأله نیز به سه شکل بررسی می‌شود. در شکل اول یا بدون پیش‌پردازش، P، s و t با هم داده می‌شوند و ما باید قابل دید بودن شیء را مشخص کنیم. در دسته دوم یا SOQ، P و s از قبل مشخص هستند وما می‌توانیم با انجام پیش‌پردازش، در زمان پرس‌وجو قابل دید بودن t را مشخص کنیم. دسته سوم یا TOQ، مسائلی هستند که در ابتدا فقط P مشخص است و s و t در زمان پرس‌وجو تعیین می‌شوند.

در شکل بدون پیش‌پردازش فقط حالتی را که t پاره‌خط است بررسی می‌کنیم، زیرا حالتی که t نقطه است می‌توان به روش بخش ۲.۳ حل کرد. در حالت اول فرض کنید P یک چندضلعی ساده است. در این صورت همان‌طور که در بخش ۲.۲.۲ ذکر شد، می‌توان در زمان خطی چندضلعی قابل دید ضعیف s را محاسبه کرد. چون چندضلعی $\mathrm{V}(s)$، ساده و با تعداد رأس $O(n)$ است، می‌توان در همان زمان خطی قرارگیری همه یا بخشی از t را در $\mathrm{V}(s)$ تشخیص داد. بنابراین مسأله در این حالت در زمان خطی قابل پاسخ‌گویی است.

در حالت دوم، فرض کنید P حفره‌دار است. همان‌طور که پیش از این گفته شد، پیچیدگی $\mathrm{V}(s)$ در این حالت، $O(n^۴)$ است، ولی $\mathrm{V}(s)$ از قرارگیری $O(n^۲)$ مثلث در صفحه به‌وجود آمده است که می‌توان در زمان $O(n^۲)$ آن‌ها را تعیین کرد. بنابراین برای قابل دید بودن t کافی است وجود اشتراک بین t و حداقل یکی از این مثلث‌ها مشخص شود که در همین زمان قابل انجام است. بنابراین در این حالت مسأله در زمان $O(n^۲)$ قابل پاسخ‌گویی است.

در شکل دوم مسأله، یعنی SOQ، مانند شکل بدون پیش‌پردازش، چندضلعی قابل دید ضعیف s محاسبه می‌شود. در حالت اول که P ساده است، $\mathrm{V}(s)$ نیز ساده بوده و در زمان $O(n)$ قابل محاسبه است. برای بررسی قابل دید بودن t از s، اگر t نقطه باشد، کافی است $\mathrm{V}(s)$ را برای جایابی نقطه پیش‌پردازش کنیم و در زمان پرس‌وجو وضعیت قرارگیری نقطه t را در $\mathrm{V}(s)$ بررسی کنیم. در حالتی که t پاره‌خط است، $\mathrm{V}(s)$ را برای جایابی نقطه و شلیک پرتو پیش‌پردازش می‌کنیم. در زمان پرس‌وجو، ابتدا قرارگیری یکی از دو سر t را در $\mathrm{V}(s)$ بررسی می‌کنیم. در صورتی که نقطه داخل $\mathrm{V}(s)$ باشد، t از s قابل دید است. در غیر این‌صورت پرتوی را از همان سر t شلیک می‌کنیم و اولین محل برخورد آن را با $\mathrm{V}(s)$ می‌یابیم. اگر برخوردی وجود داشته باشد و محل برخورد روی t باشد، t از s قابل دید ضعیف است و در غیر این‌صورت t از s قابل دید ضعیف نیست. پیش‌پردازش $\mathrm{V}(s)$ برای جایابی نقطه و شلیک پرتو به ترتیب به زمان $O(n\log n)$ و $O(n)$ نیاز دارد. با انجام این پیش‌پردازش‌ها، جایابی نقطه و شلیک

پرتو در زمان $O(\log n)$ انجام می‌شود. بنابراین زمان پیش‌پردازش لازم برای حل مسأله $O(n \log n)$ و زمان پرس‌وجو، $O(\log n)$ خواهد بود.

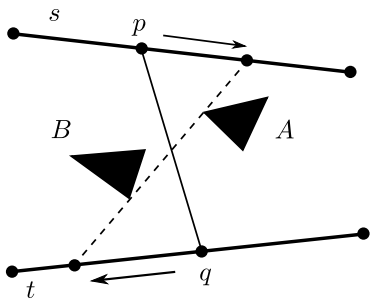
در حالت دوم مسأله SOQ که P یک چندضلعی حفره‌دار است، بعد از تعیین $\mathrm{V}(s)$ با پیچیدگی $O(n^۴)$، آن را برای جایابی نقطه، و یال‌های $O(n^۲)$ مثلث تشکیل دهنده آن را برای شلیک پرتو پیش‌پردازش می‌کنیم. جایابی نقطه در این چندضلعی به $O(n^۴ \log n)$ زمان پیش‌پردازش نیاز دارد. برای شلیک پرتو می‌توان از روش Chan [۱۲] استفاده کرد. در این صورت با صرف زمان $O(n^۴ \operatorname{polylog} n)$ برای پیش‌پردازش و حافظه $O(n^۴)$، می‌توان تلاقی پاره‌خط t را با یال‌های $\mathrm{V}(s)$، همانند حالت قبل، تشخیص داد. در مجموع اگر t نقطه باشد به زمان پیش‌پردازش $O(n^۴ \log n)$ و اگر t پاره‌خط باشد، به زمان پیش‌پردازش $O(n^۴ \operatorname{polylog} n)$ و حافظه $O(n^۴)$ نیاز است. پاسخ‌گویی به پرس‌وجو در حالتی که t نقطه باشد در زمان $O(\log n)$ و در حالتی که t پاره‌خط باشد در زمان $O(\operatorname{polylog} n)$ انجام می‌شود. در بخش ۴.۳ روش تعیین دید دو شیء پرس‌وجو را ارائه می‌کنیم که پیچیدگی پیش‌پردازش آن بسیار کمتر از این روش است، ولی زمان پاسخ به پرس‌وجو در آن بیشتر می‌شود. واضح است که می‌توان برای کاهش زمان و حافظه پیش‌پردازش در این حالت نیز، از آن روش استفاده کرد.

۴.۳ آزمون قابلیت دید دو شیء پرس‌وجو

در این بخش، پیچیده‌ترین حالت تعیین قابلیت دید، یعنی وقتی هر دو شیء غیرنقطه‌ای بوده و در زمان پرس‌وجو مشخص شوند بررسی می‌گردد. بدیهی است در این حالت، فقط می‌توان از ویژگی‌های صحنه برای پیش‌پردازش مسأله استفاده کرد که کار تشخیص قابلیت دید را دشوارتر می‌کند. برای تشخیص وجود رابطه دید ضعیف بین دو پاره‌خط، ابتدا به دنبال خاصیتی هستیم که وجود آن بین دو پاره‌خط، شرط لازم و کافی برای وجود رابطه دید ضعیف باشد. این خاصیت را می‌توان با استفاده از لم زیر شناسایی کرد.

لم ۳.۳ در چندضلعی حفره‌دار P، دو پاره‌خط s و t یکدیگر را به صورت ضعیف می‌بینند، اگر سر یکی از دو پاره‌خط، پاره‌خط دیگر را ببیند و یا یالی از گراف توسعه‌یافته قابلیت دید P هم s و هم t را قطع کند.

اثبات. $\Rightarrow$: واضح است که اگر یک سر s یا t پاره‌خط دیگر را ببیند، s و t به‌صورت ضعیف یکدیگر را می‌بینند. همچنین یال‌های گراف توسعه‌یافته قابلیت دید موانع صحنه را قطع نمی‌کنند، بنابراین اگر s و t هر دو یک یال این گراف را قطع کنند، در راستای همین یال، این دو پاره‌خط یکدیگر را به‌صورت ضعیف می‌بینند.



شکل ۳.۳ : دید ضعیف بین دو پاره‌خط.

⇐: فرض کنید s و t یکدیگر را به‌صورت ضعیف ببینند. بنابراین نقطه‌های p و q به ترتیب روی s و t قرار دارند که یکدیگر را می‌بینند. همان‌طور که در شکل ۳.۳ دیده می‌شود، اگر ما p و q را روی s و t در دو جهت مخالف بلغزانیم، به‌گونه‌ای که موانع صحنه را قطع نکنند، دو حالت ممکن است رخ بدهد. در حالت اول حداقل یکی از p و q به انتهای پاره‌خط خود می‌رسند، که نشان دهنده آن است که یک سر یکی از پاره‌خط‌ها، پاره‌خط دیگر را می‌بینند. در حالت دوم، که در شکل دیده می‌شود، بین دو مانع متوقف شده است. در این حالت pq بخشی از یالی از گراف توسعه‌یافته قابلیت دید است که بر دو مانع مماس می‌باشد. بنابراین در این حالت، هم s و هم t یک یال گراف توسعه‌یافته قابلیت دید را قطع می‌کنند. □

لم ۳.۳، روشی را برای تشخیص وجود دید ضعیف بین دو پاره‌خط ارائه می‌کند که در لم زیر توضیح داده می‌شود.

لم ۴.۳ یک چندضلعی حفره‌دار با تعداد رأس n را می‌توان در زمان $O(n^{2+\epsilon})$ با استفاده از $O(n^2)$ حافظه به‌گونه‌ای پیش‌پردازش کرد که وجود دید ضعیف بین هر دو پاره‌خط دلخواه پرس‌وجو را در زمان $O(n)$ تشخیص داد.

اثبات. بر طبق لم ۳.۳، ما باید دو حالت را بررسی کنیم. حالت اول این‌که یک سر یکی از پاره‌خط‌ها، پاره‌خط دیگر را ببینند. طبق آن‌چه در بخش ۲.۳ گفته شد، تعیین قابل دید بودن یک پاره‌خط از یک نقطه با استفاده از روش Asano و همکارانش [۷]، با پیش‌پردازش چندضلعی در زمان $O(n^2)$، در زمان پرس‌وجوی $O(n)$ قابل انجام است، که ما این کار را برای هر چهار سر پاره‌خط‌ها تکرار می‌کنیم.

حالت دوم، بررسی قطع شدن s و t توسط یکی از یال‌های گراف توسعه‌یافته قابلیت دید است. این کار به روش ذکر شده در بخش ۱.۳ قابل انجام است. چون تعداد یال‌های گراف توسعه‌یافته قابلیت دید، $O(n^2)$ است، برای ساخت داده‌ساختار لازم، به زمان پیش‌پردازش $O(n^{2+\epsilon})$ و حافظه $O(n^2)$ نیاز است. با استفاده از این داده‌ساختار، می‌توان تعیین کرد که آیا یالی از گراف

توسعه‌یافته قابلیت دید وجود دارد که هم s و هم t را قطع کند یا خیر، و در صورت لزوم چنین یالی را بیابد. زمان لازم برای این کار $O(n)$ است. واضح است که از جمع این زمان‌ها، لم اثبات می‌شود. □

نتیجه بالا به‌راحتی قابل تعمیم به حالتی است که s و t دو چندضلعی محدب با پیچیدگی (اندازه) ثابت باشند. طبق لم ۲.۳، برای تشخیص وجود دید ضعیف بین s و t، کافی است روش بالا برای هر ترکیب زوج یال‌های آن دو تکرار شود. چون اندازه s و t محدود است، می‌توان قضیه زیر را برای تشخیص وجود دید ضعیف بین دو چندضلعی محدب نتیجه گرفت.

قضیه ۱.۳ یک چندضلعی حفره‌دار با تعداد رأس n را می‌توان در زمان $O(n^{۲+\epsilon})$ با استفاده از $O(n^۲)$ حافظه به‌گونه‌ای پیش‌پردازش کرد که وجود دید ضعیف بین هر دو چندضلعی محدب با پیچیدگی ثابت، در زمان $O(n)$ قابل تشخیص باشد.

۵.۳ خلاصه

در این فصل مسأله آزمون قابلیت دید دو شیء از یکدیگر را مطالعه کردیم. در ابتدا الگوریتم‌هایی برای شکل‌های ساده‌تر مسأله براساس راه‌حل‌های موجود ارائه کردیم. در ادامه نشان دادیم مسأله در حالتی که دو شیء پاره‌خط یا چندضلعی محدب بوده و فقط محیط قابل پیش‌پردازش باشد، از همه حالت‌ها پیچیده‌تر است. در انتها با استفاده از روشی که برای حل مسأله یافتن پاره‌خط متلاقی با دو پاره‌خط پرس‌وجو ارائه دادیم، مسأله را در پیچیده‌ترین حالت حل کرده و به زمان پرس‌وجوی $O(n)$ و حافظه $O(n^۲)$ رسیدیم. در فصل ۵ نشان خواهیم داد که چگونه برای این مسأله می‌توان به توازنی بین حافظه و زمان پرس‌وجو رسید.

فصل ۴

چندضلعی قابل دید ناظر نقطه‌ای در داخل چندضلعی حفره‌دار

در بخش ۴.۱.۲ روش‌های موجود را برای محاسبه چندضلعی قابل دید یک ناظر نقطه‌ای به همراه یک مرحله پیش‌پردازش عنوان کردیم و پیچیدگی زمانی آن‌ها را مقایسه نمودیم. یکی از مشکلاتی که در همه این الگوریتم‌ها وجود دارد، زمان بالای پاسخ به پرس‌وجو در بدترین حالت است. مثلاً اگر تعداد حفره‌های چندضلعی $O(n)$ باشد، یا مسأله در آرایشی از پاره‌خط‌ها طرح شود، و نیز اندازه چندضلعی قابل دید نقطه پرس‌وجو $O(n)$ باشد، هم الگوریتم Pocchiola و Vegter [۵۷] و هم الگوریتم زارعی و قدسی [۶۵]، شبیه روش بدون پیش‌پردازش عمل می‌کنند و در زمان $O(n \log n)$ چندضلعی قابل دید را محاسبه می‌کنند. در مورد روش Vegter [۶۳] نیز می‌توان نمونه‌ای از مسأله را تولید کرد که بیش از آن‌چه برای گزارش چندضلعی قابل دید نیاز است، زمان صرف می‌کند.

مشکل دیگری که در مورد این الگوریتم‌ها وجود دارد، عدم امکان تعیین برخی از خاصیت‌های چندضلعی قابل دید، بدون محاسبه خود چندضلعی است. نمونه این خاصیت‌ها می‌تواند اندازه چندضلعی قابل دید باشد. یعنی در این الگوریتم‌ها ابتدا باید خود چندضلعی قابل دید محاسبه شود، و سپس اندازه آن مشخص شود.

در این بخش، ما روشی را ارائه می‌کنیم که چندضلعی قابل دید را برای یک نقطه پرس‌وجو در زمان $O(\log n)$ محاسبه می‌کند. در صورتی که نیاز به گزارش چندضلعی قابل دید باشد، می‌توان در زمان $O(\log n + k)$ این کار را انجام داد که k اندازه چندضلعی قابل دید است. زمان و حافظه مورد نیاز برای پیش‌پردازش به ترتیب $O(n^4 \log n)$ و $O(n^4)$ است. این الگوریتم قابلیت تعیین شکل کلی چندضلعی (بدون تعیین دقیق بعضی از رأس‌ها) و یا اندازه آن و سایر خصوصیات کلی چندضلعی را در

زمان $O(\log n)$ دارد. این ویژگی الگوریتم، به ما اجازه می‌دهد بتوانیم نتیجه این الگوریتم را برای مسائل بعدی (مثلاً بررسی تلاقی یک مجموعه پاره‌خط با چندضلعی قابل دید) پیش‌پردازش کنیم و پاسخ آن مسائل را در زمان کمتری بیابیم.

به‌دلیل بالا بودن زمان و حافظه مصرفی، در ادامه، الگوریتم را به‌گونه‌ای تغییر می‌دهیم که بتوان بین حافظه مصرف شده و زمان پرس‌وجو توازنی برقرار کرد. با این تغییرات می‌توان با صرف حافظه $O(m)$ و زمان $O(m\log(1+\sqrt{m}/n))$ برای پیش‌پردازش، چندضلعی قابل دید یک نقطه پرس‌وجو را در زمان $O(n^2\log(1+\sqrt{m}/n)/\sqrt{m})$ محاسبه کرد. در این رابطه‌ها m پارامتری دلخواه است به‌طوری‌که $n^2 \le m \le n^4$. در صورتی که گزارش دقیق چندضلعی قابل دید لازم باشد، می‌توان این کار را نیز در زمان اضافی $O(k)$ انجام داد. به روشی مشابه می‌توان لیست مرتب شده تعدادی نقطه در صفحه را حول یک نقطه پرس‌وجو در همین زمان محاسبه کرد. زمان و حافظه مصرفی برای پیش‌پردازش نیز به همین اندازه است.

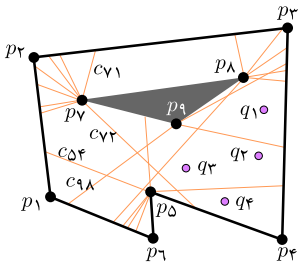
در بخش ۱.۴ روش ارائه شده برای محاسبه چندضلعی قابل دید در زمان $O(\log n)$ را تشریح می‌کنیم و در بخش ۲.۴ چگونگی ایجاد توازن بین حافظه مصرفی و زمان پرس‌وجو را توضیح می‌دهیم.

۱.۴ محاسبه چندضلعی قابل دید در زمان لگاریتمی

در این بخش به دنبال محاسبه چندضلعی قابل دید یک نقطه پرس‌وجو در داخل یک چندضلعی حفره‌دار هستیم. چون می‌توان یال‌های مرز چندضلعی را از هم جدا کرد و آرایشی از پاره‌خط‌ها به‌دست آورد، ما در این بخش و بخش بعد، صحنه‌ای را در نظر می‌گیریم که از تعدادی پاره‌خط تشکیل شده است. فرض کنید $O = \{o_1, \ldots, o_n\}$ تعدادی پاره‌خط در صفحه باشند که فقط در نقاط انتهایی ممکن است یکدیگر را قطع کنند. این پاره‌خط‌ها یال‌های مرز صحنه هستند. همچنین فرض کنید q نقطه‌ای در این صحنه باشد. هدف ما محاسبه $\mathrm{V}(q)$ به‌طور کارا با صرف زمان و حافظه لازم در پیش‌پردازش است.

ایده اصلی برای این کار عبارت است از تقسیم صحنه به زیربخش‌هایی که در هر کدام از آن‌ها چندضلعی قابل دید، شکل یکسانی دارد. این ایده پیش از این، توسط Bose و همکارانش [۱۱] و نیز Aronov و همکارانش [۵] برای چندضلعی ساده استفاده شده است. از آن‌جایی که چندضلعی قابل دید هر نقطه یکتاست، نمی‌توان صحنه را به‌گونه‌ای تقسیم کرد که در هر زیربخش چندضلعی قابل دید یکسان باشد. بلکه تقسیم‌بندی فقط می‌تواند به‌گونه‌ای باشد که شکل کلی یا به‌طور دقیق‌تر ساختار ترکیبیاتی چندضلعی قابل دید یکسان باشد. ساختار ترکیبیاتی چندضلعی قابل دید یک نقطه را ترتیب یال‌ها و رئوسی از صحنه که از آن نقطه دیده می‌شود تعریف می‌کنیم و با $\mathcal{V}(q)$ نمایش می‌دهیم.

فرض کنید مجموعه نقاط انتهایی پاره‌خط‌های O با $P = \{p_1, \ldots, p_l\}$ نمایش داده شود که در آن $l \le 2n$. اگر نقطه q در صحنه حرکت کند، می‌توان نشان داد زمانی $\mathcal{V}(q)$ تغییر می‌کند که q



شکل ۱.۴ : تقسیم‌بندی قابلیت دید یک چندضلعی حفره‌دار. چندضلعی قابل دید نقاط $q_۱$، $q_۲$ و $q_۴$ ساختار ترکیبیاتی متفاوتی دارد، درحالی‌که چندضلعی قابل دید نقاط $q_۲$ و $q_۳$ ساختار ترکیبیاتی یکسانی دارند.

از مرزهایی با نام قیود بحرانی[۱] عبور کند. قید بحرانی c_{ij} پرتوی است که از p_i در خلاف جهت p_j ساطع می‌شود (شکل ۱.۴). در طی حرکت، اگر پرتو به مانعی برخورد کند، متوقف می‌شود، ولی در غیراینصورت پرتو تا بینهایت ادامه پیدا می‌کند. وقتی q از میان یک قید بحرانی c_{ij} عبور کند، ترتیب دیده شدن p_i و p_j از q عوض می‌شود، ولی بقیه بخش‌های $\mathcal{V}(q)$ تغییر نمی‌کند.

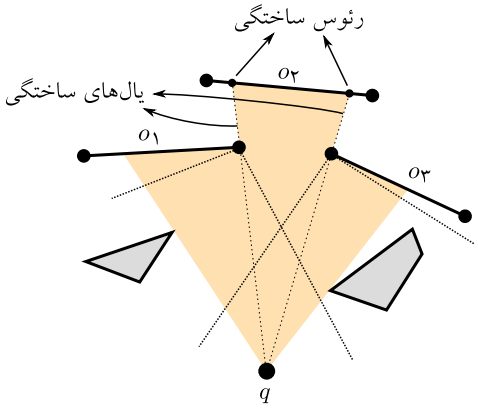
آرایش قیود بحرانی یک چندضلعی حفره‌دار در صفحه، آن را به تعدادی ناحیه تقسیم می‌کند. به این آرایش، تقسیم‌بندی قابلیت دید[۲] می‌گوییم. در هر کدام از ناحیه‌های تقسیم‌بندی قابلیت دید، $\mathcal{V}(q)$ ثابت بوده و تفاوت $\mathcal{V}(q)$ بین دو ناحیه مجاور $O(۱)$ است. چون قیود بحرانی، پرتوها یا پاره‌خط‌هایی هستند که بر روی $O(n^۲)$ خط قرار دارند، تعداد ناحیه‌های تشکیل شده از آرایش آن‌ها حداکثر $O(n^۴)$ ناحیه است. لم زیر خلاصه نتایج ذکر شده است.

لم ۱.۴ یک صحنه تشکیل شده از یک چندضلعی حفره‌دار یا تعدادی پاره‌خط، با پیچیدگی کلی $O(n)$، قابل تقسیم به حداکثر $O(n^۴)$ ناحیه است، به‌گونه‌ای که تمام نقاط q داخل یک ناحیه، $\mathcal{V}(q)$ یکسان داشته باشند و برای هر دو نقطه $q_۱$ و $q_۲$ در دو ناحیه مجاور، $\mathcal{V}(q_۱)$ با $O(۱)$ تغییر قابل تبدیل به $\mathcal{V}(q_۲)$ است.

با استفاده از نتیجه بالا می‌توان روشی را برای محاسبه سریع چندضلعی قابل دید یک نقطه پرس‌وجو به‌دست آورد. ابتدا با انجام یک پیمایش عمق اول[۳]، یک تور را که از همه نواحی عبور کند و از هر یال آرایش قیود حداکثر دو بار عبور کند به‌دست می‌آوریم. سپس برای یک ناحیه دلخواه، در زمان $O(n \log n)$، $\mathcal{V}$ را محاسبه می‌کنیم.

به‌دلیل وجود تفاوت کم بین $\mathcal{V}$ در نواحی مجاور و برای استفاده بهینه از حافظه، از یک

[۱] Critical constraints
[۲] Visibility decomposition
[۳] Depth first traversal



شکل ۲.۴ : بخشی از چندضلعی قابل دید نقطه q. بخش رنگی، $\mathrm{V}(q)$ را نشان می‌دهد. یال‌ها و رئوس ساختگی در شکل مشخص شده‌اند.

داده‌ساختار ماندگار[۴]، مثل درخت قرمز–سیاه ماندگار استفاده می‌کنیم. درخت قرمز–سیاه ماندگار، که توسط Sarnak و Tarjan [۵۹] معرفی شده است، درختی قرمز–سیاه است که توانایی به‌خاطر سپردن نسخه‌های قبلی خود را دارد. یعنی بعد از انجام m به‌روز رسانی در یک مجموعه n تایی مرتب شده، هر عنصری از نسخه t درخت، که $1 \leq t \leq m$، در زمان $O(\log n)$ قابل دسترسی است. درخت قرمز–سیاه ماندگار با استفاده از حافظه $O(m+n)$ در زمان $O((m+n)\log n)$ ساخته می‌شود.

بعد از ساخت $\mathcal{V}$ برای اولین ناحیه و ذخیره آن در درخت ماندگار، با توجه به تور ایجاد شده، ناحیه‌ها یکی پس از دیگری ملاقات شده و درخت ماندگار به‌روز رسانی می‌شود. بعد از به‌روز رسانی، شماره نسخه درخت را برای آن ناحیه ذخیره می‌کنیم تا هر زمان که به $\mathcal{V}$ آن ناحیه احتیاج داشتیم به سرعت به درخت دسترسی پیدا کنیم. طبق آن‌چه گفته شد زمان ساخت درخت برای همه ناحیه‌ها برابر $O((n^4+n)\log n) = O(n^4 \log n)$ و حافظه مصرفی برابر $O(n^4+n) = O(n^4)$ خواهد بود. علاوه بر محاسبه $\mathcal{V}$، نواحی تشکیل شده از آرایش قیود را برای جایابی نقطه نیز پیش‌پردازش می‌کنیم که این کار با صرف زمان و حافظه $O(n^4)$ قابل انجام است [۱۵].

برای محاسبه چندضلعی قابل دید یک نقطه پرس‌وجوی q، ابتدا ناحیه‌ای را که نقطه در آن قرار می‌گیرد تعیین می‌کنیم. سپس در درخت ماندگار، $\mathcal{V}$ آن ناحیه را می‌یابیم. این کارها در زمان $O(\log n)$ قابل انجام است. نتیجه ذخیره شده در $\mathcal{V}$، یال‌ها و رئوس قابل دید چندضلعی است. غیر از یال‌ها و رئوس $\mathrm{V}(q)$، $\mathcal{V}$ شامل یال‌ها و رئوسی است که به ترتیب جزء O و P نیستند. این یال‌ها و رئوس را به ترتیب یال‌های ساختگی[۵] و رئوس ساختگی[۶] می‌نامیم (شکل ۲.۴). یال‌ها و رئوس ساختگی با یک بار

[۴] Persistent data structure
[۵] Constructed edge
[۶] Constructed vertex

پیمایش $\mathcal{V}$ در زمان $O(|\mathrm{V}(q)|)$ قابل محاسبه هستند. بنابراین قضیه زیر به اثبات می‌رسد.

قضیه ۱.۴ یک صحنه تشکیل شده از یک چندضلعی حفره‌دار یا تعدادی پاره‌خط، با پیچیدگی کلی $O(n)$ را می‌توان در زمان $O(n^۴ \log n)$ و حافظه $O(n^۴)$ پیش‌پردازش کرد، به‌گونه‌ای که برای هر نقطه q اشاره‌گری به درختی قرمز-سیاه که حاوی $\mathcal{V}(q)$ است در زمان $O(\log n)$ برگردانده شود. علاوه بر این، می‌توان $\mathrm{V}(q)$ را در زمان $O(\log n + |\mathrm{V}(q)|)$ گزارش کرد.

۲.۴ برقراری توازن بین حافظه و زمان پرس‌وجو

در بخش قبل دیدیم که چگونه با استفاده از $O(n^۴)$ حافظه می‌توان در زمان لگاریتمی، $\mathcal{V}(q)$ را برای نقطه پرس‌وجوی q محاسبه کرد. به دلیل زیاد بودن این حافظه در این بخش می‌خواهیم رابطه‌ای بین حافظه و زمان پرس‌وجو به‌وجود آوریم تا در کاربردهایی که امکان صرف چنین حافظه‌ای نیست، با اضافه شدن به زمان پرس‌وجو، در حد امکان مصرف حافظه کاهش یابد.

این کار با ترکیب الگوریتم Asano و همکارانش [۷] که در بخش ۴.۱.۲ توضیح داده شد و الگوریتم بخش قبلی حاصل می‌شود. همانطور که در بخش ۴.۱.۲ و قضیه ۳.۲ دیدیم، می‌توان با پیش‌پردازشی در مدت زمان $O(n^۲)$، چندضلعی قابل دید نقطه پرس‌وجوی دلخواه را در زمان $O(n)$ محاسبه کرد. با ترکیب این الگوریتم، با الگوریتم با زمان لگاریتمی بخش قبل می‌توان به توازن بین حافظه و زمان پرس‌وجو رسید.

برای فهم دقیق‌تر روش کار، آشنایی با دو مفهوم مورد نیاز است. اولین مفهوم، تبدیل دوگان و استفاده از آن برای مرتب کردن زاویه‌ای نقاط حول یک نقطه در زمان $O(n)$ است که در بخش ۴.۱.۲ معرفی شد. مفهوم دوم، $(۱/r)$-برش است که در ادامه توضیح داده می‌شود.

۱.۲.۴ $(۱/r)$-برش

مجموعه L از n خط را در نظر بگیرید. یک $(۱/r)$-برش[۷] از L $(۱ \le r \le n)$، عبارت است از تعدادی مثلث مجزا (که برخی بی‌کران هستند)، به‌طوری‌که کل صفحه را می‌پوشانند و هر یک از این مثلث‌ها در داخل خود حداکثر n/r خط L را قطع می‌کنند. این مثلث‌ها به همراه مجموعه خطوطی که هریک قطع می‌کنند در زمان $O(nr)$ قابل محاسبه هستند.

اولین الگوریتم برای محاسبه برش، هرچند ناکارا، توسط Clarkson [۱۹] ارائه شد. او نشان داد یک نمونه‌برداری تصادفی از خطوط L با اندازه r با احتمال زیاد یک برش خوب تولید می‌کند. الگوریتم

[۷] $(۱/r)$-cutting

کارای ساخت $(1/r)$-برش در صفحه با اندازه $O(r^2)$ را در مقالات Chazelle و Friedman [۱۶] و de Berg و Schwarzkopf [۲۲] می‌توان یافت.

برای این‌که یک $(1/r)$-برش در الگوریتم ما قابل استفاده باشد، باید خاصیتی را که منظم بودن[8] می‌نامیم داشته باشد. یک $(1/r)$-برش را منظم می‌گوییم اگر هر خط دلخواه در صفحه، $O(r)$ مثلث برش را قطع کند. در ادامه ثابت می‌کنیم برشی که توسط Matoušek [۴۸] ارائه شده است، خاصیت منظم بودن را برآورده می‌کند و بنابراین در این الگوریتم می‌توان از آن استفاده کرد.

ابتدا الگوریتم‌های ارائه شده توسط Chazelle و Friedman [۱۶] و Matoušek [۴۸] را مرور می‌کنیم. Chazelle و Friedman [۱۶] الگوریتمی تصادفی معرفی کردند که در زمان متوسط $O(nr)$، $(1/r)$-برش را محاسبه می‌کند. همچنین آن‌ها نشان دادند که با استفاده از روش Raghavan [۵۸] و Spencer [۶۱] می‌توان الگوریتم را به‌صورت قطعی درآورد، هرچند زمان اجرای الگوریتم در این حالت $O(rn^{10/3})$ خواهد شد. Matoušek [۴۸] الگوریتم Chazelle و Friedman [۱۶] را به عنوان زیربنای الگوریتم خود انتخاب کرد و با الگوریتمی تودرتو، $(1/r)$-برش را در زمان $O(nr)$ محاسبه کرد.

توضیح زیربنای الگوریتم Matoušek، بدین صورت است. برای مجموعه L از n خط، هدف ما یافتن $(1/r)$-برش Ξ با اندازه $O(r^2)$ است. در مرحله اول، مجموعه نمونه تصادفی $R \subset L$ را می‌سازیم که هر عضو L با احتمال r/n در R انتخاب می‌شود. سپس آرایش $\mathcal{A}(R)$ خطوط R را محاسبه کرده و مثلث‌بندی کانونی[9] آرایش را به‌دست می‌آوریم. مثلث‌بندی کانونی یا مثلث‌بندی با پایین‌ترین رأس از رسم تمام قطرهای پایین‌ترین رأس در هر ناحیه c از آرایش به‌دست می‌آید.

$\mathcal{T}(R)$ را مثلث‌بندی کانونی آرایش خطوط R بگیرید. مثلث‌بندی $\mathcal{T}(R)$ هنوز قادر به تضمین ضریب برش r نیست، اما می‌توانیم این برش را با یک لایه تقسیم دیگر ترکیب کنیم تا به یک $(1/r)$-برش مناسب برسیم. مثلث دلخواه $t \in \mathcal{T}(R)$ را در نظر بگیرید. به مثلث t، مثلث نسل اول می‌گوییم. مجموعه خطوطی از L که توسط t قطع می‌شوند را با L_t نمایش می‌دهیم. عدد $r_t = r|L_t|/n$ میزان افزونگی $|L_t|$ از مقدار n/r را نشان می‌دهد. به ازای هر مثلث $t \in \mathcal{T}(R)$، با استفاده از یک روش دلخواه ناکارا، یک $(1/r_t)$-برش Ξ_t برای L_t محاسبه می‌کنیم. در این حالت، کارایی برش برای ما اهمیت ندارد و فرض می‌کنیم تعداد زیرمثلث‌های تولید شده (که به آن‌ها مثلث‌های نسل دوم می‌گوییم)، به ازای یک ثابت C، r_t^C باشد. اجتماع همه مثلث‌های نسل دوم، برش نهایی Ξ مطلوب ما است. لم زیر که در [۴۸] به اثبات می‌رسد، تعداد مثلث‌های Ξ را مشخص می‌کند.

لم ۲.۴ امید ریاضی تعداد مثلث‌های Ξ، $O(r^2)$ است.

شهود لم بالا این است که تعداد مثلث‌هایی که برای آن‌ها $r_t \geq k$ به طور نمایی با افزایش k کاهش می‌یابد.

[8] Regularity
[9] Canonical triangulation

الگوریتم ذکر شده در بالا، زیربنای الگوریتمی است که Matoušek [۴۸] ارائه می‌کند. در لم زیر نشان می‌دهیم که امید ریاضی تعداد مثلث‌های قطع شده توسط هر خط دلخواه، خطی است.

لم ۳.۴ برای هر خط دلخواه l، امید ریاضی تعداد مثلث‌های $t \in \Xi$ که توسط l قطع می‌شوند، $O(n)$ است.

اثبات. امید ریاضی تعداد خطوط R، $\Theta(r)$ است. با استفاده از قضیه ناحیه[۱۰] [۱۸]، امید ریاضی تعداد مثلث‌های نسل اول که توسط l قطع می‌شوند نیز $\Theta(r)$ است. زیرمجموعه مثلث‌های مثلث‌بندی $\mathcal{T}(R)$ و برش Ξ که توسط l قطع می‌شوند، به ترتیب با $\mathcal{T}_l(R)$ و Ξ_l نشان می‌دهیم. همچنین فرض کنید $E(r^C)$ امید ریاضی r_t^C به‌ازای هر مثلث $t \in \mathcal{T}(R)$ باشد. داریم:

$$|\Xi_l| \leq \sum_{t \in \mathcal{T}_l(R)} r_t^C$$

با گرفتن امید ریاضی از دو طرف داریم:

$$\begin{aligned}
\mathrm{E}(|\Xi_l|) &\leq \mathrm{E}\left(\sum_{t \in \mathcal{T}_l(R)} r_t^C\right) \\
&\leq \mathrm{E}(|\mathcal{T}_l(R)|)\mathrm{E}(r^C) \\
&\leq \mathrm{E}(|\mathcal{T}_l(R)|)\frac{\mathrm{E}\left(\sum_{t' \in \mathcal{T}(R)} r_{t'}^C\right)}{\mathrm{E}(|\mathcal{T}(R)|)}
\end{aligned}$$

چون $\mathrm{E}(\sum_{t' \in \mathcal{T}(R)} r_{t'}^C)$ امید ریاضی تعداد مثلث‌های Ξ است، که طبق اثبات Matoušek برابر $O(r^۲)$ است، و $\mathrm{E}(|\mathcal{T}(R)|)$ امید ریاضی تعداد مثلث‌های نسل اول است که برابر $\Theta(r^۲)$ است، خواهیم داشت:

$$\begin{aligned}
\mathrm{E}(|\Xi_l|) &\leq \mathrm{E}(|\mathcal{T}_l(R)|)\frac{\mathrm{E}\left(\sum_{t' \in \mathcal{T}(R)} r_{t'}^C\right)}{\mathrm{E}(|\mathcal{T}(R)|)} \\
&= O(r)\frac{O(r^۲)}{\Theta(r^۲)} \\
&= O(r),
\end{aligned}$$

[۱۰] Zone theorem

که همان چیزی است که می‌خواهیم. □

به صورت شهودی، چون تعداد متوسط مثلث‌های نسل دوم در هر مثلث نسل اول $O(۱)$ است و l به طور متوسط $O(r)$ مثلث نسل اول را قطع می‌کند، تعداد متوسط مثلث‌های قطع شده $O(r)$ است، یعنی $\mathrm{E}(|\Xi_l|) = O(r)$. لم بالا نشان می‌دهد که برشی که توسط Chazelle و Friedman [۱۶] ساخته می‌شود خاصیت منظم بودن را دارد. چون این الگوریتم به عنوان زیربنای الگوریتم Matoušek [۴۸] استفاده می‌شود، می‌توانیم ثابت کنیم که برش ساخته شده توسط الگوریتم Matoušek نیز خاصیت منظم بودن را دارد. در لم زیر وجود یک $(۱/r)$-برش منظم عنوان می‌شود که در بخش بعدی برای رسیدن به توازن بین حافظه و زمان پرس‌وجو استفاده می‌شود.

لم ۴.۴ مجموعه L از n خط و پارامتر $r \leq n$ داده شده‌اند. می‌توان در مدت زمان $O(nr)$ یک $(۱/r)$-برش منظم با اندازه $O(r^۲)$ برای L ساخت و برای هر مثلث برش، مجموعه خطوطی که آن را قطع می‌کند، محاسبه نمود.

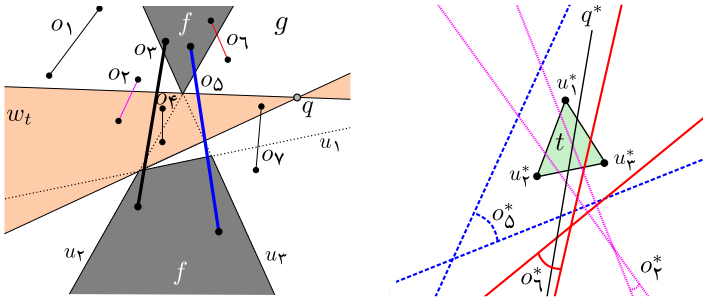
۲.۲.۴ الگوریتم با توازن بین حافظه و زمان پرس‌وجو برای محاسبه چندضلعی قابل دید

در این قسمت نشان می‌دهیم چگونه می‌توان برای محاسبه چندضلعی قابل دید، بین حافظه و زمان پرس‌وجو توازن برقرار کرد. فرض کنید همانند بخش قبل، O مجموعه پاره‌خط‌های مانع و P مجموعه نقاط انتهایی این پاره‌خط‌ها باشد. آرایش خطوط دوگان اعضای P را در صفحه دوگان با $\mathcal{A}(P^*)$ نمایش می‌دهیم. بر اساس این آرایش، یک $(۱/r)$-برش منظم ایجاد می‌کنیم که با $\mathcal{R}$ نمایش می‌دهیم. طبق تعریف منظم بودن برش، هر خط دلخواه حداکثر $O(r)$ مثلث از مثلث‌های $\mathcal{R}$ را قطع می‌کند.

نقطه دلخواه q را در صفحه و خط دوگان q را، که با q^* نمایش می‌دهیم، در نظر بگیرید. محل برخورد q^* با یال‌های $\mathcal{R}$، آن را به $O(r)$ پاره‌خط تقسیم می‌کند. این پاره‌خط‌ها در صفحه اصلی، $O(r)$ گوه دوگانه[۱۱] مجزا به مرکز q هستند که کل صفحه را می‌پوشانند. مثلث t از برش $\mathcal{R}$ را در نظر بگیرید که توسط q^* قطع شده است (شکل ۳.۴- الف). w_t را گوه دوگانه‌ای بگیرید که دوگان آن اشتراک q^* و t است (شکل ۳.۴- ب). همچنین $\mathcal{V}(q,t)$ را بخشی از $\mathcal{V}(q)$ بگیرید که به w_t محدود می‌شود. برای محاسبه $\mathcal{V}(q)$ کافی است $\mathcal{V}(q,t)$ را برای همه مثلث‌هایی که توسط q^* قطع می‌شوند جداگانه محاسبه کنیم و سپس آن‌ها را به ترتیب به یکدیگر بچسبانیم.

فرض کنید $u_۱$، $u_۲$ و $u_۳$ سه خطی در صفحه اصلی باشند که نقاط دوگان آن‌ها $u_۱^*$، $u_۲^*$ و $u_۳^*$ رئوس مثلث t باشند. آرایش این خطوط صفحه را به ۷ ناحیه تقسیم می‌کند. در شکل ۳.۴- ب این

[۱۱] Double wedge



(ب) صفحه اصلی (الف) صفحه دوگان

شکل ۳.۴ : الف. در صفحه دوگان، مثلث t توسط q^* قطع شده است. دوگان تعدادی از پاره‌خط‌های مانع نیز نشان داده شده است. ب. آرایش خطوط u_1، u_2 و u_3 و ناحیه‌های f و g و نقطه q و پاره‌خط‌های O در صفحه اصلی.

ناحیه‌ها دیده می‌شوند که از میان آن‌ها دو ناحیه تیره‌تر رسم شده‌اند. نقاط این دو ناحیه یا بالای u_1، u_2 و u_3 قرار دارند یا پایین این خطوط. فرض کنید f اجتماع این دو ناحیه تیره‌تر باشد و g مکمل f. خط دوگان هیچ نقطه‌ای روی f مثلث t را قطع نمی‌کند، زیرا یا بالا یا پایین همه رئوس t قرار دارد. در مقابل، دوگان هر نقاط روی g، که در شکل روشن‌تر رسم شده است، t را قطع می‌کند. دو نتیجه اولیه از این تعاریف این است که q در g قرار دارد و w_t گوه دوگانه‌ای به مرکز q است که دو مرز آن بر f مماس شده است (شکل ۳.۴- ب).

مجموعه پاره‌خط‌هایی از O که در صفحه دوگان، گوه دوگانه‌شان t را قطع می‌کند، شامل دو دسته پاره‌خط است. دسته اول پاره‌خط‌هایی هستند که یک یا هر دو سر آن‌ها در g قرار دارد. به این پاره‌خط‌ها، پاره‌خط‌های بسته t می‌گوییم و با مجموعه CS_t نمایش می‌دهیم. دسته دوم، پاره‌خط‌هایی هستند که g را قطع می‌کنند ولی دو سر آن‌ها در f قرار دارد، یعنی پاره‌خط‌هایی که یک سر آن‌ها بالای u_1، u_2 و u_3 است و سر دیگر، پایین u_1، u_2 و u_3 است. این پاره‌خط‌ها را پاره‌خط‌های باز t می‌نامیم و با مجموعه OS_t نمایش می‌دهیم.

شکل ۳.۴- الف، مثلث t را در صفحه دوگان نشان می‌دهد که توسط q^* قطع شده است. در شکل ۳.۴- ب، ناحیه‌های f، g و مجموعه‌ای از پاره‌خط‌هایی که g را قطع می‌کند، نشان داده شده است. g شامل q و حداقل یک رأس از هر پاره‌خط بسته (مثل o_2 و o_6) است. در مقابل f، که تیره‌تر رنگ آمیزی شده است، شامل حداکثر یک رأس از هر پاره‌خط بسته و هر دو رأس سایر پاره‌خط‌های O است. اگر یک سر یک پاره‌خط در یک ناحیه تیره و سر دیگر در ناحیه تیره دیگر باشد، آن پاره‌خط یک پاره‌خط باز است (مثل o_5). همانطور که در شکل ۳.۴- الف دیده می‌شود، گوه دوگانه دوگان پاره‌خط‌های باز، مثلث را به طور کامل در بر می‌گیرند، ولی گوه دوگانه دوگان پاره‌خط‌های بسته، فقط

بخشی از مثلث را می‌پوشانند.

با استفاده از مشاهدات بالا و این مطلب که پاره‌خط‌های O فقط در نقاط انتهایی ممکن است یکدیگر را قطع کنند، نتیجه می‌گیریم که پاره‌خط‌های OS_t ناحیه روشن شکل را به $|OS_t| + 1$ بخش تقسیم می‌کنند. فرض کنید $R_t = \{r_t^1, \ldots, r_t^{|OS_t|+1}\}$ مجموعه این بخش‌ها را نشان دهد. در هر بخش r_t^k، مجموعه پاره‌خط‌های $CS_t^k \subset CS_t$ قرار دارند. چون $\mathcal{V}(q,t)$ به بخش r_t^kای محدود است که q در آن قرار گرفته، برای محاسبه $\mathcal{V}(q,t)$، ابتدا باید r_t^k را پیدا کنیم و سپس $\mathcal{V}(q,t)$ را که به r_t^k و w_t محدود شده است تعیین کنیم.

ابتدا فرض کنید که بخش $r_t^k \in R_t$ را که q در آن قرار گرفته است، می‌دانیم. هر نقطه‌ای که در r_t^k قرار گرفته باشد، فقط پاره‌خط‌هایی را می‌بیند که در CS_t^k است. اگر در مرحله پیش‌پردازش، برای هر بخش r_t^k از روش بخش ۱.۴ با مجموعه موانع CS_t^k استفاده کنیم، می‌توانیم $\mathcal{V}(q,t)$ محدود به w_t را در زمان $O(\log(|CS_t^k|)) = O(\log n)$ محاسبه کنیم. برای جلوگیری از مصرف بیش از حد حافظه، این کار فقط برای بخش‌هایی انجام می‌شود که CS_t^k در آن‌ها ناتهی است. در این‌صورت، برای هر مثلث t در مرحله پیش‌پردازش، حافظه $O((n/r)^4)$ و زمان $O((n/r)^4 \log(n/r))$ صرف شده است.

در زمان پرس‌وجو، با دریافت نقطه q، بخش r_t^k را که q در آن قرار می‌گیرد، یافته و در صورتی که $CS_t^k \neq \emptyset$، با استفاده از داده‌ساختار بخش ۱.۴، $\mathcal{V}(q,t)$ تعیین می‌کنیم. اگر $CS_t^k = \emptyset$، r_t^k فقط از دو پاره‌خط $o_i, o_j \in OS_t$ تشکیل شده است، بنابراین می‌توان به راحتی $\mathcal{V}(q,t)$ را که به w_t محدود شده است، در زمان $O(1)$ محاسبه کرد.

برای تکمیل روش باید چگونگی ساخت سه داده‌ساختار مورد نیاز الگوریتم، یعنی R_t، CS_t^k و CS_t را در مرحله پیش‌پردازش مشخص کنیم. CS_t با اندازه $O(n/r)$ مجموعه پاره‌خط‌هایی است که دوگان حداقل یک سر آن‌ها t را قطع می‌کند. این مجموعه در زمان ساخت برش و در همان مدت $O(nr)$ قابل تعیین است. اما R_t با پیچیدگی $O(n)$ فضای زیادی را نیاز دارد که اگر بخواهیم برای همه مثلث‌ها آن را حساب کنیم، زمان و فضای پیش‌پردازش را بالا می‌برد.

برای محاسبه بخش r_t^k که نقطه p در آن قرار دارد، بر اساس نوع t از دو روش استفاده می‌کنیم. برای $O(r)$ مثلث بی‌کران برش، از لیست نیمه مرتب شده برای R_t استفاده می‌کنیم. در این لیست نیمه مرتب شده، R_t به $O(n/r)$ زیرلیست تقسیم می‌شود که هر کدام اندازه $O(r)$ دارند و برای هر $i < j$، بخش‌های زیرلیست $R_t(i)$ در سمت چپ بخش‌های زیرلیست $R_t(j)$ قرار دارند. این لیست نیمه مرتب را می‌توان در زمان $O(n \log(n/r))$ و صرف حافظه $O(n)$ به‌دست آورد که در مجموع برای همه مثلث‌های بی‌کران، به زمان $O(rn \log(n/r))$ و حافظه $O(rn)$ نیاز است. جستجو در این لیست نیمه مرتب شده نیز در زمان $O(r + \log(n/r))$ قابل انجام است.

برای سایر مثلث‌های برش، که کران‌دار هستند، ما تفاوت R_t و $R_{t'}$ را برای هر دو مثلث مجاور t و t' محاسبه می‌کنیم. این تفاوت‌ها در دو لیست مرتب شده $M_{t\to t'}$ و $M_{t'\to t}$ قابل نگه‌داری هستند. هر

$M_{t\to t'}$، بخش‌هایی از R_t را که به هنگام جابه‌جایی از مثلث t به t' در $R_{t'}$ به بخش‌های کوچکتر تقسیم شده و یا با بخش‌های دیگر ترکیب شده‌اند به صورت مرتب شده نگه‌داری می‌کند. این داده‌ساختار وقتی استفاده می‌شود که ما از مثلث t به t' می‌رویم، و نیز می‌دانیم که نقطه p در بخش r_t^k قرار دارد. ابتدا در $M_{t\to t'}$ به دنبال r_t^k می‌گردیم. اگر جستجو ناموفق بود، به این معناست که r_t^k در مثلث t' نیز وجود دارد و p در همان بخش قرار دارد. ولی اگر جستجو موفق بود، در $M_{t'\to t}$ به دنبال بخشی که شامل p است می‌گردیم. با این‌کار می‌توانیم بخشی را که p در آن قرار گرفته مشخص کنیم.

برای هر دو مثلث مجاور t و t'، فقط یک نوع از پاره‌خط‌های O می‌توانند بخشی را در $M_{t\to t'}$ قرار دهند. برای هر پاره‌خط o_i، اگر $o_i \in CS_t$ و $o_i \in OS_{t'}$ باید به $OS_{t'}$ اضافه شود و بنابراین یک تقسیم اتفاق می‌افتد. بخشی که تقسیم می‌شود به $M_{t\to t'}$ اضافه شده و دو بخش تولید شده جدید به $M_{t'\to t}$ اضافه می‌شوند. همین کار باید برای تغییر مکان از t' به t نیز انجام شود تا دو داده‌ساختار به‌درستی کامل شوند. چون اندازه $M_{t\to t'}$ از $|CS_t| + |CS_{t'}| = O(n/r)$ بزرگتر نیست، فضای مورد نیاز برای ساخت داده‌ساختارها برای همه مثلث‌ها حداکثر $O(rn)$ است. همچنین زمان لازم برای تعیین $r_{t'}^l$ از روی r_t^k برابر $O(\log(n/r))$ است.

تنها داده‌ساختار باقیمانده، CS_t^k است که باید برای هر بخش r_t^k مشخص شود. برای مشخص کردن CS_t^k، برای هر نقطه انتهایی p_i از پاره‌خط o_j، تمام بخش‌هایی که p_i در آن‌ها قرار می‌گیرد محاسبه می‌کنیم. کار را با مثلث بی‌کرانی شروع می‌کنیم که توسط p_i^* قطع می‌شود. می‌دانیم که می‌توان در مثلث بی‌کران t_∞، ناحیه $r_{t_\infty}^k$ را که p_i در آن قرار دارد، در زمان $O(r + \log(n/r))$ یافت. چون یک سر o_j مثلث t_∞ را قطع می‌کند، o_j جزو پاره‌خط‌های $CS_{t_\infty}^k$ است. در ادامه کار، همسایه t_∞، t_1، را در نظر می‌گیریم که p_i^* آن را قطع می‌کند. با استفاده از $M_{t_\infty\to t_1}$ و $M_{t_1\to t_\infty}$ می‌توان در زمان $O(\log(n/r))$ بخش $r_{t_1}^l$ای را که p_i در آن قرار می‌گیرد، تعیین کرد. این فرایند برای تمام مثلث‌هایی که p_i^* آن‌ها را قطع می‌کند تکرار شده و در نهایت با صرف زمان $O(r\log(n/r))$ برای تمام بخش‌های r_t^k که o_j در آن‌ها قرار می‌گیرد، o_j به CS_t^k اضافه می‌شود. تکرار این کار برای نقاط انتهایی همه پاره‌خط‌های O در مدت $O(rn\log(n/r))$، داده‌ساختار CS_t^k را برای همه بخش‌ها تکمیل می‌کند.

در زمان پرس‌وجو، برای تعیین $\mathcal{V}(q)$، کافی است روش بالا برای q تکرار شود و برای همه مثلث‌های t که توسط q^* قطع می‌شوند، بخش r_t^kای که q در آن قرار می‌گیرد مشخص می‌کنیم که این کار در مجموع به زمان $O(r\log(n/r))$ احتیاج دارد. با استفاده از بخش‌های تعیین شده، می‌توانیم در زمان $O(\log(n/r))$ بخشی از $\mathcal{V}(q)$ که به w_t محدود شده است را مشخص کنیم. با کنار هم قرار دادن این بخش‌های $\mathcal{V}(q,t)$، می‌توان $\mathcal{V}(q)$ را به‌طور کامل مشخص کرد. از آن‌چه گفته شد، می‌توان قضیه زیر را نتیجه گرفت.

قضیه ۲.۴ یک صحنه تشکیل شده از یک چندضلعی حفره‌دار یا تعدادی پاره‌خط، با پیچیدگی کلی $O(n)$ را می‌توان در داده‌ساختاری با اندازه $O(m)$ در زمان $O(m\log(\sqrt{m}/n))$ پیش‌پردازش کرد

($n^۲ \leq m \leq n^۴$)، به‌گونه‌ای که برای هر نقطه q، $\mathcal{V}(q)$ را در زمان $O(n^۲ \log(\sqrt{m}/n)/\sqrt{m})$ محاسبه کرد. علاوه بر این، $\mathrm{V}(q)$ را می‌توان در زمان $O(n^۲ \log(\sqrt{m}/n)/\sqrt{m} + |\mathrm{V}(q)|)$ گزارش کرد.

۳.۴ کاربردها

روش ارائه شده در بخش قبل برای محاسبه چندضلعی قابل دید یک نقطه، کاربردهای زیادی دارد. در این بخش به‌طور خلاصه نتیجه حاصل از بکارگیری این روش را برای دو مسأله نشان می‌دهیم. مسأله اول، نگه‌داری چندضلعی قابل دید یک ناظر نقطه‌ای متحرک است. چون ضمن حرکت ناظر، چندضلعی قابل دید تغییر می‌کند، و محاسبه آن از ابتدا نیز هزینه بالایی دارد، در مسائلی که ناظر در حال حرکت است، به‌روزرسانی چندضلعی قابل دید اهمیت زیادی دارد.

با نگاهی به روش بخش ۱.۴ مشاهده می‌کنیم که زمانی چندضلعی قابل دید یک نقطه از لحاظ ساختاری تغییر می‌کند که نقطه از یکی از قیود بحرانی عبور کند. بنابراین برای تعیین مکانی که ساختار چندضلعی قابل دید تغییر می‌کند، باید محلی را که ناظر از یک قید بحرانی می‌گذرد، مشخص کنیم. ناحیه‌ای که ناظر در حال حاضر در آن قرار دارد یک چندضلعی محدب است، بنابراین در صورتی که ناظر بر روی یک خط راست حرکت کند، می‌توان در زمان $O(\log n)$ محل برخورد آن با مرز این چندضلعی را تعیین کرد. این محل، جایی است که ناظر از یک قید بحرانی عبور می‌کند. با تعیین قیدی که ناظر از آن عبور کرده است، می‌توان ناحیه بعدی را که ناظر در آن قرار می‌گیرد مشخص کرد و چندضلعی قابل دید را نیز به‌روز رساند. همه این کارها در زمان $O(\log n)$ قابل انجام است. برای به‌روزرسانی چندضلعی قابل دید، کافی است بر اساس قید بحرانی قطع شده، یال و رأس مناسب را به ساختار ترکیبیاتی چندضلعی قابل دید اضافه و یا از آن حذف کنیم.

در حالت دوم الگوریتم (توازن بین حافظه و زمان پرس‌وجو)، وقتی نقطه ناظر روی خط راست حرکت می‌کند، دوگان آن حول نقطه‌ای ثابت دوران پیدا می‌کند. برای یافتن اولین محل تغییر در چندضلعی قابل دید، اولین محل تغییر را در تمام مثلث‌هایی که خط دوگان قطع می‌کند پیدا کرده و در بین آن‌ها محلی را که زودتر ناظر به آن می‌رسد، انتخاب می‌کنیم. در هر مثلث اولین محل تغییر به روش قبلی پیدا می‌شود. اگر اولین محل تغییر مثلث‌ها به همراه محل خروج از مثلث‌های فعلی و ورود به مثلث‌های جدید را در صف اولولیتی قرار دهیم، در زمان $O(\log r + \log(n/r)) = O(\log n)$ می‌توانیم اولین محل برخورد را پیدا کنیم و چندضلعی قابل دید را به‌روزرسانی کنیم. زمان ساخت صف اولویت، $O(r \log r + r \log(n/r)) = O(r \log n)$ است. با تغییر پارامترها مشابه مسأله قبلی، قضیه زیر نتیجه گرفته می‌شود.

قضیه ۳.۴ یک صحنه تشکیل شده از یک چندضلعی حفره‌دار یا تعدادی پاره‌خط، با پیچیدگی کلی $O(n)$

را می‌توان در داده‌ساختاری با اندازه $O(m)$ در زمان $O(m\log(\sqrt{m}/n))$ پیش‌پردازش کرد، به‌گونه‌ای که برای هر نقطه q که بر روی یک خط حرکت می‌کند، $\mathrm{V}(q)$ را در زمان $O(\frac{n^2}{\sqrt{m}}\log n + k\log n)$ به‌روزرسانی کرد که k در این رابطه تعداد تغییراتی است که در ساختار $\mathrm{V}(q)$ رخ می‌دهد. علاوه بر این، می‌توان اولین مکانی را که ساختار $\mathrm{V}(q)$ تغییر می‌کند در زمان $O(\log n)$ تعیین کرد و چندضلعی قابل دید را به‌روزرسانی کرد.

مسأله دوم که با استفاده از روش بخش‌های قبل قابل حل است، محاسبه چندضلعی قابل دید ضعیف یک پاره‌خط است. برای این‌کار، از یک سر پاره‌خط شروع کرده و همانند مسأله قبل، چندضلعی قابل دید نقطه را به‌روز می‌رسانیم. با این تفاوت که با هر تغییر در چندضلعی قابل دید، رئوس و اضلاعی که به تازگی از نقطه ناظر قابل دید هستند را در دنباله‌ای که چندضلعی قابل دید ضعیف را نگه‌داری می‌کند، ذخیره می‌کنیم. قضیه زیر نتیجه حاصل از این روش را خلاصه می‌کند.

قضیه ۴.۴ یک صحنه تشکیل شده از یک چندضلعی حفره‌دار یا تعدادی پاره‌خط، با پیچیدگی کلی $O(n)$ را می‌توان در داده‌ساختاری با اندازه $O(m)$ در زمان $O(m\log(\sqrt{m}/n))$ پیش‌پردازش کرد، به‌گونه‌ای که برای هر پاره‌خط rs، $\mathrm{WV}(rs)$ را در زمان $O(\frac{n^2}{\sqrt{m}}\log n + |\mathrm{V}(rs)\log n|)$ محاسبه کرد که k در این رابطه تعداد تغییراتی است که در ساختار چندضلعی قابل دید در داخل rs رخ می‌دهد.

۴.۴ خلاصه

در این فصل مسأله محاسبه چندضلعی قابل دید ناظر نقطه‌ای را مورد مطالعه قرار دادیم. در آغاز نشان دادیم می‌توان صفحه را به بخش‌هایی تقسیم کرد، به‌گونه‌ای که شکل کلی چندضلعی قابل دید در هر بخش ثابت باشد. با استفاده از این خاصیت، الگوریتمی با زمان پیش‌پردازش $O(n^4\log n)$ و زمان پرس‌وجوی $O(\log)$ ارائه دادیم. به دلیل بالا بودن زمان پیش‌پردازش و حافظه مورد نیاز برای الگوریتم و برای کاهش این مقادیر، الگوریتمی را ارائه دادیم که امکان برقراری توازن بین حافظه و زمان پرس‌وجو را فراهم می‌کرد. این الگوریتم به عنوان پیش‌نیاز به شکل خاصی از برش احتیاج داشت که آن را برش منظم نامیدیم و ثابت کردیم برش ساخته شده توسط یکی از الگوریتم‌های موجود، منظم است.

در انتهای فصل، دو کاربرد ابتدایی از الگوریتم‌های ارائه شده را ذکر کردیم. الگوریتم اول نگه‌داری چندضلعی قابل دید یک ناظر نقطه‌ای متحرک است و الگوریتم دوم محاسبه چندضلعی قابل دید ضعیف یک پاره‌خط. در فصل بعد خواهیم دید که چگونه با استفاده از الگوریتم ارائه شده در این فصل، می‌توان به توازن بین حافظه و زمان پرس‌وجو برای حل مسأله آزمون قابلیت دید دو پاره‌خط رسید.

فصل ۵

آزمون قابلیت دید دو شیء از هم (۲)

در فصل ۳ مسأله آزمون قابل دید بودن دو شیء را بررسی کردیم. در آنجا مسأله را برای انواع مختلف اشیاء (نقطه، پاره‌خط و چندضلعی) و در حالت‌های بدون پیش‌پردازش، SOQ یا پیش‌پردازش صحنه و ناظر، و TOQ یا پیش‌پردازش فقط صحنه مطالعه و راه‌حل‌هایی ارائه نمودیم. در این فصل به دنبال تکمیل راه‌حل مسأله برای اشیاء غیرنقطه‌ای و در حالت TOQ هستیم. به‌طور دقیق‌تر می‌خواهیم توازنی را بین حافظه و زمان پرس‌وجو برقرار کنیم، یعنی امکانی را فراهم نماییم تا با افزایش حافظه مصرفی، زمان پرس‌وجو را کاهش دهیم.

در بخش ۴.۳ نشان دادیم که با پیش‌پردازش چندضلعی حفره‌دار در زمان $O(n^{۲+\epsilon})$ و با استفاده از $O(n^۲)$ حافظه، می‌توان قابلیت دید بین دو شیء را در زمان $O(n)$ تشخیص داد. در لم ۳.۳ دیدیم که این مسأله از دو زیرمسأله دیگر تشکیل شده است. زیرمسأله اول یا CISD یافتن یالی از میان $O(n^۲)$ یال گراف توسعه یافته قابلیت دید صحنه است با این شرط که هر دو شیء را قطع کند و زیرمسأله دوم بررسی قابل دید بودن یکی از اشیاء از یکی از رئوس شیء دیگر.

در لم ۱.۳ توضیح دادیم که مسأله CISD امکان برقراری توازن بین حافظه و زمان پرس‌وجو را فراهم می‌کند، ولی برای زیرمسأله دوم چنین امکانی را نداشتیم. این نکته ما را به سمت یافتن روشی برای محاسبه چندضلعی قابل دید یک نقطه با امکان توازن بین حافظه و زمان پرس‌وجو سوق داد، تا این‌که توانستیم راه‌حلی با این قابلیت بیابیم، و توضیح آن در فصل ۴ آمد. در این فصل چگونگی استفاده از این راه‌حل را در مسأله آزمون قابلیت دید دو شیء برای رسیدن به توازن توضیح می‌دهیم.

در زیر یکبار دیگر لم اصلی تعیین قابلیت دید بین دو شیء را می‌بینیم.

لم ۱.۵ در چندضلعی حفره‌دار P، دو پاره‌خط s و t یکدیگر را به صورت ضعیف می‌بینند، اگر سر یکی از دو پاره‌خط، پاره‌خط دیگر را ببیند و یا یالی از گراف توسعه‌یافته قابلیت دید هم s و هم t را

قطع کند.

در این لم، ما دو زیرمسأله داریم. زیرمسأله (۱) آیا یکی از دو پاره‌خط، چندضلعی قابل دید یکی از دو سر پاره‌خط دیگر را قطع می‌کند؟ و زیرمسأله (۲) آیا یالی از گراف توسعه یافته قابلیت دید، هم s و هم t را قطع می‌کند؟ برای زیرمسأله (۲) نتیجه زیر را داریم که مستقیماً از لم ۱.۳ نتیجه گرفته می‌شود. همانطور که در این نتیجه دیده می‌شود، برخلاف لم ۱.۳ در اینجا m باید بین $n^۲$ و $n^۴$ باشد که این به دلیل تعداد یال‌های گراف توسعه یافته قابلیت دید چندضلعی است که $O(n^۲)$ است. به همین صورت زمان پرس‌وجو نیز تغییر یافته است.

نتیجه ۱.۵ یک چندضلعی حفره‌دار با n ضلع را می‌توان در زمان $O(n^{۲+\epsilon} + m\log^{\epsilon} n)$ و با صرف حافظه $O(m)$ به‌گونه‌ای پیش‌پردازش کرد که برای هر زوج پاره‌خط دلخواه s و t بتوان در زمان $O(\frac{n^۲}{\sqrt{m}}\log^۴(۱ + \frac{\sqrt{m}}{n}))$ وجود یالی از گراف توسعه یافته قابلیت دید که هر دوی s و t را قطع کند، مشخص کرد. در این رابطه‌ها m پارامتری با شرط $n^۲ \le m \le n^۴$ است.

همچنین می‌توان از الگوریتمی که Chan [۱۲] اخیراً ارائه کرده استفاده کرد، و ضریب n^{ϵ} را حذف کرد. در این صورت زمان پیش‌پردازش به $O(m\,\mathrm{polylog}\,n)$ به‌طور متوسط و زمان پرس‌وجو به $O(\frac{n^۲}{\sqrt{m}}\,\mathrm{polylog}\,n)$ به‌طور متوسط تغییر می‌یابند.

۱.۵ تعیین قابل دید بودن پاره‌خط از نقطه

اکنون به زیرمسأله (۱) توجه می‌کنیم که بررسی تلاقی یک پاره‌خط با چندضلعی قابلیت دید یک نقطه است. به‌طور دقیق‌تر، چندضلعی حفره‌دار P، نقطه s و پاره‌خط t داده شده‌اند. می‌خواهیم بدانیم که آیا t توسط s دیده می‌شود یا خیر. به عبارت دیگر «آیا t چندضلعی قابلیت دید s را قطع می‌کند؟» در ادامه این فصل توضیح می‌دهیم که چگونه می‌توان از نتیجه فصل ۴ استفاده کرد و این مسأله را حل کرد. ابتدا یک بار دیگر قسمت اول قضیه ۲.۴ را مرور می‌کنیم.

قضیه ۱.۵ یک صحنه تشکیل شده از یک چندضلعی حفره‌دار یا تعدادی پاره‌خط، با پیچیدگی کلی $O(n)$ را می‌توان در داده‌ساختاری با اندازه $O(m)$ در زمان $O(m\log(\sqrt{m}/n))$ پیش‌پردازش کرد $(n^۲ \le m \le n^۴)$، به‌گونه‌ای که برای هر نقطه q، $\mathcal{V}(q)$ را در زمان $O(n^۲\log(\sqrt{m}/n)/\sqrt{m})$ محاسبه کرد.

در روش ارائه شده برای محاسبه چندضلعی قابل دید، ما برای هر کدام از ناحیه‌های دارای $\mathcal{V}$ ثابت، یک درخت دودویی متوازن داریم که در زمان پرس‌وجو ریشه این درخت برگردانده می‌شود. بنابراین می‌توان از این درخت برای پردازش‌های بعدی روی چندضلعی قابل دید نقطه استفاده کرد.

زمانی که t، $\mathrm{V}(s)$ را قطع کند، دو حالت قابل تصور است. حالت اول این‌که حداقل یکی از دو سر t داخل $\mathrm{V}(s)$ قرار بگیرد. در حالت دوم هیچ‌یک از دو سر t داخل چندضلعی قرار نمی‌گیرند، بلکه هر دو بیرون $\mathrm{V}(s)$ هستند، ولی یک نقطه میانی t با $\mathrm{V}(s)$ اشتراک دارد. حالت اول را (i) و حالت دوم را (ii) می‌نامیم. حالت (i) به سادگی با یک جایابی نقطه در چندضلعی ستاره‌ای $\mathrm{V}(s)$ در زمان $O(\log n)$ قابل انجام است. در این حالت باید توجه داشت که نتیجه‌ای که الگوریتم فصل ۴ در قالب یک درخت دودویی متوازن به ما داده است، $\mathcal{V}(s)$ است و نه $\mathrm{V}(s)$، ولی به راحتی با یک جستجوی دودویی در درون درخت، می‌توان دو یالی از $\mathrm{V}(s)$ را که پشت یا جلوی هر یک از دو سر t قرار می‌گیرند، یافته و در زمان ثابت بررسی کرد که آیا این یال‌ها مانع دیده شدن هر یک از دو سر t هستند یا خیر.

حالت (ii) پیچیده‌تر است و حل آن نیاز به بررسی بیشتر دارد. بخش بعدی به حل این حالت اختصاص دارد.

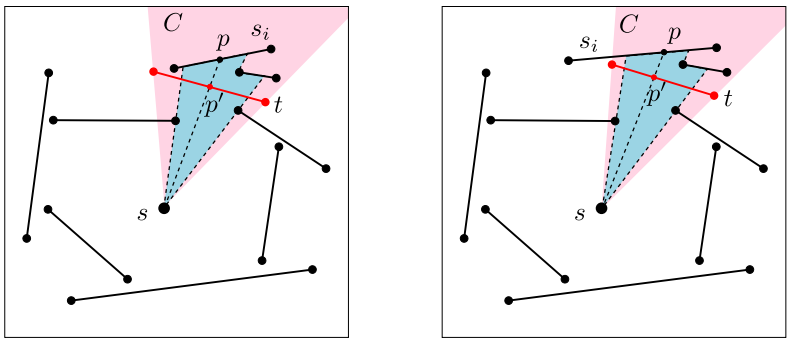
۱.۱.۵ تعیین قابل دید بودن نقطه میانی پاره‌خط از نقطه

در این بخش حالت (ii) از مسأله قابل دید بودن پاره‌خط از نقطه را بررسی می‌کنیم. فرض کنید $\{s_1, \ldots, s_n\}$ مجموعه یال‌های چندضلعی حفره‌دار P باشند. برای نقطه s و دو پاره‌خط t و s_i، دو عبارت P_1 به صورت «s و t در یک طرف خط حامل s_i قرار دارند» و عبارت P_2 به صورت «s و s_i در دو طرف خط حامل t قرار دارند» را در نظر بگیرید. همچنین فرض کنید C کنجی باشد که از اتصال s به دو سر t و امتداد دادن آن‌ها حاصل می‌شود. چون s داخل چندضلعی P قرار دارد، در ادامه فرض می‌کنیم $V(s)$ محدود است. در صورتی که s در میان مجموعه‌ای از پاره‌خط‌ها باشد، کافی است ابتدا فضای اطراف مجموعه پاره‌خط‌ها را با چهار پاره‌خط دیگر محدود کنیم. ابتدا لم زیر را اثبات می‌کنیم.

لم ۲.۵ در یک چندضلعی حفره‌دار، نقطه‌ای میانی از t از s قابل دید است اگر نقطه‌ای مثل p روی پاره‌خطی مثل s_i باشد، به‌طوری که p در C قرار داشته و از s قابل دید باشد و حداقل یکی از دو عبارت P_1 و P_2 برقرار باشد.

اثبات. $\Rightarrow$: فرض کنید نقطه p روی پاره‌خط s_i در C بوده و از s قابل دید باشد و همچنین P_1 برقرار باشد. در اینصورت پاره‌خط sp، t را قطع می‌کند و در نتیجه محل برخود sp با t از s قابل دید است. در حالتی که به جای P_1، P_2 برقرار باشد نیز همین شرط برقرار است.

$\Leftarrow$: اگر نقطه‌ای میانی از t مثل p' در شکل‌های ۱.۵ از s قابل دید باشد، پاره‌خطی مثل s_i پشت t (نسبت به s) و نقطه‌ای بر روی s_i قرار دارد که از s قابل دید است (بدون حضور t) و p روی شعاعی قرار دارد که از s به سمت p' ساطع می‌شود. چون s_i و p مجزا هستند و یکدیگر را قطع نمی‌کنند، خط حامل یکی، دیگری را قطع نمی‌کند، زیرا در غیر اینصورت این دو متلاقی خواهند بود (شکل‌های ۱.۵-الف و ب). به علاوه چون sp، t را قطع می‌کند، p در C قرار دارد و یکی از این دو حالت اتفاق می‌افتد:



ب الف

شکل ۱.۵ : نقطه‌ای میانی از t از s قابل دید است. ناحیه صورتی‌رنگ C و ناحیه آبی‌رنگ $\mathrm{V}(s)$ است. الف. s و t در یک طرف خط حامل s_i هستند. ب. s و s_i در دو طرف خط حامل t هستند.

یا s و دو سر t در یک طرف خط حامل s_i قرار دارند (شکل ۱.۵– الف) و یا خط حامل t ،s و s_i را جدا می‌کند (شکل ۱.۵– ب). □

حال با استفاده از این لم، مسأله تشخیص دیده شدن نقطه میانی t از s را حل می‌کنیم. مجموعه $S_{P_1 \vee P_2}$ را زیرمجموعه‌ای از S بگیرید که برای پاره‌خط‌های آن حداقل یکی از دو شرط P_1 یا P_2 برقرار است. روش کلی کار بدین صورت است که ما باید $\mathcal{V}(s)$ را حساب کنیم و بخشی از آن را که داخل C قرار می‌گیرد یافته، و بررسی کنیم که آیا پاره‌خطی مانند s_i در این بخش وجود دارد که در $S_{P_1 \vee P_2}$ باشد. یافتن پاره‌خطی مانند s_i به این صورت در زمان محدودی که برای ما وجود دارد، کار آسانی نیست. به‌همین دلیل راه‌حل را کمی تغییر می‌دهیم.

در ابتدا همه پاره‌خط‌هایی که در $\mathcal{V}(s)$ قرار دارند ولی در $S_{P_1 \vee P_2}$ قرار ندارند را حذف می‌کنیم. آن‌چه باقی می‌ماند، زیرمجموعه‌ای از $S_{P_1 \vee P_2}$ است که در $\mathcal{V}(s)$ قرار دارند، و رئوسی که ابتدا و انتهای حضور این پاره‌خط‌ها را در $\mathcal{V}(s)$ مشخص می‌کنند. توجه کنید که این رئوس می‌توانند رئوس انتهایی خود پاره‌خط‌ها باشد، و یا رئوس انتهایی پاره‌خط‌های دیگر، حتی رئوس پاره‌خط‌هایی که شرایط P_1 و P_2 برای آن‌ها برقرار نیست. با محدود کردن $\mathcal{V}(s)$ به پاره‌خط‌های مطلوب ما، کافی است آن قسمت از $\mathcal{V}(s)$ را که داخل C است مشخص کنیم. اگر این قسمت غیر تهی باشد، بخش میانی t از s قابل دید است، و اگر این قسمت تهی باشد، بخش میانی t از s قابل دید نیست.

مشکل باقیمانده فراهم کردن امکانی است که بتوان در زمان پرس‌وجو و در مدت زمان مناسبی، پاره‌خط‌های نامطلوب را از $\mathcal{V}(s)$ حذف کرد. این کار را با ترکیب داده‌ساختار چندلایه جستجوی نیم‌صفحه‌ها و داده‌ساختار محاسبه چندضلعی قابل دید انجام می‌دهیم.

ابتدا توجه کنید که پاره‌خط‌هایی که برای آن‌ها شرط P_1 یا P_2 برقرارند را می‌توان با داده‌ساختارهای چندلایه جستجوی نیم‌صفحه‌ها شناسایی کرد. برای P_1 ما به دنبال پاره‌خط‌هایی هستیم که هم s و هم هر دو سر t در بالا یا در پایین خط حامل آن پاره‌خط‌ها باشند. اگر ما نقاط دوگان خطوط حامل پاره‌خط‌های صحنه را برای سه لایه جستجوی نیم‌صفحه‌ها در داده‌ساختار $\mathcal{D}_{P_1}$ پیش‌پردازش کنیم و در زمان پرس‌وجو یک‌بار با نیم‌صفحه‌های بالای خطوط دوگان s و دو سر t و یک‌بار با نیم‌صفحه‌های پایین خطوط دوگان s و دو سر t جستجو را انجام دهیم، حاصل این جستجو پاره‌خط‌هایی هستند که خطوط حامل آن‌ها یا بالای s و دو سر t و یا پایین s و دو سر t هستند.

پاره‌خط‌های P_2، پاره‌خط‌هایی هستند که دو سر آن‌ها در نیم‌صفحه‌ای از خط حامل t قرار دارند که s قرار ندارد. بنابراین برای P_2 کافی است دو سر پاره‌خط‌های صحنه را برای دو لایه جستجوی نیم‌صفحه‌ها در داده‌ساختاری به نام $\mathcal{D}_{P_2}$ پیش‌پردازش کنیم، در لایه اول سر سمت راست پاره‌خط‌ها و در لایه دوم سر سمت چپ پاره‌خط‌ها. در زمان پرس‌وجو نیم‌صفحه‌ای از خط حامل t که s در آن قرار ندارد را در دو لایه جستجو استفاده می‌کنیم، حاصل پاره‌خط‌هایی از صحنه هستند که P_2 برای آن‌ها برقرار است.

بعد از یافتن مجموعه پاره‌خط‌هایی که برای آن‌ها یکی از شروط P_1 و P_2 برقرار است، باید بتوانیم به سرعت وضعیت $\mathcal{V}(s)$ را فقط برای این پاره‌خط‌ها مشخص کنیم. یعنی باید تعیین کنیم کدام پاره‌خط‌ها از آن میان در $\mathcal{V}(s)$ قرار دارند. این کار با ترکیب هر یک از داده‌ساختارهای $\mathcal{D}_{P_1}$ و $\mathcal{D}_{P_2}$ با داده‌ساختار محاسبه چندضلعی قابل دید نقطه قابل انجام است. داده‌ساختارهای $\mathcal{D}_{P_1}$ و $\mathcal{D}_{P_2}$، درخت‌های چندلایه‌ای هستند که در لایه انتهایی آن‌ها، هر گره معرف مجموعه‌ای از پاره‌خط‌ها هستند. اگر ما برای هر گره لایه انتهایی این دو درخت داده‌ساختار جدیدی با نام $\mathcal{D}_V$ محاسبه و به گره متصل کنیم به شکلی که $\mathcal{D}_V$ بخش‌هایی از $\mathcal{V}(s)$ را که فقط شامل پاره‌خط‌های آن گره هستند برگرداند، در زمان پرس‌وجو می‌توانیم در مدت زمان کوتاهی $\mathcal{V}(s)$ محدود شده به پاره‌خط‌های $S_{P_1 \vee P_2}$ را محاسبه کنیم.

بعد از تعیین $\mathcal{V}(s)$ محدود شده به $S_{P_1 \vee P_2}$ که با $\mathcal{V}_{P_1 \vee P_2}(s)$ نمایش می‌دهیم، باید بخشی از آن را که در C قرار می‌گیرد جدا کنیم و تعیین کنیم که آیا تهی است یا خیر. جدا کردن بخشی از $\mathcal{V}_{P_1 \vee P_2}(s)$ که در C است، به راحتی و با دو جستجوی دودویی در $\mathcal{D}_V$ قابل انجام است. بررسی تهی بودن این بخش هم در زمان ثابت صورت می‌گیرد. در صورتی که پاره‌خطی در $\mathcal{V}_{P_1 \vee P_2}(s)$ محدود به C وجود داشته باشد، نقطه‌ای میانی از t از s قابل دید است و در غیر اینصورت t قابل دید نیست.

قضیه زیر اثبات دقیقی از آن‌چه گذشت ارائه می‌دهد.

قضیه ۲.۵ یک چندضلعی حفره‌دار با n رأس را می‌توان در زمان $O(m \log^3 n \log(1 + \sqrt{m}/n))$ در داده‌ساختاری به اندازه $O(m \log^3 n)$ پیش‌پردازش کرد، به‌طوری که در زمان پرس‌وجوی $O(n^2 \log^3 n \log(1 + \sqrt{m}/n)/\sqrt{m})$ بتوان قابل دید بودن نقطه‌ای میانی از پاره‌خط پرس‌وجوی t را از نقطه پرس‌وجوی s مشخص کرد. در این رابطه‌ها m پارامتری با شرط $n^2 \leq m \leq n^4$ است.

اثبات. فرض می‌کنیم $m = \omega(n^۲)$، زیرا در غیر این‌صورت ما چندضلعی قابل دید s را بدون هیچ پیش‌پردازشی در زمان $O(n \log n)$ حساب کرده و در زمان $O(n)$ تلاقی آن را با t بررسی می‌کنیم. واضح است که شرایط قضیه برقرار می‌باشند.

طبق آن‌چه گفته شد، ابتدا داده‌ساختارهای $\mathcal{D}_{P_۱}$ و $\mathcal{D}_{P_۲}$ را می‌سازیم که داده‌ساختارهایی چندلایه برای جستجوی نیم‌صفحه‌هاست و به‌ازای نقطه s و پاره‌خط t، پاره‌خط‌هایی را به ما می‌دهند که $P_۱$ یا $P_۲$ برای آن‌ها برقرار است. در لایه انتهایی $\mathcal{D}_{P_۱}$ و $\mathcal{D}_{P_۲}$، هر گره معرف تعدادی پاره‌خط است که مجموعه کانونی آن گره نامیده می‌شود.

در مرحله دوم، برای مجموعه کانونی هر گره در آخرین لایه $\mathcal{D}_{P_۱}$ و $\mathcal{D}_{P_۲}$، داده‌ساختار $\mathcal{D}_V$ را می‌سازیم. ایده کلی روشی است که در فصل ۴ استفاده شد ولی در اینجا با کمی تفاوت استفاده می‌شود. در فصل ۴ ما مجموعه‌ای از پاره‌خط‌ها داشتیم و $\mathcal{V}(s)$ را که از همه این پاره‌خط‌ها تشکیل می‌شود، برای هر نقطه پرس‌وجو محاسبه کردیم. ولی در این مسأله، ما به $\mathcal{V}(s)$ محدود شده به زیرمجموعه‌ای از پاره‌خط‌ها نیاز داریم.

فرض کنید دو مجموعه S و $S' \subset S$ داده شده‌اند. می‌خواهیم داده‌ساختاری را بسازیم که برای هر نقطه پرس‌وجوی s، $\mathcal{V}(s)$ محدود شده به S' را در زمان کوتاهی محاسبه کند. ابتدا مسأله را در حالت زمان لگاریتمی بررسی می‌کنیم. تفاوت راه حل این مسأله، با راه حل مسأله در فصل ۴ در قیود بحرانی دو مسأله است. در فصل ۴ به ازای هر دو رأس قابل دید، دو قید بحرانی از این رئوس رسم می‌شد تا زمانی که به مرز چندضلعی برخورد کنند. در مقابل، برای مسأله فعلی، فقط قیود بحرانی را برای دو رأس قابل دید از پاره‌خط‌های S' یا یک رأس از پاره‌خط‌های S' و یک رأس از پاره‌خط‌های $S \setminus S'$ رسم می‌کنیم. در حالت اول برای هر دو رأس، قید بحرانی در داخل چندضلعی رسم می‌شود، و در حالت دوم فقط یک قید بحرانی از رأسی که پاره‌خط آن در S' نیست، رسم می‌شود.

بقیه مراحل الگوریتم مشابه فصل ۴ است. اگر $|S| = n$ و $|S'| = w$، تعداد قیود بحرانی $O(nw)$ و تعداد نواحی با $\mathcal{V}$ ثابت، $O(n^۲w^۲)$ است. زمان پیش‌پردازش و حافظه مصرفی به ترتیب $O(n^۲w^۲ \log n)$ و $O(n^۲w^۲)$ و زمان پرس‌وجو برابر $O(\log n)$ خواهد بود.

با در نظر گرفتن شکل جدید قیود بحرانی، می‌توان این نتیجه را به حالت توازن بین حافظه و زمان پیش‌پردازش تعمیم داد. در فصل ۴ برای رسیدن به توازن، ابتدا یک $(۱/r)$–برش برای خطوط دوگان نقاط انتهایی پاره‌خط‌ها محاسبه و در مرحله بعد برای هر مثلث این برش، پاره‌خط‌هایی که خطوط دوگان نقاط انتهایی آن‌ها مثلث را قطع کرده‌اند به روش اول با زمان لگاریتمی پیش‌پردازش شدند. حافظه مصرف شده برابر $O(r^۲(n/r)^۴) = O(n^۴/r^۲)$ و زمان پیش‌پردازش برابر $O(n^۴ \log n/r^۲)$ و زمان پرس‌وجو نیز برابر $O(r \log(n/r))$ بود.

اگر بخواهیم مشابه همین روش را برای مسأله فعلی اعمال کنیم، باید شکل جدیدی از $(۱/r)$–برش را استفاده کنیم. اگر خطوط دوگان نقاط انتهایی پاره‌خط‌های S را رسم کنیم، برش جدید

باید دارای این شرط باشد که هر مثلث برش حداکثر $1/r$ خطوط دوگان از مجموعه S و حداکثر $1/r$ خطوط دوگان از مجموعه S' را قطع کند. این کار با تغییر الگوریتم‌های برش موجود قابل انجام است. به عنوان مثال، می‌توانیم از الگوریتم Matoušek [۴۸] استفاده کنیم.

در الگوریتم Matoušek، برای ساخت یک $(1/r)$–برش برای مجموعه L با اندازه n، ابتدا یک زیرمجموعه تصادفی به اندازه r از L انتخاب می‌شود، و بعد از مثلث‌بندی آرایش این مجموعه خطوط، مثلث‌هایی که دارای شرط برش نیستند (بیشتر از n/r خط را قطع می‌کنند) با اجرای یک مرحله بهبود، و با استفاده از یک الگوریتم ضعیف‌تر برش، به مثلث‌های کوچک‌تر تقسیم می‌شوند. Matoušek نشان می‌دهد که این مثلث‌بندی خواص $(1/r)$–برش را دارد.

همین ایده را می‌توان برای دو مجموعه L و L' با اندازه‌های n و w استفاده کرد؛ یعنی یک زیرمجموعه تصادفی به اندازه r از L و یک زیرمجموعه تصادفی به اندازه r از L' می‌سازیم و آرایش این مجموعه خطوط را مثلث‌بندی می‌کنیم. در مرحله بعد، هر مثلثی که بیش از n/r خط از L یا بیش از w/r خط از L' را قطع کند، با استفاده از یک الگوریتم ضعیف‌تر برش، به مثلث‌های کوچک‌تر تقسیم می‌شود. در نهایت ما $O(r^2)$ مثلث خواهیم داشت که هرکدام حداکثر $1/r$ خطوط L و $1/r$ خطوط L' را قطع می‌کند.

با استفاده از $(1/r)$–برش به‌دست آمده، می‌توان مشابه فصل ۴، الگوریتم با زمان لگاریتمی را برای هر مثلث به‌طور جداگانه استفاده نمود و به توازن بین حافظه و زمان پرس‌وجو رسید. با این کار حافظه مصرف شده $O(n^2w^2/r^2)$ و زمان پیش‌پردازش $O(n^2w^2\log(1+nw/r^2)/r^2) = O(n^2w^2\log(1+n/r)/r^2)$ و زمان پرس‌وجو $O(r\log(1+nw/r^2)) = O(r\log(1+n/r))$ می‌باشد، با شرط $1 \le r \le n$. با تغییر پارامترها، حافظه مصرف شده $O(m)$ و زمان پیش‌پردازش $O(m\log(1+m/nw))$ و زمان پرس‌وجو $O(nw\log(m/nw)/\sqrt{m})$ خواهند بود به‌گونه‌ای که m باید در شرط $w^2 \le m \le w^2n^2$ صدق کند.

اکنون از داده‌ساختار به‌دست آمده می‌توانیم برای مسأله اصلی استفاده کنیم. مجموعه همه پاره‌خط‌ها را مجموعه S و مجموعه کانونی هر گره در لایه انتهایی $\mathcal{D}_{P_1}$ و $\mathcal{D}_{P_2}$ را مجموعه S' قرار می‌دهیم و مسأله محاسبه $\mathcal{V}(s)$ محدود شده به S' را حل می‌کنیم. نتیجه به‌دست آمده داده‌ساختار لازم برای تعیین قابل دید بودن نقطه‌ای میانی روی t از نقطه s است.

در زمان پرس‌وجو، ابتدا به سراغ $\mathcal{D}_{P_1}$ می‌رویم و یک بار در نیم‌صفحه‌های بالای خطوط دوگان دو سر t و خط دوگان s و بار دیگر در نیم‌صفحه‌های پایین خطوط دوگان دو سر t و خط دوگان s در $\mathcal{D}_{P_1}$ جستجو می‌کنیم. نتیجه شامل تعدادی مجموعه کانونی است که برای هر کدام داده‌ساختار $\mathcal{D}_V$ محاسبه شده است. در ادامه در هر کدام از این داده‌ساختارها جستجو را ادامه می‌دهیم و ابتدا $\mathcal{V}(s)$ را برای هر یک یافته و سپس بررسی می‌کنیم که آیا پاره‌خطی در یکی از آن‌ها وجود دارد که در کنج

C قرار بگیرد یا خیر. اگر برای یکی از این $\mathcal{V}(s)$ ها چنین پاره‌خطی یافت شد، جواب مسأله اصلی مثبت است. در غیر اینصورت به سراغ داده‌ساختار $\mathcal{D}_{P_۲}$ می‌رویم و مشابه همین کار را تکرار می‌کنیم. تفاوت کار در این داده‌ساختار در نیم‌صفحه‌های استفاده شده برای جستجو است. در هر دو لایه $\mathcal{D}_{P_۲}$، نیم‌صفحه‌ای از خط حامل t را استفاده می‌کنیم که s در آن قرار ندارد.

تحلیل زمان و حافظه: برای $\mathcal{D}_{P_۱}$ و $\mathcal{D}_{P_۲}$ از داده‌ساختار جستجوی چندلایه نیم‌صفحه با زمان لگاریتمی Matoušek [۴۹] استفاده می‌کنیم. ما در این‌جا این داده‌ساختار را برای صفحه توضیح می‌دهیم، ولی برای ابعاد بالاتر نیز قابل استفاده می‌باشد. در این داده‌ساختار یک ثابت ρ انتخاب شده و براساس آن درخت جستجو ساخته می‌شود. هر گره درخت، $\rho^۲$ فرزند دارد. به هر گره در سطح i، $O(n/\rho^{۲i})$ نقطه نسبت داده می‌شوند که این نقاط زیرمجموعه‌ای از نقاط پدر گره است. همچنین هر گره به ازای هر یک از فرزندان خود، بخشی از نقاط خود را به عنوان دو مجموعه کانونی آن فرزند در داده‌ساختاری ذخیره می‌کند. این مجموعه‌های کانونی، نقاطی است که اطمینان داریم بالا یا پایین صفحه پرس‌وجو قرار می‌گیرند. عمق درخت تا زمانی ادامه پیدا می‌کند که به ازای یک پارامتر دلخواه r هر برگ به تعداد n/r نقطه داشته باشد. واضح است که به این طریق، ارتفاع درخت $O(\log r)$ خواهد بود. در زمان پرس‌وجو، به ازای نیم‌صفحه داده شده، مسیری از ریشه به یکی از برگ‌ها طی می‌شود و در ضمن طی این مسیر، پاسخ جستجو مجموعه‌های کانونی گره‌های طی شده است.

Matoušek [۴۹] نشان داد که با انتخاب $r = n/\log^۲ n$ و استفاده از الگوریتم با حافظه خطی در برگ‌ها، داده‌ساختار جستجوی نیم‌صفحه با اندازه $O(nr) = O(n^۲/\log^۲ n)$ در زمان $O(n^۲ \log^\epsilon n/\log^۲ n)$ ساخته می‌شود. در این حد ϵ یک عدد مثبت دلخواه است. زمان پرس‌وجو در این داده‌ساختار $O(\log n)$ خواهد بود. ویژگی این داده‌ساختار این است که می‌توان هر یک از مجموعه‌های کانونی گره‌ها را برای پیش‌پردازش‌های بعدی استفاده کرد و داده‌ساختارهای جستجوی چندلایه ساخت. با این کار، به‌ازای هر لایه اضافه، یک ضریب لگاریتم به زمان پیش‌پردازش، حافظه و زمان پرس‌وجو اضافه می‌شود.

علت انتخاب مقدار r توسط Matoušek کاهش زمان پیش‌پردازش و حافظه بود. چون در الگوریتم ما عامل تعیین کننده این پارامترها داده‌ساختار $\mathcal{D}_V$ است، ما اندکی در داده‌ساختار Matoušek تغییر ایجاد و آن را ساده‌تر می‌کنیم. ما $r = n/\log n$ انتخاب می‌کنیم و بدین صورت دیگر احتیاجی به پیش‌پردازش نقاط گره‌ها نیست. چون هر گره حداکثر $O(\log n)$ نقطه خواهد داشت که می‌توان وجود تک تک نقاط را در بازه مورد نظر بررسی کرد.

حال پیچیدگی کلی حاصل از ترکیب $\mathcal{D}_{P_۱}$ و $\mathcal{D}_{P_۲}$ با داده‌ساختار $\mathcal{D}_V$ را محاسبه می‌کنیم. در مورد $\mathcal{D}_{P_۱}$، سه لایه داده‌ساختار جستجوی نیم‌صفحه و لایه بعد $\mathcal{D}_V$ است. در سطح i از هر لایه $\mathcal{D}_{P_۱}$، $\rho^{۲i}$ گره داریم که هر یک $۲\rho^۲$ مجموعه کانونی با اندازه $O(n/\rho^i)$ دارند. در لایه انتهایی، به ازای w نقطه اندازه $\mathcal{D}_V$ برابر $O(w^۲n^۲/u^۲)$ با شرط $۱ \le u \le n$ است. بنابراین داریم:

$$
\begin{aligned}
S(n) &= \sum_{i=0}^{\lceil \log r \rceil} 2\rho^2\rho^{2i} \sum_{j=0}^{\lceil \log r \rceil - i} 2\rho^2\rho^{2j} \sum_{k=0}^{\lceil \log r \rceil - i - j} 2\rho^2\rho^{2k} O((n/\rho^{i+j+k})^2 n^2/u^2) \\
&\leq \sum_{i=0}^{\lceil \log r \rceil} 2\rho^2\rho^{2i} \sum_{j=0}^{\lceil \log r \rceil - i} 2\rho^2\rho^{2j} \sum_{k=0}^{\lceil \log r \rceil - i - j} c_1\rho^{2k} n^4/\rho^{2i+2j+2k} u^2 \\
&\leq \sum_{i=0}^{\lceil \log r \rceil} 2\rho^2\rho^{2i} \sum_{j=0}^{\lceil \log r \rceil - i} 2\rho^2\rho^{2j} c_1 n^4 \log r/\rho^{2i+2j} u^2 \\
&\leq \sum_{i=0}^{\lceil \log r \rceil} 2\rho^2\rho^{2i} \sum_{j=0}^{\lceil \log r \rceil - i} c_2\rho^{2j} n^4 \log r/\rho^{2i+2j} u^2 \\
&\leq \sum_{i=0}^{\lceil \log r \rceil} 2\rho^2\rho^{2i} c_2 n^4 \log^2 r/\rho^{2i} u^2 \\
&\leq c_3 n^4 \log^3 r/u^2 \\
&= O(n^4 \log^3 n/u^2)
\end{aligned}
$$

زمان پیش‌پردازش نیز به همین روش محاسبه می‌شود و چون زمان ساخت $\mathcal{D}_V$ یک ضریب $\log(1+n/u)$ از میزان حافظه بیشتر است، این ضریب در زمان ساخت داده‌ساختار کلی نیز اعمال می‌شود، پس زمان ساخت داده‌ساختار $O(n^4 \log^3 n \log(1+n/u)/u^2)$ خواهد بود. در زمان پرس‌وجو لایه اول تعداد $O(\log n)$ مجموعه کانونی به عنوان نتیجه دارد که جستجو در هر کدام از این مجموعه‌ها ادامه پیدا می‌کند. به همین ترتیب لایه دوم و سوم تعداد مجموعه‌های کانونی را به $O(\log^2 n)$ و $O(\log^3 n)$ می‌رساند. در نهایت در هر کدام از این مجموعه‌ها و با استفاده از $\mathcal{D}_V$ نسبت داده شده به آن، در زمان کلی $O(u \log(1+n/u) \log^3 n)$ همه $\mathcal{V}(s)$ ها برای نقطه پرس‌وجوی s محاسبه می‌شود. با تغییر پارامتر مناسب، حدود تعیین شده در قضیه اثبات می‌گردد.

□

با حل حالت (ii) مسأله قابل دید بودن پاره‌خط از نقطه، می‌توان نتیجه زیر را عنوان کرد.

نتیجه ۲.۵ با پیش‌پردازش یک چندضلعی حفره‌دار با n رأس در زمان $O(m \log^3 n \log(1+\sqrt{m}/n))$ و با حافظه $O(m \log^3 n)$، می‌توان قابل دید بودن یک پاره‌خط پرس‌وجو را از یک نقطه پرس‌وجو در زمان $O(n^2 \log^3 n \log(1+\sqrt{m}/n)/\sqrt{m})$ تشخیص داد. در این رابطه‌ها داریم $n^2 \leq m \leq n^4$.

حال به سراغ مسأله اصلی می‌رویم. دیدیم که مسأله آزمون قابلیت دید ضعیف دو پاره‌خط به دو زیرمسأله شکسته شد. زیرمسأله اول، یافتن یالی از گراف توسعه یافته قابلیت دید صحنه بود که هر دو پاره‌خط پرس‌وجو را قطع کند. نتیجه حل این زیرمسأله در نتیجه ۱.۵ آمد. زیرمسأله دوم، تعیین قابل

دید بودن یکی از دو پاره‌خط از یک سر پاره‌خط دیگر بود. نتیجه حل این مسأله نیز در نتیجه ۲.۵ آمده است. با ترکیب این دو نتیجه می‌توان قضیه زیر را نتیجه گرفت.

قضیه ۳.۵ می‌توان یک چندضلعی حفره‌دار با n رأس را در داده‌ساختاری به اندازه $O(m\log^{۳} n)$ در زمان $O(n^{۲+\epsilon} + m\log^{۳} n\log(۱ + \sqrt{m}/n))$ پیش‌پردازش کرد، به‌گونه‌ای که بتوان در زمان $O(n^{۲}\log^{۳} n\log(۱+\sqrt{m}/n)/\sqrt{m})$ قابل دید بودن ضعیف دو پاره‌خط (شیء) پرس‌وجو را از یکدیگر تشخیص داد. در این رابطه‌ها ϵ عدد مثبت دلخواه است و $n^{۲} \le m \le n^{۴}$.

اگر حافظه مصرفی نتیجه ۱.۵ و نتیجه ۲.۵، با هم مقایسه شود، دیده می‌شود که به ازای مقدار یکسان m، حافظه لازم برای حل مسأله یافتن یال متلاقی با دو پاره‌خط کمتر است. اگر مقدار m را در دو نتیجه به‌گونه‌ای انتخاب کنیم که حافظه دو الگوریتم با هم برابر شود، زمان پرس‌وجوی نتیجه ۱.۵ کاهش می‌یابد و در نتیجه زمان تعیین کننده، زمان پرس‌وجوی نتیجه ۲.۵ خواهد بود، که این مطلب در قضیه ۳.۵ منعکس شده است.

قابل ذکر است که می‌توان از الگوریتمی که Chan [۱۲] اخیراً ارائه نموده استفاده کرد، و ضریب n^{ϵ} را حذف کرد. در این صورت زمان پیش‌پردازش به $O(m\,\mathrm{polylog}\,n)$ به‌طور متوسط و زمان پرس‌وجو به $O(\frac{n^{۲}}{\sqrt{m}}\,\mathrm{polylog}\,n)$ به‌طور متوسط تغییر می‌یابند.

۲.۵ خلاصه

در این فصل، دوباره مسأله فصل ۳ یعنی آزمون قابلیت دید دو شیء را مورد مطالعه قرار دادیم. هدف ما در این فصل کاستن زمان پیش‌پردازش و حافظه لازم برای الگوریتم با پرداخت هزینه بیشتر در زمان پرس‌وجو بود. با استفاده از الگوریتمی که در فصل ۴ برای محاسبه چندضلعی قابل دید نقطه ارائه کرده بودیم و با اعمال تغییراتی در آن، و نیز استفاده کارا از الگوریتم‌های جستجوی بازه، توانستیم به توازن بین حافظه و زمان پیش‌پردازش در مسأله آزمون قابلیت دید دو شیء برسیم. به طور دقیق‌تر، با انتخاب پارامتر m با شرط $n^{۲} \le m \le n^{۴}$، با داده‌ساختاری به اندازه $O(m\log^{۳} n)$، می‌توانیم مسأله آزمون قابلیت دو شیء را در زمان $O(n^{۲}\log^{۳} n\log(۱+\sqrt{m}/n)/\sqrt{m})$ حل کنیم.

فصل ٦

محاسبه چندضلعی قابل دید جزئی

در این فصل، تعریف قابلیت دید جزئی را مطرح کرده و براساس این تعریف مسائلی را عنوان و بررسی می‌کنیم. به‌بیان دقیق‌تر، ما دو مسأله «محاسبه چندضلعی قابل دید جزئی یک نقطه در چندضلعی حفره‌دار» و مسأله «یافتن بیشینه تعداد اشیائی که توسط شعاع‌های ساطع شده از یک نقطه قطع می‌شوند» را مطالعه می‌کنیم. در مورد مسأله اول، نشان می‌دهیم که چگونه می‌توان روش محاسبه چندضلعی قابل دید در فصل ۴ را به قابلیت دید جزئی تعمیم داد و در نهایت به توازن بین حافظه و زمان پرس‌وجو رسید. در ادامه از این نتیجه استفاده می‌کنیم و مسأله دوم را حل می‌کنیم.

١.٦ مقدمه

در ابتدا توضیحاتی در مورد شهود قابلیت دید جزئی و مسائل مرتبطی که پیش از این مطرح شده می‌دهیم. محیطی را تصور کنید که اشیاء نیمه شفاف در آن قرار دارند و نه به طور کامل نور را از خود عبور می‌دهند و نه مانع نور هستند. در این محیط، ناظر اشیاء پشت شیء نیمه شفاف را به طور مبهم می‌بیند. در فیزیک گفته می‌شود که چنین موادی درصدی از نور رسیده به خود را عبور می‌دهند و درصدی را منعکس می‌کنند. با توجه به این نکته، نور پس از عبور از تعدادی مانع، بخش زیادی از شدت خود را از دست می‌دهد و از ناظر قابل دریافت نیست. بنابراین قابلیت دید جزئی با تعیین پارامتر k تعریف می‌شود. در این شکل از قابلیت دید، ناظر فقط اشیائی را می‌بیند که حداکثر k مانع نیمه شفاف بین آن‌ها و ناظر وجود داشته باشد.

اگر قابلیت دید را به سایر حوزه‌های علمی گسترش دهیم، می‌توان کاربردهای زیادی برای قابلیت دید جزئی یافت. به عنوان مثال Fulek و همکارانش [۲۹] نشان دادند، براساس مدلی که Liu و

Hung [۴٦] برای سرویس موقعیت‌یابی بی‌سیم ارائه و ثبت امتیاز کرده‌اند، علائم ارسالی از یک حس‌گر حداکثر می‌توانند از k مانع عبور کنند و اگر بیش از k مانع بین حس‌گر و ایستگاه اصلی وجود داشته باشد، این علائم توسط ایستگاه اصلی دریافت نمی‌شود.

بجز این، Fulek و همکارانش [۲۹] مسأله‌ای که با قابلیت دید جزئی ارتباط دارد مطالعه کردند. برای توضیح کار آن‌ها، ابتدا باید چند نماد را تعریف کنیم. برای یک مجموعه S از n شیء در صفحه و نقطه p، $\tau(p,S)$ تعداد بیشینه اشیائی را نشان می‌دهد که توسط شعاع‌های ساطع شده از p قطع می‌شوند. برای مجموعه S، $\tau(S)$ کمینه مقدار $\tau(p,S)$ برای همه نقاط ممکن p می‌باشد. بیشینه مقدار $\tau(S)$ برای همه مجموعه اشیاء n عضوی S در صفحه را با $\tau(n)$ نمایش می‌دهیم. مسأله‌ای که آن‌ها بررسی کردند، یافتن حد بالا و پایین برای همه مقادیر $\tau(n)$ است. مسأله دیگری که به آن پرداختند، نحوه محاسبه $\tau(S)$ برای مجموعه دلخواه S است که نشان دادند در زمان $O(n^4 \log n)$ قابل انجام است.

در طی سه سال اخیر، چند کار دیگر در زمینه قابلیت دید انجام شده که زمینه مسائل مطرح شده در آن‌ها، مشابه قابلیت دید جزئی تعریف شده توسط ما است. اولین بار، Aichholzer و همکارانش [۳] مفهوم k–مودم[۱] را مطرح کردند. آن‌ها مسأله موزه هنر را تعمیم دادند و نگهبان‌هایی را در نظر گرفتند که بتوانند اشیاء پشت حداکثر k مانع را ببینند. انگیزه آن‌ها برای طرح این مسأله، پوشش فضای یک ساختمان توسط مودم‌های بی‌سیم بود. بر طبق مقاله آن‌ها، برد یک مودم بی‌سیم، هم به فاصله و هم به تعداد دیوارهای بین فرستنده و گیرنده ارتباط پیدا می‌کند که در ساختمان‌ها تأثیر تعداد دیوارها بسیار محسوس‌تر از فاصله است. هدف آن‌ها یافتن حدود پایین و بالای تعداد مودم مورد نیاز برای پوشش یک چندضلعی توسط k–مودم‌ها است.

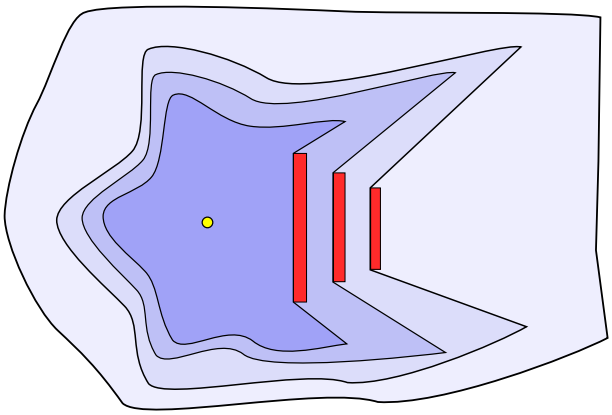
این کار، شروعی شد برای مجموعه‌ای از کارهای بعدی که مفهوم k–مودم را استفاده می‌کردند. Aichholzer و همکارانش [۳] ابتدا مسأله را در چندضلعی‌های یکنوا و متعامد یکنوا بررسی کردند و آرایشی از مودم‌ها را ارائه کردند که کل چندضلعی را پوشش می‌داد. بعد از آن Fabila-Monroy و همکارانش [۲۷] همین مسأله را در محیط‌های دیگر مثل مجموعه پاره‌خط‌ها، مجموعه خطوط و مجموعه پاره‌خط‌های عمودی و افقی مطالعه کردند. شماعی و همکارانش [٦۰] حد بالای مودم‌های لازم را برای چندضلعی‌های یکنوا کاهش دادند. علاوه‌براین، حد بالای تعداد مودم‌های مورد نیاز را برای چندضلعی‌های متعامد ارائه کردند. همچنین Ballinge و همکارانش [۹] نیز روی حدود بالا و پایین این مسأله در چندضلعی، زنجیره‌های چندضلعی‌گون و مجموعه پاره‌خط کار کردند. البته آن‌ها از عنوان k–فرستنده[۲] برای این مفهوم استفاده کردند.

Martins [۴۷] در رساله دکترای خود، مفهوم k–مودم را به‌کار برد و تعدادی از مسائل هندسه محاسباتی را حل کرد. از جمله راه‌حلی تقریبی برای مسأله موزه هنر با نگهبان‌های k–مودم ارائه کرد.

---

[۱] k-Modem

[۲] k-Transmitters



شکل ١.٦ : چندضلعی‌های قابل دید جزئی یک نقطه.

در ضمن این کار، مفهوم چندضلعی قابل دید k–مودم را تعریف و الگوریتمی برای محاسبه آن در زمان $O(n^2)$ برای یک چندضلعی دلخواه ارائه کرد. با تعاریفی که تاکنون ارائه شد، واضح است که چندضلعی قابل دید k–مودم، همان چندضلعی قابل دید جزئی با پارامتر k است.

مفهوم دیگری که پیش از این استفاده شده و با قابلیت دید جزئی ارتباط دارد، قابلیت دید میله‌ای k [3] است. در این شکل از قابلیت دید، تعدادی میله افقی وجود دارند و دو میله یکدیگر را می‌بینند اگگر پاره‌خطی عمودی آن دو را به هم وصل کند. این پاره‌خط حداکثر می‌تواند تعداد k میله دیگر را در فاصله بین دو میله قطع کند. این مفهوم را اولین بار Dean و همکارانش [٢٣] معرفی کردند و خواص گراف قابلیت دید مجموعه‌ای از میله‌ها را با این تعریف بررسی کردند. بعد از آن، Felsner و Massow [٢٨] و Hartke و همکارانش [٣٧] نیز در همین زمینه کار کردند.

٢.٦ تعریف مسأله

مجموعه S را در نظر بگیرید که شامل l چندضلعی محدب نیمه شفاف در صفحه است و در مجموع تعداد n رأس دارد. به این چندضلعی‌ها شیء می‌گوییم. می‌خواهیم بخش قابل دید صفحه را از یک ناظر نقطه‌ای در حضور این اشیاء محاسبه کنیم. این مسأله به محاسبه چندضلعی قابل دید نقطه مربوط می‌شود.

از آن‌جایی که اشیاء نیمه شفاف هستند، بخشی از نور از آن‌ها عبور می‌کند و در مقابل، شدت نور عبور کرده کاهش می‌یابد. برای سادگی بیشتر مسأله، فرض می‌کنیم کاهش شدت نور، فقط هنگام عبور نور از شیء اتفاق می‌افتد و به فاصله‌ای که در شیء طی می‌کند، بستگی ندارد. در این صورت،

[3] Bar k-visibility

صفحه به ناحیه‌هایی به شکل چندضلعی تقسیم می‌شود که هرکدام با شدت متفاوتی از ناظر قابل دید هستند (شکل ١.٦ را ببینید). هدف ما یافتن این چندضلعی‌هاست.

با استفاده از مفهوم قابلیت دید جزئی، برای نقطه q و پارامتر k، چندضلعی قابل دید جزئی q با پارامتر k را با $\mathrm{V}_k(q)$ نمایش می‌دهیم و بدین صورت تعریف می‌کنیم: مجموعه همه نقاطی از صفحه که قیود بحرانی آن نقاط به q حداکثر k شیء را قطع کند. به‌وضوح برای $j < k$، $\mathrm{V}_j(q)$ در داخل $\mathrm{V}_k(q)$ قرار دارد. همچنین $\mathrm{V}_0(q)$ همان چندضلعی قابل دید عادی نقطه است که در فصل ۴ نحوه محاسبه آن ارائه شد. با در نظر گرفتن تعریف $\tau(S)$ می‌توان نقطه‌ای مانند q را در صفحه یافت که به ازای $k \geq \tau(S)$، $\mathrm{V}_k(q)$ شامل همه نقاط صفحه باشد.

فرض کنید r_q شعاعی باشد که از q ساطع می‌شود. $\tau(r_q)$ را تعداد اشیائی که توسط r_q قطع می‌شوند تعریف می‌کنیم. $\tau(q)$ را $\max_{r_q} \tau(r_q)$ قرار می‌دهیم. $\tau(q)$ همان $\tau(q, S)$ است که توسط Fulek و همکارانش [۲۹] تعریف شد. با استفاده از مفهوم چندضلعی قابل دید جزئی، می‌توان $\tau(q)$ را به‌گونه‌ای دیگر نیز تعریف کرد. به این شکل، $\tau(q)$ کوچک‌ترین مقدار k است که $\mathrm{V}_k(q)$ شامل همه نقاط صفحه می‌باشد.

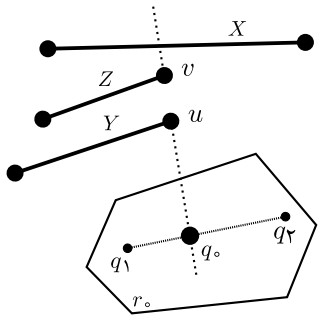
٣.٦ محاسبه چندضلعی قابل دید جزئی نقطه در زمان لگاریتمی

در این بخش نشان می‌دهیم چگونه می‌توان چندضلعی قابل دید جزئی یک نقطه را محاسبه کرد. راه حل ما به دو بخش زمان پیش‌پردازش و زمان پرس‌وجو تقسیم می‌شود. در زمان پیش‌پردازش مجموعه S داده می‌شود و ما داده‌ساختارهایی را برای پاسخ‌گویی به پرس‌وجو می‌سازیم. در زمان پرس‌وجو، نقطه q و پارامتر k داده می‌شوند و ما در زمان لگاریتمی $\mathrm{V}_k(q)$ را محاسبه می‌کنیم.

ایده اصلی مشابه روشی است که برای محاسبه چندضلعی قابل دید در بخش ۱.۴ استفاده کردیم. ابتدا صفحه را به ناحیه‌هایی تقسیم می‌کنیم به‌گونه‌ای که برای هر ناحیه، چندضلعی قابل دید جزئی نقطه q با پارامتر k شکلی ثابت داشته باشد. شباهت بین چندضلعی‌های قابل دید جزئی را ترتیب یکسان در یال‌ها و رئوس دیده شده از ناظر تعریف می‌کنیم. برای نقطه q و پارامتر k، لیست مرتب شده یال‌ها و رئوس قابل دید S را با $\mathcal{V}_k(q)$ نمایش می‌دهیم.

تقسیم صفحه به نواحی ذکر شده، با استفاده از قیود بحرانی رئوس موانع انجام می‌شود. تعریف قید بحرانی در این فصل کمی با تعریف فصل ۴ متفاوت است. به ازای دو رأس u و v، قید بحرانی c_{uv} پرتوی است که از u در خلاف جهت v ساطع می‌شود و تا رسیدن به مرز بیرونی چندضلعی حفره‌دار ادامه پیدا می‌کند. لم زیر خاصیت این قیود بحرانی را توضیح می‌دهد.

لم ١.٦ چندضلعی حفره‌دار P و دو نقطه q_1 و q_2 را در صفحه را در نظر بگیرید به‌گونه‌ای که q_1 و q_2



شکل ۲.٦ : قیود بحرانی مکان‌هایی را مشخص می‌کنند که تغییرات در قابلیت دید جزئی اتفاق می‌افتد.

روی هیچ قید بحرانی P قرار نگیرند. $\mathcal{V}_k(q_۱)$ و $\mathcal{V}_k(q_۲)$ به ازای یک مقدار k با هم متفاوتند، اگر یکی از قیود بحرانی P دو رأس $q_۱$ و $q_۲$ را از هم جدا کند.

اثبات. $\Rightarrow$: اگر قید بحرانی c_{uv}، دو رأس $q_۱$ و $q_۲$ را از هم جدا کند (شکل ۳.٦)، به ازای یک مقدار k، یکی از دو نقطه v را می‌بیند ولی دیگری نمی‌بیند و یا در صورت دیده شدن v در هر دو نقطه، ترتیب دیده شدن u و v متفاوت است. بنابراین $\mathcal{V}_k(q_۱)$ و $\mathcal{V}_k(q_۲)$ متفاوت هستند.

$\Leftarrow$: فرض کنید $\mathcal{A}$ آرایش قیود بحرانی P باشد. فرض خلف می‌کنیم که دو نقطه $q_۱$ و $q_۲$ وجود دارند که در یک ناحیه از $\mathcal{A}$ هستند و به ازای یک مقدار مشخص k، $\mathcal{V}_k(q_۱)$ و $\mathcal{V}_k(q_۲)$ با هم فرق می‌کنند. نقطه q را روی $q_۱$ قرار داده و به سمت $q_۲$ حرکت دهید. فرض کنید اولین جایی که $\mathcal{V}_k(q)$ تغییر می‌کند، نقطه $q_۰$ باشد. این تغییر به صورت اضافه یا کم شدن یک یال در لیست یال‌های قابل دید q است.

بدون از دست رفتن کلیت قضیه، فرض کنید این تغییر به شکل اضافه شدن یال Z بین دو یال X و Y باشد. این به معنای آن است که قبل از رسیدن به $q_۰$، تعداد اشیاء بین q و هر نقطه از Z حداقل $k+۱$ بوده است، ولی در $q_۰$، این تعداد حداکثر k است. این حالت فقط زمانی روی می‌دهد که قبل از رسیدن به $q_۰$ پاره‌خطی مانع دیده شدن Z باشد و در $q_۰$ این مانع از بین رفته باشد. همانطور که در شکل ۳.٦ دیده می‌شود، $q_۰$ باید روی قید بحرانی دو پاره‌خط (در شکل Y و Z) قرار داشته باشد. این مطلب خلاف فرض اولیه است که $q_۱$ و $q_۲$ در یک ناحیه قرار دارند. □

با استفاده از لم بالا، کافی است به ازای هر ناحیه در $\mathcal{A}$، $\mathcal{V}_k(p)$ را برای یک نقطه p در آن ناحیه محاسبه کنیم. به این روش می‌توان به قضیه زیر دست یافت.

قضیه ۱.٦ مجموعه‌ای از اشیاء با اندازه n را می‌توان در داده‌ساختاری با اندازه $O(n^۴)$ در زمان $O(n^۴ \log n)$ قرار داد، به‌گونه‌ای که به‌ازای هر نقطه پرس‌وجوی q و پارامتر صحیح k، $\mathrm{V}_k(q)$ را در زمان $O(\log n + |\mathrm{V}_k(q)|)$ محاسبه کرد.

اثبات. روش کلی کار بدین صورت است که ابتدا آرایش $\mathcal{A}$ را ساخته و صفحه را به نواحی محدبی تقسیم می‌کنیم که چندضلعی‌های قابل دید در هر کدام شکل ثابتی دارند. برای این تقسیم‌بندی و با پیمایش عمق اول ناحیه‌ها، دوری را به‌دست می‌آوریم که از هر ناحیه $\mathcal{A}$ عبور می‌کند و از هر یال $\mathcal{A}$ حداکثر دو بار می‌گذرد. هر دو ناحیه متوالی در این دور، در $\mathcal{A}$ مجاور هم هستند و چندضلعی‌های قابل دید جزئی آن‌ها بسیار شبیه به هم هستند. چندضلعی‌های قابل دید جزئی این دو ناحیه، بسته به خط جداکننده آن‌ها، فقط به ازای یک مقدار k با هم تفاوت دارند و به ازای سایر مقادیر یکسانند. علاوه‌براین، این اختلاف در یک یال و رأس است.

اگر برای نگه‌داری چندضلعی‌های قابل دید جزئی ناحیه‌ها از داده‌ساختار ماندگار[٤] استفاده کنیم، می‌توانیم حافظه و زمان ساخت داده‌ساختار لازم را به مقدار زیادی کاهش دهیم. توضیحات مربوط به داده‌ساختار ماندگار در بخش ۱.۴ آمده است.

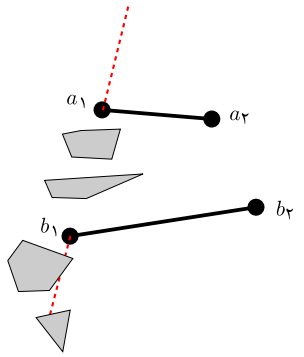
در ابتدا به ازای یک ناحیه دلخواه در دور، چندضلعی‌های قابل دید جزئی یک نقطه p در ناحیه را برای همه مقادیر k با شرط $0 \leq k \leq n$ محاسبه می‌کنیم و هریک را در یکی از $n+1$ داده‌ساختار ماندگار قرار می‌دهیم. چون $\mathcal{V}_k(p)$ یک لیست مرتب شده است، می‌توان آن را بدون هیچ ابهامی در داده‌ساختار ماندگار که به شکل درخت جستجوی دودویی است قرار داد.

در مرحله بعد، به ناحیه بعدی در دور رفته و مقدار k را می‌یابیم که به ازای آن $\mathcal{V}_k(p)$ از ناحیه قبلی به ناحیه فعلی تغییر می‌کند و این تغییر را در داده‌ساختار ماندگار مربوط به k اعمال می‌کنیم و $\mathcal{V}_k(p)$ جدید را به ناحیه فعلی نسبت می‌دهیم. به این طریق، چندضلعی‌های قابل دید جزئی ناحیه جدید در داده‌ساختارهای ماندگار خواهند بود و زمان و حافظه مصرف شده برای ناحیه جدید $O(\log n)$ و $O(1)$ می‌باشد. برای تعیین ریشه درخت‌هایی که چندضلعی‌های قابل دید جزئی هر ناحیه را نگه‌داری می‌کنند، یک داده‌ساختار ماندگار دیگر نیز لازم داریم. در این داده‌ساختار، $n+1$ اشاره‌گر که هر کدام چندضلعی قابل دید جزئی را برای یکی از مقادیر k مشخص می‌کند وجود دارند. چون به ازای هر تغییر ناحیه در دور، فقط یکی از این اشاره‌گرها به‌روزرسانی می‌شود، زمان و حافظه لازم برای نگه‌داری این داده‌ساختار نیز برای هر ناحیه $O(\log n)$ و $O(1)$ است.

در زمان پرس‌وجو، برای نقطه q، ابتدا ناحیه‌ای را که q در آن قرار می‌گیرد یافته، و در داده‌ساختار ماندگار مربوط به مقدار k، $\mathcal{V}_k(q)$ را می‌یابیم. بسته به نیاز، می‌توان همین داده‌ساختار را برگرداند یا این‌که در زمان متناسب با $|\mathrm{V}_k(q)|$، $\mathrm{V}_k(q)$ را محاسبه و برگرداند.

تحلیل زمان و حافظه: ساخت $\mathcal{A}$ که متشکل از $O(n^2)$ خط است، احتیاج به زمان $O(n^4)$ دارد. در همین مدت زمان، می‌توان دوری را ساخت که از همه ناحیه‌ها عبور کند. محاسبه و ذخیره $\mathcal{V}_k(p)$ به ازای $0 \leq k \leq n$ برای ناحیه اول در دور، در مدت $O(n^2 \log n)$ انجام می‌پذیرد. محاسبه چندضلعی‌های قابل دید جزئی برای سایر ناحیه‌ها و ذخیره در داده‌ساختارهای ماندگار در مدت زمان کلی $O(n^4 \log n)$

[٤] Persistent data structure



شکل ٣.٦ : بهبود آرایش $\mathcal{A}$ با کوچک کردن قیود بحرانی. در این شکل $k_{\max} = 3$.

و با حافظه $O(n^4)$ صورت می‌گیرد. همچنین باید $\mathcal{A}$ را برای جایابی نقطه پیش‌پردازش کرد که این کار در زمان $O(n^4)$ و با صرف حافظه $O(n^4)$ قابل انجام است [١٥].

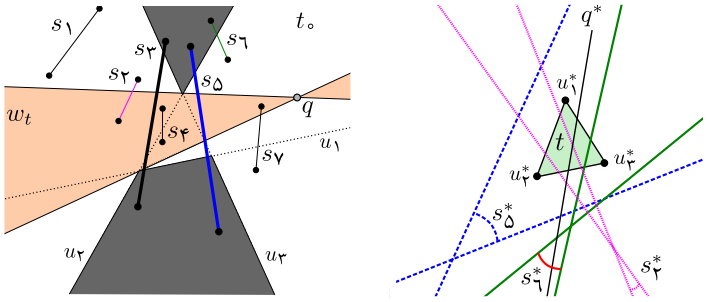
در زمان پرس‌وجو، یک جایابی نقطه و یک جستجو در داده‌ساختارهای ماندگار لازم است تا $\mathcal{V}_k(q)$ یافته شود، که همه این‌ها در زمان $O(\log n)$ انجام می‌شود. اگر نیازمند محاسبه دقیق $\mathrm{V}_k(q)$ باشیم، این کار در زمان اضافی $O(|\mathrm{V}_k(q)|)$ انجام می‌شود. □

در این‌جا مناسب است نکته‌ای در مورد بهینه‌سازی آرایش $\mathcal{A}$ عنوان کنیم. اگر بیشینه مقدار ممکن k در پرس‌وجوها مشخص باشد، و برابر $k_{\max}$ باشد، زمانی که قید بحرانی دو نقطه a_1 و b_1 را رسم می‌کنیم، کافی است شعاع را از b_1 تا زمانی امتداد دهیم که نقاط روی آن، حداکثر $k_{\max}$ شیء را قبل از a_1 ببینند و به‌محض این‌که شیء $k_{\max}+1$ام قطع شد، دیگر خط واصل را امتداد ندهیم. همین مطلب در مورد امتداد خط از سمت a_1 نیز صحیح است (شکل ٣.٦ را ببینید).

٤.٦ برقراری توازن بین حافظه و زمان پرس‌وجو در الگوریتم

در این بخش، تلاش می‌کنیم تا همانند آنچه در بخش ٢.٤ دیدیم، بین حافظه و زمان پرس‌وجوی الگوریتم بخش قبل توازنی برقرار کنیم. نکته کلیدی اصلی برای رسیدن به چنین توازنی، همانطور که در بخش ٢.٤ عنوان شد، استفاده از یک $(1/r)$-برش منظم است. این برش، دارای این خاصیت است که هر خط دلخواه در صفحه، $O(r)$ مثلث از $O(r^2)$ مثلث برش را قطع می‌کند. نحوه ساخت چنین برشی را در بخش ١.٢.٤ دیدیم. لم زیر خصوصیات این برش را خلاصه می‌کند که پیش از این اثبات شد.

لم ٢.٦ n خط در صفحه داده شده‌اند. به‌ازای پارامتر ورودی r، با شرط $1 \le r \le n$ می‌توان صفحه را به $O(r^2)$ مثلث تقسیم کرد، به‌گونه‌ای که هر مثلث توسط $O(n/r)$ خط قطع شود و هر خط جدید



(ب) صفحه اصلی (الف) صفحه دوگان

شکل ۴.٦ : الف. در صفحه دوگان، مثلث t، دوگان ناحیه t_0 توسط q^* قطع می‌شود. در شکل دوگان تعدادی از پاره‌خط‌ها دیده می‌شوند. ب. t_0، q و یال‌هایی از اشیاء در صفحه اصلی.

دلخواه حداکثر $O(r)$ مثلث را قطع کند.

در ادامه، نحوه برقراری توازن را توضیح می‌دهیم. مجموعه P را مجموعه رئوس S در نظر بگیرید که شامل n نقطه است. دوگان نقاط P، n خط هستند که مجموعه این خطوط را با P^* نمایش می‌دهیم. در آغاز $(1/r)$-برشی با اندازه $O(r^2)$ برای این مجموعه خطوط می‌سازیم، به‌گونه‌ای که هر مثلث برش $O(n/r)$ خط از P^* را قطع کند.

نقطه پرس‌وجوی q را در نظر بگیرید. دوگان این نقطه، q^*، خطی است که $O(r)$ مثلث برش (منظم) را قطع می‌کند. پاره‌خط اشتراک q^* با هر کدام از مثلث‌ها، دوگان یک گوه دوگانه به مرکز q است و مجموعه این گوه‌های دوگانه، همه صفحه را به $O(r)$ بخش تقسیم می‌کند. حال کافی است در هر کدام از این گوه‌های دوگانه، $\mathcal{V}_k(q)$ را محاسبه و آن‌ها را به ترتیب به هم بچسبانیم تا نتیجه $\mathcal{V}_k(q)$ کامل شود. $\mathcal{V}_k(q)$ محدود شده به گوه دوگانه حاصل از اشتراک q^* و t را با $\mathcal{V}_k(q,t)$ نمایش می‌دهیم.

در شکل ۴.٦- الف، مثلث t از برش، در صفحه دوگان دیده می‌شود که توسط q^* قطع شده است. طبق تعریف برش، t توسط $O(n/r)$ خط دیگر نیز قطع می‌شود که در صفحه اصلی، نقاط انتهایی $O(n/r)$ یال از اشیاء S هستند. ما این یال‌های S را پاره‌خط‌های بسته t می‌نامیم و با CS_t نمایش می‌دهیم. طبق این تعریف، دوگان حداقل یکی از دو سر هر پاره‌خط CS_t، t را قطع می‌کند. بجز پاره‌خط‌های CS_t، ممکن است پاره‌خط‌های دیگری نیز در $\mathcal{V}_k(q)$ حضور داشته باشند (مثلا پاره‌خط s_5 در شکل ۴.٦- ب). این یال‌های S را پاره‌خط‌های باز t می‌نامیم و با OS_t نمایش می‌دهیم. طبق تعریف، OS_t شامل همه پاره‌خط‌هایی است که گوه دوگانه دوگان آن‌ها t را به‌طور کامل می‌پوشاند.

ناحیه t_0 که دوگان آن t است، همان‌طور که در شکل ۴.٦- ب دیده می‌شود، صفحه را به سه ناحیه تقسیم می‌کند. یک ناحیه که در شکل به رنگ سفید است، ناحیه‌ای است که q و حداقل یک سر پاره‌خط‌های CS_t قرار دارند (مثلاً پاره‌خط‌های s_6 و s_7. در مقابل، دو ناحیه دیگر نیز داریم، که در

شکل به رنگ تیره نشان داده شده‌اند و حداکثر یک سر پاره‌خط‌های CS_t و هر دو سر سایر پاره‌خط‌های S در آن قرار دارند. اگر یک سر یک پاره‌خط در یک ناحیه تیره و سر دیگر در ناحیه تیره دیگر باشد، آن پاره‌خط جزء پاره‌خط‌های OS_t است (مثلاً پاره‌خط‌های s_3 و s_5).

می‌توان به‌راحتی دید که پاره‌خط‌های OS_t ناحیه سفیدرنگ را به $|OS_t|+1$ بخش تقسیم می‌کنند. فرض کنید $R_t = \{r_t^1, \ldots, r_t^{|OS_t|+1}\}$ مجموعه این بخش‌ها باشد. در هر بخش r_t^k زیرمجموعه پاره‌خط‌های $CS_t^k \subset CS_t$ قرار دارد. هر بخش r_t^k دارای دو دیواره از پاره‌خط‌های OS_t است.

به دلیل آن‌که ما دید نقطه q را محدود به گوه دوگانه مثلث t نیاز داریم، می‌توانیم پاره‌خط‌های OS_t را مانند خطوط نامتناهی فرض کنیم. با این کار، دید نقطه q محدود شده به گوه دوگانه تغییری نمی‌کند. با توجه به این‌که در دید جزئی، پاره‌خط‌های OS_t مانند بخش ۲.۴ به عنوان یک مانع کامل عمل نمی‌کنند و ناظر می‌تواند بخش‌های پشت پاره‌خط‌ها را ببیند، باید در الگوریتم تغییراتی اعمال کنیم.

۱.۴.٦ نحوه محاسبه و نگه‌داری داده‌ساختارهای لازم

در بخش ۲.۴ دیدیم که باید سه داده‌ساختار لازم را نگه‌داری کنیم و روش‌های نگه‌داری را توضیح دادیم. در این بخش تنها دو داده‌ساختار از این میان برای ما لازمند که عبارتند از CS_t و R_t. CS_t مجموعه پاره‌خط‌هایی است که حداقل یک سر آن‌ها در ناحیه سفید t_0 قرار می‌گیرند. در صفحه دوگان، حداقل یکی از دو مرز گوه دوگانه دوگان این پاره‌خط‌ها t را قطع می‌کند. این پاره‌خط‌ها در زمان ساخت برش به همراه خود برش محاسبه می‌شوند. پس CS_t برای هر مثلث t در برش در زمان ساخت برش قابل محاسبه است.

در مقابل، R_t با اندازه $O(n)$ به‌طور جداگانه برای هر مثلث قابل محاسبه و نگه‌داری نیست. زیرا حافظه در اختیار ما برای هر مثلث حداکثر $O((n/r)^4)$ است که به ازای برخی از مقادیر r از $O(n)$ کمتر است. برای محاسبه و نگه‌داری R_t ها از شباهت این مجموعه‌ها در مثلث‌های مجاور هم در برش استفاده می‌کنیم. البته چون در این مسأله ما نیاز به جستجو در همه ناحیه‌های R_t داریم، مانند بخش ۲.۴ عمل نمی‌کنیم. بلکه در این حالت، از داده‌ساختار ماندگار برای نگه‌داری این ناحیه‌ها در هر مثلث استفاده می‌کنیم. چون تفاوت ناحیه‌ها در R_t و $R_{t'}$ به ازای دو مثلث مجاور t و t'، $O(n/r)$ است، اندازه داده‌ساختار ماندگار و زمان ساخت آن به ترتیب برابر خواهد بود با $O(nr)$ و $O(nr\log(n/r))$.

بعد از محاسبه R_t برای همه ناحیه‌های همه مثلث‌ها، نوبت به پیش‌پردازش پاره‌خط‌ها می‌رسد. همان‌طور که گفته شد، این‌بار پیش‌پردازش لازم با پیش‌پردازشی که در بخش ۲.۴ انجام شده، اندکی فرق می‌کند. دلیل آن هم این است که در این مسأله، پاره‌خط‌های OS_t نمی‌توانند نقش مانع کامل را بازی کنند و دید ناظر را محدود کنند. بلکه فقط درصدی از قدرت دید را کم می‌کنند. برای حل این مشکل، به‌جای اینکه در هر ناحیه r_t^k، آرایش قیود بحرانی به‌طور جداگانه رسم شود، این آرایش برای

همه پاره‌خط‌های CS_t و در کل ناحیه سفید رنگی که توسط دوگان t مشخص شده است، رسم می‌شود. تفاوت این آرایش با آرایش محاسبه شده در بخش ۳.٦ در این است که علاوه بر پاره‌خط‌های CS_t، پاره‌خط‌های OS_t نیز وجود دارند که تاثیری در تعداد قیود بحرانی ندارند و فقط نقش مانع نیمه شفاف را در آرایش بازی می‌کنند. ما برای این‌که پاره‌خط‌های OS_t پیچیدگی آرایش را افزایش ندهند، تا هنگام پاسخ‌گویی به پرس‌وجو، آن‌ها را نادیده می‌گیریم.

بنابراین آرایش $\mathcal{A}$ از قیود بحرانی نقاط انتهایی پاره‌خط‌های CS_t را محاسبه می‌کنیم. برای تکمیل کار، نقاط برخورد خطوط دوگان رئوس مثلث t را نیز به این مجموعه اضافه کرده و قیود بحرانی را برای نقاط انتهایی پاره‌خط‌های CS_t و این نقاط در $\mathcal{A}$ در نظر می‌گیریم. سپس در $\mathcal{A}$ مانند قبل یک تور را که از همه ناحیه‌ها عبور کند ساخته و برای یک نقطه دلخواه q در هر ناحیه و برای همه مقادیر k، $1 \leq k \leq n$، $\mathcal{V}_k(q)$ را محاسبه و مانند بخش ۳.٦ آن‌ها را در داده‌ساختارهای ماندگار قرار می‌دهیم. ناحیه r_t^kای که نقطه q در آن انتخاب شده است، در این حالت برای ما مهم است، بنابراین این ناحیه را نیز در کنار اطلاعات دیگر نگه‌داری می‌کنیم.

۲.۴.٦ پاسخ به پرس‌وجو

برای نقطه پرس‌وجوی q، و برای هر مثلث t که توسط q^* قطع می‌شود، در داده‌ساختارهای ماندگار می‌توانیم $\mathcal{V}_k(q,t)$ را بیابیم. این کار با جایابی نقطه در آرایش $\mathcal{A}$ مربوط به مثلث t و جستجو در داده‌ساختارهای ماندگار انجام می‌شود. نکته‌ای که باقی می‌ماند این است که ممکن است تعداد موانعی که q واقعاً در هر جهت می‌بیند با تعدادی که در $\mathcal{V}_k(q,t)$ ذخیره شده یکی نباشد. این مشکل به این دلیل رخ داده که ما پاره‌خط‌های OS_t را حذف کردیم. برای حل این مشکل، کافی است ناحیه r_t^i که q در آن قرار گرفته با ناحیه r_t^k که p در آن قرار گرفته و در زمان پیش‌پردازش $\mathcal{V}_k(p,t)$ برای آن محاسبه شده است، مقایسه کنیم. تعداد موانع OS_t که در بین این دو ناحیه قرار گرفته‌اند میزان اختلاف دید این دو نقطه را نشان می‌دهند. مثلاً اگر w مانع بین q و p وجود داشته باشد و p سمت چپ q قرار داشته باشد، قسمت سمت چپ $\mathcal{V}_k(p,t)$، معادل $\mathcal{V}_{(k+w)}(q,t)$ و قسمت سمت راست $\mathcal{V}_k(p,t)$ معادل $\mathcal{V}_{(k-w)}(q,t)$ خواهد بود. حال با استفاده از این توضیحات $\mathcal{V}_k(q,t)$ را در زمان $O(\log(n/r))$ برای هر مثلث محاسبه می‌کنیم. جواب نهایی حاصل اتصال جواب‌های بخش‌های مختلف است. چون تعداد مثلث‌هایی که توسط q^* قطع می‌شوند $O(r)$ است، $\mathcal{V}_k(q)$ در زمان $O(r\log(n/r))$ محاسبه می‌شود.

به‌طور خلاصه، زمان پیش‌پردازش الگوریتم $O(n^4\log(n/r)/r^2)$، حافظه مصرف شده $O(n^4/r^2)$، و زمان پیش‌پردازش برابر $O(r\log(n/r))$ است. اگر حافظه مصرف شده را با m نمایش دهیم، خواهیم داشت:

قضیه ۲.٦ با پیش‌پردازش مجموعه S از اشیاء با پیچیدگی کلی n در مدت زمان $O(m\log(1+\sqrt{m}/n))$

با صرف حافظه $O(m)$ با شرط $n^۲ \leq m \leq n^۴$، برای نقطه پرس‌وجوی q، می‌توان چندضلعی قابل دید جزئی با پارامتر k، $\mathcal{V}_k(q)$ را در زمان $O(n^۲ \log(۱ + \sqrt{m}/n)/\sqrt{m})$ محاسبه کرد. به‌علاوه اگر $\mathrm{V}_k(q)$ خواسته شده باشد، می‌توان در زمان $O(n^۲ \log(۱ + \sqrt{m}/n)/\sqrt{m} + |\mathrm{V}_k(q)|)$ آن را محاسبه کرد.

۵.٦ چندضلعی قابل دید جزئی نقطه متحرک

با استفاده از ایده محاسبه چندضلعی قابل دید جزئی، می‌توان روشی را برای محاسبه و به‌روزرسانی چندضلعی قابل دید جزئی نقطه متحرک ارائه داد. در قضیه زیر جزئیات این روش دیده می‌شود.

قضیه ۳.٦ چندضلعی قابل دید جزئی یک نقطه متحرک در صفحه که بر خط راست حرکت می‌کند در زمان $O(n^۲ \log n/\sqrt{m} + i \log n)$ قابل محاسبه و نگه‌داری است، که i تعداد تغییرات روی داده در شکل همه چندضلعی‌های قابل دید جزئی است. زمان پیش‌پردازش و حافظه مصرفی $O(m \log(۱ + \sqrt{m}/n))$ و $O(m)$ است، که m پارامتری با شرط $n^۲ \leq m \leq n^۴$ می‌باشد.

اثبات. ابتدا حالت $m = n^۴$ را در نظر می‌گیریم. در آغاز، نقطه پرس‌وجوی q در داخل ناحیه محدب c از آرایش قیود بحرانی قرار دارد. در صورتی‌که q روی خط راست حرکت کند، در زمان $O(\log n)$ می‌توان تعیین کرد که q از کدام یال c از آن خارج می‌شود. وقتی q از c خارج شد، به ازای یکی از مقادیر k، $\mathcal{V}_k(q)$ تغییر می‌کند. در صورتی که این تغییر در چندضلعی قابل دید جزئی مورد نظر ما بود، آن را به‌روزرسانی کرده و کار را به همین شکل ادامه می‌دهیم.

اکنون حالتی را فرض کنید که $m = o(n^۴)$. در این حالت ما در آغاز برای محاسبه چندضلعی قابل دید جزئی q از ساختار $(۱/r)$-برش استفاده کرده‌ایم. وقتی q روی خط راست حرکت می‌کند، دوگان آن، q^*، حول نقطه‌ای ثابت دوران پیدا می‌کند. در این صورت ما به دنبال یافتن اولین محلی هستیم که ترتیب قطع شدن خطوط دوگان رئوس S توسط q^* تغییر کند. برای این کار کافی است، اولین محل تغییر را به‌طور جداگانه در هر کدام از مثلث‌های قطع شده توسط q^* محاسبه کنیم، که روش کار در بالا برای حالت $m = n^۴$ توضیح داده شد. بجز این مکان خروج q^* از هر یک از مثلث‌ها نیز مهم است. زمان اولین تغییر در مثلث‌ها را با در نظر گرفتن زمان خروج از مثلث‌ها در یک صف اولویت قرار می‌دهیم. اولین زمان را از صف خارج می‌کنیم و متناسب با آن، برحسب مثلثی که تغییر در آن اتفاق افتاده، بخشی از چندضلعی قابل دید جزئی q را به‌روزرسانی می‌کنیم. سپس زمان تغییر بعدی در این مثلث را محاسبه کرده و در صف قرار می‌دهیم. در صورتی که q^* از مثلث خارج شده است، زمان اولین تغییر در مثلث یا مثلث‌های جدید را در صف قرار می‌دهیم. چون تعداد مثلث‌ها $O(r)$ تا است و یافتن اولین تغییر و قرار دادن آن در صف برای همه مثلث‌ها در مجموع زمان $O(r(\log(n/r) + \log r)) = O(r \log n)$ را صرف خواهد کرد. پردازش سایر

وقایع تغییر در زمان $O(\log n)$ قابل انجام است. با تغییر پارامتر $r = n^۲/\sqrt{m}$ به نتیجه مطلوب می‌رسیم. □

نکته‌ای که می‌توان در مورد زمان نگه‌داری این الگوریتم ذکر کرد این است که اگر ما پارامتر k را در زمان پیش‌پردازش داشته باشیم، یعنی بدانیم باید $\mathrm{V}_k(q)$ را محاسبه کنیم، می‌توانیم آرایش قیود بحرانی را به‌گونه‌ای بهبود بخشیم که فقط وقایعی پردازش شوند که حتما تغییری در شکل $\mathrm{V}_k(q)$ ایجاد می‌کنند. در این صورت هزینه نگه‌داری چندضلعی قابل دید جزئی کاهش چشمگیری می‌یابد.

٦.٦ بیشینه تعداد اشیاء قطع شده

در این بخش، چگونگی حل مسأله‌ای که در ظاهر شباهتی به مسأله محاسبه چندضلعی قابل دید جزئی ندارد، با استفاده از ایده مطرح شده در این فصل را عنوان می‌کنیم. مسأله مربوط به تعیین بیشینه تعداد اشیائی می‌شود که توسط شعاع‌های ساطع شده از نقطه پرس‌وجوی q قطع می‌شوند. طبق تعاریفی که در ابتدای فصل آمد، این مقدار همان $\tau(q)$ است.

قضیه ۴.٦ برای هر نقطه q، می‌توان $\tau(q)$ را در زمان $O(n^۲ \log(۱ + \sqrt{m}/n)/\sqrt{m})$ محاسبه کرد، به شرطی که صحنه را در مدت $O(m \log(۱ + \sqrt{m}/n))$ و با صرف حافظه $O(m)$ پیش‌پردازش کنیم. m پارامتری است دلخواه که در شرط $n^۲ \le m \le n^۴$ صدق می‌کند.

اثبات. باز هم در ابتدا حالت $m = n^۴$ را در نظر بگیرید. ابتدا قیود بحرانی بین همه زوج نقاط را رسم کرده و صفحه را به $O(n^۴)$ ناحیه تقسیم می‌کنیم. واضح است که در هر ناحیه، همه نقاط q، $\tau(q)$ یکسانی دارند. پس کافی است دوری را بین ناحیه‌ها تشکیل داده و به ترتیب τ را در ناحیه‌های دور محاسبه کنیم. از آن‌جایی که اختلاف τ در دو ناحیه مجاور یک واحد است، به راحتی می‌توان در زمان $O(n^۴ \log n)$، مقدار τ را برای همه ناحیه‌ها محاسبه کرد.

اکنون حالت $m = o(n^۴)$ را در نظر بگیرید. در این حالت ما در زمان پیش‌پردازش برای هر مثلثی که در ساختار $(۱/r)$–برش وجود دارد، مشابه روشی که برای محاسبه چندضلعی قابل دید جزئی به‌کار بردیم، روش حالت $m = n^۴$ را اجرا می‌کنیم. تفاوت کار در این‌جا این است که به‌جای ذخیره‌سازی $\mathcal{V}_k(q)$ برای هر ناحیه، عدد $\tau(q)$ را برای آن ناحیه محاسبه کنیم که در نتیجه دیگر نیازی به داده‌ساختار ماندگار برای نگه‌داری چندضلعی‌های قابل دید جزئی نداریم. در زمان پرس‌وجو کافی است با استفاده از داده‌ساختاری که برای هر مثلث t قطع شده توسط دوگان نقطه پرس‌وجو محاسبه کرده‌ایم، $\tau(q)$ را محدود به گوه دوگانه‌ای که دوگان آن اشتراک q و t است محاسبه کنیم. زمان پیش‌پردازش و پرس‌وجو و حافظه مصرف شده معادل زمان پیش‌پردازش و پرس‌وجو و حافظه مصرف شده در الگوریتم محاسبه

چندضلعی قابل دید جزئی است. □

٧.٦ خلاصه

در این فصل، ابتدا مفهوم قابلیت دید جزئی تعریف شد و نشان دادیم که این مفهوم می‌تواند چه کاربردهایی داشته باشد. در ادامه نحوه محاسبه چندضلعی قابل دید جزئی یک نقطه پرس‌وجو را محاسبه کردیم. دو الگوریتم برای محاسبه این چندضلعی، یکی با زمان لگاریتمی و دیگری با توازن بین حافظه و زمان پرس‌وجو ارائه شدند. سپس نشان دادیم که چگونه می‌توان همین روش را برای نقطه در حال حرکت استفاده کرد و چندضلعی قابل دید جزئی نقطه را به‌روز رسانی کرد. در پایان دیدیم که با توسعه الگوریتم محاسبه چندضلعی قابل دید جزئی، می‌توان برای نقطه q و مجموعه S، $\tau(q, S)$ را محاسبه کرد.

فصل ۷

آلفا–رؤیت‌پذیری

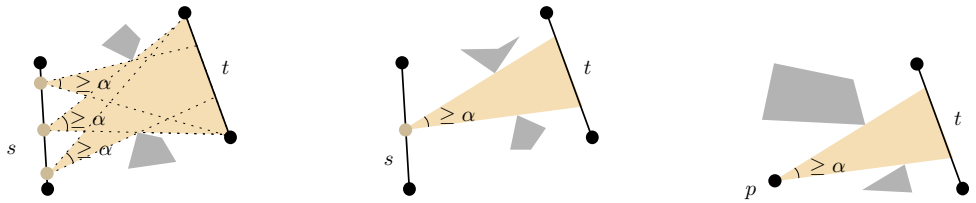
در این فصل مفهوم جدیدی را به نام «آلفا–رؤیت‌پذیری» در حوزه قابلیت دید تعریف می‌کنیم. این مفهوم به ما امکان تقریب زدن قابلیت دید را می‌دهد، به این معنی که اشیاء دور یا اشیاء ریز قابل دید نیستند. در ادامه با بررسی بیشتر این تعریف به نتایج بسیار جالبی می‌رسیم. مثلا آلفا–رؤیت‌پذیری یک پاره‌خط از یک نقطه را می‌توانیم در زمان لگاریتمی تشخیص دهیم. زمان پیش‌پردازش صحنه برای این کار زمان $O(n \log n)$ است که نسبت به قابلیت دید عادی بسیار کمتر است. به‌عنوان مثال اگر از روشی که در فصل ۵ ارائه شد استفاده کنیم، برای رسیدن به زمانی نزدیک به لگاریتمی، به زمان پیش‌پردازشی به اندازه $O(n^4 \log^4 n)$ نیازمندیم.

در ادامه فصل، تعریف ضعیف و کامل بودن دید را به آلفا–رؤیت‌پذیری گسترش می‌دهیم و نشان می‌دهیم که چگونه می‌توان مسائل پرس‌وجوی قابلیت دید را با کارایی خیلی بهتری در این مدل حل نمود.

۱.۷ مقدمه

مجموعه $\mathcal{S}$ از n پاره‌خط در صفحه را در نظر بگیرید. این پاره‌خط‌ها یکدیگر را قطع نمی‌کنند، مگر احتمالا در نقاط انتهایی. چون یک چندضلعی حفره‌دار نیز متشکل از چنین مجموعه پاره‌خطی است، $\mathcal{S}$ می‌تواند چندضلعی‌های حفره‌دار را نیز مدل کند. فرض کنید α عدد مثبتی باشد. مفهوم «آلفا–رؤیت‌پذیری» که ما در ادامه مسائلی را در ارتباط با آن مطرح می‌کنیم، به‌صورت زیر تعریف می‌شود.

- دید نقطه: پاره‌خط t از نقطه p قابل دید α است، اگر p با زاویه‌ای حداقل مساوی α، t را ببیند.



الف ب پ

شکل ۱.۷ : الف. پاره‌خط t از نقطه p آلفا–رؤیت‌پذیر است. ب. پاره‌خط t از پاره‌خط s آلفا–رؤیت‌پذیر ضعیف است. پ. پاره‌خط t از پاره‌خط s آلفا–رؤیت‌پذیر کامل است.

به بیان دیگر اگر مثلثی وجود داشته باشد که یک رأس آن روی p و ضلع روبرو به آن روی t و زاویه رأس p حداقل α باشد (شکل ۱.۷– الف).

- دید پاره‌خط: پاره‌خط t از پاره‌خط s قابل دید ضعیف α است، اگر نقطه‌ای بر s باشد که t از آن قابل دید α باشد (شکل ۱.۷– ب). پاره‌خط t از پاره‌خط s قابل دید کامل α است، اگر t از همه نقاط s قابل دید α باشد (شکل ۱.۷– پ).

- گراف قابلیت دید: برای مجموعه $\mathcal{S}$، گراف قابلیت دید ضعیف (یا قوی) α را بدین صورت تعریف می‌کنیم. گراف جهت دار G_α که رئوس آن پاره‌خط‌های $\mathcal{S}$ هستند و برای هر دو پاره‌خط s و t در $\mathcal{S}$، یالی از s به t وجود دارد اگر t از s قابل دید ضعیف (یا قوی) α باشد.

مفهوم آلفا–رؤیت‌پذیری، مفهومی است که در دنیای بیرون کاملا طبیعی به‌نظر می‌رسد. به عنوان مثال، کمترین زاویه‌ای که یک فرد قادر به دیدن یک شیء بدون استفاده از ابزار است، از نسبت طول موج نور به قطر مردمک چشم تعیین می‌شود، که این مقدار از یک مقدار ثابت (در حدود $۱۰^{-۴}$) کمتر است. نتیجه این‌که اشیائی که خیلی کوچکند یا خیلی دور هستند توسط انسان قابل دیدن نیستند. به عنوان مثال موی انسان در فاصله‌ای به اندازه دو تا سه برابر اندازه یک دست، تا چشم، قابل دیدن نیست. همه ابزارهای نوری و دیجیتالی تصویر نیز چنین محدودیت‌هایی دارند، که با تفکیک‌پذیری آن‌ها مشخص می‌شود. مدل آلفا–رؤیت‌پذیری قادر است چنین محدودیت‌هایی را در نظر گرفته و جایگزین کاربردی‌تری برای قابلیت دید عادی باشد. مقدار α در این مدل را می‌توان برای تقریب زدن عدم دقت ابزارهایی که براساس قابلیت دید اندازه‌گیری‌هایی انجام می‌دهند نیز به‌کار برد.

در حوزه مسائل یافتن کوتاه‌ترین مسیر، تحقیقات زیادی برای تقریب زدن کوتاه‌ترین مسیر انجام شده است. همچنین رابطه بسیار نزدیکی بین قابلیت دید و مسائل کوتاه‌ترین مسیر وجود دارد. ولی تا قبل از این کار، تحقیق قابل ذکری در مورد تقریب زدن قابلیت دید نشده بود. هدف کار ما در این فصل، سوق‌دهی تحقیقات به سمت تقریب زدن قابلیت دید بود و می‌تواند به انجام کارهای دیگری در این زمینه کمک کند.

۲.۷ نتایج کسب شده

بعد از تعریف مدل آلفا–رؤیت‌پذیری، تعدادی از مسائل قابلیت دید در این مدل بررسی و حل شدند. ایده اصلی ما در این کار، کاهش همه جهت‌های ممکن قابلیت دید به $O(۱/\alpha)$ جهت است. سپس با استفاده از ابزارهای هندسی مانند ذوزنقه‌بندی، نقشه کوتاه‌ترین مسیر، شلیک پرتو و جستجوی بازه‌ها، مسائل قابلیت دید را در این جهت‌های کاسته شده حل نمودیم. خلاصه نتایجی که به‌دست آوردیم در زیر می‌آید. مجموعه $\mathcal{S}$ در زیر، مجموعه پاره‌خط‌هایی در صفحه است، که به‌صورت ورودی داده می‌شود.

- داده‌ساختاری را ارائه کردیم که پاسخ پرس‌وجوهایی به شکل «آیا پاره‌خط $t \in \mathcal{S}$ از نقطه p آلفا–رؤیت‌پذیر است؟» را می‌دهد. اگر α ثابت باشد، مجموعه $\mathcal{S}$ در زمان $O(n \log n)$ در داده‌ساختاری به اندازه $O(n)$ پیش‌پردازش شده و پرس‌وجوهای ذکر شده را در مدت $O(\log n)$ پاسخ می‌دهد. برای حالتی که α ثابت نبوده و در زمان پرس‌وجو مشخص می‌شود نیز روش متفاوتی را ارائه نمودیم.

- نشان دادیم که اندازه گراف آلفا–رؤیت‌پذیری (ضعیف و قوی) مجموعه $\mathcal{S}$ خطی است و می‌توان آن را در زمان $O(n \log n)$ محاسبه کرد. در نتیجه می‌توان با پیش‌پردازشی در مدت $O(n \log n)$، پرس‌وجوهایی به شکل «آیا پاره‌خط $t \in \mathcal{S}$ از پاره‌خط $s \in \mathcal{S}$ آلفا–رؤیت‌پذیر (ضعیف یا قوی) است؟» را در زمان $O(۱)$ پاسخ گفت.

- نشان دادیم که چگونه می‌توان $\mathcal{S}$ را در مدت $O(n \log n)$ در داده‌ساختاری به اندازه $O(n)$ پیش‌پردازش کرد، تا بتوان پرس‌وجوهایی به شکل «آیا پاره‌خط $t \in \mathcal{S}$ از پاره‌خط دلخواه s در صفحه آلفا–رؤیت‌پذیر (ضعیف یا قوی) است؟» را در زمان $O(\log n)$ پاسخ گفت.

نکته قابل توجه این‌که تفاوت کلیدی بین قابلیت دید عادی (یعنی زمانی که $\alpha = ۰$) و آلفا–رؤیت‌پذیری در اندازه گراف قابلیت دید (ضعیف یا قوی) پاره‌خط‌ها در صفحه است. در قابلیت دید عادی، اندازه گراف قابلیت دید $O(n^۲)$ است، در حالی‌که در آلفا–رؤیت‌پذیری اندازه آن خطی است. این تفاوت، آلفا–رؤیت‌پذیری را هم از لحاظ نظری و هم از لحاظ کاربردی، وقتی با مجموعه‌های ورودی بزرگ کار می‌کنیم، مطلوب می‌سازد.

نکته دیگری که باید در مورد پیچیدگی‌های عنوان شده در بالا ذکر کرد، این است که بجز در مواردی که صراحتاً ذکر شده مقدار α ثابت نیست، در سایر موارد فرض می‌شود مقدار α ثابت است و یا محدود به مقدار ثابتی است و در حدود ذکر شده، α حذف می‌شود. اگر بخواهیم حدود زمانی و فضایی را دقیق‌تر ذکر کنیم، یک ضریب $۱/\alpha$ نیز باید به این حدود اضافه کنیم.

۳.۷ داده‌ساختارها والگوریتم‌های مورد نیاز

در این قسمت به‌طور خلاصه ابزارهای هندسی مورد نیاز در سایر بخش‌های این فصل را توضیح می‌دهیم.

۱.۳.۷ شلیک پرتو در میان پاره‌خط‌ها

یکی از شکل‌های مسأله شلیک پرتو، شلیک در میان مجموعه‌ای از پاره‌خط‌ها است. مجموعه n پاره‌خط در صفحه داده شده‌اند. داده‌ساختاری را بسازید که برای هر پرتو r، اولین پاره‌خطی که توسط r قطع می‌شود، برگرداند. Chan [۱۲] با استفاده از داده‌ساختار چند لایه جستجوی بازه‌ها نتیجه زیر را در مورد این مسأله به‌دست آورده است.

قضیه ۱.۷ مجموعه‌ای از n پاره‌خط در صفحه داده شده است. داده‌ساختاری با اندازه $O(n\log^{۲} n)$ در زمان $O(n\log^{۳} n)$ قابل ساخت است که می‌تواند اولین برخورد بین یک پرتو پرس‌وجوی دلخواه را با پاره‌خط‌ها در زمان متوسط $O(\sqrt{n}\log^{۲} n)$ بیابد.

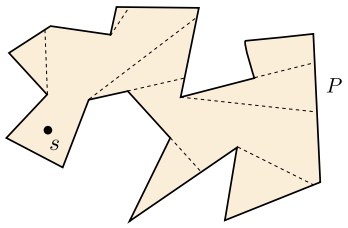
۲.۳.۷ نمودار ذوزنقه‌بندی ساده شده

مجموعه $\mathcal{S}$ از n پاره‌خط و جهت d داده شده است. یک تقسیم‌بندی صفحه به تعدادی ناحیه را در نظر بگیرید که هر ناحیه در تقسیم‌بندی بزرگترین ناحیه ممکن باشد که همه نقاط آن در جهت d پاره‌خط یکسانی را می‌بینند. این تقسیم‌بندی را می‌توان بدین صورت ساخت: از هر نقطه انتهایی هر پاره‌خط $\mathcal{S}$، پرتوی در جهت خلاف d رسم شود تا پاره‌خطی دیگر را قطع کند. از آن‌جایی که این تقسیم‌بندی بسیار شبیه به ذوزنقه‌بندی یک مجموعه پاره‌خط است، آن را نمودار ذوزنقه‌بندی ساده شده می‌نامیم و با $\mathcal{T}_d$ نمایش می‌دهیم. قضیه زیر را می‌توان برای ساخت ذوزنقه‌بندی ساده شده استفاده کرد که براساس الگوریتم ذوزنقه‌بندی اثبات می‌شود.

قضیه ۲.۷ مجموعه $\mathcal{S}$ از n پاره‌خط در صفحه و جهت d داده شده‌اند. می‌توان نمودار ذوزنقه‌بندی ساده شده $\mathcal{T}_d$ را با استفاده از روش جاروب صفحه در جهت d در زمان $O(n\log n)$ و با حافظه $O(n)$ ساخت.

۳.۳.۷ نقشه کوتاه‌ترین مسیر

برای نقطه s در چندضلعی ساده P، نقشه کوتاه‌ترین مسیر که با $\mathrm{SPM}(s)$ نشان می‌دهیم، از تقسیم‌بندی P به ناحیه‌هایی حاصل می‌شود که در هر ناحیه، رئوس میانی کوتاه‌ترین مسیر از هر نقطه تا s ثابت



شکل ۲.۷ : چندضلعی P و نقطه s. نقشه کوتاه‌ترین مسیر s داخل P با نقطه‌چین مشخص شده است.

هستند (شکل ۲.۷). ثابت شده است که اندازه $\mathrm{SPM}(s)$ خطی است و در زمان خطی قابل ساخت است [۳۴]. اگر $\mathrm{SPM}(s)$ را برای جایابی نقطه پیش‌پردازش کنیم، می‌توانیم برای هر نقطه پرس‌وجوی p آخرین رأس از P در کوتاه‌ترین مسیر از s به p را در زمان $O(\log n)$ بیابیم.

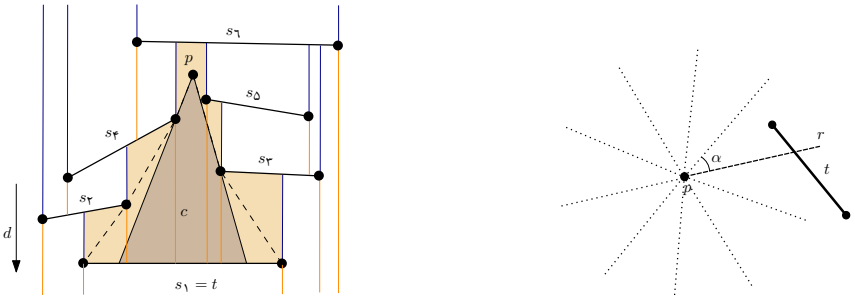
۴.۳.۷ شلیک پرتو در چندضلعی‌گون

چندضلعی‌گون (یا چندضلعی با اضلاع منحنی) با توسعه تعریف چندضلعی ساخته می‌شوند [۲۴]. چندضلعی‌گون S از چندضلعی P با تعویض یک یا چند ضلع P و قرار دادن منحنی‌هایی به‌جای آن اضلاع حاصل می‌شود. هریک از این منحنی‌ها باید دارای این شرط باشند که ناحیه محدود به منحنی و پاره‌خط واصل دو انتهای آن محدب باشد. قضیه زیر در مورد شلیک پرتو در چندضلعی‌گون است که توسط Melissaratos و Souvaine [۵۰] اثبات شده است.

قضیه ۳.۷ چندضلعی‌گون S با n ضلع داده شده است. داده‌ساختاری با اندازه $O(n)$ در زمان $O(n)$ قابل ساخت است که برای هر پرتوی پرس‌وجوی r، اولین برخورد r را با S در زمان $O(\log n)$ می‌یابد.

۴.۷ قابلیت دید نقطه

در آغاز بررسی مسائل آلفا–رؤیت‌پذیری، مسائل مربوط به قابلیت دید نقطه را بررسی می‌کنیم که ساده‌تر بوده و در مسائل پیچیده‌تر قابل استفاده‌اند. ابتدا مسأله آزمون قابل دید بودن یک پاره‌خط از نقطه پرس‌وجوی p وقتی α ثابت باشد را بررسی می‌کنیم و در حالت بعدی، همین مسأله را وقتی که α ثابت نبوده و در زمان پرس‌وجو مقدار آن مشخص شود حل می‌کنیم.



الف ب

شکل ۳.۷ : الف. پاره‌خط t از نقطه p آلفا–رؤیت‌پذیر است و پرتو r، t را قطع می‌کند. ب. ذوزنقه‌بندی یک مجموعه پاره‌خط در جهت d.

۱.۴.۷ آزمون قابلیت دید با α ثابت

فرض کنید α عدد ثابت مثبتی باشد. در این بخش نشان می‌دهیم که چگونه می‌توان مشخص کرد که آیا پاره‌خط پرس‌وجوی $t \in \mathcal{S}$ از نقطه پرس‌وجوی p در صفحه آلفا–رؤیت‌پذیر است.

قضیه ۴.۷ می‌توان مجموعه $\mathcal{S}$ را در داده‌ساختاری با اندازه $O(n)$ در زمان $O(n \log n)$ پیش‌پردازش کرد، به‌گونه‌ای که آزمون‌های آلفا–رؤیت‌پذیری در زمان پرس‌وجوی $O(\log n)$ انجام پذیر باشند.

اثبات. ابتدا مجموعه $\lceil \frac{2\pi}{\alpha} \rceil$ پرتو که از p ساطع شده و زاویه بین هر دو پرتو متوالی α است (بجز احتمالاً یک زوج که کمتر از α است)، در نظر بگیرید (شکل ۳.۷– الف). D را مجموعه جهت‌های این پرتوها قرار دهید. شرط لازم برای آلفا–رؤیت‌پذیری t از p این است که حداقل یکی از پرتوهای رسم شده از p، t را قطع کند. فرض کنید r چنین پرتوی باشد و d جهت پرتو r باشد. اکنون نمودار ذوزنقه‌بندی $\mathcal{S}$ در جهت d را همانند شکل ۳.۷– ب در نظر بگیرید. مشاهده می‌شود که p در ذوزنقه‌ای قرار دارد که t را در جهت d می‌بیند. بنابراین برای آلفا–رؤیت‌پذیری t از p، در یکی از ذوزنقه‌بندی‌های $\mathcal{S}$ در جهت‌های D، باید p در ذوزنقه‌ای قرار بگیرد که t از آن دیده می‌شود.

بعد از برقرار بودن این شرط، باید بررسی شود که آیا زاویه دید p نسبت به t حداقل α است یا خیر. نمودار ذوزنقه‌بندی ساده شده $\mathcal{S}$ در جهت d، $\mathcal{T}_d(\mathcal{S})$، را در نظر بگیرید. نقطه p در این نمودار، در ناحیه‌ای مانند c قرار می‌گیرد که برای همه نقاط c در جهت d، پاره‌خط t دیده می‌شود (شکل ۳.۷(ب)). توجه کنید که ناحیه c یک چندضلعی ساده است. اگر کوتاه‌ترین فاصله p تا دو سر t را در درون c رسم کنیم، دو زنجیره محدب تشکیل می‌دهند. زاویه‌ای که از یال اول دو زنجیره تشکیل می‌شود، برای ما بزرگترین بخش قابل دید t از p درون c را مشخص می‌کند (شکل ۳.۷(ب) را ببینید). بنابراین برای یافتن بزرگترین بخش قابل دید t از p که شامل جهت d باشد، کافی است اولین رأس در کوتاه‌ترین مسیر از p به سمت هر یک از دو سر t در داخل c را بیابیم.

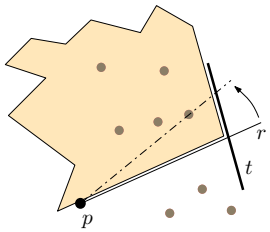
اندازه نمودار ذوزنقه‌بندی در جهت d، خطی است و در زمان $O(n \log n)$ قابل محاسبه است. با استفاده از داده‌ساختار ذوزنقه‌بندی، می‌توان ذوزنقه‌ای که p در داخل آن قرار می‌گیرد را یافته و پاره‌خطی که در جهت d از p دیده می‌شود را شناسایی کرد. نقشه کوتاه‌ترین مسیر به دو سر هر پاره‌خط در داخل ناحیه متناظرش، c، اندازه‌ای متناسب با اندازه c دارد. چون مجموع اندازه ناحیه‌ها در نمودار ذوزنقه‌بندی ساده شده $O(n)$ است، همه نقشه‌های کوتاه‌ترین مسیر به حافظه $O(n)$ نیاز دارند و در زمان کلی $O(n \log n)$ محاسبه می‌شوند. این فرایند برای همه جهت‌های مجموعه D انجام می‌شود.

برای پاسخ به یک پرس‌وجوی خاص، با دریافت نقطه p، برای همه جهت‌های d در مجموعه D، دو نمودار ذوزنقه‌بندی $\mathcal{S}$ نقطه p را جایابی می‌کنیم. ذوزنقه‌ای که در زمان $O(\log n)$ یافته می‌شود، پاره‌خطی را به ما می‌دهد که p در جهت d می‌بیند. اگر این پاره‌خط، پاره‌خط مورد نظر ما، t، نباشد، به سراغ جهت بعدی می‌رویم. ولی اگر این پاره‌خط، t بود، براساس نقشه کوتاه‌ترین مسیر مربوط به ناحیه‌ای از نمودار ذوزنقه‌بندی ساده شده که p در آن قرار می‌گیرد، بزرگترین بخشی از t که از p قابل دید است، یافته و بررسی می‌کنیم که آیا این زاویه این بخش حداقل α است یا خیر. اگر زاویه حداقل α بود، پاسخ پرس‌وجو مثبت است و t از p آلفا–رؤیت‌پذیر است. اگر زاویه کمتر از α بود، به سراغ جهت بعدی در مجموعه D می‌رویم و همین کار را برای آن تکرار می‌کنیم. اولین رأس در کوتاه‌ترین مسیرهای p به دو سر t در زمان $O(\log n)$ یافته می‌شوند. چون تعداد همه جهت‌ها مقداری ثابت، یعنی $O(1/\alpha)$ است، زمان کلی پرس‌وجو $O(\log n)$ است. □

از اثبات قضیه بالا می‌توان به نتیجه زیر در مورد گزارش همه پاره‌خط‌هایی که از نقطه پرس‌وجوی p آلفا–رؤیت‌پذیر هستند، رسید.

نتیجه ۱.۷ مجموعه $\mathcal{S}$ از n پاره‌خط را می‌توان در داده‌ساختاری با اندازه $O(n)$ در زمان $O(n \log n)$ پیش‌پردازش کرد و همه پاره‌خط‌های آلفا–رؤیت‌پذیر از نقطه پرس‌وجوی p را در زمان $O(\log n)$ گزارش کرد. به‌علاوه تعداد پاره‌خط‌های آلفا–رؤیت‌پذیر از هر نقطه p، $O(1/\alpha)$ است.

اثبات. اگر پاره‌خطی از p آلفا–رؤیت‌پذیر باشد، باید توسط حداقل یکی از پرتوهای رسم شده از p در جهت D قطع شود. برای هر جهت d در مجموعه D، ذوزنقه‌ای را که p در ذوزنقه‌بندی $\mathcal{S}$ در آن قرار می‌گیرد یافته و پاره‌خط t را که به ذوزنقه مربوط می‌شود در نظر می‌گیریم. با استفاده از نقشه‌های کوتاه‌ترین مسیر، اگر زاویه دید p نسبت به t حداقل α بود، t در گزارش نهایی می‌آید. چون تعداد همه جهت‌ها $O(1/\alpha)$ است، تعداد پاره‌خط‌های گزارش شده $O(1/\alpha)$ خواهد بود که مقداری است ثابت. تعیین قابل دید بودن هر پاره‌خط در جهت d زمان $O(\log n)$ نیاز دارد که در نتیجه زمان کلی نیز $O(\log n)$ خواهد بود. □



شکل ۴.۷ : یافتن اولین محل تغییر دید p در جهت r، وقتی r پادساعت‌گرد می‌گردد.

۲.۴.۷ آزمون قابلیت دید با α غیرثابت

در این بخش، مسأله بخش قبلی را با این فرض که α غیرثابت است و مقدار آن در زمان پرس‌وجو مشخص می‌شود بررسی می‌کنیم. بنابراین ما باید مجموعه $\mathcal{S}$ را به‌گونه‌ای پیش‌پردازش کنیم که با دریافت نقطه p، پاره‌خط $t \in \mathcal{S}$ و مقدار زاویه $\alpha > 0$ در زمان پرس‌وجو، مشخص کنیم که آیا t از p آلفا–رؤیت‌پذیر است یا خیر.

قضیه ۵.۷ مجموعه $\mathcal{S}$ از n پاره‌خط داده شده است. می‌توان $\mathcal{S}$ را در زمان $O(n \log^3 n)$ در داده‌ساختاری با اندازه $O(n \log^2 n)$ پیش‌پردازش کرد، به‌گونه‌ای که بتوانیم با دریافت نقطه p، پاره‌خط $t \in \mathcal{S}$ و زاویه α، در زمان متوسط $O(\sqrt{n} \log^3 n)$ مشخص کنیم آیا t از s قابل دید α است یا خیر.

اثبات. مجموعه $\lceil \frac{2\pi}{\alpha} \rceil$ پرتو که از p ساطع می‌شود و زاویه بین هر دو پرتو حداکثر α است را در نظر بگیرید (شکل ۳.۷(ب)). اگر t از s قابل دید α باشد، یکی از پرتوها باید t را قطع کند. فرض کنید r پرتوی است که t را قطع می‌کند. برای بررسی این‌که بخش قابل دید p حول محل برخورد r و t زاویه‌ای حداقل مساوی α تشکیل می‌دهد یا خیر، باید بزرگترین بخش قابل دید t از p را در آن قسمت را پیدا کنیم. اگر این بخش زاویه‌ای بزرگتر یا مساوی α داشت، پاسخ پرس‌وجو مثبت است.

بنابراین برای حل مسأله، باید دو زیرمسأله دیگر را حل کنیم. (۱) کدام پاره‌خط در جهت r از p قابل دیدن است؟ (۲) اگر پرتو r را حول p دوران دهیم، اولین تغییر در دید p در جهت r چه زمانی رخ می‌دهد؟

زیرمسأله (۱) مسأله شلیک پرتو در میان پاره‌خط‌هاست، که در قضیه ۱.۷ راه‌حل آن ارائه شد. برای حل زیرمسأله (۲)، از جستجوی بازه‌ها بدین صورت استفاده می‌کنیم. ابتدا نقاط انتهایی پاره‌خط‌های $\mathcal{S}$ را برای دو لایه جستجوی نیم‌صفحه پیش‌پردازش می‌کنیم. در ادامه، برای هر زیرمجموعه کانونی نتیجه، پوسته محدب نقاط آن محاسبه می‌شود.

بدون از دست رفتن کلیت مسأله، فرض کنید می‌خواهیم اولین تغییر را زمانی که r پادساعت‌گرد حول p می‌گردد بیابیم. با دریافت نقطه p در زمان پرس‌وجو و با داشتن پرتو r در لایه اول داده‌ساختار، نقاطی را می‌یابیم که در نیم‌صفحه‌ای از خط حامل t قرار می‌گیرند که p قرار دارد. سپس در لایه

دوم، نیم‌صفحه‌ای از خط حامل r که وقتی r پادساعت‌گرد به اندازه 180° می‌گردد پویش می‌شود (شکل ۴.۷ را ببینید).

نتیجه این دو لایه جستجو، به‌طور متوسط $O(\sqrt{n}\log^2 n)$ مجموعه کانونی است، و برای هر مجموعه ما پوسته محدب آن را محاسبه کرده‌ایم. بنابراین برای هر مجموعه، اولین محل برخورد r با آن مجموعه، وقتی r پادساعت‌گرد می‌گردد، در زمان $O(\log n)$ قابل محاسبه است. نتیجه نهایی، اولین نقطه در میان همه این اولین نقطه‌های برخورد است. بنابراین اولین محل تغییر دید در زمان متوسط $O(\sqrt{n}\log^3 n)$ یافته می‌شود. زمان کلی پرس‌وجو مجموع زمان لازم برای هر کدام از زیرمسأله‌هاست. □

نکته: با استفاده از تکنیکی که Matoušek [۴۹] ارائه کرده، می‌توان برای این مسأله به توازن بین حافظه و زمان پرس‌وجو رسید. به‌طور دقیق‌تر، با صرف حافظه $O(m)$، برای مقادیر m با شرط $n\log^2 n \le m \le n^2$، می‌توان به زمان پرس‌وجوی $O((n/\sqrt{m})\operatorname{polylog} m)$ رسید.

۵.۷ قابلیت دید ضعیف پاره‌خط

در این فصل توجه خود را از ناظر نقطه‌ای به پاره‌خط منتقل می‌کنیم و آلفا–رؤیت‌پذیری ضعیف پاره‌خط‌ها را بررسی می‌کنیم. در ابتدا گراف قابلیت دید مجموعه‌ای از پاره‌خط‌ها را تعریف می‌کنیم و نشان می‌دهیم اندازه این گراف خطی است. سپس مسائلی را که مربوط به آزمون قابلیت دید پاره‌خط است مطرح و حل می‌کنیم.

۱.۵.۷ گراف قابلیت دید

گراف آلفا–رؤیت‌پذیری ضعیف مجموعه $\mathcal{S}$ که با G_α نمایش داده می‌شود بدین صورت تعریف می‌شود. به‌ازای هر پاره‌خط در $\mathcal{S}$ رأسی در G_α وجود دارد. همچنین برای هر دو پاره‌خط s و t در $\mathcal{S}$، اگر t از s آلفا–رؤیت‌پذیر ضعیف باشد، در G_α یال جهت‌داری از رأس متناظر با s به رأس متناظر با t اضافه می‌شود. لم زیر نشان می‌دهد که اندازه گراف آلفا–رؤیت‌پذیری ضعیف خطی است.

لم ۱.۷ گراف آلفا–رؤیت‌پذیری ضعیف مجموعه $\mathcal{S}$، G_α، اندازه خطی دارد.

اثبات. مجموعه D از $O(1/\alpha)$ جهت را در نظر بگیرید که زاویه بین هر دو جهت متوالی حداکثر α است. فرض کنید پاره خط $t \in \mathcal{S}$ از $s \in \mathcal{S}$ آلفا–رؤیت‌پذیر ضعیف باشد. بنابراین نقطه‌ای مانند p روی s وجود دارد که t را با زاویه‌ای حداقل مساوی α می‌بیند. بدیهی است یکی از جهت‌های مجموعه D

مثل d در داخل زاویه دید p قرار دارد. فرض کنید r پاره‌خطی باشد که p را به t در جهت d وصل می‌کند. پاره‌خط r، پاره‌خط دیگری از $\mathcal{S}$ را قطع نمی‌کند. اکنون r را در جهت عمود بر d، جابه‌جا کنید (بدون تغییر جهت r)، تا زمانی که r به یکی از نقاط انتهایی s یا t و یا سایر پاره‌خط‌های $\mathcal{S}$ برسد. دو حالت مختلف را بررسی می‌کنیم.

حالت i، r به یکی از نقاط انتهایی s یا t برسد. در این حالت، نتیجه می‌گیریم که یک نقطه انتهایی s، t را در جهت d می‌بیند (حالتی که r به نقطه انتهایی s برسد)، و یا یک نقطه انتهایی t، s را در خلاف جهت d می‌بیند (حالتی که r به نقطه انتهایی t برسد).

حالت ii، r به یکی از نقاط انتهایی پاره‌خطی بجز s و t برسد. در این حالت، نتیجه می‌گیریم که یک نقطه انتهایی یک پاره‌خط در $\mathcal{S}$ در جهت d، t را می‌بیند.

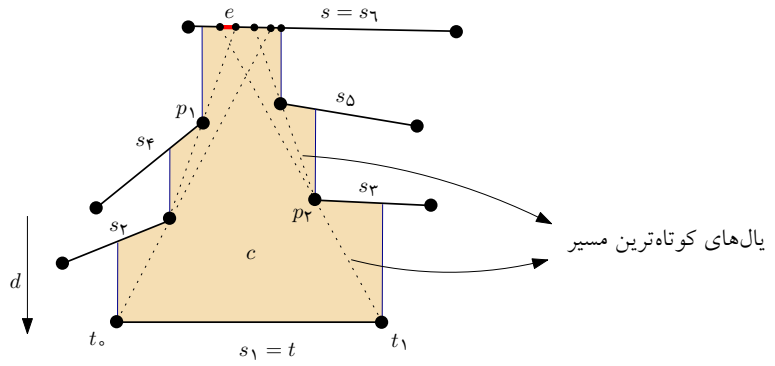
نتیجه این‌که هر حالت آلفا-رؤیت‌پذیری پاره‌خط-پاره‌خط به حالتی از قابلیت دید نقطه-پاره‌خط در جهت‌های D تبدیل می‌شود. چون هنگام حرکت کردن r، تا قبل از رسیدن به نقطه‌ای انتهایی، پاره‌خط‌های s و t تغییر نمی‌کند، و با دانستن زوج نقطه-پاره‌خط می‌توانیم زوج پاره‌خط-پاره‌خط را بیابیم، نتیجه می‌گیریم که نگاشت آلفا-رؤیت‌پذیری پاره‌خط-پاره‌خط به قابلیت دید نقطه-پاره‌خط یکتاست. چون تعداد جهت‌های مجموعه D ثابت بوده و تعداد نقاط انتهایی $\mathcal{S}$، $O(n)$ است، تعداد کل زوج‌های نقطه-پاره‌خط که نقطه، پاره‌خط را در یکی از جهات D می‌بینند، $O(n)$ و تعداد کل زوج‌های (s,t) که t از s آلفا-رؤیت‌پذیر است، $O(n)$ خواهد بود. این تعداد مساوی تعداد یال‌های G_α است. □

از نتیجه‌ای که در بالا در مورد گراف آلفا-رؤیت‌پذیری گرفتیم، استفاده می‌کنیم و داده‌ساختاری را تهیه می‌کنیم که با استفاده از آن قابل دید بودن یک پاره‌خط را از پاره‌خط دیگر بتوانیم در زمان مناسب گزارش دهیم. روش تهیه این داده‌ساختار را در بخش بعدی توضیح می‌دهیم.

۲.۵.۷ آزمون آلفا-رؤیت‌پذیری ضعیف

قضیه ۶.۷ مجموعه $\mathcal{S}$ در صفحه داده شده است. می‌توان $\mathcal{S}$ را در داده‌ساختاری با اندازه $O(n)$ در زمان $O(n\log n)$ پیش‌پردازش کرد، به‌گونه‌ای که آلفا-رؤیت‌پذیری ضعیف دو پاره‌خط s و t از $\mathcal{S}$ در زمان $O(1)$ مشخص شود.

اثبات. کار را با ساخت گراف G_α شروع می‌کنیم. مجموعه D از $O(1/\alpha)$ جهت را همانند لم ۱.۷ در نظر بگیرید. فرض کنید پاره‌خط $t \in \mathcal{S}$ از $s \in \mathcal{S}$ آلفا-رؤیت‌پذیر ضعیف باشد و p روی s، t را با زاویه‌ای حداقل مساوی α ببیند. همچنین فرض کنید d جهتی در D باشد که در زاویه دید p قرار می‌گیرد و r پاره‌خطی باشد که p را به t در جهت d وصل می‌کند. حالا r را بدون تغییر جهت دادن آن در جهت عمود بر آن، حرکت دهید تا زمانی که به نقطه‌ای انتهایی از یک پاره‌خط برسد. ناحیه‌ای که



شکل ۵.۷ : پاره‌خط s به چند قطعه پاره‌خط کوچک‌تر تقسیم شده است.

r در آن آزادانه حرکت می‌کند، بدون این‌که جهتش تغییر کند و یا پاره‌خط دیگری را قطع کند، نواری مانند w را مشخص می‌کند که هر نقطه روی s در یک طرف w نقطه‌ای روی t را در طرف دیگر w در جهت d می‌بینند. توجه کنید که w ذوزنقه‌ای در ذوزنقه‌بندی $\mathcal{S}$ در جهت d است. اگر s^* و t^* رئوس متناظر با s و t در گراف G_α باشند، اولین شرط برای وجود یالی از s^* به t^* این است که s و t دو ضلع روبه‌روی هم در یکی از ذوزنقه‌بندی $\mathcal{S}$ در یکی از جهت‌های D باشند.

این شرط، شرط لازم است، ولی کافی نیست. برای تعیین این‌که t واقعاً از s آلفا-رؤیت‌پذیر ضعیف است، باید نقطه‌ای را روی s بیابیم که با زاویه‌ای حداقل مساوی α، t را ببیند. اگر ما نقطه با بزرگترین زاویه دید را یافته و زاویه دید آن را با α مقایسه کنیم، می‌توانیم تعیین کنیم که آیا t از s آلفا-رؤیت‌پذیر ضعیف است یا خیر. برای این‌کار، ابتدا s را به بخش‌هایی تقسیم می‌کنیم که در هر بخش، مرز زاویه دید نقاط آن بخش، از دو نقطه یکتا عبور می‌کند. سپس برای هر کدام از این بخش‌ها، نقطه با بزرگترین زاویه دید را می‌یابیم. نقطه با بزرگترین زاویه دید روی s، در میان این نقاط قرار دارد.

ناحیه c را ناحیه‌ای بگیرید که در نمودار ذوزنقه‌بندی ساده شده $\mathcal{S}$ در جهت d، $\mathcal{T}_d$، به پاره‌خط t مربوط است. همچنین فرض کنید $\mathrm{SPM}(t_0)$ و $\mathrm{SPM}(t_1)$ نقشه کوتاه‌ترین مسیر دو سر t، یعنی t_0 و t_1 را در c نشان دهد. برای هر نقطه p روی s، اولین رأس در کوتاه‌ترین مسیر از p به دو سر t، زاویه دید نسبت به t که شامل d می‌شود را تشکیل می‌دهد. اگر ما s را به بخش‌هایی تقسیم کنیم که در هر بخش کوتاه‌ترین مسیر به دو سر t دارای رئوس یکسانی باشند، یا به عبارتی دیگر، شکل ترکیبیاتی کوتاه‌ترین مسیرها یکی باشد، آنگاه برای نقاط هر بخش، دو رأس تشکیل دهنده زاویه دید ثابت خواهند بود. با تعیین نقاط برخورد s با $\mathrm{SPM}(t_0)$ و $\mathrm{SPM}(t_1)$، s را به بخش‌های مذکور تقسیم می‌کنیم. فرض کنید e یکی از پاره‌خط‌های حاصل از تقسیم s باشد (شکل ۵.۷ را ببینید). هدف ما یافتن نقطه‌ای بر روی e است که بزرگترین زاویه دید را نسبت به t داشته باشد. می‌دانیم که همه نقاط روی e شکل ترکیبیاتی کوتاه‌ترین مسیر یکسانی تا دو سر t دارند. p_1 و p_2 را دو رأس اول کوتاه‌ترین مسیر این نقاط تا دو سر

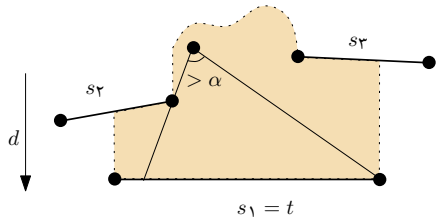
t بگیرید. زاویه دید هر نقطه p روی e تا t که شامل d هم باشد، توسط دو پرتوی که از p به سمت p_1 و p_2 رسم می‌شود، تعیین می‌گردد. اکنون مسأله ما به یافتن نقطه‌ای روی e با بزرگترین زاویه دید که از رئوس p_1 و p_2 عبور می‌کند تغییر می‌کند. این نقطه روی کوچک‌ترین دایره‌ای قرار دارد که از p_1 و p_2 می‌گذرد و e را قطع می‌کند. بنابراین اگر دایره‌ای که از p_1 و p_2 می‌گذرد و بر خط حامل e مماس است، e را نیز قطع کند، نقطه تماس دایره با e نقطه با بزرگترین زاویه دید روی e است. اگر دایره e را قطع نکند، جواب مسأله یکی از دو سر e است که به نقطه تماس دایره با خط حامل نزدیک‌تر است. اگر زاویه دید جواب یافت شده حداقل α باشد، یالی از s^* به t^* در گراف G_α اضافه می‌کنیم، و اگر چنین نبود، همین کار را برای سایر بخش‌های پاره‌خط s تکرار می‌کنیم.

اندازه مجموع همه نقشه کوتاه‌ترین مسیرها $O(n)$ است و هر یال در هر نقشه حداکثر یکی از پاره‌خط‌های $\mathcal{S}$ را قطع می‌کند. بنابراین در مجموع، پاره‌خط‌های $\mathcal{S}$ به حداکثر تعداد $O(n)$ بخش تقسیم می‌شوند. برای هر بخش، باید اولین رأس در کوتاه‌ترین مسیر به دو سر پاره‌خط قابل دید در جهت d را بیابیم که در زمان $O(\log n)$ قابل انجام است. یافتن نقطه با بزرگترین زاویه دید در هر بخش به زمان $O(1)$ نیازمند است. بنابراین G_α در زمان کلی $O(n \log n)$ محاسبه می‌شود.

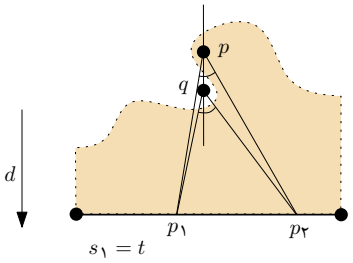
برای جهت d در D، $G_d = (V, E_d)$ را زیرگرافی از G_α بگیرید که فقط شامل یال‌های $(s^*, t^*) \in E$ باشد که s و t دو یال روبه‌روی هم در $\mathcal{T}_d$ باشند. واضح است که G_d گرافی مسطح است، زیرا می‌توان یال‌های آن را طوری رسم کرد که فقط در جهت d و عمود بر d باشند و یکدیگر را قطع نکنند. چون این گراف خطی است، می‌توان آن را در داده‌ساختاری با اندازه $O(n)$ ساخت به‌طوری که در زمان $O(1)$ بررسی کرد که آیا یالی در گراف بین دو رأس خاص وجود دارد یا خیر. نحوه ساخت این داده‌ساختار توسط Munro و Raman [۵۱] توضیح داده شده است. برای تعیین این‌که آیا t از s آلفا–رؤیت‌پذیر ضعیف است، کافی است به ازای هر $d \in D$ بررسی کنیم که آیا یال (s^*, t^*) در G_d وجود دارد یا خیر. این کار در مدت $O(1/\alpha)$ که زمانی است ثابت، انجام می‌شود. □

۳.۵.۷ آزمون آلفا–رؤیت‌پذیری ضعیف با یک پاره‌خط پرس‌وجوی دلخواه

در این بخش، مسأله بخش قبل را کمی تغییر می‌دهیم. این بار می‌خواهیم آلفا–رؤیت‌پذیری ضعیف پاره‌خط $t \in \mathcal{S}$ را از پاره‌خط دلخواه $s \notin \mathcal{S}$ بیازماییم. به‌طور دقیق‌تر، می‌خواهیم $\mathcal{S}$ را پیش‌پردازش کنیم به‌شکلی که با دریافت دو پاره‌خط $s \notin \mathcal{S}$ و $t \in \mathcal{S}$ تعیین کنیم که آیا t از s آلفا–رؤیت‌پذیر ضعیف است یا خیر. برای حل این مسأله، برای هر پاره‌خط $t \in \mathcal{S}$، مجموعه همه نقاطی از صفحه را که از آن نقاط t آلفا–رؤیت‌پذیر است تعیین می‌کنیم. اگر t از پاره‌خط s آلفا–رؤیت‌پذیر ضعیف باشد، s باید نقطه‌ای در این مجموعه داشته باشد. داده‌ساختار ناحیه آلفا–رؤیت‌پذیری t در جهت d را بدین صورت تعریف



شکل ٦.۷ : ناحیه آلفا–رؤیت‌پذیری t در جهت d.



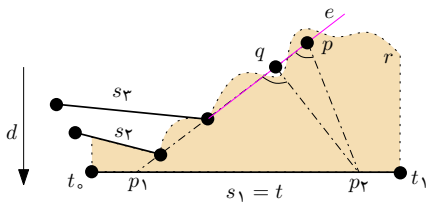
شکل ۷.۷ : نقاط p و q روی خطی در جهت d قرار دارند. p در ناحیه آلفا–رؤیت‌پذیری t در جهت d قرار دارد، ولی q بیرون این ناحیه است.

می‌کنیم: ناحیه‌ای که شامل همه نقاط صفحه است که از آن نقاط t آلفا–رؤیت‌پذیر است و جهت d در زاویه دید نقطه قرار دارد. شکل ٦.۷، ناحیه آلفا–رؤیت‌پذیری t را برای جهت رو به پایین نمایش می‌دهد.

لم ۲.۷ فرض کنید d یک جهت دلخواه و $d_\perp$ جهت عمود بر d باشد. برای هر پاره‌خط t، مرز ناحیه آلفا–رؤیت‌پذیری t در جهت d، که با r_t نشان می‌دهیم، از دو بخش $d_\perp$–یکنوا[۱] تشکیل شده است.

اثبات. توجه کنید که r_t زیرمجموعه ناحیه‌ای است که در $\mathcal{T}_d$ به t نسبت داده شده است، زیرا همه نقاط r_t در جهت d، t را می‌بینند. برای رسیدن به تناقض، فرض کنید r_t از دو بخش $d_\perp$–یکنوا تشکیل نشده است (شکل ۷.۷ را ببینید). بنابراین خطی در جهت d وجود دارد که مرز r_t را در سه نقطه یا بیشتر قطع می‌کند (بخش بالای t را دو بار قطع می‌کند). ما می‌توانیم روی این خط دو نقطه p و q را طوری انتخاب کنیم که اولاً p در r_t باشد و q در r_t نباشد و ثانیاً بردار $\overrightarrow{pq}$ به سمت t اشاره کند. می‌دانیم اندازه زاویه دید نقطه p، β، حداقل برابر مقدار α است و $\overrightarrow{pq}$ درون این زاویه قرار دارد. $p_۱$ و $p_۲$ را دو نقطه برخورد مرز زاویه دید p با t بگیرید. واضح است که زاویه $\angle p_۱qp_۲ \geq \alpha$ و $\triangle p_۱qp_۲ \subseteq \triangle p_۱pp_۲$ و هیچ مانعی در مثلث $\triangle p_۱qp_۲$ قرار ندارد. بنابراین q نیز t را با زاویه‌ای حداقل مساوی α می‌بیند و این

[۱] $d_\perp$-monotone



شکل ۸.۷ : نقاط p و q دو نقطه روی e هستند. p در ناحیه آلفا-رؤیت‌پذیری t در جهت d قرار دارد ولی q بیرون این ناحیه است.

برخلاف فرض اولیه است که $q \notin r_t$. نتیجه این‌که r_t از دو بخش $d_\perp$-یکنوا تشکیل شده است. □

از اثبات لم بالا، نتیجه زیر حاصل می‌شود.

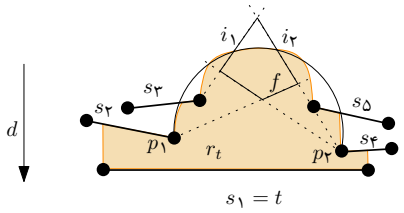
نتیجه ۲.۷ ناحیه آلفا-رؤیت‌پذیری t در جهت d، حفره ندارد.

اکنون به سراغ ساخت ناحیه آلفا-رؤیت‌پذیری برای پاره‌خط‌های $\mathcal{S}$ می‌رویم. در ابتدا نشان می‌دهیم حافظه لازم برای نگه‌داری همه این نواحی خطی است.

لم ۳.۷ مجموع پیچیدگی ناحیه‌های آلفا-رؤیت‌پذیری برای همه پاره‌خط‌های $\mathcal{S}$ در جهت d خطی است.

اثبات. فرض کنید c ناحیه‌ای از $\mathcal{T}_d$ باشد که به t نسبت داده شده است و r_t ناحیه آلفا-رؤیت‌پذیری t در جهت d باشد. می‌دانیم $r_t \subseteq c$. نقشه کوتاه‌ترین مسیر دو سر t را در c می‌سازیم. ادعای اول ما این است که مرز r_t هر یال نقشه کوتاه‌ترین مسیر را حداکثر یک‌بار قطع می‌کند. فرض کنیم چنین نباشد و مطابق با شکل ۸.۷، برای یکی از دو سر t، مثلاً $t_۰$ یالی مانند e در نقشه کوتاه‌ترین مسیر $t_۰$ وجود داشته باشد که مرز r_t، e را حداقل در دو جا قطع کند. می‌توانیم دو نقطه p و q را روی e طوری انتخاب کنیم که $p \in r_t$ و $q \notin r_t$ و بردار $\overrightarrow{pq}$ به سمت t اشاره کند. فرض کنید $p_۱$ و $p_۲$ دو نقطه برخورد مرز زاویه دید p با t باشند، به نحوی که $p_۱$ از $p_۲$ به $t_۰$ نزدیک‌تر باشد. چون p و q روی یالی از نقشه کوتاه‌ترین مسیر $t_۰$ قرار دارند، هر دو از $p_۱$ قابل دید هستند. همچنین $p_۱$ روی نقطه برخورد t و خط حامل e قرار دارد. می‌دانیم که $\angle p_۱pp_۲ \geq \alpha$. اکنون زاویه $\angle p_۱qp_۲$ را در نظر بگیرید. مشاهده می‌شود که $\angle p_۱qp_۲$ از $\angle p_۱pp_۲$ بزرگ‌تر بوده و خالی است. بنابراین q می‌تواند t را با زاویه‌ای حداقل مساوی α ببیند و چون q در c قرار دارد، باید در r_t نیز باشد که خلاف فرض اولیه است که $q \notin r_t$. بنابراین فرض خلف باطل است و مرز r_t هر یال نقشه کوتاه‌ترین مسیر را حداکثر یک‌بار قطع می‌کند.

اکنون مثالی را که در شکل ۹.۷ دیده می‌شود، در نظر بگیرید. فرض کنید $i_۱$ و $i_۲$ دو نقطه برخورد متوالی مرز r_t با دو نقشه کوتاه‌ترین مسیر دو سر t باشد. بنابراین $i_۱$ و $i_۲$ دو رأس در مرز یک



شکل ۹.۷ : اثبات لم ۳.۷، نقاط i_1 و i_2 دو محل برخورد مرز r_t با نقشه‌های کوتاه‌ترین مسیر دو سر t است.

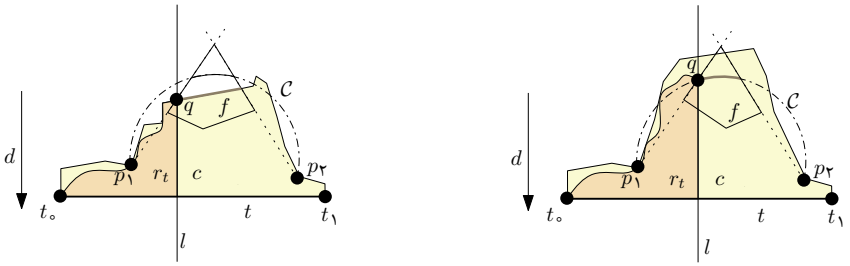
چندضلعی f خواهند بود که حاصل روی هم قرار گرفتن نقشه‌های کوتاه‌ترین مسیر است. ادعای دوم ما این است که بین i_1 و i_2، پیچیدگی مرز r_t برابر $O(|f|)$ است. این مطلب بدان خاطر است که برای هر نقطه در f، شکل ترکیبیاتی کوتاه‌ترین مسیرها به سمت دو سر t ثابت است و در نتیجه برای هر نقطه در f زاویه دید به سمت دید توسط دو نقطه ثابت، مثلاً p_1 و p_2، تعیین می‌شود. مجموعه همه نقاطی از f که t را از میان p_1 و p_2 با زاویه‌ای حداقل برابر با α می‌بینند، در داخل یا مرز کمانی است که از p_1 و p_2 می‌گذرد و اندازه زاویه محاطی آن به رأس‌های p_1 و p_2، α است. بنابراین اشتراک r_t با f مساوی اشتراک کمان مذکور با f است. این اشتراک اندازه‌ای حداکثر مساوی $2|f|$ دارد، زیرا هر یال f حداکثر دو بار کمان را قطع می‌کند. بنابراین مرز r_t بین i_1 و i_2 نمی‌تواند اندازه‌ای بیش از $O(|f|)$ داشته باشد.

با استفاده از دو ادعای بالا، می‌توانیم پیچیدگی r_t را محاسبه کنیم. مرز r_t هر یال نقشه‌های کوتاه‌ترین مسیر را حداکثر یک‌بار قطع می‌کند و داخل هر ناحیه f پیچیدگی r_t مساوی $O(|f|)$ است. واضح است که r_t یال‌های ناحیه f را نمی‌تواند در ناحیه‌های مجاور f قطع کند. بنابراین هر یال حداکثر یک‌بار در پیچیدگی r_t مؤثر است. پس پیچیدگی r_t، $O(|c|)$ است و چون مجموع پیچیدگی ناحیه‌های $\mathcal{T}_d$ خطی است، مجموع پیچیدگی ناحیه‌های آلفا–رؤیت‌پذیری در جهت d برای همه پاره‌خط‌ها $O(n)$ است. □

در لم بالا نشان دادیم که حافظه لازم برای نگه‌داری ناحیه‌های آلفا–رؤیت‌پذیری، اندازه خطی دارد. در ادامه و در لم زیر نحوه محاسبه این ناحیه‌ها را در زمان $O(n \log n)$ توضیح می‌دهیم.

لم ۴.۷ ناحیه آلفا–رؤیت‌پذیری در جهت d را برای همه پاره‌خط‌های $\mathcal{S}$، می‌توان در زمان $O(n \log n)$ و با حافظه $O(n)$ محاسبه کرد.

اثبات. ابتدا $\mathcal{T}_d$ را برای مجموعه $\mathcal{S}$ محاسبه می‌کنیم. سپس برای هر پاره‌خط $t \in \mathcal{S}$، نقشه‌های کوتاه‌ترین مسیر دو سر t را در ناحیه c نسبت داده شده به t در $\mathcal{T}_d$ رسم می‌کنیم. فرض کنید t_0 و t_1 دو سر t باشند و $\mathrm{SPM}(t_0)$ و $\mathrm{SPM}(t_1)$ نقشه‌های کوتاه‌ترین مسیر t_0 و t_1 باشند. ناحیه آلفا–رؤیت‌پذیری t در جهت d را r_t بنامید. طبق تعریف، دو سر t در r_t هستند. مرز پایینی r_t خود t است. مرز بالایی r_t را



الف ب

شکل ۱۰.۷ : الف. پردازش نقطه رویداد q وقتی q داخل c است. ب. پردازش نقطه رویداد q وقتی q روی مرز c است.

به‌تدریج با جاروب کردن خط l موازی d از t_0 تا t_1 می‌سازیم. چون مرز بالایی r_t یکنواست، l همواره مرز بالایی r_t را در یک نقطه قطع می‌کند. نقاط رویداد مهم برای ما برخورد مرز r_t با $\mathrm{SPM}(t_0)$ و $\mathrm{SPM}(t_0)$ و مرز c است.

فرض کنید نقطه فعلی در حال پردازش q باشد، یعنی محل برخورد خط l با مرز بالایی r_t، و p_1 و p_2 اولین رأس کوتاه‌ترین مسیرها از q تا t_0 و t_1 باشد. در آغاز $q = t_0$ و کوتاه‌ترین مسیرها از q تا t_0 و t_1 دو پاره‌خطی است که q را به t_0 و t_1 وصل می‌کند. بنابراین p_1 و p_2 به ترتیب t_0 و t_1 هستند.

فرض کنید نقطه رویداد جاری q است و ما مرز r_t را از t_0 تا q حساب کرده‌ایم. همچنین فرض کنید q روی مرز یا داخل چندضلعی f که از روی هم قرار دادن دو نقشه کوتاه‌ترین مسیر t_0 و t_1 ساخته شده است، قرار دارد. مجموعه نقاطی که t را با زاویه‌ای حداقل مساوی α از میان p_1 و p_2 می‌بینند، در داخل یا روی مرز کمانی از دایره $\mathcal{C}$ هستند که از p_1 و p_2 می‌گذرد و زاویه محاطی کمان مساوی α است و این کمان بالای p_1p_2 است. اگر c روی مرز c نباشد (شکل ۱۰.۷– الف)، زاویه دید آن دقیقا α است و روی یالی از نقشه‌های کوتاه‌ترین مسیر است که به سمت p_1 یا p_2 است. بنابراین دایره $\mathcal{C}$ آن یال را در نقطه q قطع می‌کند و وارد f می‌شود. در این حالت $\mathcal{C}$ مرز r_t را تعیین می‌کند. در مقابل، اگر q روی مرز c باشد (شکل ۱۰.۷– ب)، q درون $\mathcal{C}$ خواهد بود و مرز c مرز r_t را تعیین می‌کند. در حالت اول، ما اولین نقطه تلاقی $\mathcal{C}$ با $\mathrm{SPM}(t_0)$ و $\mathrm{SPM}(t_1)$ و c را به عنوان نقطه رویداد بعدی ذخیره می‌کنیم. در حالت دوم، اولین نقطه برخورد c با $\mathrm{SPM}(t_0)$ و $\mathrm{SPM}(t_1)$ و $\mathcal{C}$ به عنوان نقطه رویداد بعدی ذخیره می‌کنیم. در هر دو حالت، یالی که q را به نقطه رویداد بعدی متصل می‌کند، به r_t اضافه می‌کنیم. این یال بسته به این‌که از مرز $\mathcal{C}$ یا مرز c انتخاب شده باشد، کمانی از دایره یا پاره‌خط است.

تحلیل زمانی: $\mathcal{T}_d$ در زمان $O(n \log n)$ ساخته می‌شود. $\mathrm{SPM}(t_0)$ و $\mathrm{SPM}(t_1)$ در زمان $O(n)$ محاسبه می‌شود. یافتن اولین نقطه برخورد $\mathcal{C}$ با $\mathrm{SPM}(t_0)$ و $\mathrm{SPM}(t_1)$ زمان $O(|f|)$ لازم دارد که در مجموع در مدت $O(|c|)$ برای همه دایره‌ها قابل انجام است. محاسبه نقاط برخورد دایره‌ها با c نیز در

زمان کلی $O(|c|)$ انجام می‌پذیرد. بنابراین ناحیه آلفا–رؤیت‌پذیری برای همه پاره‌خط‌ها در جهت d در زمان $O(n \log n)$ و با صرف حافظه $O(n)$ محاسبه می‌شود. □

اکنون با استفاده از ناحیه آلفا–رؤیت‌پذیری یک پاره‌خط، وجود رابطه بین دو پاره‌خط $s \notin \mathcal{S}$ و $t \in \mathcal{S}$ را تعیین می‌کنیم. قضیه زیر روش کار را توضیح می‌دهد.

قضیه ۷.۷ مجموعه $\mathcal{S}$ از n پاره‌خط داده شده است. می‌توان $\mathcal{S}$ را در داده‌ساختاری با اندازه $O(n)$ در زمان $O(n \log n)$ پیش‌پردازش کرد، به‌گونه‌ای که با دریافت دو پاره‌خط پرس‌وجوی $s \notin \mathcal{S}$ و $t \in \mathcal{S}$، در زمان $O(\log n)$ تعیین کنیم که آیا t از s آلفا–رؤیت‌پذیر ضعیف است یا خیر.

اثبات. همانند قبل، ابتدا مجموعه D از $O(1/\alpha)$ جهت را انتخاب می‌کنیم به شکلی که زاویه بین هر دو جهت متوالی حداکثر α باشد. اگر t از s آلفا–رؤیت‌پذیر ضعیف باشد، نقطه‌ای مانند q روی s و جهت $d \in D$ وجود دارد که t از q آلفا–رؤیت‌پذیر است و q می‌تواند t را در جهت d در داخل زاویه دید خود ببیند. بنابراین q در ناحیه آلفا–رؤیت‌پذیری t در جهت d قرار دارد. اکنون مسأله به بررسی تلاقی s با هر یک از ناحیه‌های آلفا–رؤیت‌پذیری t در هریک از جهت‌های D کاسته می‌شود.

برای حل مسأله، در زمان پیش‌پردازش برای همه جهت‌های $d \in D$ و همه پاره‌خط‌های $t \in \mathcal{S}$ ناحیه آلفا–رؤیت‌پذیری t در جهت d را محاسبه کرده و آن را برای شلیک پرتو پیش‌پردازش می‌کنیم. ناحیه آلفا–رؤیت‌پذیری، ناحیه‌ای محدود و بدون حفره است، که مرز آن از تعدادی پاره‌خط و کمان دایره تشکیل شده است. بنابراین بر طبق تعریف چندضلعی‌گون، این ساختار یک چندضلعی‌گون است و ما با استفاده از قضیه ۳.۷ می‌توانیم آن را برای شلیک پرتو پیش‌پردازش کنیم.

در زمان پرس‌وجو، با دریافت پاره‌خط‌های $s \notin \mathcal{S}$ و $t \in \mathcal{S}$، ابتدا ناحیه آلفا–رؤیت‌پذیری t در جهت d، r_t را می‌یابیم. می‌خواهیم بدانیم که آیا s با r_t اشتراکی دارد یا خیر. اگر s_0 و s_1 دو سر s باشند، ابتدا بررسی می‌کنیم که آیا s_0 در r_t است یا خیر. چون الگوریتم شلیک پرتو [۵۰]، الگوریتم جایابی نقطه را نیز در زمینه خود دارد، می‌توانیم از همان الگوریتم استفاده کنیم و تعیین کنیم که آیا s_0 در r_t است یا خیر. اگر s_0 در r_t بود، جواب سؤال اصلی، مثبت است و t از s آلفا–رؤیت‌پذیر ضعیف است. اگر s_0 در r_t نبود، الگوریتم شلیک پرتو را از s_0 و در جهت s_1 اجرا می‌کنیم تا ببینیم اولین محل برخورد پرتو با r_t کجاست. اگر محل برخورد روی s باشد، s با r_t اشتراک دارد و t از s آلفا–رؤیت‌پذیر ضعیف است. در غیر این‌صورت t از s در جهت d آلفا–رؤیت‌پذیر ضعیف نیست و باید به سراغ جهت دیگری در مجموعه D برویم. اگر برای همه جهت‌ها این کار تکرار شد و s با هیچ کدام از ناحیه‌ها اشتراک نداشت، t از s آلفا–رؤیت‌پذیر ضعیف نیست.

تحلیل زمان و حافظه: محاسبه ناحیه‌های آلفا–رؤیت‌پذیری، زمان $O(n \log n)$ و حافظه $O(n)$ را نیاز دارد. پیش‌پردازش ناحیه‌ها برای شلیک پرتو نیز به همین زمان و حافظه نیاز دارد. زمان پرس‌وجو

برای یافتن جایابی نقطه و شلیک پرتو $O(\log n)$ است. □

۶.۷ قابلیت دید کامل پاره‌خط

در این بخش به موضوع آلفا–رؤیت‌پذیری کامل می‌پردازیم. می‌گوییم پاره‌خط t از پاره‌خط s آلفا–رؤیت‌پذیر کامل است اگر و فقط اگر t از همه نقاط s آلفا–رؤیت‌پذیر باشد. مشابه تعریف گراف آلفا–رؤیت‌پذیری ضعیف، گراف آلفا–رؤیت‌پذیری کامل تعریف می‌شود. توجه کنید که گراف آلفا–رؤیت‌پذیری کامل، زیرگرافی از گراف آلفا–رؤیت‌پذیری ضعیف است و بنابراین اندازه آن خطی است. به روشی مشابه قضیه ۶.۷ می‌توان گراف آلفا–رؤیت‌پذیری کامل یک مجموعه پاره‌خط را محاسبه کرد. اما در بخش بعدی ما با استفاده از ناحیه‌های آلفا–رؤیت‌پذیری، این گراف را محاسبه می‌کنیم، زیرا توضیح آن ساده‌تر و کوتاه‌تر خواهد بود.

۱.۶.۷ آزمون آلفا–رؤیت‌پذیری کامل

مسأله‌ای که در این بخش حل می‌کنیم تعیین آلفا–رؤیت‌پذیری کامل پاره‌خطی از $\mathcal{S}$ از پاره‌خط دیگری از $\mathcal{S}$ است. به‌طور دقیق‌تر ما مجموعه $\mathcal{S}$ را پیش‌پردازش می‌کنیم، تا با دریافت دو پاره‌خط پرس‌وجوی s و t در $\mathcal{S}$، مشخص کنیم که آیا t از s آلفا–رؤیت‌پذیر کامل است یا خیر.

قضیه ۸.۷ مجموعه پاره‌خط‌های $\mathcal{S}$ داده شده است. می‌توان $\mathcal{S}$ را در داده‌ساختاری با اندازه $O(n)$ پیش‌پردازش کرد، به‌گونه‌ای که بتوان آزمون‌های آلفا–رؤیت‌پذیری کامل بین دو پاره‌خط $\mathcal{S}$ را در زمان $O(1)$ پاسخ داد.

اثبات. مانند قبل، مجموعه D از $O(1/\alpha)$ جهت را انتخاب می‌کنیم و برای هر جهت $d \in D$ از نتیجه لم ۴.۷ استفاده می‌کنیم تا ناحیه‌های آلفا–رؤیت‌پذیری همه پاره‌خط‌های $\mathcal{S}$ را محاسبه کنیم. مرز ناحیه آلفا–رؤیت‌پذیری t، r_t از مجموعه‌ای از پاره‌خط‌ها و کمان‌ها تشکیل شده است. پاره‌خط‌های مرز r_t پاره‌خط‌هایی از $\mathcal{S}$ یا بخش‌هایی از آن‌ها هستند که مانع دیدن نقاط پشت خود می‌شوند. پاره‌خط t از این بخش‌های پاره‌خط‌های $\mathcal{S}$ آلفا–رؤیت‌پذیر است. بنابراین اگر t از پاره‌خطی آلفا–رؤیت‌پذیر کامل باشد، باید اجتماع همه بخش‌هایی از s که در ناحیه‌های آلفا–رؤیت‌پذیری t در همه جهات D قرار می‌گیرند، برابر خود s شود.

برای جهت $d \in D$ و پاره‌خط $t \in \mathcal{S}$، همه پاره‌خط‌های $e \in r_t$ را پیمایش می‌کنیم. فرض کنید e بخشی از $s \in \mathcal{S}$ باشد. e را به مجموعه مرتب $I_{s,d}$ اضافه می‌کنیم. مجموعه مرتب $I_{s,d}$، همه بازه‌هایی از

s را ذخیره می‌کند که در r_t به ازای جهت d شرکت دارند، یعنی از نقاط آن‌ها t آلفا–رؤیت‌پذیر است و جهت d در زاویه دید آن‌هاست. بعد از پیمایش همه ناحیه‌های همه پاره‌خط‌ها، ما برای هر پاره‌خط $O(1/\alpha)$ مجموعه مرتب داریم، که نقاط هر مجموعه t را با زاویه‌ای حداقل مساوی α می‌بینند. اگر t از s آلفا–رؤیت‌پذیر کامل باشد، اجتماع همه این بازه‌ها s می‌شود. اجتماع این بازه‌ها را می‌توان در زمان $O(k\log(1/\alpha))$ محاسبه کرد، که k در این‌جا تعداد کل بخش‌هایی از s است که در ناحیه‌های قابلیت دید t استفاده شده‌اند و حداکثر $O(n)$ است. اگر اجتماع این بخش‌ها s باشد، یالی را در گراف آلفا–رؤیت‌پذیری کامل $\mathcal{S}$ از رأس متناظر با s به رأس متناظر با t اضافه می‌کنیم. در مجموع، زمان پیش‌پردازش لازم برای ساخت گراف، $O(n\log(1/\alpha)/\alpha)$ خواهد بود، زیرا تعداد کل بخش‌های همه پاره‌خط‌ها $O(n/\alpha)$ است.

برای پاسخ دادن به پرس‌وجوها، اگر از نکته‌ای که در قضیه ۶.۷ ذکر شد، استفاده کنیم، می‌توانیم گراف را به $O(1/\alpha)$ زیر گراف مسطح تقسیم کنیم و پرس‌وجو را در زمان $O(1/\alpha)$ پاسخ دهیم، که زمانی است ثابت. □

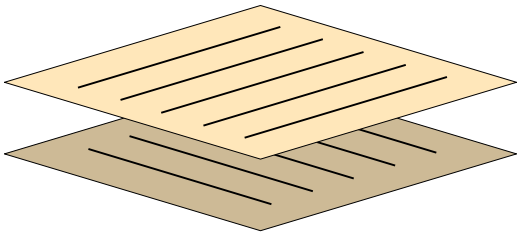
۲.۶.۷ آزمون آلفا–رؤیت‌پذیری کامل با یک پاره‌خط پرس‌وجوی دلخواه

در این بخش، مسأله قبل را کمی تغییر داده و آن را با این فرض حل می‌کنیم که s جزء پاره‌خط‌های مجموعه $\mathcal{S}$ نباشد. در این مسأله نیز، از ناحیه‌های آلفا–رؤیت‌پذیری استفاده می‌کنیم. اگر t از s آلفا–رؤیت‌پذیر کامل باشد، اجتماع همه اشتراک‌های s با ناحیه‌های آلفا–رؤیت‌پذیری t مساوی s خواهد بود.

قضیه ۹.۷ مجموعه پاره‌خط‌های $\mathcal{S}$ داده شده است. می‌توان $\mathcal{S}$ را در داده‌ساختاری با اندازه $O(n)$ پیش‌پردازش کرد، به‌گونه‌ای که با دریافت دو پاره‌خط پرس‌وجوی $s \notin \mathcal{S}$ و $t \in \mathcal{S}$، در مدت زمان $O(n)$ تعیین کرد آیا t از s آلفا–رؤیت‌پذیر کامل است یا خیر.

اثبات. در زمان پیش‌پردازش، همانند قبل، مجموعه D از $O(1/\alpha)$ جهت را انتخاب می‌کنیم. به ازای هر $d \in D$، با استفاده از لم ۴.۷، ناحیه آلفا–رؤیت‌پذیری همه پاره‌خط‌های $t \in \mathcal{S}$ را محاسبه می‌کنیم. در زمان پرس‌وجو، با دریافت پاره‌خط دلخواه s، برای هر کدام از $d \in D$، بازه‌هایی از s که از آن‌ها t آلفا–رؤیت‌پذیر است را در $I_{s,d}$ قرار می‌دهیم. سپس اجتماع همه این بازه‌ها را محاسبه می‌کنیم. اگر این نتیجه با s مساوی بود، t آلفا–رؤیت‌پذیر است، و در غیر اینصورت آلفا–رؤیت‌پذیر نیست.

زمان پیش‌پردازش $O(n\log n)$ و حافظه مصرفی $O(n)$ است. در زمان پرس‌وجو، ما $O(1/\alpha)$ مجموعه مرتب بازه داریم، که اگر تعداد کلی بازه‌ها k باشد، اجتماع مجموعه‌ها در مدت $O(k\log(1/\alpha))$



شکل ۱۱.۷ : آرایشی از n پاره‌خط در فضای سه‌بعدی که گراف آلفا–رؤیت‌پذیری آن‌ها اندازه $\Theta(n^2)$ دارد.

محاسبه می‌شود. در بدترین حالت، مقدار k، $O(n)$ خواهد بود. □

۷.۷ قابلیت دید در فضای سه‌بعدی

در هندسه محاسباتی و بخصوص مسائل قابلیت دید، یک قاعده سرانگشتی وجود دارد که پیچیدگی مسأله در ابعاد بزرگتر، به‌صورت نمایی رشد پیدا می‌کند. هر چند این قاعده همواره صحیح نیست، ولی مسائل زیادی وجود دارند که از چنین قاعده‌ای تبعیت می‌کنند. بنابراین نگاه اولیه ما به موضوع تعمیم آلفا–رؤیت‌پذیری، افزایش پیچیدگی مسائل مطرح در قابلیت دید است.

در واقع این نگاه نسبت به موضوع، بی‌دلیل هم نیست و ما می‌توانیم نشان دهیم که این مسائل در فضای سه‌بعدی به خوبی مسائل معادل در صفحه نیستند و پیچیدگی زمانی و حافظه مصرفی آن‌ها بالا می‌رود. مثال شکل ۱۱.۷ نشان می‌دهد که گراف آلفا–رؤیت‌پذیری که در صفحه اندازه خطی داشت و این یکی از مزیت‌های برتری آلفا–رؤیت‌پذیری نسبت به قابلیت دید عادی بود، در فضای سه‌بعدی به همان خوبی عمل نمی‌کند و می‌توان نمونه‌هایی را ساخت که اندازه گراف قابلیت دید در آن‌ها $\Theta(n^2)$ است، یعنی معادل اندازه گراف قابلیت دید عادی در بدترین حالت. این بدان معناست که تعریف آلفا–رؤیت‌پذیری به شکل فعلی، کارایی چندانی در فضای سه‌بعدی ندارد و اگر به دنبال تقریب زدن قدرت دید مشابه روش آلفا–رؤیت‌پذیری باشیم، باید تعاریف جدیدی در این زمینه ارائه کنیم.

اما روش ساخت نمونه‌ای با اندازه گراف آلفا–رؤیت‌پذیری $\Theta(n^2)$ بدین صورت است. دو صفحه در فضا به موازات هم و عمود بر محور z، در نظر بگیرید. روی صفحه پایینی $n/2$ پاره‌خط موازی محور x و روی صفحه بالایی $n/2$ پاره‌خط موازی محور y قرار می‌دهیم. می‌توانیم زاویه α و فاصله بین دو صفحه را به‌گونه‌ای انتخاب کنیم که همه پاره‌خط‌های یک صفحه، از همه پاره‌خط‌های صفحه دیگر آلفا–رؤیت‌پذیر ضعیف باشند. بنابراین لم ۱.۷ مبنی بر خطی بودن گراف آلفا–رؤیت‌پذیری،

برای فضای سه‌بعدی قابل تعمیم نیست.

این مثال نشان می‌دهد که باید تعریف آلفا–رؤیت‌پذیری برای فضای سه‌بعدی را تغییر دهیم. مسأله چگونگی ارائه تعریفی جایگزین برای آلفا–رؤیت‌پذیری در فضای سه‌بعدی، می‌تواند زمینه جالبی برای سایر پژوهش‌ها در این زمینه باشد.

۸.۷ خلاصه

در این فصل مفهوم جدیدی در ارتباط با تقریب زدن قابلیت دید بین اشیاء ارائه شد. مفهوم آلفا–رؤیت‌پذیری، در ابتدا برای تقریب قابل دید بودن یک پاره‌خط از یک نقطه تعریف شد و سپس به اشیاء پیچیده‌تر تعمیم داده شد. مشابه قابلیت دید عادی، آلفا–رؤیت‌پذیری ضعیف و کامل، بین دو پاره‌خط تعریف شد.

در ادامه فصل، نشان دادیم که گراف‌های آلفا–رؤیت‌پذیری ضعیف و کامل یک مجموعه پاره‌خط (چندضلعی حفره‌دار)، خطی است و الگوریتمی برای محاسبه آن در مدت زمان $O(n \log n)$ ارائه دادیم. این مطلب، هم از لحاظ نظری و هم از لحاظ عملی، زمانی که با مجموعه‌های بزرگ داده‌ای کار می‌کنیم بسیار قابل توجه است. همچنین در این فصل، با استفاده از تعاریف جدید، راه‌حل‌هایی برای تعدادی از مسائل قابلیت دید ارائه شد. در انتها نیز نشان دادیم که این نتایج بر پایه این تعاریف قابل تعمیم به فضای سه‌بعدی نیست و باید به دنبال تعاریف جدیدی برای فضای سه‌بعدی بود.

فصل ۸

نتیجه‌گیری

۱.۸ نتایج به‌دست آمده

در این پژوهش به بررسی مسائل مربوط به قابلیت دید پرداخته شد. این مسائل در دو دسته کلی قرار می‌گیرند. محاسبه چندضلعی قابل دید نقطه (در محیط‌ها و با شرایط مختلف)، آزمون قابل دید بودن (با تعاریف مختلف قابل دید بودن و اشیاء مختلف مورد آزمون).

اولین مسأله مورد بررسی، آزمون قابل دید ضعیف بودن دو شیء از یکدیگر در یک چندضلعی‌حفره‌دار بود. این مسأله در حالت‌های مختلف بدون پیش‌پردازش و با پیش‌پردازش و با اشیاء مختلف بررسی و حل شد. نتایج مربوط به این مسأله در جدول ۱.۸ آمده است.

در ادامه کار، مسأله محاسبه چندضلعی قابل دید یک نقطه در درون چندضلعی حفره‌دار بررسی شد. در آغاز الگوریتمی با زمان لگاریتمی برای محاسبه چندضلعی قابل دید پیشنهاد شد. سپس برای کاهش حافظه مصرفی، توازنی بین حافظه و زمان پرس‌وجو برقرار شد. در ادامه کاربردهایی از ایده ارائه شده، در مسائل دیگر دیده شد و برای آن مسائل نیز راه حل ارائه شد. جدول ۲.۸ نتایج مربوط به این قسمت را خلاصه می‌کند.

موضوع دیگری که در این رساله مطرح شد، تعریف قابلیت دید جزئی و کاربردهای آن بود. در ابتدا الگوریتمی برای محاسبه چندضلعی قابل دید جزئی یک نقطه پرس‌وجو ارائه کردیم و سپس نشان دادیم روش حل مسأله، قابل تعمیم به مسأله‌ای که در شبکه‌های حس‌گر کاربرد دارد می‌باشد.

در انتهای رساله، مفهوم قابلیت دید آلفا تعریف شد و برخی از مسائلی که براساس این تعریف قابل طرح هستند بررسی و حل شدند. نتایج این بخش در جدول ۳.۸ به‌طور خلاصه آمده‌اند.

جدول ۱.۸ : نتایج به‌دست آمده برای مسأله آزمون قابل دید ضعیف بودن دو شیء*

شرایط مسأله	زمان پیش‌پردازش	حافظه	زمان پرس‌وجو
چندضلعی ساده، بدون پیش‌پردازش، نقطه ← شیء	–	$O(n)$	$O(n)$
چندضلعی حفره‌دار، بدون پیش‌پردازش، نقطه ← شیء	–	$O(n)$	$O(n \log n)$
چندضلعی ساده، SOQ، نقطه ← شیء	$O(n)$	$O(n)$	$O(\log n)$
چندضلعی حفره‌دار، SOQ، نقطه ← شیء	$O(n \log n)$	$O(n)$	$O(\log n)$
چندضلعی ساده، TOQ، نقطه ← نقطه	$O(n)$	$O(n)$	$O(\log n)$
چندضلعی حفره‌دار، SOQ، نقطه ← نقطه	$O(m^{1+\epsilon})$	$O(m)$	$O(n^{1+\epsilon}/\sqrt{m})$
چندضلعی ساده، بدون پیش‌پردازش، شیء ← شیء	–	$O(n)$	$O(n)$
چندضلعی حفره‌دار، بدون پیش‌پردازش، شیء ← شیء	–	$O(n)$	$O(n^2)$
چندضلعی ساده، SOQ، شیء ← شیء	$O(n \log n)$	$O(n)$	$O(\log n)$
چندضلعی حفره‌دار، SOQ، شیء ← نقطه	$O(n^4 \log n)$	$O(n^4)$	$O(\log n)$
چندضلعی حفره‌دار، SOQ، شیء ← شیء	$O(n^{4+\epsilon})$	$O(n^4)$	$O(n^\epsilon)$
چندضلعی حفره‌دار، TOQ، شیء ← شیء	$O(m^{1+\epsilon})$	$O(m)$	$O(n^{1+\epsilon}/\sqrt{m})$

* پارامتر دلخواه m با شرط $n^2 \leq m \leq n^4$ انتخاب می‌شود.

جدول ۲.۸ : نتایج به‌دست آمده برای مسأله محاسبه چندضلعی قابل دید نقطه در چندضلعی حفره‌دار*

شرایط مسأله	زمان پیش‌پردازش	حافظه	زمان پرس‌وجو
محاسبه چندضلعی قابل دید نقطه	$O(n^4 \log n)$	$O(n^4)$	$O(\log n)$
محاسبه چندضلعی قابل دید نقطه	$O(m \log(1+\sqrt{m}/n))$	$O(m)$	$O(n^2 \log(1+\sqrt{m}/n)/\sqrt{m})$
محاسبه نگه‌داری چندضلعی قابل دید نقطه متحرک	$O(m \log(1+\sqrt{m}/n))$	$O(m)$	$O(n^2 \log n\sqrt{m} + k \log n)$
محاسبه چندضلعی قابل دید ضعیف پاره‌خط t	$O(m \log(1+\sqrt{m}/n))$	$O(m)$	$O(n^2 \log n\sqrt{m} + \lvert \mathrm{V}(t) \rvert \log n)$

* پارامتر دلخواه m با شرط $n^2 \leq m \leq n^4$ انتخاب می‌شود.

۲.۸ پژوهش‌های آینده و مسائل باز

آن‌چه پیش از این در این گزارش ملاحظه شد، خلاصه‌ای از مطالعات انجام شده در ضمن تحقیقات و نیز کارهایی که تاکنون انجام شده است بود. در این فصل زمینه‌ها و فعالیت‌های مناسب برای ادامه تحقیقات را مشخص می‌کنیم.

۱.۲.۸ تحلیل پیچیدگی‌ها

یکی از نکاتی که در مورد مسائل قابلیت دید هنوز جای کار زیادی دارد، بررسی پیچیدگی این مسائل است. در بیشتر مسائل قابلیت دید، حد پایین دقیقی برای مسأله به اثبات نمی‌رسد؛ یعنی در این مسائل فقط یک حد بالا برای حل مسأله ارائه می‌شود ولی کارا بودن الگوریتم به اثبات نمی‌رسد.

جدول ۳.۸ : نتایج به‌دست آمده در زمینه قابلیت دید آلفا

شرایط مسأله	زمان پیش‌پردازش	حافظه	زمان پرس‌وجو
آزمون قابلیت دید آلفا نقطه ← پاره‌خط، α ثابت	$O(n \log n)$	$O(n)$	$O(\log n)$
آزمون قابلیت دید آلفا نقطه ← پاره‌خط، α غیرثابت	$O(n \log^{۳} n)$	$O(n \log^{۲} n)$	$O(\sqrt{n} \log^{۳} n)$
گزارش همه پاره‌خط‌های قابل دید آلفا از نقطه	$O(n \log n)$	$O(n)$	$O(\log n)$
ساخت گراف قابل دید آلفا (ضعیف/کامل)	–	$O(n)$	$O(n \log n)$
آزمون قابلیت دید آلفا (ضعیف/کامل)، پاره‌خطی از $\mathcal{S}$← پاره‌خطی از $\mathcal{S}$	$O(n \log n)$	$O(n)$	$O(۱)$
آزمون قابلیت دید آلفا (ضعیف/کامل)، پاره‌خطی دلخواه ← پاره‌خطی از $\mathcal{S}$	$O(n \log n)$	$O(n)$	$O(\log n)$

نمونه این مسائل، مسأله محاسبه چندضلعی قابل دید یک نقطه پرس‌وجو در داخل چندضلعی حفره‌دار و عدم وجود حد پایینی برای آن است. به‌عنوان مثال، ما هنوز نمی‌دانیم که آیا الگوریتم ارائه شده برای محاسبه چندضلعی قابل دید در زمان لگاریتمی که نیازمند زمان پیش‌پردازش $O(n^{۴} \log n)$ است، کاراست یا خیر و آیا الگوریتمی با زمان پیش‌پردازش کمتر نیز ممکن است وجود داشته باشد. ما می‌دانیم که Chazelle [۱۳] حد پایینی را برای مسأله جستجوی بازه‌ها ارائه کرده است که شاید با این مسأله مرتبط باشد، اما آیا می‌توان در این مسأله نیز حد پایینی را مشخص کرد. این مسأله یکی از کارهای مناسب برای ادامه تحقیقات در این زمینه است که در صورت رسیدن به نتیجه، کارایی الگوریتم فعلی را به اثبات می‌رساند.

از مسائل دیگری که در مورد تحلیل پیچیدگی الگوریتم‌ها مناسب به نظر می‌رسد، تحلیل پیچیدگی الگوریتم‌های موجود در حالت متوسط است. ممکن است برخی از الگوریتم‌های موجود برای محاسبه چندضلعی قابل دید یا مسائل مشابه، در بدترین حالت مناسب نباشند، ولی اگر بتوان اثبات کرد که در حالت متوسط پیچیدگی قابل قبولی دارند، کاربرد بیشتری خواهند داشت.

به عنوان نمونه، ما در الگوریتم خود نشان دادیم در بدترین حالت، زمان پیش‌پردازش الگوریتم $O(n^{۴} \log n)$ است. حال اگر بتوانیم اثبات کنیم که در حالت متوسط، این زمان بسیار کمتر از $O(n^{۴} \log n)$ است، بهبود زیادی به الگوریتم خود داده‌ایم. به همین شکل می‌توان به کار Pocchiola و Vegter [۵۷] اشاره کرد که زمان پیش‌پردازش آن وابسته به گراف قابلیت دید است. پس اگر اثبات کنیم که گراف قابلیت دید در حالت متوسط اندازه کمتری از $O(n^{۲})$ دارد، زمان پیش‌پردازش این الگوریتم نیز در حالت متوسط کاهش خواهد یافت.

۲.۲.۸ الگوریتم‌های قابلیت دید جنبشی

یکی از کارهایی که در چند سال اخیر روی مسائل هندسه محاسباتی انجام شده است، جنبشی کردن الگوریتم‌هاست. مثلاً الگوریتم جنبشی محاسبه پوسته محدب[۱] تعدادی نقطه، پوسته محدب را زمانی

[۱] Convex hull

که نقاط در حال حرکت هستند به‌صورت بهینه نگه‌داری و به‌روزرسانی می‌کند. در مورد جنبشی کردن مسائل قدرت دید مهمترین کار، پایان‌نامه Hall-Holt [۳۶] است که به موانع محدب محدود شده است. کار دیگری نیز توسط Hornus و Puech [۳۹] برای محاسبه چندضلعی قابل دید ناظر نقطه‌ای انجام شده است، ولی این روش چندان کارا نیست و ممکن است وقایع زیادی را پردازش کند که تأثیری در چندضلعی قابل دید نمی‌گذارند. بنابراین چون نتایج موجود برای چنین مسائلی رضایت‌بخش نیست، مسائلی مثل نگه‌داری چندضلعی قابل دید یک نقطه یا پاره‌خط در حضور موانع متحرک در زمان لگاریتمی و تعیین قابل دید بودن دو شیء در حضور موانع متحرک را می‌توان به لیست کارهای آتی اضافه کرد.

۳.۲.۸ سایر مسائل مشابه

علاوه بر مسائل ذکر شده در بالا، مسائل دیگری نیز در هندسه محاسباتی وجود دارند که با زمینه مورد توجه ما ارتباط دارند و می‌توان بسته به نوع ناظر و محیطی که در آن قرار می‌گیرد و نوع دید تعریف شده زیرمسائل متعددی را به‌وجود آورد که برخی از آن‌ها مورد بررسی قرار نگرفته است و یا الگوریتم کارایی برای آن یافت نشده است. این موضوعات که می‌توانند در کنار زمینه اصلی کار مورد مطالعه قرار بگیرند در زیر فهرست می‌شوند.

- بررسی مسأله موزه هنر با پاره‌خط‌های نگهبان:

  مسأله موزه هنر با نگهبان‌های نقطه‌ای، مسأله‌ای است که کار زیادی بر روی آن انجام شده است. ثابت شده است که تعیین تعداد کمینه نگهبان در چندضلعی ساده NP–سخت است ولی در مورد چندضلعی‌های ساده متعامد[۲] هنوز مسأله باز است. به همین جهت شکل‌های مختلفی از مسأله برای چندضلعی‌های متعامد بررسی و راه‌حل‌هایی ارائه شده است. در مورد نگهبان‌های به شکل پاره‌خط، مسأله تا حدی کار شده است. مثلاً Aggarwal [۲] در پایان‌نامه دکترای خود نشان داده است که در چندضلعی‌های متعامد، ممکن است تعداد $\lceil(3n+4)/16\rceil$ پاره‌خط نگهبان لازم باشد تا چندضلعی به‌طور کامل پوشش داده شود. همچنین این تعداد برای پوشش هر چندضلعی متعامد کافی است.

  یکی از مسائلی که می‌توان در این حیطه تعریف کرد تعیین تعداد کمینه پاره‌خط نگهبان برای حالت‌های خاص چندضلعی است. علاوه بر این می‌توان فضای قابل دید پاره‌خط نگهبان را به یک زاویه محدود کرد یا شرایط دیگری را روی دید آن قرار داد که به این ترتیب مسائل جدیدی تعریف می‌شوند که مطالعه بیشتر برای شناسایی خصوصیات آن‌ها لازم خواهد بود.

[۲] Simple orthogonal polygon

- تعمیم الگوریتم‌های قدرت دید ارائه شده به فضای سه بعدی.
- بررسی مسائل قدرت دید با اضافه شدن خاصیت انعکاس به موانع.
- تعقیب اشیاء در چندضلعی‌های حفره‌دار.

پیوست ا

فهرست نمادها

مفهوم	نماد
چندضلعی قابل دید نقطه p	$\mathrm{V}(p)$
ساختار ترکیبیاتی $\mathrm{V}(p)$	$\mathcal{V}(p)$
چندضلعی قابل دید جزئی نقطه p با عبور از i مانع	$\mathrm{V}_i(p)$
ساختار ترکیبیاتی $\mathrm{V}_i(p)$	$\mathcal{V}_i(p)$
گراف قابلیت دید چندضلعی P	$\mathrm{VG}(P)$
بیشینه اشیاء مجموعه S که توسط شعاع‌های ساطع شده از نقطه p قطع می‌شوند	$\tau(p,S)$
$\min_p \tau(p,S)$	$\tau(S)$
$\max_{\forall S:\lvert S\rvert=n} \tau(S)$	$\tau(n)$
شعاع به مبدأ a گذرنده از b	$\vec{ab}$
دوگان نقطه (خط) a	a^*
آرایش مجموعه خطوط L	$\mathcal{A}(L)$
نقشه کوتاه‌ترین مسیر نقطه p	$\mathrm{SPM}(p)$

پیوست ب

واژه‌نامه فارسی به انگلیسی

آرایش ..... arrangement
آرایش پاره‌خط‌ها ..... line segment arrangement
بستار ترایایی ..... transitive closure
برش ..... cutting
بی‌کران ..... unbounded
پادساعت‌گرد ..... counterclockwise
پرتو ..... ray
پرس‌وجو ..... query
پشته ..... stack
پوسته محدب ..... convex hull
پویش ..... sweep
پویش زاویه‌ای ..... angular sweep
پیچش ..... winding
پیچیدگی ..... complexity
پیش‌پردازش ..... preprocessing
پیمایش عمق اول ..... depth first traversal
ترتیب جزئی ..... partial order
ترتیب کلی ..... total order
تقسیم‌بندی قابلیت دید ..... visibility decomposition
توازن بین حافظه و زمان پرس‌وجو ..... space/query-time tradeoff

جایابی نقطه . . . . . point location
جستجوی بازه . . . . . range search
جستجوی بازه چند لایه . . . . . multi-level range search
جستجوی بازه مثلث . . . . . triangle range search
جستجوی بازه نیم‌صفحه . . . . . half-plane range search
چندضلعی حفره‌دار . . . . . polygon with holes
چندضلعی ساده . . . . . simple polygon
چندضلعی قابل دید . . . . . visibility polygon
چندضلعی قابل دید جزئی . . . . . partial visibility polyon
چندضلعی قابل دید ضعیف . . . . . weak visibility polyon
چندضلعی قابل دید کامل . . . . . complete visibility polygon
چندضلعی گون . . . . . splinegon
حافظه . . . . . space
حذف خطوط مخفی . . . . . hidden line elimination
خط حامل . . . . . supporting line
خنثی . . . . . undo
داده‌ساختار ماندگار . . . . . persistent data structure
دالان . . . . . corridor
دوگان . . . . . dual
ذوزنقه‌بندی . . . . . trapezoidal
رأس . . . . . vertex
رأس ساختگی . . . . . constructed vertex
زنجیره چندضلعی‌گون . . . . . polygonal chain
ساختار ترکیبیاتی . . . . . combinatorial structure
ساعت‌گرد . . . . . clockwise
ستاره‌ای . . . . . star-shaped
شلیک پرتو . . . . . ray shooting
فرستنده . . . . . transmitter
قابل دید . . . . . visible
قابلیت دید جزئی . . . . . partial visibility
قابلیت دید ضعیف آلفا . . . . . weak α-visibility

قابلیت دید کامل آلفا ..... complete α-visibility
قابلیت دید میله‌ای k ..... bar k-visibility
قطعه‌های کانونی ..... canonical pieces
قضیه ناحیه ..... zone theorem
قید بحرانی ..... critical constraint
کران‌دار ..... bounded
کنج ..... cone
گراف جهت‌دار ..... directed graph
گراف قابلیت دید ..... visibility graph
گراف قابلیت دید توسعه‌یافته ..... extended visibility graph
گره ..... node
گوه دوگانه ..... double wedge
مثلث‌بندی کانونی ..... canonical triangulation
متعامد یکنوا ..... monotone orthogonal
مجتمع دیداری ..... visibility complex
مجموعه کانونی ..... canonical set
محدب ..... convex
مرتب‌سازی توپولوژیکی ..... topological order
مقعر ..... reflex
منظم ..... regular
موزه هنر ..... art gallery
نقشه کوتاه‌ترین مسیر ..... shortest path map
نقطه انتهایی ..... end-point
نیمه شفاف ..... semi-transparent
واقعه ..... event
هسته ..... kernel
یال ..... edge
یال ساختگی ..... constructed edge
یکنوا ..... monotone

پیوست پ

واژه‌نامه انگلیسی به فارسی

پویش زاویه‌ای ..... angular sweep
آرایش ..... arrangement
موزه هنر ..... art gallery
قابلیت دید میله‌ای k ..... bar k-visibility
کران‌دار ..... bounded
قطعه‌های کانونی ..... canonical pieces
مجموعه کانونی ..... canonical set
مثلث‌بندی کانونی ..... canonical triangulation
ساعت‌گرد ..... clockwise
ساختار ترکیبیاتی ..... combinatorial structure
قابلیت دید کامل آلفا ..... complete α-visibility
چندضلعی قابل دید کامل ..... complete visibility polygon
پیچیدگی ..... complexity
کنج ..... cone
یال ساختگی ..... constructed edge
رأس ساختگی ..... constructed vertex
محدب ..... convex
پوسته محدب ..... convex hull
دالان ..... corridor
پادساعت‌گرد ..... counterclockwise

قید بحرانی ..... critical constraint
برش ..... cutting
پیمایش عمق اول ..... depth first traversal
گراف جهت‌دار ..... directed graph
گوه دوگانه ..... double wedge
دوگان ..... dual
یال ..... edge
نقطه انتهایی ..... end-point
واقعه ..... event
گراف قابلیت دید توسعه‌یافته ..... extended visibility graph
جستجوی بازه نیم‌صفحه ..... half-plane range search
حذف خطوط مخفی ..... hidden line elimination
هسته ..... kernel
آرایش پاره‌خط‌ها ..... line segment arrangement
یکنوا ..... monotone
متعامد یکنوا ..... monotone orthogonal
جستجوی بازه چند لایه ..... multi-level range search
گره ..... node
ترتیب جزئی ..... partial order
قابلیت دید جزئی ..... partial visibility
چندضلعی قابل دید جزئی ..... partial visibility polyon
داده‌ساختار ماندگار ..... persistent data structure
جایابی نقطه ..... point location
چندضلعی حفره‌دار ..... polygon with holes
زنجیره چندضلعی‌گون ..... polygonal chain
پیش‌پردازش ..... preprocessing
پرس‌وجو ..... query
جستجوی بازه ..... range search
پرتو ..... ray
شلیک پرتو ..... ray shooting
مقعر ..... reflex

منظم . . . . . regular
نیمه شفاف . . . . . semi-transparent
نقشه کوتاه‌ترین مسیر . . . . . shortest path map
چندضلعی ساده . . . . . simple polygon
حافظه . . . . . space
توازن بین حافظه و زمان پرس‌وجو . . . . . space/query-time tradeoff
چندضلعی گون . . . . . splinegon
پشته . . . . . stack
ستاره‌ای . . . . . star-shaped
خط حامل . . . . . supporting line
پویش . . . . . sweep
مرتب‌سازی توپولوژیکی . . . . . topological order
ترتیب کلی . . . . . total order
بستار ترایایی . . . . . transitive closure
فرستنده . . . . . transmitter
ذوزنقه‌بندی . . . . . trapezoidal
جستجوی بازه مثلث . . . . . triangle range search
بی‌کران . . . . . unbounded
خنثی . . . . . undo
رأس . . . . . vertex
مجتمع دیداری . . . . . visibility complex
تقسیم‌بندی قابلیت دید . . . . . visibility decomposition
گراف قابلیت دید . . . . . visibility graph
چندضلعی قابل دید . . . . . visibility polygon
قابل دید . . . . . visible
قابلیت دید ضعیف آلفا . . . . . weak α-visibility
چندضلعی قابل دید ضعیف . . . . . weak visibility polyon
پیچش . . . . . winding
قضیه ناحیه . . . . . zone theorem

کتاب‌نامه

[1] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 1–56. American Mathematical Society, Providence, RI, 1999.

[2] A. Aggarwal. *The art gallery problem: Its variations, applications, and algorithmic aspects*. PhD thesis, Department of Computer Science, Johns Hopkins University, Baltimore, MD, 1984.

[3] O. Aichholzer, R. Fabila-Monroy, D. Flores-Peñaloza, T. Hackl, C. Huemer, J. Urrutia, and B. Vogtenhuber. Modem illumination of monotone polygons. In *Abstracts of 25th European Workshop on Computational Geometry*, pages 167–170. Brussels, Belgium, 2009.

[4] B. Aronov, L. J. Guibas, M. Teichmann, and L. Zhang. Visibility queries and maintenance in simple polygons. *Discrete & Computational Geometry*, 27(4):461–483, 2002.

[5] B. Aronov, L. J. Guibas, M. Teichmann, and L. Zhang. Visibility queries and maintenance in simple polygons. *Discrete & Computational Geometry*, 27(4):461–483, 2002.

[6] T. Asano. Efficient algorithms for finding the visibility polygons for a polygonal region with holes. *Transactions of IECE of Japan*, E68:557–559, 1985.

[7] T. Asano, T. Asano, L. J. Guibas, J. Hershberger, and H. Imai. Visibility of disjoint polygons. *Algorithmica*, 1:49–63, 1986.

[8] D. Avis and G. T. Toussaint. An optimal algorithm for determining the visibility of a polygon from an edge. *IEEE Transactions on Computers*, C-30(12):910–1014, 1981.

[9] B. Ballinger, N. Benbernou, P. Bose, M. Damian, E. D. Demaine, V. Dujmović, R. Flatland, F. Hurtado, J. Iacono, A. Lubiw, P. Morin, V. Sacristán, D. Souvaine, and R. Uehara. Coverage with k-transmitters in the presence of obstacles. In *Proceedings of the 4th international conference on Combinatorial optimization and applications - Volume Part II*, COCOA'10, pages 1–15, Berlin, Heidelberg, 2010. Springer-Verlag.

[10] B. K. Bhattacharya, S. K. Ghosh, and T. Shermer. A linear time algorithm to remove winding of a simple polygon. *Computational Geometry: Theory & Applications*, 33(3):165–173, 2006.

[11] P. Bose, A. Lubiw, and J. Munro. Efficient visibility queries in simple polygons. *Computational Geometry: Theory & Applications*, 23(3):313–335, 2002.

[12] T. Chan. Optimal partition trees. *Discrete & Computational Geometry*, 47(4):661–690, 2012.

[13] B. Chazelle. Lower bounds on the complexity of polytope range searching. *Journal of the American Mathematical Society*, 2:637–666, 1989.

[14] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(5):485–524, 1991.

[15] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 9(2):145–158, 1993.

[16] B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990.

[17] B. Chazelle and L. J. Guibas. Visibility and intersection problems in plane geometry. *Discrete & Computational Geometry*, 4:551–581, 1989.

[18] B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25:76–90, 1985.

[19] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete & Computational Geometry*, 2:195–222, 1987.

[20] L. Davis and M. Benedikt. Computational models of space: isovists and isovist fields. *Computer Graphics and Image Processing*, 11:49–72, 1979.

[21] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Theory & Applications*. Springer-Verlag, 3rd edition, 2008.

[22] M. de Berg and O. Schwarzkopf. Cuttings and applications. *International Journal of Computational Geometry and Applications*, 5:343–355, 1995.

[23] A. M. Dean, W. Evans, E. Gethner, J. Laison, M. A. Safari, and W. T. Trotter. Bar k-visibility graphs: Bounds on the number of edges, chromatic number, and thickness. In *Graph Drawing*, volume 3843 of *Lecture Notes in Computer Science*, pages 73–82. Springer Berlin Heidelberg, 2006.

[24] D. P. Dobkin and D. L. Souvaine. Computational geometry in a curved world. *Algorithmica*, 5:421–457, 1990.

[25] H. ElGindy. Efficient algorithms for computing the weak visibility polygon from an edge. Technical Report MS-CIS-86-04, University of Pennsylvania, Philadelphia, USA, 1986.

[26] H. ElGindy and D. Avis. A linear algorithm for computing the visibility polygon from a point. *Journal of Algorithms*, 2:186–197, 1981.

[27] R. Fabila-Monroy, A. R Vargas, and J. Urrutia. On modem illumination problems. In *XIII Spanish Workshop on Computational Geometry*, pages 9–19, 2009.

[28] S. Felsner and M. Massow. Parameters of bar k-visibility graphs. *Journal of Graph Algorithms and Applications*, 12(1):5–27, 2008.

[29] R. Fulek, A. Holmsen, and J. Pach. Intersecting convex sets by rays. *Discrete & Computational Geometry*, 42(3):343–358, 2009.

[30] H. N. Gabow and R. E. Tarjan. A linear time algorithm for a special case of disjoint set union. *Journal of Computer Systems and Science*, 30:209–221, 1985.

[31] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry: Theory & Applications*, 5:165–185, 1995.

[32] M. Ghodsi, A. Maheshwari, M. Nouri, J.-R. Sack, and H. Zarrabi-Zadeh. α-visibility. In *13th Scandinavian Symposium and Workshops on Algorithm Theory*, pages 1–12, 2012.

[33] S. K. Ghosh and D. M. Mount. An output sensitive algorithm for computing visibility graphs. In *Proceedings 28th IEEE Symposium on the Foundations of Computer Science*, pages 11–19, 1987.

[34] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[35] L. J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem. *SIAM Journal on Computing*, 26(4):1120–1138, August 1997.

[36] O. A. Hall-Holt. *Kinetic visibility*. PhD thesis, Stanford University, Stanford, CA, USA, 2002.

[37] S. G. Hartke, J. Vandenbussche, and P. Wenger. Further results on bar k-visibility graphs. *SIAM Journal on Discrete Mathematics*, 21(2):523–531, April 2007.

[38] P. J. Heffernan and J. S. B. Mitchell. An optimal algorithm for computing visibility in the plane. *SIAM Journal on Computing*, 24(1):184–201, 1995.

[39] S. Hornus and C. Puech. A simple kinetic visibility polygon. In *18th European Workshop on Computational Geometry*, pages 27–30, 2002. Warsaw Uni.

[40] R. Inkulu and S. Kapoor. Visibility queries in a polygonal region. *Computational Geometry: Theory & Applications*, 42(9):852–864, 2009.

[41] B. Joe. On the correctness of a linear-time visibility polygon algorithm. *International Journal of Computer Mathematics*, 32:155–172, 1990.

[42] B. Joe and R. B. Simpson. Correction to Lee's visibility polygon algorithm. *BIT*, 27:458–473, 1987.

[43] M. Keil, D. M. Mount, and S. K. Wismath. Visibility stabs and depth-first spiralling on line segments in output sensitive time. *International Journal of Computational Geometry and Applications*, 10(5):535–552, 2000.

[44] D. T. Lee. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing*, 22:207–221, 1983.

[45] D. T. Lee and A. K. Lin. Computing the visibility polygon from an edge. *Computer Vision, Graphics, and Image Processing*, 34:1–19, 1986.

[46] Ch.-Ts. Liu and T.-Y. Hung. Method of building a locating service for a wireless network environment. patent no. 7203504, April 2007. www.freepatentsonline.com/7203504.html.

[47] A. M. Martins. *Geometric optimization on visibility problems: metaheuristic and exact solutions*. PhD thesis, Doutoramento em Matemática, Universidade de Aveiro, 2009.

[48] J. Matoušek. Cutting hyperplane arrangements. *Discrete & Computational Geometry*, 6:385–406, 1991.

[49] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete & Computational Geometry*, 10(2):157–182, 1993.

[50] E. A. Melissaratos and D. L. Souvaine. Shortest paths help solve geometric optimization problems in planar regions. *SIAM Journal on Computing*, 21(4):601–638, 1992.

[51] J. I. Munro and V. Raman. Succinct representation of balanced parentheses and static trees. *SIAM Journal on Computing*, 31(3):762–776, 2001.

[52] M. Nouri and M. Ghodsi. Weak visibility of two segments in planar polygonal scenes. Submitted to Information and Computation.

[53] M. Nouri and M. Ghodsi. Space–query-time tradeoff for computing the visibility polygon. In *Proceedings of the Third International Workshop on Frontiers in Algorithmics*, pages 120–131, 2009.

[54] M. Nouri and M. Ghodsi. Partial visibility polygon with semi-transparent objects. In *Abstracts of the 26th European Workshop on Computational Geometry*, pages 233–236, 2010.

[55] M. Nouri, A. Zarei, and M. Ghodsi. Weak visibility of two objects in planar polygonal scenes. In *Proceedings of the 2007 International Conference on Computational Science and its Applications*, pages 68–81, 2007.

[56] M. Nouri-Baygi and M. Ghodsi. Space/query-time tradeoff for computing the visibility polygon. *Computational Geometry: Theory & Applications*, 46(3):371–381, 2013.

[57] M. Pocchiola and G. Vegter. The visibility complex. *International Journal of Computational Geometry and Applications*, 6(3):279–308, 1996.

[58] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37(2):130–143, 1988.

[59] N. Sarnak and R. E. Tarjan. Planar point location using persistent search trees. *Communications of ACM*, 29(7):669–679, July 1986.

[60] M. S. Shamaee, A. Mohades, and M. Eskandari. The illumination of polygonal regions with modems. In *Proceedings of the Second Conference on Contemporary Issues in Computer and Information Science*, pages 68–71, 2011.

[61] J. Spencer. *Ten lectures on the probabilistic method*. SIAM, 1987.

[62] S. Suri and J. O'Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. In *Proceedings of the 2nd Annual ACM Symposium on Computational Geometry*, pages 14–23, 1986.

[63] G. Vegter. The visibility diagram: a data structure for visibility problems and motion planning. In *Proceedings of the 2nd Scandinavian Workshop on Algorithm Theory*, volume 447 of *Lecture Notes in Computer Science*, pages 97–110. Springer-Verlag, 1990.

[64] S. K. Wismath. Computing the full visibility graph of a set of line segments. *Information Processing Letters*, 42(5):257–261, 1992.

[65] A. Zarei and M. Ghodsi. Query point visibility computation in polygons with holes. *Computational Geometry: Theory & Applications*, 39(2):78–90, 2008.

Abstract

The study of visibility is at least 100 years old, when in 1913 Brunn proved a theorem about the kernel of a set. By now, visibility has become one of the most studied notions in computational geometry. The reasons are two-fold: 1) such problems arise naturally in areas where computational geometry tools and algorithms find applications including: computer graphics, robotics, motion planning, geographic information systems, computer games, computer-aided architecture, and pattern recognition; and where 2) their solutions are required, or serve as building blocks in the development of solutions to other problems, such as shortest paths or motion planning problems.

Many natural problem instances arise and have been extensively studied in two and higher dimensions. In this thesis, we revisited some of the problems. Our target was to achieve much better results for these problems, comparing to the old results. Some other problems we studied were not studied before. The main problems we focused on are as follows: the problem of detecting weak visibility between two objects, the problem of computing the visibility polygon of a query point, the problem of computing the partial visibility polygon of a point and problems related to α-visibility.

Keywords: Visibility problems, Visibility polygon, Visibility detection, Partial visibility, α-visibility.

New Results on Different Types of Visibility Problems

by

Mostafa Nouri Baygi

Submitted in Partial Fulfillment
of the Requirements
for the Degree of
Ph.D
in
Computer Engineering (Software)

Under supervision of
Dr. Mohammad Ghodsi

January, 2013

Computer Engineering Department
Sharif University of Technology
Tehran