یافتن کوتاه‌ترین مسیر برای مشاهده یک پاره‌خط در یک ناحیه چندضلعی

الهه شبان [1]، مصطفی نوری بایگی [2]

[1] دانشجوی کارشناسی ارشد، دانشکده مهندسی کامپیوتر، دانشگاه فردوسی مشهد، el.shaban@mail.um.ac.ir
[2] استادیار، دانشکده مهندسی کامپیوتر، دانشگاه فردوسی مشهد، nouribaygi@um.ac.ir

چکیده

پیدا کردن کوتاه‌ترین مسیر برای مشاهده یک شیء یک مسأله پرکاربرد در هندسه محاسباتی است. از جمله کاربردهای آن می‌توان به وضعیتی که دیدن یا دیده شدن توسط شیء هدف اهمیت دارد اشاره کرد. به عنوان مثال هنگامی که بخواهیم با شیء هدف ارتباط برقرار کنیم یا آن را بازرسی کنیم؛ با این شرط که نحوه ارتباط با شیء هدف به صورت خط دید باشد. نقطه مبدأ s را در یک ناحیه چندضلعی P با h-1 مانع در نظر بگیرید. می‌خواهیم با انجام پیش‌پردازش بر روی ورودی، کوتاه‌ترین مسیر از نقطه s به نقطه دلخواهی در P را پیدا کنیم؛ به طوری که پاره‌خط دلخواه l از آن نقطه قابل دیدن باشد.

برای حل این مسأله در این مقاله ما دو راه حل ارائه کردیم. در راه حل نخست با صرف زمان پیش‌پردازش $O(n^{4+\varepsilon})$ مسأله در زمان $O(nh)$ قابل حل خواهد بود. در راه حل پیشنهادی دوم با افزایش زمان پیش‌پردازش به $O(n^8)$ توانستیم مسأله را در زمان $O(\log n)$ حل کنیم.

کلمات کلیدی

کوتاه‌ترین مسیر، دیدن، ناحیه چندضلعی

۱- مقدمه

پیدا کردن کوتاه‌ترین مسیر یک مسأله قدیمی در حوزه هندسه محاسباتی بوده و الگوریتم‌های کارایی برای آن ارائه شده است[1].

در یک نمونه از این مسائل به دنبال یافتن مسیری از شیء مبدأ به نقطه‌ای هستیم که از آن نقطه بتوان شیء هدف را دید. در نسخه پرس‌وجو از این مسأله در یک ناحیه چندضلعی[1] در حالتی که شیء هدف نقطه باشد، توسط آرکین و همکاران مورد مطالعه قرار گرفته است[2]. در راه حل پیشنهاد شده با صرف زمان پیش‌پردازش $O(n^2 2^{\alpha(n)}\log n)$ می‌توان مسأله را در زمان $O(k\log^2 n)$ حل کرد، به طوری که k اندازه چندضلعی پدیداری[2] نقطه هدف است. همچنین آن‌ها نشان دادند که با ساخت سریع‌ترین نقشه دید[3] با اندازه $O(n^7)$ در زمان $O(n^8\log n)$ می‌توان این مسأله را برای هر نقطه دلخواه در زمان $O(\log n)$ حل کرد.

ونگ نیز با ارائه ساختمان داده‌ای با اندازه $O(n\log h + h^2)$ مسأله را در زمان $O(h\log h\log n)$ حل کرد [3]. این مسأله در چندضلعی ساده با ساخت ساختمان‌داده با اندازه $O(n)$ در زمان $O(\log n)$ قابل حل است [4].

در این مقاله ما به مطالعه مسأله در حالتی پرداختیم که شیء هدف پاره‌خط باشد. این مسأله تاکنون در ناحیه چندضلعی مورد مطالعه قرار نگرفته است. اگر پاره‌خط در یک چندضلعی ساده قرار گرفته باشد، می‌توان از روشی مشابه روش ارائه شده برای حل مسأله در حالتی که شیء هدف نقطه باشد استفاده کرد[7]. در این حالت با صرف زمان $O(n^3 \log n)$ و فضای $O(n^3)$ می‌توان مسأله را در زمان $O(n\log n)$ حل کرد [12].

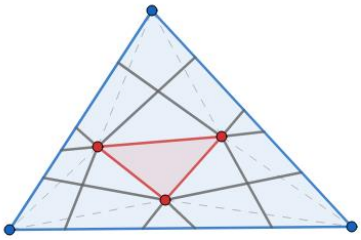
۱-۱- تعریف مسأله

فرض کنید P یک ناحیه چندضلعی با h-1 مانع و n رأس باشد و نقطه مبدأ s داده‌شده باشد. هدف این است که با پیش‌پردازش ورودی بتوان کوتاه‌ترین مسیر از نقطه s به نقطه‌ای که بتوان از آن پاره‌خط دلخواه l را در زمان پرس‌وجو مشاهده کرد، را گزارش کرد. دو نقطه p و q زمانی برای یکدیگر قابل دیدن هستند که پاره‌خط pq کاملاً داخل P قرار بگیرد.

در راه‌حل پیشنهادی یک الگوریتم با زمان پرس‌وجو $O(nh)$ ارائه می‌کنیم. سپس در ادامه به توضیح روشی می‌پردازیم که با افزایش زمان پیش‌پردازش زمان پرس‌وجو را کاهش می‌دهد. ساختار ادامه مقاله بدین شرح است: در بخش ۲ به توضیح مفاهیم پرداختیم سپس در بخش ۳ دو الگوریتم ارائه شده برای حل مسأله را توضیح می‌دهیم. بخش ۴ به نتیجه‌گیری اختصاص دارد.

۲- مفاهیم

۲-۱- تقسیم‌بندی پدیداری[۴]



شکل (۱): تقسیم‌بندی پدیداری

اگر p را یک نقطه در ناحیه چندضلعی P در نظر بگیریم، توالی پدیداری نقطه p ترتیبی از رئوس و یال‌های P است که برای P قابل دیدن است. تقسیم‌بندی P به سلول‌هایی به طوری که کلیه نقاط درون یک سلول دارای توالی پدیداری یکسانی باشند، تقسیم‌بندی پدیداری نامیده می‌شود. توالی دید هر دو سلول مجاور در یک رأس اختلاف دارند و مرز این دو سلول، محدودیت بحرانی[۵] نامیده می‌شود شکل (۱).

تقسیم‌بندی دید ناحیه چندضلعی دارای $O(n^2)$ محدودیت بحرانی و $O(n^4)$ سلول است [5].

۲-۲- جست‌وجوی محدوده[۶]

در مسأله جست‌وجو محدوده، مجموعه نقاط p داده شده است. به ازای یک محدوده Q در فضا به دنبال اطلاعاتی در مورد زیرمجموعه‌ای از p که داخل Q قرار می‌گیرد هستیم. این اطلاعات می‌تواند شامل گزارش نقاطی که درون محدوده قرار می‌گیرد و یا شمارش آن‌ها باشد. در این مقاله ما p را مجموعه نقاط در صفحه و Q را نیم‌صفحه در نظر می‌گیریم.

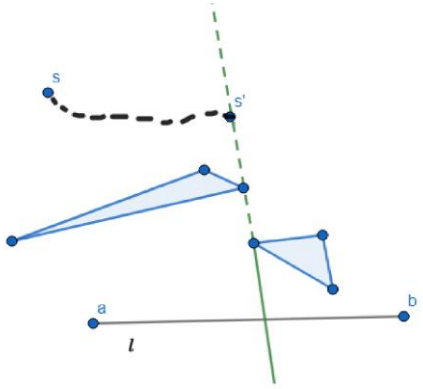
با استفاده از یک ساختمان داده با اندازه $O(n^{2+\varepsilon})$ که در زمان مشابه قابل ساخت است، می‌توان هر پرس‌وجویی را در زمان $O(\log n)$ پاسخ داد[9]. نتایج جست‌وجوی محدوده به صورت مجموعه مجزایی از زیرمجموعه‌های کانونی[۷] است. می‌توان این مجموعه‌ها را مورد پردازش قرار داد به طوری که؛ چندین مرتبه جست‌وجوی محدوده بر روی مجموعه نقاط اولیه داده‌شده، قابل انجام باشد[10]. در جست‌وجوی محدوده به صورت چندسطحی زمان پرس‌وجو به ازای هر سطح به صورت ضریب لگاریتمی افزایش می‌یابد.

۳- راه حل

در این قسمت از مقاله قصد داریم توضیح دهیم که چطور با انجام پیش‌پردازش بر روی ورودی می‌توان جواب مسأله را برای هر پاره‌خط دلخواه l حل کرد. در ابتدا نشان می‌دهیم که علاوه بر پیدا کردن کوتاه‌ترین مسیر برای مشاهده نقاط انتهایی پاره‌خط، کافی است کوتاه‌ترین مسیر تا امتداد محدودیت‌های بحرانی که پاره‌خط l را قطع می‌کنند را به دست آورد و از میان مسیرهای به دست آمده کوتاه‌ترین را گزارش کرد.

لم۱. با داشتن تقسیم‌بندی پدیداری و با شروع از نقطه مبدأ s نزدیک‌ترین نقطه که از آن می‌توان پاره‌خط دلخواه l را مشاهده کرد یا روی امتداد محدودیت بحرانی که با پاره‌خط l برخورد می‌کند قرار دارد و یا نقطه‌ای است که از آن، یکی از دو سر l قابل دیدن است.

اثبات. فرض کنید با حرکت از نقطه شروع s، s' نزدیک‌ترین جایی باشد که یک نقطه از پاره‌خط l قابل دیدن باشد. اگر s' حداقل یکی از دو سر l را ببیند، حکم ثابت می‌شود. در غیر این صورت فرض کنید s' هیچ یک از دو سر l را نمی‌بیند. اگر روی مسیر s تا s' کمی به عقب برگردیم به دلیل خاصیت s'، دیگر l قابل مشاهده نیست. اما چون s' هیچ یک از دو سر l را نمی‌بیند، این امر فقط در صورتی امکان‌پذیر است که s' روی امتداد محدودیت بحرانی که l را قطع می‌کند، قرار داشته باشد شکل(۲).



شکل(۲) اثبات لم۱. نقطه s' بر روی امتداد یک محدودیت بحرانی که پاره‌خط l را قطع کرده است قرار دارد.

۳-۱- راه حل اول

در این قسمت قصد داریم مشابه روشی که در مقاله[6] آمده است، روشی را برای حل مسأله ارائه کنیم.

در زمان پیش‌پردازش بعد از ساخت تقسیم‌بندی پدیداری، کوتاه‌ترین مسیر تا همه محدودیت‌های بحرانی را حساب می‌کنیم[2]. در زمان پرس‌وجو بعد از دریافت پاره‌خط l کوتاه‌ترین مسیر برای مشاهده یک سر پاره‌خط را حساب می‌کنیم [3]. به سمت دیگر l حرکت می‌کنیم. محدودیت‌های بحرانی که با پاره‌خط l برخورد دارند را به دست می‌آوریم. کوتاه‌ترین مسیرها تا این پاره‌خطها را در نظر می‌گیریم. کوتاه‌ترین مسیر برای مشاهده نقطه انتهایی دیگر l را محاسبه می‌کنیم. از میان مسیرهای به دست آمده کوتاه‌ترین را انتخاب می‌کنیم. در ضمن با حرکت از یک نقطه انتهایی l به سمت نقطه انتهایی دیگر آن، آن دسته از محدودیت‌های بحرانی برای ما اهمیت دارد که عبور از آن‌ها در یک جهت، باعث می‌شود یک رأس جدید دیده شود.

به منظور به دست آوردن محدودیت‌های بحرانی که با پاره‌خط l برخورد دارند از جست‌وجوی محدوده استفاده می‌کنیم. ابتدا خط حامل پاره‌خط l را در نظر می‌گیریم و آن را d می‌نامیم. آن دسته از محدودیت‌های بحرانی، خط d را قطع می‌کنند که نقطه انتهایی سمت چپ آن‌ها پایین خط d و نقطه انتهایی سمت راست آن‌ها بالای خط d واقع‌شده باشد و یا نقطه انتهایی سمت چپ آن‌ها بالای خط d و نقطه انتهایی سمت راست آن‌ها پایین خط d واقع‌شده باشد. برای هر دو دسته بایستی فرایندی که در ادامه شرح داده می‌شود تکرار شود.

مجموعه محدودیت‌های بحرانی حاصل برای هر دسته را می‌توان با دو سطح جستجوی محدوده نیم‌صفحه به‌دست آورد. با اضافه کردن یک سطح دیگر جستجوی محدوده می‌توان محدودیت‌های بحرانی که با عبور از آن‌ها یک رأس حذف می‌شود را از مجموعه حاصل حذف کرد.

سپس از مجموعه پاره‌خط‌های به دست آمده از مرحله قبل، آن دسته از پاره‌خط‌هایی را انتخاب می‌کنیم که خط حامل آن‌ها پاره‌خط l را قطع می‌کند. برای انجام این کار ابتدا پاره‌خط l و خطوط حامل به دست آمده از مرحله قبل را به فضای دوگان می‌بریم. در فضای دوگان با انجام جست‌وجوی محدوده پاره‌خط‌های مورد نظر را می‌توان یافت.

۳-۱-۱- تحلیل الگوریتم

کوتاه‌ترین مسیر برای مشاهده یک نقطه را می‌توان در زمان $O(hlognlogh)$ به دست آورد[3].

جست‌وجوی پنج سطحی شرح داده‌شده با صرف زمان پیش‌پردازش $O(k^{2+\varepsilon})$ و زمان پرس‌وجوی $O(log^5 n)$ قابل انجام است به طوری که k تعداد کل محدودیت‌های بحرانی است. از آنجایی که تعداد محدودیت‌های بحرانی در یک ناحیه چندضلعی $O(n^2)$ است، زمان پیش‌پردازش از مرتبه $O(n^{4+\varepsilon})$ خواهد بود. کوتاه‌ترین مسیر تا یک محدودیت بحرانی را می‌توان در زمان $O(logn)$ به دست آورد[2]. پس زمان پیش‌پردازش از مرتبه $O(n^{4+\varepsilon})$ خواهد بود. پس اگر پاره‌خط l، m محدودیت بحرانی را قطع کند، مسأله با زمان پیش‌پردازش $O(n^4)$ و زمان پرس‌وجو $O(m+log^5 n)$ قابل حل خواهد بود.

۳-۲- راه حل دوم

تعداد محدودیت‌های بحرانی که توسط پاره‌خط مورد پرس‌وجو قطع می‌شود، m، در بدترین حالت می‌تواند از پیچیدگی $O(nh)$ باشد[5]. بنابراین زمان پرس‌وجو می‌تواند زیاد باشد. به این دلیل در ادامه مقاله قصد داریم با افزایش زمان پیش‌پردازش زمان پرس‌وجو را کاهش دهیم.

بدین منظور در زمان پیش‌پردازش، کلیه حالت‌هایی که خط حامل پاره‌خط l محدودیت‌های بحرانی را قطع می‌کند را به کلاس‌هایی افراز می‌کنیم؛ به طوری که هر کلاس بیانگر مجموعه یکسانی از محدودیت‌های بحرانی قطع‌شده با ترتیب یکسانی است. کافی است در زمان پیش‌پردازش برای هر کلاس جواب را محاسبه کنیم و در زمان پرس‌وجو با مشخص کردن کلاسی که خط حامل پاره‌خط داده‌شده به آن تعلق دارد، می‌توان جواب را گزارش کرد.

هر پاره‌خط را در فضای دوگان با یک گوه دوتایی[۸] می‌توان نشان داد شکل(۳). همه محدودیت‌های بحرانی را به فضای دوگان می‌بریم. این کار باعث می‌شود، فضای دوگان به سلول‌هایی تقسیم شود. خط حامل پاره‌خط l را در نظر می‌گیریم. در فضای دوگان این خط تبدیل به نقطه می‌شود. این نقطه درون گوه‌های دوتایی قرار می‌گیرد که در واقع دوگان

محدودیت‌های بحرانی می‌باشند که توسط خط حامل پاره‌خط l قطع‌شده است. در واقع هر سلول حاصل از تقسیم‌بندی گوه‌های دوتایی نشان دهنده نقاطی است که خطوط مربوط به آن‌ها در فضای اولیه مجموعه مشخصی از محدودیت‌های بحرانی را قطع کرده است.

برای هر سلول زمانی ترتیب محدودیت‌های بحرانی تغییر می‌کند که از خط متصل‌کننده دو مرکز[9] گوه دوتایی آن‌ها عبور کند و همچنین مرکز هر یک از گوه‌های دوتایی بایستی در گوه دوتایی دیگری قرار داشته باشد. زیرا زمانی ترتیب محدودیت‌های بحرانی عوض می‌شود که در فضای اولیه محدودیت‌های بحرانی مرتبط با این دو گوه دوتایی یکدیگر را قطع کنند. در نتیجه این خطوط را هم به تقسیم‌بندی اضافه می‌کنیم. در تقسیم‌بندی حاصل، هر سلول بیانگر کلاسی است که در آن خط حامل پاره‌خط l با ترتیب مشخصی مجموعه‌ای از محدودیت‌های بحرانی را قطع می‌کند.

محدودیت‌های بحرانی را برچسب‌گذاری می‌کنیم. برای هر سلول یک درخت جستجوی باینری می‌سازیم به طوری که برگ‌ها به ترتیب محدودیت‌های بحرانی قطع‌شده توسط خط حامل پاره‌خط l در درخت قرار دارند. در برگ‌های این درخت، برچسب مربوط به هر محدودیت بحرانی و فاصله آن تا نقطه s ذخیره شده است. در ریشه و هر گره میانی درخت برچسب محدودیت بحرانی که دارای کمترین فاصله تا s در میان برگ‌های زیر درخت آن است، برچسب سمت چپ‌ترین محدودیت بحرانی زیر درخت چپ آن و برچسب سمت راست‌ترین محدودیت بحرانی زیر درخت سمت راست ذخیره‌شده است.

برای یک سلول، این درخت را می‌توان با صرف زمان و فضای $O(nh)$ ساخت؛ ولی از آنجایی که حداکثر اختلاف بین هر دو سلول مجاور در یک محدودیت بحرانی می‌باشد، می‌توان با یک ساختمان داده ماندگار[10] [8] با صرف حافظه $O(1)$ و زمان $O(\log nh)$ برای هر سلول، درخت را برای آن تشکیل داد[11].

در زمان پرس‌وجو با دریافت پاره‌خط l، ابتدا خط حامل آن را در نظر می‌گیریم. دوگان این خط یک نقطه است. به وسیله الگوریتم مکان‌یابی نقطه، مشخص می‌کنیم که در کدام سلول بایستی جستجو را ادامه دهیم.

بعد از یافتن سلول با شروع از ریشه درخت ذخیره‌شده در آن، بررسی می‌کنیم که آیا پاره‌خط l یک یا چند محدودیت بحرانی را قطع می‌کند یا خیر. این کار را می‌توان با مقایسه موقعیت محل برخورد سمت چپ‌ترین و سمت راست‌ترین محدودیت بحرانی با خط حامل پاره‌خط l و موقعیت دو سر l انجام داد. این کار در زمان $O(1)$ ممکن است.

بر اساس نتیجه این مقایسه، جستجو را در یک یا هر دو زیر درخت ریشه ادامه می‌دهیم و این فرایند را برای ریشه آن زیر درخت به صورت بازگشتی تکرار می‌کنیم. ممکن است حین جستجو به این نتیجه برسیم که پاره‌خط l تمام محدودیت‌های بحرانی یک زیر درخت را قطع می‌کند. در این صورت برچسب ذخیره‌شده در ریشه این زیر درخت کاندیدای جواب است. اگر پاره‌خط l هیچ محدودیت بحرانی متعلق به آن زیر درخت را قطع نکند نیازی به ادامه جستجو در آن زیر درخت نیست.

در صورتی که در انتهای جستجو به برگ رسیدیم، محدودیت بحرانی ذخیره‌شده در آن را به عنوان کاندیدای جواب در نظر می‌گیریم. همچنین کلیه‌ی زیر درخت‌هایی که به عنوان جواب در نظر گرفتیم محدودیت بحرانی ذخیره‌شده در ریشه این زیر درخت‌ها نیز کاندیدای جواب است. به طور مثال در شکل (۴-الف) برای یک سلول دلخواه برچسب مجموعه محدودیت‌های بحرانی و فاصله آن‌ها به ترتیب از چپ به راست a=10، b=6، c=7، d=25، e=16، f=4، g=25، h=11 است. پاره‌خط l که با رنگ آبی مشخص‌شده است، خطوط b، c، d، e و f را قطع می‌کند. در شکل(۴-ب) درخت مرتبط با آن رسم شده است. با شروع از ریشه جستجو در یک یا دو زیر درخت از گره میانی که با رنگ سبز نشان داده‌شده است ادامه پیدا می‌کند. محدودیت‌های بحرانی ذخیره‌شده در گره‌های مشخص‌شده با رنگ قرمز، کاندیدای جواب هستند.

از بین محدودیت‌های بحرانی که کاندیدای جواب هستند، محدودیت بحرانی با کمترین فاصله را انتخاب می‌کنیم و کوتاه‌ترین مسیر تا آن را محاسبه می‌کنیم[2] . سپس مسأله پیدا کردن کوتاه‌ترین مسیر برای مشاهده نقطه را برای هر دو نقطه انتهایی پاره‌خط حساب می‌کنیم. از بین این سه مسیر، مسیر با کمترین فاصله را انتخاب کرده و مسیر مربوطه را گزارش می‌کنیم.

۳-۲-۱- تحلیل الگوریتم

از آنجایی که تعداد محدودیت‌های بحرانی برابر با $O(n^2)$ است، پس تعداد گوه‌های دوتایی $O(n^2)$ است. تعداد خطوط حاصل از وصل کردن مرکزهای گوه‌های دوتایی حداکثر $O(n^4)$ است. پس تعداد سلول‌های که توسط این خطوط ایجاد می‌شود، برابر با $O(n^8)$ است که با صرف زمان $O(n^8 \log nh)$ و حافظه $O(n^8)$ قابل ساخت خواهد بود.

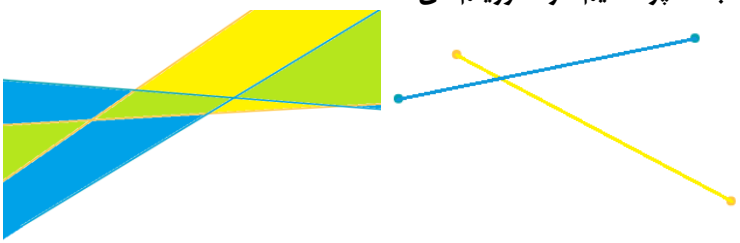
تشخیص اینکه نقطه دوگان خط حامل پاره‌خط l در کدام سلول قرار می‌گیرد، با استفاده از مکان‌یابی نقطه در زمان $O(\log n)$ ممکن است. جست‌وجو در درخت مربوط به هر سلول نیز در زمان لگاریتمی قابل انجام است؛ چراکه تعداد کل

زیر درخت‌هایی که ریشه آن‌ها کاندیدای جواب است حداکثر از مرتبه ارتفاع درخت است. با توجه به اینکه پیدا کردن کوتاه‌ترین مسیر تا نقطه‌ای که یک سر پاره‌خط را بتوان رؤیت کرد در زمان $O(\log n)$ قابل انجام است[2]، زمان پرس‌وجو از همین مرتبه خواهد بود.

ارائه‌شده با افزایش زمان پیش‌پردازش نشان دادیم که می‌توان زمان پرس‌وجو را کاهش داد. به دلیل زیاد بود زمان پیش‌پردازش راه حل دوم، می‌توان به دنبال روشی بود که این زمان را کاهش دهد.

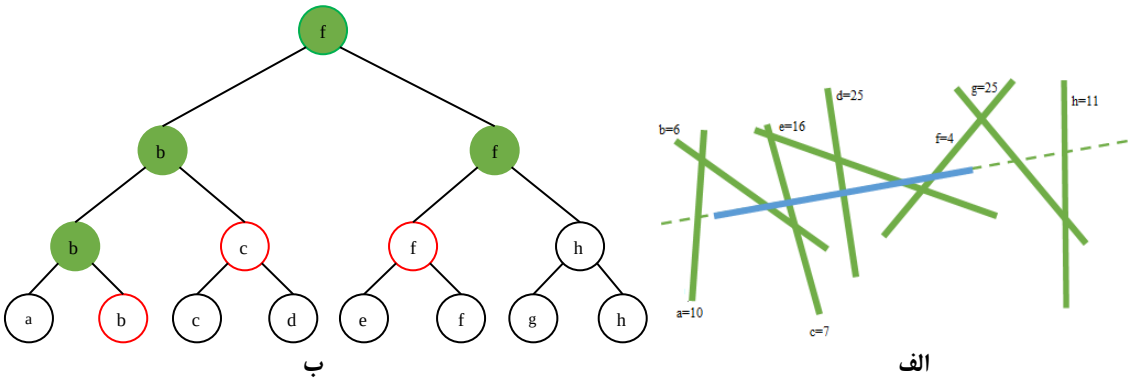
۴- نتیجه‌گیری

در این مقاله به مطالعه مسأله سریع‌ترین پرس‌وجو برای دیدن، در حالتی که شیء هدف پاره‌خط باشد پرداختیم. در الگوریتم‌های



ب الف

شکل (۳) : الف - دو پاره‌خط در فضای اولیه رسم شده است. ب- گوه دوتایی مربوط به هر پاره‌خط رسم شده است.



ب الف

شکل(۴): الف-یک نمونه برخورد پاره‌خط با محدودیت‌های بحرانی. ب-درخت متناظر با این برخورد.

مراجع

[1] Toth, Csaba D., Joseph O'Rourke, and Jacob E. Goodman, eds. *Handbook of discrete and computational geometry*. CRC press, 2017.

[2] Arkin, E. M., Efrat, A., Knauer, C., Mitchell, J. S., Polishchuk, V., Rote, G., ... & Talvitie, T. (2016). *Shortest path to a segment and quickest visibility queries*. Journal of Computational Geometry, 7(2), 77-100.

[3] Wang, H. (2019). *Quickest visibility queries in polygonal domains*. Discrete & Computational Geometry, 62(2), 374-432.

[4] Knauer, C., & Rote, G. (2005). *Shortest inspection-path queries in simple polygons*.

[5] King, J. (2013). *Fast vertex guarding for polygons with and without holes*. Computational Geometry, 46(3), 219-231.

[6] Bygi MN, Daneshpajouh S, Alipour S, Ghodsi M. *Weak visibility counting in simple polygons*. Journal of Computational and Applied Mathematics. 2015 Nov 1;288:215-22.

[7] Khosravi, R. and Ghodsi, M., 2005, March. *The fastest way to view a query point in simple polygons*. In *EuroCG* (pp. 187-190).

[8] Mehta, D. P., & Sahni, S. (2004). *Handbook of data structures and applications.* Chapman and Hall/CRC.

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

[10] Dobkin, D.P. and Edelsbrunner, H., 1987. *Space searching for intersecting objects*. Journal of Algorithms, 8(3), pp.348-361.

[11] Aronov, Boris, et al. "*Visibility queries and maintenance in simple polygons*." *Discret. Comput. Geom.* 27.4 (2002): 461-483.

[12] Bygi, Mojtaba Nouri, and Mohammad Ghodsi. "*Weak Visibility Queries in Simple Polygons*." *CCCG*. 2011.

پانویس‌ها

[1] Polygonal domain
[2] Visibility polygon
[3] Quickest visibility map
[4] Visibility decomposition
[5] Critical constraint
[6] Range search
[7] Canonical subsets
[8] Double wedge
[9] Center
[10] Persistent data structure