طراحی هندسی کامپیوتری صفحات ۱ ـ ۳۰

پویا آقاحسینی

۲۹ آذر ۱۳۹۷

فهرست مطالب

پیش‌گفتار ۹

۱ مقدمه ۱۱

نوع داده مجرد ۱۱

مسئله محاسباتی ۱۲

حل مسئله کامپیوتری ۱۳

مدل‌سازی ۱۳

مراحل حل یک مسئله ۱۳

آلگوریتم ۱۴

چند سؤال و جواب اولیه ۱۴

طراحی آلگوریتم‌ها ۱۵

آنالیز یک آلگوریتم ۱۶

مقایسه کران بالا و کران پایین ۱۷

اعتبارسنجی آلگوریتم ۱۷

۲ ساختمان داده‌های هندسی ۱۹

عملگرهای روی بردارها ۱۹

نمونه‌هایي از اشیاء هندسي ۲۰

نمونه‌هايي از عملگرهاي هندسي ۲۱

نکاتی راجع‌به فاصله ۲۱

۳ روش افزایشی ۲۳

مقدمه ۲۳

مرتب‌سازی درجی ۲۴

ساختن چندضلعي ستاره شکل ۲۶

هسته چندضلعی ۲۶

تعریف چندضلعی ۲۷

رویت پذیری ۲۸

نگارخانه هنر ۲۹

تمرین ۳۳

پوش محدب ۳۴

فهرست تصاویر

۱.۲ شکل ۱ . . . . . . ۲۰
۲.۲ سمت چپ، آنچه ما مي‌پنداريم و سمت راست، آنچه الگوريتم مي‌بيند. ۲۱
۳.۲ فاصله اقليدسي در فضاي ۱، ۲، ۳ و ۴ بعدي ۲۲
۴.۲ متر منهتن ۲۲

۱.۳ يک مرکز اطلاعات ۲۷
۲.۳ شکل سمت چپ، دسته¬بندي چندضلعي¬ها به چندضلعي¬هاي ساده و غير ساده. شکل سمت راست: دسته بندي چندضلعي ها به چندضلعي محدب و غير محدب ۲۸
۳.۳ : نقطه A براي نقطه B قابل رؤيت نيست يعني نقاط A و B نمي‌توانند يکديگر را ببينند. نقطه C براي D قابل رؤيت است. نقطه E براي نقطه F به وضوح قابل رؤيت است. ۳۰
۴.۳ هسته در چندضلعي‌هاي مختلف به رنگ تيره نشان داده شده است. (الف) هسته در يک چندضلعي محدب. (ب) هسته در يک چندضلعي پروانه‌اي شکل. (ج) هسته در يک چندضلعي ستاره‌اي شکل که پروانه‌اي شکل نمي‌باشد. (د) يک چندضلعي غير ستاره‌اي شکل که هسته آن تهي است. ۳۰
۵.۳ اجراي الگوريتم چندضلعي ستاره‌اي شکل براي مجموعه‌اي از نقاط ۳۱
۶.۳ اجراي مراحل الگوريتم ساخت چندضلعي ستاره‌اي شکل ۳۲
۷.۳ رويت‌پذيري خاصيت تعدي ندارد. ۳۳

فهرست جداول

پیش‌گفتار

هندسه محاسباتي يا به تعبير فرانسوی‌ها هندسه آلگوريتمي [1] از دهه هفتم قرن نوزدهم به صورت يک رشته تحقيقاتي مستقل رسما متولد شد. اگر چه تولد اين رشته جديد را به پايان نامه دكتري ميشل شاموس [2] كه بعداً با نام مشترک خود وي و استادش پاراپرتا [3] به صورت اولين كتاب در زمينه هندسه محاسباتي منتشر شد [4] ربط مي دهند، ولي تحقيق و پژوهش در مورد بسياري از مفاهيم اساسي اين رشته از قبيل دياگرام ورونوي، مثلث بنديها . . . به سده‌هاي قبل‌تر بر ميگردد.

هندسه محاسباتي در طول عمر رسمي كمتر از ۵۰ ساله خود، با گسترش بسيار زياد، نظرات پژوهشگران و محققان بسياري را بهخود جلب كرده است. طرح مسائل بسيار زيبا كه صورت ساده و قابل فهمي دارند و كاربردهاي بسيار گسترده در اكثر علوم مهندسي، كامپيوتر، رباتيک، جغرافيا، بايوانفورماتيک، مدل سازي . . . از يک سو، و ارائه راه حل هاي الگوريتمي و قابل پياده سازي و اجرا توسط ماشين و نيز بكارگيري قوتها و توانائي هاي آلگوريتم، ساختمان داده ها و هندسه تئوريک از سوي ديگر هندسه محاسباتي را به يک رشته تحقيقاتي مدرن، به روز و پرجاذبه تبديل كرده است.

اگر چه بسياري از مسائل هندسه محاسباتي سخت و پر چالش هستند ولي معمولاً صورت آنها ساده و قابل فهم است. به همين دليل همچون يک اقيانوس عميق در عين حال با سواحل كم عمق و زيبا، براي محققان و پژوهشگران بسيار جذاب مي باشد.

كاربردهاي هندسه محاسباتي بسيار گسترده است. چندان دور از واقعيت نيست اگر ادعا كنيم؛ هندسه محاسباتي اين توانائي را دارد كه براي بسياري از مشكلات و مسائل در زمينه‌هاي مهندسي و كاربردي با روش هاي الگوريتمي يک مدل درست كند. اگر چه هميشه حل اين مدل به سادگي صورت نمي پذيرد.

و اما از سوي ديگر طراحي و تحليل آلگوريتم براي دانشجويان علوم و مهندسي كامپيوتر و حتي رياضي، اصلي و پايه اي ترين درس در دوره ي كارشناسي است. دانشجوياني كه اين درس را به خوبي فرا مي گيرند در مشاغل مختلف مهندسي و برنامه نويسي در كار با كامپيوتر مهارت خواهند داشت. بنابراين بالا بردن مهارت دانشجويان در زمينه طراحي و تحليل آلگوريتم براي دانشكده هاي رياضي و علوم و مهندسي كامپيوتر بسيار مهم و استراتژيک مي باشد.

مسائل كتاب حاضر بارها در مقطع كارشناسي و بعضاً كارشناسي ارشد در دانشكده رياضي و علوم كامپيوتر تدريس شده است، و به عنوان يک درس كمكي براي فراگيري مفاهيم و روشهاي طراحي و تحليل الگوريتم با كمک مثال هايي از هندسه مفيد مي باشد.

در واقع مباحث اين كتاب نه مي تواند جايگزين درس طراحي و تحليل الگوريتم شود، كه از كليت مباحث درس طراحي الگوريتم برخوردار نيست و نه مي تواند جايگزين درس هندسه محاسباتي شود، كه تمام مباحث آن را پوشش نمي دهد. بلكه كتاب حاضر به عنوان مرجع درسي با نام "هندسه آلگوريتمي" و يا "آگلوريتم هاي هندسي" و به منظور تقويت درک دانشجويان از مباحث طراحي الگورتيم و نيز آشنائي با مسائل هندسه محاسباتي براي دانشجويان سال آخر دوره‌ي كارشناسي و يا سال اول كارشناسي ارشد علوم و مهندسي كامپيوتر قابل ارائه مي باشد.

[1] Geometrie algorithmique
[2] Michael Shamos
[3] Franco P.Preparata
[4] Computational Geometry - An Introduction

در کتاب پیش روي به بهانه آموزش طراحي و تحلیل آلگوریتم ها مثال هاي جذاب و زیبائي از هندسه محاسباتي طرح و حل مي‌شود. بعضي از مسائل از مقالات و آخرین تحقیقات روز انتخاب شده است و دانشجویان را با مرزهاي دانش آشنا مي شوند. و بعضي دیگر از کتب رایج هندسه محاسباتي.

واقعیت این است که این کتاب بیشتر یک کتاب تحلیل الگوریتم و روش‌هاي حل مسئله است تا بررسي ساختمان داده‌ها. دلیل آن این است که در حل مسائل عمدتا به تحلیل آلگوریتم پرداخته شده است و از مباحث ساختمان داده‌ها نسبتا سطحي عبور کرده‌ایم.

در حقیقت پرداختن به تحلیل و الگوریتم و بررسي ساختمان داده‌ها بطور هم زمان مقداري موجب خلط مباحث مي‌شود، و خواننده نه به درستي با روش ها و آلگوریتم‌هاي حل مسئله آشنا مي‌شود و نه با شیوه‌هاي انتخاب و تحلیل ساختمان داده‌ها.

به همین دلیل در این کتاب ترجیح داده ام که بررسي آلگوریتم‌ها را انتخاب کنم و تحلیل ساختمان داده‌ها را به فرصتي دیگر احاله دهیم.

فصل ۱

مقدمه

در این فصل بطور مختصر به بررسي چند سؤال ابتدايي ولي بسيار مهم مي‌پردازيم.

به این سؤالات معمولا در درس ساختمان داده‌ها و طراحي آلگوريتم مقدماتي به تفصيل پاسخ داده مي‌شود. در اینجا نیز به منظور يادآوري، به شكلي بسيار مختصر به آنها مي‌پردازيم. این سؤالات عبارتند از:

- نوع داده مجرد چیست و چه استفاده اي دارد؟
- ساختمان داده ها چیست و چه کاربردي دارد؟
- آلگوريتم چيست؟
- مفهوم آناليز آلگوريتم به چه معني است؟
- مسئله محاسباتي به چه نوع مسئله اي مي گوييم؟

نوع داده مجرد

یک نوع داده مجرد [1] يك مدل رياضي براي داده‌ها است که تعدادي عمليات روي آن تعريف مي‌شود. بنابراین هر نوع داده مجرد متشكل است از مجموعه‌اي از مقادير (داده‌ها) و مجموعه‌اي از عمليات بر روي آن‌ها. این مجموعه داده‌ها و عمليات روي آن، يك ساختار رياضي تشكيل مي‌دهد كه با كمك آن يك ساختمان داده‌ها [2] پياده‌سازي مي‌شود.

یک نوع داده مجرد شامل تعدادي عملگر است. این عملگرها توسط آلگوريتم بكار گرفته مي‌شوند. بنابراين براي هر یک نوع داده مجرد، دسته‌اي از عملگرها وجود دارد.

يك مثال ساده از داده‌هاي مجرد مي تواند مجموعه اعداد صحيح با عملگرهاي (+،−،×،%،/) باشد.

آرايه، مثال ساده يك‌ـ‌بعدي از یک نوع داده مجرد مي‌باشد که بصورت زير تعريف مي‌شود:

- مجموعه عناصر: دنباله‌اي با طول ثابت (مجموعه‌اي مرتب) از عناصر كه همگي از يك نوع اند.
- عمليات اصلي: دست‌يابي مستقيم به هر عنصر آرايه، طول آرايه

[1] Abstract Data Type- ADT

[2] Data Structure همواره ساختمان داده ها ترجمه شده است. لاكن اخيرا بعضي همكاران ترجمه داده ساختار را صحيح تر دانسته اند. از آنجا كه در ادبيات ترجمه ساختمان داده ها بيشتر بكار رفته است و تقريبا رايج شده است ما نيز از این كلمه استفاده مي كنيم.

به عنوان مثال‌هايي براي یک نوع داده مجرد مي توان: صف، لیست، صف اولویت، مجموعه، درخت، دنباله نام برد.
ساختمان داده‌ها یک مدل براي سازماندهي داده‌ها است. ساختمان داده‌ها عبارت است از ساختن تعدادي بلوک که بخش‌هاي یک آلگوریتم را بهم متصل مي‌کند، برکارايي آلگوریتم تاثير مي‌گذارد و براي اجراي یک آلکوریتم سهولت ایجاد مي‌کند. به همین دلیل یادگيري ساختمان داده‌ها بسیار اساسي و مهم است.
ساختمان داده‌ها یک روش براي ساخت، تغيير و دسترسي به داده‌ها است. به تعبيري رسمي‌تر، ساختمان داده‌ها عبارت است از سه مؤلفه:

۱. یک ساختار پایدار که داده‌هاي مجرد در آن نگهداري مي‌شوند.

۲. مجموعه‌اي از عملگرها براي بکارگيري نوعي خاص از داده‌هاي مجرد

۳. اجراي عملگرهاي تعریف شده بر روي داده‌ها و ساختارهاي نگهداري‌شده (ادامه صفحه ۲۴ کتاب نوشته شود.)

ساختمان داده‌ها را به طور مختصر مي‌توان بصورت زیر دسته‌بندي کرد:

۱. ساختارهای خطی [۳]
   (آ) لیست های فشرده [۴] :
      i. رشته‌ها [۵]
      ii. آرایه‌ها [۶]
      iii. پشته‌ها [۷]
      iv. صف‌ها [۸]
   (ب) لیست‌های پیوندی [۹]
      i. انواع لیستهاي يك طرفه و دو طرفه و مدور
      ii. پشته و صف به روش لیستهاي پیوندي

۲. ساختارهای غیرخطی [۱۰]
   (آ) گراف‌ها [۱۱]
   (ب) درخت‌ها [۱۲]

مسئله محاسباتی

یک مسئله محاسباتي مسئله‌اي است که مي‌توان آن را با کمک یک مجموعه از دستورات پي در پي ریاضي نوشت و سپس توسط کامپیوتر حل نمود. البته توجه داشته باشید که بعضي مسائل محاسباتي هستند که توسط کامپیوتر قابل حل نیستند؛ مانند یافتن یک دور بسته با حداقل طول که از تمام نقاط یک مجموعه داده شده دقیقا یک بار بگذرد.

---

[۳] Linear Structures
[۴] Dense Lists
[۵] Strings
[۶] Arrays
[۷] Stacks
[۸] Queues
[۹] Linked Lists
[۱۰] Non-Linear Structures
[۱۱] Graphs
[۱۲] Trees

بنابراين، يک مسئله محاسباتي مي‌تواند بصورت مجموعه‌اي از دستورات نوشته شود؛ براي هر دستور يک راه حل وجود دارد و قابل پياده‌سازي مي‌باشد.

يک مسئله محاسباتي داراي يک ورودي است و يک خروجي که وابسته به ورودي مي باشد.

مثال۱: عددn داده شده است. بررسي کنيد اين عدد اول است يا خير.

در اين مثال عدد صحيح n ورودي است و پاسخ بله يا خير که مي توان بجاي آن صفر و يک در نظر گرفت به عنوان خروجي

مثال۲: مجموعه‌اي از نقاط در صفحه داده شده است. يک چندضلعي ساده با کمک اين نقاط بسازيد.

در اين مسئله محاسباتي مجموعه اي از نقاط صفحه ورودي مسئله است و يک چندضلعي ساده که با کمک نقاط ورودي ساخته شده خروجي مسئله.

براي حل يک مسئله محاسباتي، به يک نوع داده مجرد، يک ساختمان بر روي آن و مجموعه‌اي از عملگرها که ساختمان مربوطه از اين عملگرها پشتيباني کند، نياز داريم. استفاده از اين عملگرها به حل مسئله توسط ماشين مي‌انجامد.

حل مسئله کامپیوتری

حل مسئله توسط کامپيوتر، يکي از مهارت‌هاي ارزشمند است که مي‌تواند يکي از اهداف اصلي دانش علوم کامپيوتر باشد و دانشجوي علوم کامپيوتر بايستي آن را در دوران تحصيل خود کسب کند. در حقيقت توانايي حل مسائل مختلف توسط کامپيوتر مي‌تواند مهارت اصلي يک دانش‌آموخته رشته علوم کامپيوتر تلقي شود. که طبيعتا مهارت ارزشمندي نيز مي باشد.

امروزه عمدتا با مسائلي سر و کار داريم که از داده‌هاي حجيم استفاده مي‌کنند؛ بنابراين نه تنها بايد براي حل آنها از روش‌هاي کامپيوتري استفاده نمود ، بلکه بايد در مورد سازماندهي داده‌ها نيز بررسي‌هاي لازم را انجام داد. از سوي ديگر، بسياري از مسائل، يکبار پيش‌پردازش مي‌شوند؛ ولي بارها و بارها براي داده‌هاي مختلف مورد سؤال قرار مي‌گيرند. در اين گونه مسائل، سرعت در پاسخ داده بسيار مهم است. بنابراين حل اين گونه مسائل نه تنها با کامپيوتر، بلکه با روش‌هاي کامپيوترهاي بسيار سريع بايد صورت پذيرد و زمان در حل اين گونه مسائل بسيار مهم است. يک دانش آموخته علوم کامپيوتر مي‌بايست لوازم ضروري اين مهارت را داشته باشد. يکي از اين لوازم، طراحي آلگوريتم خصوصا آلگوريتم‌هاي هندسي مي باشد.

مدل‌سازی

براي حل يک مسئله توسط کامپيوتر بايد مسئله در قالب يک مدل در آيد که با سازوکارهاي کامپيوتر قابل درک باشد؛ لذا بايد براي آن يک مدل ساخت و يا به عبارت ديگر، بايد آن را مدل‌سازي کرد. براي اين کار، بايد پارامترهاي اصلي مسئله را شناخت و از آن‌ها براي تعريف مدل مسئله استفاده نمود.

اگر بتوان مدل‌هاي اصلي مسئله را به کار گرفت، مي توان مدل مسئله را ساده کرد و آنگاه حل مسئله نيز ساده‌تر خواهد شد. براي مدل‌سازي بايستي پارامترهاي جزئي تا حد ممکن ناديده گرفت و پارامترهاي اصلي را تعيين و برد.

براي حل مدل مسئله بايد از يک آلگوريتم استفاده کرد. اين الگوريتم مشخص مي‌کند که داده‌هاي مسئله بايد توسط چه ساختمان داده اي بيان شوند و روي اين ساختمان داده چه اعمالي صورت گيرد.

مراحل حل یک مسئله

- بيان مسئله [۱۳] که عبارت است از تعريف صورت مسئله به شکلي واضح و دقيق و تعيين ورودي و خروجي

[۱۳] Problem Statement

- تدوين يك مدل [۱۴] كه عبارت است از تدوين يک مدل مناسب براي مسئله، حذف پارامترهاي غيرضروري كه امكان حذف آنها وجود دارد و تعيين پارامترهاي اصلي.

براي مدل سازي يک مسئله معمولا دو سؤال مطرح مي شود که پاسخ به آنها به تدوين يک مدل مناسب براي مسئله منجر خواهد شد.

– براي مدل‌سازي يک مسئله، راه حل اول که ساده‌تر به نظر مي‌رسد، اين است که مسئله ديگري مشابه آن بيابيم و با اعمال تغييراتي که لازم است، آن را تبديل به مدلي براي حل مسئله خودمان کنيم. براي استفاده از اين روش بايد اطلاعات خود را در مورد مدل‌هاي مختلف حل مسئله و نمونه هاي مسائلي که مدل و حل شده است افزايش دهيم.

– روش ديگر اين است که مستقيما تا حد ممکن پارامترهاي مسئله را ساده و سپس براي آن يک مدل طراحي کنيم. بخش اول اين کار را ساده‌سازي مسئله مي‌گويند. يعني تبديل مسئله سخت با پارامترهاي متعدد به مسئله‌اي ساده با پارمترهاي اندک. هرچه ساده‌سازي مسئله بهتر صورت پذيرد، طراحي مدل براي آن ساده‌تر خواهد بود. براي اينکار لازم است مسئله و پارمترها بطور دقيق آناليز شوند و ميزان تأثير پارامترها در پاسخ مسئله معين گردد.

آلگوريتم

تعاريف متعددي در مورد آلگوريتم وجود دارد. در يک تعريف ساده مي توان گفت:
آلگوريتم عبارت است از يک روش براي حل يک مسئله محاسباتي. يا آلگورتيم عبارت است از مجموعه‌اي متناهي از قواعد و دستورات که مراحل حل يك مسئله توسط کامپيوتر را بيان مي‌کند.
براي انجام هر نوع پردازش در کامپيوتر آلگوريتم لازم است؛ لذا از اين ديدگاه، علم کامپيوتر يعني طراحي و تحليل آلگوريتم.
يک آلگوريتم را مي‌توان به روش‌هاي مختلف بيان و دستورات آن را با کمک کامپيوتر محاسبه کرد. آلگوريتم و ساختمان داده‌ها با يکديگر ارتباط متقابل دارند. آلگوريتم، ساختمان داده‌ها را انتخاب مي‌کند و عملگرهاي آن را بکار مي‌گيرد. ساختمان داده‌ها مانند بلوک‌هايي هستند که مراحل آلگوريتم با کمک آن‌ها ساخته مي‌شود. يك آلگوريتم بايد از سادگي و قطعيت برخوردار باشد تا بتواند توسط کامپيوتر اجرا شود. يك آلگوريتم به زباني که براي ماشين قابل فهم است، ترجمه و پياده سازي مي‌شود. اين عمل را اجراي آلگوريتم گويند.
اكنون به طرح چند سؤال و پاسخ مختصر آن مي پردازيم. اگر پاسخ ها بسيار کلي و سريع به نظر رسيد و نتوانست شما را متقاعد کند بايد به کتب طراحي الگوريتم مراجعه نماييد.

چند سؤال و جواب اوليه

- يك آلگوريتم را چگونه طراحي کنيم؟

طراحي آلگوريتم يک علم و در عين حال يک هنر است که با حل مسئله و نيز تجربه بدست مي‌آيد. در اين کتاب سعي مي‌شود بعضي از روش‌هاي مرسوم مورد بررسي قرار گيرد و با حل نمونه هاي متعدد، اين مهارت افزايش يابد.

- درستي يك آلگوريتم به چه معني است؟

درستي يک آلگوريتم بدين معني است که آلگوريتم بايد بتواند تمام نمونه‌هاي مسئله را حل کند. بعضي مواقع براي اثبات درستي يک آلگوريتم اثبات لازم است. بعضي مواقع نيز آلگوريتم بازگوکننده مراحل يک مسئله است و اثبات درستي، در خود راه حل مسئله مستتر مي‌باشد.

---
[۱۴] Modeling

- چگونه يك آلگوريتم را تجزيه و تحليل كنيم؟

مسئله تجزیه و تحلیل یک آلگوریتم، مسئله مهمي است که نتیجه آن ارزشیابي آلگوریتم‌ها مي‌باشد. با تجریه و تحلیل یک الگوریتم، کارایي آن مشخص مي‌شود.

- عملي‌بودن يك آلگوريتم را چگونه بررسي كنيم؟

عملي‌بودن یک آلگوریتم با پیاده‌سازي آن قابل بررسي است.اگر آلگوریتمي قابل پیاده‌سازي بود و توانست نمونه‌هاي مسئله مربوطه را حل کند، براي حل مسئله یک روش عملي ارائه داده است.

بنابراین براي حل مسئله ابتدا طراحي یک مدل لازم است که مانند یک الگو به بررسي، تحلیل و حل مسئله کمک مي کند. سپس طراحي يك آلگوريتم در آن مدل و استفاده از ساختمان داده مناسب براي پیاده سازي آلگوریتم. اثبات درستي و صحت آلگوریتم و تجزیه و تحلیل آلگورتیم مراحل مهمي است که بدنبال طراحي آلگوریتم بایستي مد نظر قرار گیرد. در بخش هاي زیر در مورد این مفاهیم توضیحات بیشتري خواهیم داد.

طراحی آلگوریتم‌ها

ساختار و مراحل مقدماتي طراحي يك آلگوريتم را به صورت زير مي‌توان بيان نمود.

۱. مراحل ورودي [۱۵] تعیین ورودي آلگوریتم که بایستي دقیق، معین و قابل بررسي باشد.

مرحله جایگذاري [۱۶] تعریف تعدادي زیرمسئله براي حل مسئله اصلي، چینش دقیق زیرمسائل و تبیین یک مدل و استراتژي براي ورورد به مسئله

۲. مرحله تصمیم‌گیري [۱۷] انتخاب یک مسیر اصلي براي حل مسئله یا به عبارت دیگر طراحي یک استراتژي از قبیل حریصانه، تقسیم و حل، بازگشت به عقب و .... براي حل مسئله

۳. مرحله تکرار [۱۸] ممکن است براي حل مسئله نیاز به بررسي و تکرار حل زیر مسائل باشد.

۴. مرحله خروجي [۱۹] لازم است مسئله یک خروجي نهایي مشخص داشته باشد که در انتها به آن اشاره مي شود.

يك آلگوريتم بايد داراي ويژگي هاي زير باشد. (Knuth-1973)

۱. متناهي بودن [۲۰] آلگوریتم بایستي در زماني قابل قبول تمام شود و کار حل مسئله به انتها برسد.

۲. معین بودن [۲۱] يك آلگوريتم بايد بدون ابهام، قطعي و در هر مرحله واضح و روشن باشد.

۳. کلیت [۲۲] تمام حالات خاص مسائل را در بر بگیرد.

۴. کارایي [۲۳] آلگوریتم باید تمام نمونه هاي مسئله را به درستي حل کند.

---

[۱۵] Input Step
[۱۶] Assignment Step
[۱۷] Decission Step
[۱۸] Iteration Step
[۱۹] Output Step
[۲۰] Finiteness
[۲۱] Definiteness
[۲۲] Generality
[۲۳] Effectiveness

۵. ورودي [۲۴] يك آلگوريم بايستي داراي يک ورودي باشد.

۶. خروجي [۲۵] يك آلگوريتم بايد كاري انجام دهد و خروجي داشته باشد.

آناليز یک آلگوریتم

پس از طراحي يک آلگوريتم براي حل يک مسئله، همواره يک سؤال مطرح است: "آيا آلگوريتم بهتري براي حل مسئله وجود دارد؟"
به عبارت ديگر، براي حل هر مسئله ممکن است تعدادي آلگوريتم وجود داشته باشد. سؤال اين است که "کدام آلگوريتم براي حل مسئله مناسب‌تر است؟" آناليز يک آلگوريتم، عبارت است از اندازه‌گيري کارايي آن؛ بدين معني که يک آلگوريتم تا چه حد و چه موقع قابل اجرا است و اجراي آن به چه ميزان منابع نياز دارد. همچنين زمان و حافظه مورد نياز آلگوريتم به صورت تابعي از اندازه ورودي الگوريتم بايد بررسي گردد. با کمک آناليز آلگوريتم، کارآيي ساختمان داده‌هاي مورد نياز تعيين مي‌شود. مي‌توان زمان موثر دو آلگوريتم براي حل يک مسئله را مقايسه کرد و بدين روش، الگوريتم‌ها مقايسه و رتبه‌بندي مي‌شوند.
يک روش ساده براي مقايسه آلگوريتم‌ها، اجرا و سپس مقايسه آنها است؛ اگرچه مقايسه زمان اجراي آلگوريتم‌ها خود مشکلاتي دارد و سؤلات متعددي را به دنبال دارد. از جمله اينکه:

- چه داده‌اي مورد استفاده قرار گيرد.
- چه کامپيوتري براي اجرا در نظر گرفته شود.
- آلگوريتم با چه زبان برنامه نويسي نوشته شود.

بنابراين باتوجه به وجود پارمترهاي متعدد که هريک در اجرا و سرعت و نيز ميزان استفاده از فضاي حافظه کامپيوتر تاثيردارد مقايسه آلگوريتم‌ها کار خيلي ساده‌اي نيست.

براي آناليز يک آلگوريتم لازم است کارآيي آن مورد بررسي قرار گيرد. با بررسي کارآيي يک آلگوريتم مشخص مي‌شود که الگوريتم چه موقع قابل استفاده خواهد بود. از سوي ديگر، آناليز آلگوريتم‌هاي مختلف، امکان مقايسه و درجه‌بندي آنها را در هنگام حل يک مسئله واحد فراهم مي‌آورد.
آناليز يک آلگوريتم بطور کلي به معني بررسي دو نکته در مورد آلگوريتم است: فضا (حافظه) و زمان مورد نياز آلگوريتم در هنگام حل يک مسئله. در اين کتاب، عمدتا به زمان مورد نياز يک آلگوريتم براي حل يک مسئله توجه خواهد شد.
به منظور محاسبه زمان يك آلگوريتم، لازم است عملگرها و تعداد تكرار آنها مورد بررسي قرار گيرد. به عنوان مثال، در بررسي مسئله محاسبه پوش محدب چند ضلعي، رؤس چند ضلعي را ورودي مسئله در نظر مي گيريم و تعداد دفعاتي كه اضلاع چندضلعي مورد مراجعه قرار مي‌گيرند، به عنوان زمان آلگوريتم محاسبه مي‌شود. به عنوان مثال ديگر، در مرتب سازي و جستجو، هنگامي كه يک آرايه از اعداد به صورت ورودي مسئله مورد نظر قرار مي‌گيرد، تعداد دفعاتي كه اعداد بررسي و يا مقايسه مي‌شوند، به عنوان زمان آلگوريتم محاسبه مي‌شود. زمان كلي يك آلگوريتم برابر است با مجموع زمان مراحل اجراي آن. اين زمان با کمک محاسبه مجموع زمان مورد نياز براي انجام عملگرهاي آلگوريتم مشخص مي‌شود.
از آنجا كه زمان اجرا بستگي به مدل محاسبه دارد و نمي‌تواند به کامپيوتر و وسايل سخت‌افزاري ديگر متكي باشد، بنابراين بايد مدل واحدي را براي محاسبه آلگوريتم‌هاي متفاوت در حل يك مسئله به كار برد. اين مدل كه قاعدتا بايد مستقل از نوع و مدل و توانايي كامپيوتر باشد، پيچيدگي زمان الگوريتم ناميده مي‌شود.

---
[۲۴] Input
[۲۵] Output

مقایسه کران بالا و کران پایین

آناليز آلگوريتم، علاوه بر تعيين كران بالا و كران پائين، زمان حل يک مسئله توسط آن آلگوريتم را نيز تعيين مي‌كند. هنگامي كه كران بالا و كران پائين زمان مسئله را داشته باشيم، آن‌ها را با هم مقايسه مي‌كنيم و ميزان مفيد بودن آلگوريتم را مي‌سنجيم. با محاسبه پيچيدگي حد بالا مي توان به اين سوال پاسخ داد كه آيا آلگوريتم براي حل تمام نمونه‌هاي مسئله مناسب است يا خير. همچنين محاسبه پيچيدگي حد بالا نيز نشان مي‌دهد كه آلگوريتم براي حل كدام نمونه از مسائل مي‌تواند ارزشمند باشد.

اعتبارسنجی آلگوریتم

اعتبار سنجي آلگوريتم عبارت است از نشان‌دادن اينكه آلگوريتم براي كليه ورودي‌هاي داده شده صحيح عمل مي‌كند يا خير. براي سنجش اعتبار آلگوريتم، معمولا نمونه‌هاي خيلي خاص مسئله در نظر گرفته مي‌شود و توانايي آلگوريتم در حل اين نمونه‌ها مورد تحليل قرار مي‌گيرد.
پس از نهايي‌شدن اين مراحل، مي‌توان با استفاده از يک زبان برنامه نويسي مناسب، الگوريتم را پياده‌سازي كرد.

فصل ۲

ساختمان داده‌های هندسی

کوچکترین شيء هندسي، نقطه است. یک رابطه يك به يك بين نقاط در صفحه و زوج‌هاي مرتب (x, y) وجود دارد؛ به این صورت که يك نقطه را در صفحه مي‌توان با زوج مرتب (x, y) نمايش داد. همچنين زوج مرتب (x, y) مي‌تواند به عنوان يك بردار در صفحه نيز در نظر گرفته شود. در حقيقت مبدأ ابتداي اين بردار و نقطه (x, y) انتهاي آن خواهد بود. با اين تعبير، یک بردار، يك پاره‌خط جهت‌دار است که از مبدأ به سمت نقطه (x, y) امتداد دارد. مبدأ مختصات يعني نقطه $(0, 0)$ بردار صفر ناميده و با 0 نشان داده مي‌شود.

عملگرهای روی بردارها

مي‌توان روي بردارها که به صورت فوق تعريف شدند، تعدادي عملگر تعريف نمود.

عملگر جمع : اگر $\vec{a} = (x_1, y_1)$ و $\vec{b} = (x_2, y_2)$ آنگاه عملگر جمع را بصورت $\vec{a} + \vec{b} = (x_2 + x_2, y_1 + y_2)$ تعريف مي‌کنيم.

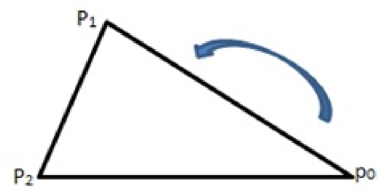
از نظر هندسي، با کمک بردارهاي $\vec{a}$ و $\vec{b}$ مي‌توان یک متوازي‌الاضلاع ساخت که $\vec{a} + \vec{b}$ قطر آن را تشكيل مي‌دهد.

عملگر ضرب عددي: اگر t يك عدد حقيقي باشد، آنگاه تعريف مي‌کنيم: $t\vec{a} = (tx, ty)$

اگر $t > 0$ آنگاه $t\vec{a}$ و $\vec{a}$ هم‌جهت خواهند بود؛ در غير اين صورت، جهت آن‌ها در خلاف جهت يكديگر مي‌باشد.

عملگر تفاضل : اگر $\vec{a} = (x_1, y_1)$ و $\vec{b} = (x_2, y_2)$ آنگاه تعريف مي‌کنيم: $\vec{a} + \vec{b} = \vec{a} + (\vec{-b})$

پاره‌خط جهت‌دار ab **:** عبارت است از پاره‌خطي که ابتداي آن ثابت و در نقطه a، انتهاي آن در نقطه b و جهت آن نيز از a به سمت b باشد. سه نقطهa و b و c را در نظر مي‌گيريم. گوييم مثلث $\triangle abc$ كه توسط اين سه نقطه ساخته مي‌شود، چرخش مثبت يا چرخش چپ (چرخش منفي يا چرخش راست) دارد؛ اگر و فقط اگر نقطه c سمت چپ (سمت راست) پاره‌خط جهت‌دار ab باشد. به عبارت ديگر، اگر $\theta_a cb$ زاويه‌اي باشد که از چرخش $\vec{ca}$ بصورت پادساعت گرد به سمت $\vec{cb}$ بدست آمده است، آنگاه مثلث $\triangle abc$ چپ گرد (راست گرد) است يا چرخش مثبت (چرخش منفي) دارد؛ اگر و فقط اگر $(180 < \theta_{acb} < 360)$ $(0 < \theta_{acb} < 180)$



شکل ۲.۱: شکل ۱

اگر $\vec{A}$ و $\vec{B}$ دو بردار باشند، $|A \times B|$ برابر است با مساحت متوازي‌الاضلاع ساخته‌شده به وسيله دو بردار.

به عبارت دیگر، اگر $\vec{A} = a - c$ و $\vec{A} = b - c$ آنگاه مساحت مثلث $\triangle abc$ عبارت است از نصف بردار $A \times B$

$$\frac{1}{2}A \times B = \begin{vmatrix} i & j & k \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix} = (a_x b_y - a_x b_y)i + (a_x b_z - a_z b_x)j + (a_x b_y - a_y b_x)k$$

اگر این بردارها در دو بعد باشند، آنگاه $(b_z = a_z = 0)$

در این صورت، حاصل‌ضرب دو بردار عبارت است از بردار نرمال صفحه مثلث حاصل از دو بردار.
مقدار این بردار برابر است با $|A \times B| = (A_0 B_1 - A_1 B_0)$
و این برابر است با دترمینان زیر:

$$2 \times Area(a, b, c) = \begin{vmatrix} a_4 & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix} = (b_x - a_x)(c_y - a_y) - (b_y - a_y)(c_x - a_x)$$

اگر مقدار این دترمینان مثبت باشد، نقاط a, b, c چپ‌گرد اند.
اگر مقدار این دترمینان منفي باشد، نقاط a, b, c راست‌گرد اند.
اگر مقدار این دترمینان صفر باشد، نقاط a, b, c هم‌راستا اند.
مساحت یک چندضلعي را مي‌توان با تقسیم آن به تعدادي مثلث و استفاده از روش فوق بدست آورد.

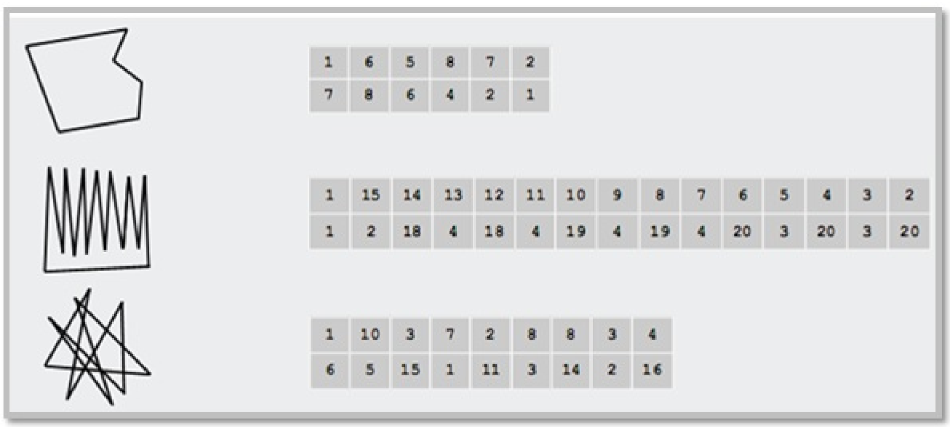
نمونه‌هایي از اشیاء هندسي

نقطه: دو عدد (x, y) خط: دو عدد a, b $(ax + by = 1)$
پاره‌خط: دو نقطه
چندضلعي: دنباله‌اي از نقاط
مثلث، مستطیل، دایره، کره، مخروط

نمونه‌هايي از عملگرهاي هندسي

برخي عملگرهاي اوليه عبارتند از:

- آيا نقطه درون چندضلعي قرار دارد؟
- مقايسه‌ي بخش‌هايي از دو خط
- فاصله ميان دو نقطه
- آيا دو پاره‌خط اشتراک دارند؟
- اگر سه نقطه p_1, p_2, p_3 داده شده باشد، آيا $p_1 - p_2 - p_3$ يک حرکت پادساعت‌گرد است؟
- آيا چندضلعي داده‌شده ساده است؟



شکل ۲.۲: سمت چپ، آنچه ما مي‌پنداريم و سمت راست، آنچه الگوريتم مي‌بيند.

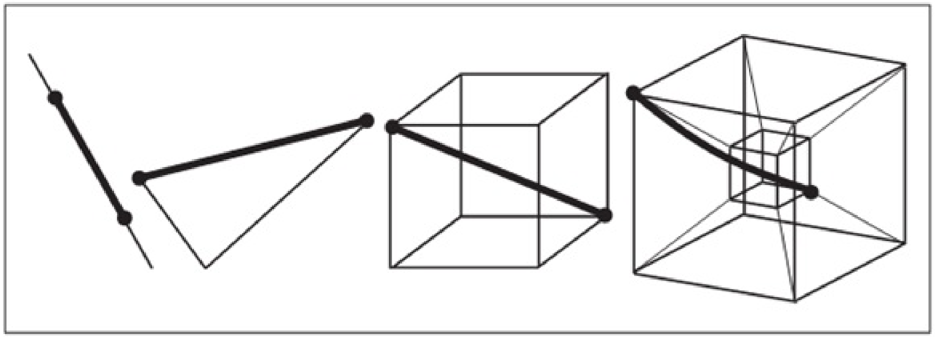
نکاتی راجع‌به فاصله [1]

تعريف فضاي متريک: به مجموعه‌اي گفته مي‌شود که نوعي فاصله (متر) ميان اعضاي آن تعريف شده باشد.
مثال‌هاي فضاي متريک:

۱. متر (فاصله) اقليدسي: يک خلبان هلي کوپتر فاصله ها به صورت متر اقليدسي مي‌سنجد.

$$d = \sqrt{(x_1 - x0)^2 + (y_1 - y_0)^2}$$

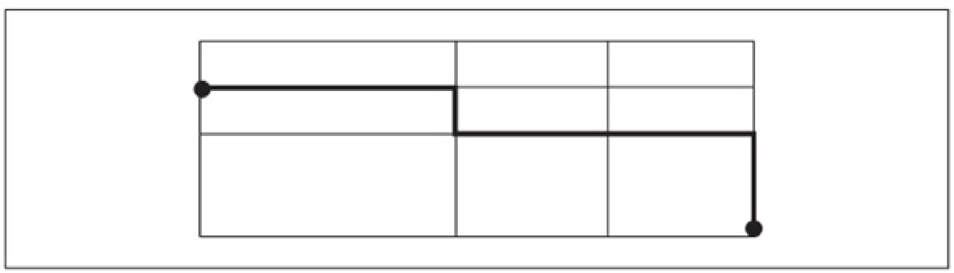
[1] Distance



شکل ۳.۲: فاصله اقلیدسي در فضاي ۱، ۲، ۳ و ۴ بعدي

۲. متر Manhattan : خیابان‌هاي محله منهتن نیویورک بدین صورت آرایش یافته است. معمولا تاکسي‌هاي محله منهتن فاصله یک نقطه از نقطه دیگر را با متر منهتن مي‌سنجند. (در فضای nــ بعدی)

$$d(x-y)=\sum_{k=0}^{n}|xk-yk|$$



شکل ۴.۲: متر منهتن

۳. متر (فاصله) ماکزیمم: عبارت است از مؤلفه ماکزیمم اختلاف دو نقطه:

$$d=Maxd(x,y)$$

فصل ۳

روش افزایشی

مقدمه

اجازه دهید قبل از ورود به مباحث جدي این فصل، کار را با یک توضیح ساده که در عین حال مي تواند مفهوم آلگوریتم افزایشي را به خوبي منتقل نماید شروع کنیم.

فرض کنید شخصي در حال دیدن یک فیلم پلیسي است که قتلي در آن اتفاق افتاده است. بیننده از ابتدا که شروع به دیدن فیلم مي‌کند، یک فرضیه در ذهن دارد که چه کسي مرتکب قتلي شده که در ابتداي فیلم رخ داده و چگونه و چرا این امر صورت گرفته است. در طول دیدن فیلم، هر نشانه‌ي جدید ممکن است فرضیه اول را تایید یا تکمیل کند و یا نیاز باشد در آن تجدید نظر شود. همچنین ممکن است حتي رها شده و مجددا فرموله شود. تنها در پایان فیلم است که تمام سرنخ‌ها بدست مي‌آیند و بیننده راز جرم و جنایت را حل کرده است. البته با این فرض که بیننده به اندازه کافي باهوش است و نویسنده فیلم نیز داستان را صحیح و منصفانه تعریف کرده است.

الگوریتم‌هاي افزایشي نیز چنین وضعیتي دارند. در بعضي موارد، الگوریتم تنها قادر به بیان و توضیح وضعیت فعلي به عنوان یک راه حل مسئله است. اگر چه ممکن است این راه حل، مخالف پاسخ نهایي باشد. این وضعیت مي‌تواند زماني اتفاق افتد که بخشي از ورودي‌ها خیلي ناقص است و یا یک وضعیت منسجم براي ورودي‌ها وجود ندارد. بنابراین حدس‌زدن پاسخ نهایي در طول حل مسئله، همیشه ما را به پاسخ نهایي رهنمون نمي‌کند.

براي ساختن یک چند ضلعي با کمک تعدادي نقطه داده‌شده، تا زماني که تمام ورودي‌ها پردازش نشوند، مرز چندضلعي بدست نمي‌آید. گاهي ممکن است با ورود تعدادي نقطه، ساده‌بودن چندضلعي به هم بخورد و تا زماني که همه ورودي پردازش نشده است، چند ضلعي مشخص نمي‌شود. همان‌طور که در مثال دیدن فیلم جنایي، فرضیه‌هاي بدست‌آمده در مراحل مختلف، ممکن است بارها با اشتباه روبرو شوند و یا سرنخ‌هاي اولیه و بینش‌هاي سازمان‌یافته نتوانند به فرآیند بدست‌آوردن پاسخ نهایي کمک کنند. در این مثال نیز هرگونه حدسي براي جواب ممکن است غلط از کار درآید.

مثال ۱

یک مثال ساده محاسباتي دیگر از روش افزایشي درجي، مي‌تواند پیداکردن کوچکترین عدد، در یک آرایه از اعداد صحیح باشد. در ابتدا اولین عدد را براي پاسخ مسئله کاندید مي‌نماییم و سپس جلو مي‌رویم. با هر عدد جدید از آرایه، پاسخ قبلي را به روز مي‌کنیم و هرگاه به عددي کوچکتر برخورد کردیم، آن را جایگزین پاسخ قبلي مي‌نماییم. کوچکترین عدد صحیح در میان اعداد دیده‌شده، پاسخ مسئله تا کنون خواهد بود؛ بنابراین تا آخرین عدد، هر حدسي براي پاسخ مسئله ممکن است غلط باشد و پاسخ مسئله وقتي بدست مي‌آید که تمام اعداد آرایه بررسي شده باشند.

روش محاسباتي افزايشي ، يک روش محاسباتي نسبتا ساده است که در حل مسائل محاسباتي و از جمله هندسه محاسباتي متعددي بکار گرفته مي‌شود. اين روش محاسباتي ويژگي هاي زير را دارد:

- روش افزايش درجي، يک روش مناسب براي حل مسائل محاسباتي و هندسه محاسباتي مي‌باشد.
- براي حل يک مسئله با روش افزايشي ابتدا کار را با يک ورودي آغاز مي‌کنيم و در هر مرحله، ورودي مسئله را يکي افزايش مي‌دهيم.
- در هر مرحله، با افزايش ورودي، پاسخ بدست آمده در مرحله قبلي را به‌روز مي‌کنيم.
- در انتها، با اتمام ورودي‌ها، پاسخ مسئله نهايي به دست مي‌آيد.
- در بعضي مسائل، تا انتها نمي‌توان پاسخ صحيح مسئله را ديد.
- در طول الگوريتم، با بررسي داده‌ها در هر مرحله، پاسخ مسئله مرحله به مرحله ساخته مي‌شود. پاسخ مسئله تنها در انتهاي الگوريتم قابل رؤيت خواهد بود.

با توجه به تعاريف و توضيحات داده‌شده، اکنون وقت آن رسيده است که به توضيح نمونه‌هايي از مسائلي که با اين روش حل مي‌شوند، بپردازيم. اولين مسئله، مرتب‌سازي درجي خواهد بود که يک مسئله ساده است. در حقيقت با توضيح اين مسئله که معمولا در درس ساختمان داده‌ها عنوان مي‌شود، مروري بر توضيحات بالا خواهيم داشت و پس از آن، به مسائل هندسي خواهيم پرداخت.

مرتب‌سازی درجی

فرض کنيد آرايه‌اي از اعداد حقيقي داده شده است. مي‌خواهيم داده‌هاي درون اين آرايه را مرتب نماييم.

مرتب‌سازي به روش درجي، ساده‌ترين نوع مرتب‌سازي است. يک مثال ساده و ملموس از مرتب‌سازي به روش درجي، مرتب‌کردن کارت‌هاي بازي مي‌باشد. براي مرتب‌کردن کارت‌ها به روش درجي، ابتدا از اولين کارت شروع مي‌کنيم. در هر مرحله، يك کارت را خارج نموده و با کارت‌هايي که تا به حال مرتب شده است، يکي يکي مقايسه مي‌کنيم. سپس کارت مذکور را منتقل نموده و در محل خود قرار مي‌دهيم. اين عمل را تا زماني که ديگر کارتي باقي نماند، و در همه جاي اصلي خود قرار گرفته باشند ادامه مي دهيم.
فرض کنيد آرايه $A[0, 1, 2, \ldots, n-1]$ شامل n عدد حقيقي داده شده است. مي خواهيم اعداد اين آرايه را مرتب نماييم.
شبه کد آلگوريتم درجي اين کار را انجام مي دهد:

```
insertion_Sort(A_n)
for  i=2 to n   Do
    key = A[i];
    j = i - 1;
    while   j > 0 && A[j] > key
        A[j+1] = A[j]
        j = j - 1;
    A[j+1]= key;
```

نقطه قوت مرتب‌سازي درجي اين است که نسبتاً ساده و قابل درک و اجراي آن آسان است. در مقابل، نقطه ضعف آن اين است که براي مرتب‌کردن مجموعه‌اي بزرگ از داده‌ها مناسب نيست. چرا که براي رسيدن به پاسخ نهايي، بايد هر عدد را با کليه داده‌ها مقايسه نماييم و اين،

وقت نسبتا زيادي خواهد گرفت.
مثال: مجموعه اعداد "21 32 51 64 8 34" داده شده است. مي خواهيم با روش مرتب سازي درجی كه یک روش افزایشی است این مجموعه را مرتب نماییم.

34 8 64 51 32 21

34 8 64 51 32 21

34 8 64 51 32 21

34 8 64 51 32 21

34 8 64 51 32 21

34 8 64 51 32 21

34 8 64 51 32 21

34 8 64 51 32 21

پیچیدگي زماني مرتب‌سازي درجي

بهترين حالت [۱] : بهترين حالت براي مرتب كردن اعداد به روش افزايشي ، حالتي است كه اعداد از قبل مرتب باشند. در اين حالت هر عدد در محل صحيح خود قرار دارد و نيازي به مقايسه نيست. بنابراين پيچيدگي زماني مرتب سازي درجي $O(n)$ خواهد بود.

بدترين حالت : بدترين حالت براي مرتب كردن اعداد به روش افزايشي درجي، حالتي است كه اعداد برعكس مرتب شده باشند، در اين حالت هر عدد بايد با كليه اعداد قبلي مقايسه و آنگاه در انتهاي آرايه قرار گيرد. در اين حالت پيچيدگي زماني مرتب سازي $O(n^2)$ خواهد بود.

حالت متوسط: پيچيدگي زمان متوسط یک آلگوريتم براي شرايطي رخ مي دهد كه عناصر به صورت تصادفي داده شوند. بنابراين حالت متوسط آلگوريتم مرتب سازي درجي در حالتيكه اعداد به صورت تصادفي پخش شده باشند محاسبه مي‌شود. اگر به شبه كد بالا توجه نماييد در هر تكرار حلقه‌ي بيروني، حلقه‌ي داخلي براي يافتن محل مناسب درج عنصر جديد به طور ميانگين نصف ليست مرتب شده را پيمايش مي‌كند.به اين معني مطابق شبه كد بالا در هر تكرار حلقه‌ي بيروني، حلقه‌ي داخلي براي يافتن محل مناسب درج عنصر جديد به طور ميانگين نصف ليست مرتب شده را پيمايش مي‌كند. چرا كه به طور متوسط نيمي از عناصر در $A[1\ldots j]$ كم‌تر از Key هستند و نيمي بزرگ‌تر از آن مي‌باشند. بنابراين در حالت ميانگين از مرتبه‌ي نصف تعداد ليست مرتب شده زمان مصرف مي شود. بنابراين:

$$T(n) = 1 + \frac{2}{2} + \frac{3}{2} + \cdots + \frac{n-1}{2} = \frac{n(n-1)}{4} \approx \frac{n^2}{4} \in O(n^2)$$

بنابراين ثابت مي شود كه بطور متوسط، پيچيدگي زماني مرتب سازي درجي مشابه بدترين حال $\theta(n^2)$ مي باشد.

[۱] Best Case

ساختن چندضلعي ستاره شكل

فرض كنيد مجموعه نقاط $S = s_0, \ldots, s_0$ در صفحه داده شده است. چگونه مي‌توان با استفاده از مجموعه نقاط S يك چندضلعي ستاره‌شكل [۲] ساخت؟

با توضيحاتي كه ارائه شد و مثال هايي كه زده شد مقداري با روش افزايشي آشنا شديم. اكنون مي توانيم به مسائل هندسي و حل آن با كمک روش افزايشي بپردازيم.

در اين بخش مي خواهيم به روش افزايشي با كمک مجموعه اي از نقاط داده شده يک چندضلعي ستاره شكل بدست آوريم. براي انجام اين كار مقدمات و تعاريف نسبتا زيادي لازم است كه در زير خواهد آمد.

مقدمه و کاربرد: در يک تعريف نسبتاً ساده، مي‌توان علم هندسه را علم انجام قواعد و دستورات بر روي نقطه، خط و چندضلعي دانست. قواعدي از قبيل دوري و نزديكي، داخل و خارج، شباهت و تفاوت و ... .

در اين ميان، خط نقش و اهميت خاصي دارد. ترسيم خط راست بسيار ساده‌تر از منحني است و كاربردهاي آن در دنياي واقعي بي‌شمار است. در معماري بناهاي كلاسيک از خطوط راست استفاده مي‌شود. خيابان‌ها و اتوبان‌ها از خطوط راست تشكيل شده اند. در تئوري فيزيک كلاسيک، نور از شعاع‌هاي راست تشكيل شده است. در تئوري ديد و رؤيت پذيري، پرتو ديد و شعاع اشعه‌هاي سنسورها از خطوط راست ايجاد شده اند. در هندسه قديم و هندسه مدرن، پاره‌خط كاربردهايي بيش از خط دارد. پاره‌خط، خط راستي است كه ابتدا و انتهاي آن با كمک دو نقطه محدود شده است. به عبارت ديگر، يک پاره‌خط، از دو طرف محدود مي‌باشد. با كمک پاره‌خط‌ها، چندضلعي‌ها ساخته مي‌شوند. بسياري از محاسبات مسائل هندسي، در مورد چندضلعي‌ها مي‌باشد. چندضلعي¬ها اشكال هندسي هستند كه براي نمايش بسياري از اشياي موجود در دنياي واقعي به كار مي¬روند و اساس علم هندسه را تشكيل مي دهد.

هسته چندضلعی

در اين بخش مي خواهيم به عنوان اولين كاربرد از روش افزايشي براي يافتن چندضلعي ستاده شكل استفاده كنيم. براي اين كار لازم است ابتدا تعريف هسته را بدانيم. به منظور آشنائي با اين مفهوم مي‌توانيد يک سالن بزرگ شامل تعدادي فروشگاه و بخش‌هاي اداري متعدد را در نظر بگيريد و يک مركز اطلاعات كه در قسمتي از آن وجود دارد. افراد براي كسب اطلاع به مركز اطلاعات مراجعه مي‌كنند و پاسخ سؤالات خود را مي‌گيرند. در اينجا يک سؤال مطرح مي‌شود: "چه محلي در سالن، بهترين مكان براي اين مركز اطلاعات مي‌باشد؟" بدون شک، بهترين مكان براي اين مركز اطلاعات، محلي است كه از تمام نقاط سالن و درهاي ورودي قابل رويت باشد تا افراد به سادگي بتوانند از هر نقطه و از هر فاصله‌اي، مركز اطلاعات را ببينند و به آن مراجعه نمايند. با نگاهي متفاوت، بهترين موقعيت مكاني براي مركز اطلاعات، موقعيتي است كه از آنجا، هر شخصي كه در مركز اطلاعات حضور دارد، بتواند تمام فضاي فروشگاه را ببيند. به چنين مكاني در فروشگاه كه از آن تمام نقاط قابل رؤيت باشد، هسته فروشگاه مي‌گويند. و اگر چنين مكاني در فروشگاه وجود داشته باشد فروشگاه ستاره شكل است. نكات فوق بصورتي دقيق تر و با تعابير هندسي تعريف خواهد شد.

دريک تعريف ابتدايي، چندضلعي ساده P [۳] ، ناحيه‌اي در صفحه است كه توسط تعداد محدودي پاره‌خط متصل به هم احاطه شده است؛ به طوريكه اين مجموعه پاره‌خط‌ها، يک دور ساده ايجاد مي‌كنند. در يک چندضلعي ساده، دو پاره‌خط (ضلع) مجاور، تنها در نقاط پاياني خود (رئوس P) با يكديگر تقاطع دارند. و اشتراک پاره‌خط‌ها (اضلاع) غير مجاور تهي است. اگر چه بيان فوق مقداري مفهوم چندضلعي ساده را بيان مي كند ولي چندضلعي را به صورت دقيق‌تر و با كمک عبارات رياضي مي توان به شكل زير تعريف كرد.

---

[۲] Star-Shaped

[۳] Simple Polygon

شکل ۱.۳: یک مرکز اطلاعات

تعریف چندضلعی

یک چند¬ضلعي در صفحه، شکلي هندسي است که از یک مجموعه¬ي مرتب از نقاط در صفحه تشکیل شده است. فرض کنید $V=$ $v_0,\ldots,v_{n-1}$ ، مجموعه اي از n نقطه در صفحه باشد. همچنین فرض کنید:

$$e_1 = v_0v_1, e_2 = v_1v_2, \ldots, e_i = v_{i-1}v_i, e_n = v_{n-1}v_0,$$

n پاره خط باشند که نقاط را به هم متصل مي‌نمایند؛ آنگاه ناحیه محصورشده توسط مجموعه $C = e_1, e_2, ..., e_n$ تشکیل يك چندضلعي ساده مي‌دهد؛ اگر و تنها اگر شرایط زیر برقرار باشند.

۱. محل تلاقي هر جفت پاره‌خط مجاور، تنها نقطه مشترك آن‌ها باشد.

$$\forall i \in 1, 2, \ldots, n-1 e_i \bigcap e_{i+1} = v_i$$

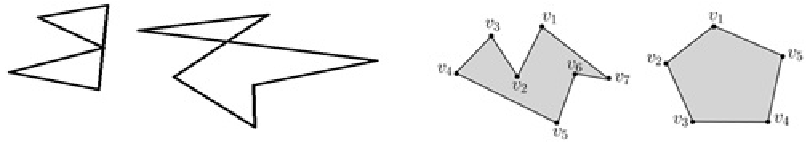
۲. هیچ دو پاره‌خط غیرمجاور یکدیگر را قطع نکنند.

$$\forall j \in 1, \ldots, n-1 j \neq i+1, i-1 \rightarrow e_i \bigcap e_j = o$$

شرط اول تضمین مي‌کند که مرز چندضلعي بسته باشد. به عبارت دیگر، انتهاي هر ضلع به ابتداي ضلع بعدي متصل شده باشد. شرط دوم نیز تضمین مي‌کند که بین هیچ دو ضلع غیر مجاور، تقاطع وجود نداشته باشد. بدین ترتیب چنانچه دو شرط فوق برقرار باشد، یک چندضلعي ساده ایجاد مي¬شود که پاره¬خط¬هاي سازندهي آن را اضلاع [۴] چندضلعي و نقاط سازنده¬ي آن را رئوس [۵] چندضلعي مي¬نامند. چنانچه از دو شرط فوق، شرط دوم برقرار نباشد، چندضلعي تشکیل¬شده یک چندضلعي غیر¬ساده مي¬باشد. مسائل هندسه¬ي محاسباتي اغلب بر چند¬ضلعي¬¬هاي ساده تمرکز مي¬کنند. در نتیجه ما نیز این رویکرد را دنبال میکنیم. شکل زیر را ملاحظه نمایید. چنانچه پاره‌خط اول و پاره‌خط انتهایي اشتراکي نداشته باشند، یک زنجیر باز [۶] خواهیم داشت. در حقیقت زنجیر باز به صورت زیر تعریف مي‌شود.
فرض کنید

$$e_1 = v_0v_1,\ e_2 = v_1v_2, \ldots,\ e_i = v_{i-1}v_i,\ e_{n-1} = v_{n-1}v_0,$$

[۴] Edge
[۵] Vertex
[۶] Open Chain



شکل ۲.۳: شکل سمت چپ، دسته‌بندي چندضلعي‌ها به چندضلعي‌هاي ساده و غير ساده. شکل سمت راست: دسته بندي چندضلعي ها به چندضلعي محدب و غير محدب

$n-1$ پاره‌خط باشند و شرايط زير برقرار باشند.

$$\forall i \in 1,2,\ldots,n-2 \quad e_i \bigcap e_{i+1} = v_i$$

$$\forall j \in 1,\ldots,n-1 \quad j \neq i+1, i-1 \quad \rightarrow \quad e_i \bigcap e_j = \varnothing$$

با استفاده از قضيه خم جردن، مي‌توان گفت چندضلعي ساده، صفحه را به دو ناحيه تقسيم مي‌کند: ناحيه‌ي کران‌دار [۷] که داخل چندضلعي قرار دارد و ناحيه‌ي بي‌کران [۸] که بيرون چندضلعي واقع شده است. معمولاً در مسائل هندسهي محاسباتي، مرز [۹] چندضلعي نيز جزء ناحيه‌ي داخل چندضلعي محسوب مي‌شود. به عبارت ديگر، بر خلاف آنچه در هندسه دبيرستاني مي دادند، يک چندضلعي در هندسه محاسباتي تنها يک مرز نيست؛ بلکه چندضلعي شامل مرز و ناحيه‌ي کران‌دار محدود شده توسط مرز مي‌باشد.

در يک دسته‌بندي مي‌توان چندضلعي‌هاي ساده را به دو دسته‌ي چندضلعي‌هاي محدب [۱۰] و چندضلعي‌هاي غيرمحدب [۱۱] طبقه‌بندي کرد. اگر چه در اين مورد بطور مفصل در بخش هاي بعدي صحبت خواهد شد. جهت آشنائي ابتدائي مي توان گفت چند‌ضلعي ساده‌ي P يک چندضلعي محدب است؛ اگر و تنها اگر هر پاره‌خطي که دو نقطه‌ي دلخواه از داخل P را به هم وصل مي‌کند، کاملاً داخل P قرار گيرد. در غير اين صورت P يک چندضلعي نامحدب است. از نگاهي ديگر، در چندضلعي محدب P، همه‌ي زواياي داخلي چندضلعي کوچکتر از π راديان مي‌باشند؛ در حالي که در يک چندضلعي نامحدب، حداقل يک زاويه‌ي داخلي بزرگتر از π راديان وجود دارد. شکل ۲ را ملاحظه کنيد.

اکنون که تعريف چندضلعي را بيان کرديم، به عنوان پيش‌نياز به يک موضوع ديگر نياز داريم و آن رؤيت پذيري است. اگر چه اين موضوع خود آنقدر بزرگ و گسترده و پرکاربرد است که به بزرگترين و جذاب ترين موضوع در هندسه محاسباتي تبديل شده است. ولي ما در اينجا به چند تعريف و نکته ابتدائي آن فقط در حد مورد نياز اشاره مي کنيم. ولي در فصول آينده مجددا به آن اشاره خواهيم داشت.

رویت پذیری

در هندسه "ديدن" و "رويت پذيري" از يک تعريف بسيار ساده آغاز و سپس به يکي از مباحث مهم و پيچيده با کاربردهاي فراوان در هندسه محاسباتي- که هم اکنون نيز به عنوان يک زمينه علمي جذاب با مسائل باز بسيار مطرح است- تبديل مي‌شود.

يک نمونه از کاربردهاي رويت‌پذيري، بهينه‌سازي حرکت ربات‌هاي جستجوگر است که يک هدف ثابت يا متحرک را دنبال مي‌کنند. در مسئله فوق، در يک فضاي بسته که با يک چندضلعي شبيه‌سازي مي‌شود تعدادي ربات با داشتن قابليت ديد، از برخورد با موانع پرهيز و به هدف دسترسي

[۷] Bounded
[۸] Unbounded
[۹] Boundary
[۱۰] Convex
[۱۱] Non-Convex

پيدا مي‌کنند. در اين مسئله، رؤيت‌پذيري در هدايت ربات‌ها کاربرد پيدا مي‌کند.
مسئله مسيريابي در سيستم‌هاي اطلاعات جغرافيايي که با يک شبکه گراف شبيه‌سازي مي‌شود نمونه ديگري از کاربردهاي رويت‌پذيري مي‌باشد. در اين مسئله، مسيريابي يک متحرک با کمک رويت‌پذيري صورت مي‌گيرد.
رويت‌پذيري در هندسه معماري نيز کاربرد دارد. طراحي پلان و ابعاد يک ساختمان، چشم انداز، جهت و نماي ساختمان و تقسيم‌بندي بخش‌هاي درون يک ساختمان با کمک رويت پذيري صورت مي‌گيرد. در حقيقت رؤيت پذيري در طراحي ديد و چشم انداز مناسب و افزايش بهره گيري از نور، تناسب بخش‌هاي مختلف داخلي نقش دارد.
مسئله رؤيت‌پذيري در طراحي و جانمايي شبکه سنسور و طراحي سيستم امنيت براي ساختمان‌ها و نيز نورپردازي سالن‌ها و نماي ساختمان‌ها کاربرد دارد.
يک کاربرد رايج ديگر مسئله رؤيت‌پذيري، امنيت و حفاظت از يک محيط توسط نگهبانان و يا دوربين‌هاي مدار بسته مي‌باشد.
مسئله‌ي معروف نگارخانه‌ي هنر و تلاش براي کاهش تعداد نگهباناني که از نگارخانه مراقبت مي‌کنند، يکي از معروف‌ترين کاربردهاي رويت‌پذيري مي‌باشد. در بخش بعدي به اين مسئله خواهيم پرداخت. ادامه اين بخش ما را از مسيرمان مقداري دور مي کند و لي با توجه به تعريف رؤيت پذيري ديدن صورت مسئله نگارخانه هنر در اينجا خالي از لطف نيست. اگر چه بعدها به اين مسئله بيشتر خواهيم پرداخت. اين کار ما را مقداري از مسيرمان منحرف مي کند ولي با توجه به تعريف رؤيت پذيري ديدن صورت مسئله گالري هنر در اينجا خالي از لطف نيست. اگر چه بعدها به اين مسئله بيشتر خواهيم پرداخت.

نگارخانه هنر

فرض کنيد يک نگارخانه با سالن‌ها و راهروهاي پيچ در پيچ وجود دارد که محل نگهداري تابلوها و آثار هنري گران‌قيمت است. به دليل اهميت و ارزش آثار هنري، نگارخانه بايد به طور دائم و کامل توسط نگهبانان يا دوربين‌هاي مداربسته تحت کنترل قرار گيرد. همچنين نکات زير در مورد نگارخانه و نگهبانان آن مطرح است:
دوربين‌ها بايد به نحوي در مکان‌هاي مختلف نگارخانه نصب شوند که تعدادشان کمينه شود؛ بدين وسيله، در هزينه‌هاي مراقبت از نگارخانه صرفه‌جويي مي‌شود. در عين حال، بايد تمام نقاط مراقبت شوند. بنابراين نبايد تعداد دوربين‌ها به اندازه‌اي کم باشد که بخش‌هايي از نگارخانه توسط آن‌ها پوشش داده نشود. علاوه بر آن، هرچه تصاوير ارسالي از دوربين‌ها کمتر باشد، نظارت و کنترل آن‌ها راحت‌تر خواهد بود. در واقع در بعضي مسائل، کاهش هم پوشاني دوربين‌ها اهميت دارد.
براي بررسي و تحليل مسئله نگارخانه هنر، مي‌توان نگارخانه را توسط يک چندضلعي با n رأس و دوربين‌ها را با نقاط مدل کرد.
مسئله‌ي نگارخانه هنر ابتدا در سال ۱۹۷۳ توسط کلي [۱۲] مطرح شد. او سؤال زير را مطرح کرد:
"چه تعداد نگهبان براي کنترل و حفاظت از يک نگارخانه مورد نياز است؟"
در مسئله کلاسيک نگارخانه هنر، شرايط زير در نظر گرفته مي شود:

- نگارخانه يک n ضلعي ساده مي باشد.
- هر نگهبان به عنوان يک نقطه‌ي ثابت در نظر گرفته مي‌شود که قادر است تمامي جهات را ببيند و داراي دامنه‌ي ديد ۳۶۰ درجه و با شعاع ديد بي‌نهايت مي‌باشد.
- نگهبان‌ها نمي‌توانند پشت ديوارها را ببينند.

مسائل جذاب، ارزشمند و در عين حال کاربردي در زمينه نگارخانه هنر وجود دارد که بعضي شرايط فوق را ناديده مي‌گيرند. به عنوان مثال، نگارخانه مي‌تواند ساده نباشد، بلکه تعدادي ستون در ميانه سالن‌ها و راهروها وجود داشته باشد. در اين حالت چندضلعي ساده نخواهد بود؛

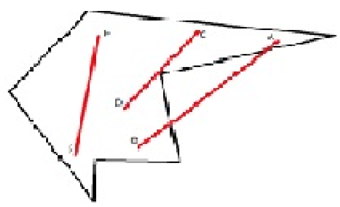
---

[۱۲] Klee

حالت دیگر آن است که دامنه دید نگهبان‌ها از ۳۶۰ درجه کمتر باشد؛ در این صورت اصطلاحا به آن آلفا گارد [۱۳] گویند. در حالتي دیگر، مي‌توان در نظر گرفت که شعاع دید نگهبان‌ها محدود باشد و یا اینکه بجاي نگهبان، لامپ در نظر بگیریم و محل لامپ‌ها را در خارج نگارخانه قرار دهیم.
در این حالت مسئله نگارخانه هنر را مي‌توان به صورت زیر نیز مطرح کرد:
"براي روشنایي کامل دیوارهاي یک ساختمان، براي مثال یک زندان، چه تعداد لامپ مورد نیاز است؟"
در مورد هرکدام از زیر مسائل فوق کارهاي تحقیقاتي بسیاري شده است و مسائل باز متنوعي وجود دارد که نظر پژوهشگران را به خود جلب مي کند.
براي مدل‌کردن مسئله کلي به تعاریف زیر نیاز داریم.
قابلیت دید [۱۴] : به فرض x و y دو نقطه درون چند ضلعي P باشند. گوییم نقطه x مي‌تواند نقطه y را ببیند ـ یا y قابل رؤیت براي x است اگر $\overline{xy} \subseteq P$ باشد.

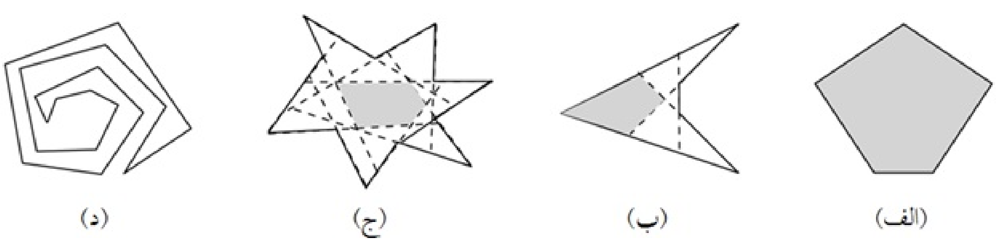
$$x\ can\ see\ y\ iff\ \overline{xy} \subseteq P$$

به عبارت دیگر، پاره‌خطي که دو نقطه x و y را به هم متصل مي‌کند، بطورکامل درون چندضلعي P قرار گیرد. توجه : عبارت بالا نشان مي‌دهد xy مي‌تواند با يك راس یا ضلع از چندضلعي P تماس داشته باشد. به عبارت دیگر xy مي‌تواند با مرز چندضلعي تماس داشته باشد.
قابلیت دید واضح : گوئیم دو نقطه x و y یکدیگر را به‌وضوح مي‌بینند یا براي یکدیگر به وضوح قابل رؤیت [۱۵] : هستند؛ اگر: $xy \subseteq$



شکل ۳.۳: : نقطه A براي نقطه B قابل رؤیت نیست یعني نقاط A و B نمي‌توانند یکدیگر را ببینند. نقطه C براي D قابل رؤیت است. نقطه E براي نقطه F به وضوح قابل رؤیت است.

$P, xy \bigcap \partial P \subseteq x, y$ بنابراین در حالتي که y به‌وضوح براي x قابل رؤیت است، xy نمي‌تواند با مرز چندضلعي تماس داشته باشد.



شکل ۴.۳: هسته در چندضلعي‌هاي مختلف به رنگ تیره نشان داده شده است. (الف) هسته در یک چندضلعي محدب. (ب) هسته در یک چندضلعي پروانه‌اي شکل. (ج) هسته در یک چندضلعي ستاره‌اي شکل که پروانه‌اي شکل نمي‌باشد. (د) یک چندضلعي غیر ستاره‌اي شکل که هسته آن تهي است.

[۱۳] α-Gard
[۱۴] Visibility
[۱۵] Clear Visibility

هسته : هسته [۱۶] يك چندضلعي P عبارت است از مجموعه كليه نقاطي از P كه تمام چندضلعي را مي‌بينند.

$$kernel(P) = x \in P | x \ can \ see \ y, \forall y \in P$$

واضح است كه وجود و بزرگي هسته يک چندضلعي، بستگي به شكل آن دارد. ممكن است هسته يک چندضلعي تهي باشد و يا با خود چندضلعي برابر باشد.

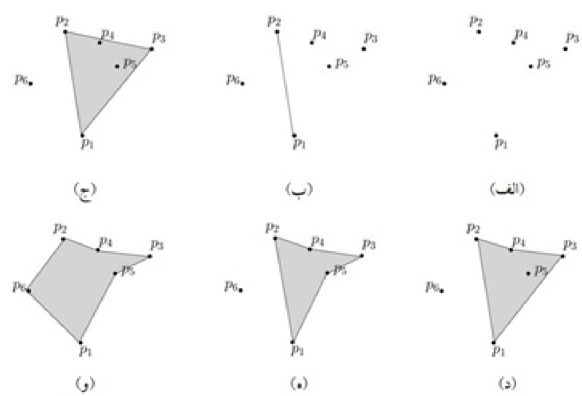
چندضلعي ستاره‌شكل: يك چندضلعي را ستاره‌شكل [۱۷] گوييم اگر هسته آن ناتهي باشد.

چندضلعي پروانه‌اي شكل: يك چندضلعي را پروانه‌اي شكل [۱۸] گوييم؛ اگر هسته آن شامل حداقل يكي از رأس‌هاي آن باشد. از آنجا كه تمام نقاط يک چندضلعي محدب مي‌توانند يكديگر را ببينند، هسته يک چندضلعي محدب با خود آن برابر است.

اكنون كه مقدمات لازم براي بررسي مسئله ساختن چندضلعي ستاره شكل را مطرح كرديم، مجددا به عنوان يادآوري صورت مسئله را مجددا ذكر مي كنيم.

فرض كنيد مجموعه نقاط $S = \{s_0, s_1, \ldots, s_{n-1}\}$ در صفحه داده شده است. چگونه مي‌توان با استفاده از مجموعه نقاط S يك چندضلعي ستاره‌شكل [۱۹] ساخت؟

همانطور كه در توضيح روش افزايشي اشاره شد، ساختن پاسخ مسئله را ابتدا با يک ورودي آغاز مي‌كنيم و در هر مرحله، ورودي مسئله را يكي افزايش داده، پاسخ قبلي را بهبود بخشيده و آن را به‌روز مي‌نماييم.



شكل ۵.۳: اجراي الگوريتم چندضلعي ستاره‌اي شكل براي مجموعه‌اي از نقاط

براي محاسبه يك چندضلعي ستاره‌شكل با كمک مجموعه نقاط $S = \{s_0, s_1, \ldots, s_{n-1}\}$ ابتدا با سه نقطه يک مثلث تشكيل مي‌دهيم و به دنبال آن مراحل ساخت يک چند چندضلعي ستاره‌شكل را دنبال مي‌كنيم. در هر مرحله نقطه جديد را به گونه‌اي به چندضلعي كه تا كنون ساخته شده است، اضافه مي‌كنيم كه چندضلعي جديد ستاره‌شكل باقي بماند. اين مسئله يک پاسخ منحصر به فرد ندارد؛ بلكه پاسخ‌هاي متعددي براي مسئله مي‌تواند وجود داشته باشد. با كمک آلگوريتمي كه در زير ارائه شده است، نوع خاصي چندضلعي ستاره‌اي بدست مي‌آيد كه هسته آن، شامل يكي از نقاط مجموعه S يعني s_0 است.

الگوريتم افزايشي براي يافتن هسته چندضلعی ستاره شكل

الگوريتم افزايشي براي يافتن هسته چندضلعي ستاره شكل براي ساختن يک چندضلعي ستاره‌شكل به گونه اي عمل مي كند كه اولين نقطه ورودي همواه يكي از رؤس پاسخ نهايي خواهد بود.

---

[۱۶] Kernel
[۱۷] Star-Shaped Polygon
[۱۸] Fan-Shaped
[۱۹] Star-Shaped

مراحل آلگوریتم به صورت زیر است.

- ابتدا با نقطه $s = s_0$ شروع مي‌کنيم.

- سپس با کمک نقاط ورودي دوم و سوم، یک مثلث مي‌سازيم که ستاره‌اي شکل است و شامل نقطه s_0 نيز مي‌باشد.

- فرض کنيد تا نقطه i ام اضافه و با کمک اين نقاط تا کنون چندضلعي ستاره‌شکل ساخته شده است. اکنون در مرحله i ام هستيم و نقطه i ام وارد شده است. نقطه s_i به گونه‌اي به چندضلعي که تا کنون ساخته شده است، اضافه مي‌شود که چندضلعي جديد ستاره‌شکل باقي بماند. براي اين کار، مرز چندضلعي موجود را از s_1 بصورت پادساعت‌گرد حول s_0 دور مي‌زنيم تا به رأسي مانند s_j برسيم که زاويه پادساعت‌گرد نقطه s_i نسبت به s_0 و خط افق بلافاصله کمتر از s_j نسبت به s_0 و خط افق باشد. در اين صورت s_i قبل از s_j قرار خواهد گرفت.

  در صورتي که دور به پايان رسيد و چنين رأسي يافت نشد، در اين حالت زاويه پادساعت‌گردِ رأس جديد از همه بزرگتر خواهد بود. در اين حالت s_i قبل از s_0 قرار خواهد گرفت.

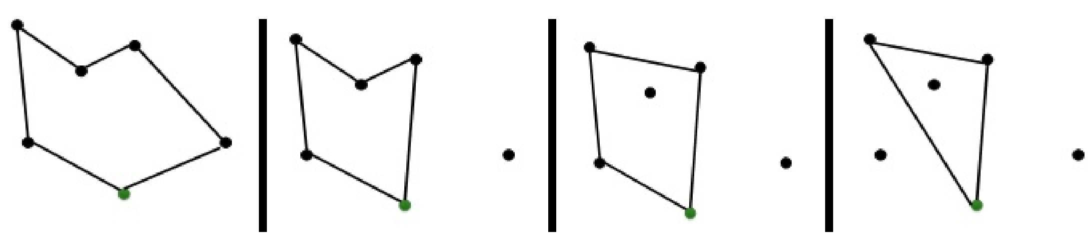
- با روش بالا نقاط بصورت پادساعت‌گرد با توجه به زاويه‌شان نسبت به s_0 به ترتيب به چندضلعي اضافه مي‌شوند. به عبارت ديگر s_p قبل از s_q قرار مي‌گيرد؛ اگر زاويه پادساعت‌گردش نسبت به s_0 و خط کوچکتر از s_q باشد.

روش مقايسه: روش مقايسه: به نظر مي رسد لازم باشد در مورد مقايسه موقعيت نقاط و تعيين محل نقطه جديدالورود که در حقيقت نکته اصلي آلگوريتم است و موجب مي‌شود چندضلعي ستاره شکل باقي بماند توضيح بيشتري داده شود. اولاً وقتي مي گوييم زاويه نقطه جديدالورود s_i منظور زاويه پادساعت گردي است که $s_i s_0$ با خط افق که از s_0 گذشته است مي سازد و آنرا با θ_{si} نشان مي‌دهيم. وقتي براي انجام مراحل آلگوريتم افزايشي و اضافه‌کردن نقطه جديد s_i مرز چندضلعي موجود را بصورت پادساعت‌گرد دور مي‌زنيم و رئوس را يکي يکي با راس جديد s_i مقايسه مي‌کنيم تا به رأسي مانند s_j برسيم که زاويه پادساعت‌گرد نقطه s_i بلافاصله کمتر از زاويه s_j است. روش مقايسه دو نقطه بدين صورت است که:

گوئيم نقطه $s_p = (r_p, \theta_p)$ کوچکتر از نقطه $s_q = (r_q, \theta_q)$ است و به صورت
$s_p < s_q$ نشان مي‌دهيم اگر : $\theta_q > \theta_p$ يا $\theta_q = \theta_p$ و $r_q > r_p$

با اين روش، نقاط روي محيط چندضلعي به صورت پادساعت‌گرد، به ترتيب زاويه از کوچک به بزرگ مرتب مي‌شوند.

فرض کنيد در مرحله iام چندضلعي ستاره‌شکل به صورت $\{s_0, s_1, \dots, s_{i-1}\}$ باشد که رئوس به ترتيب پادساعت‌گرد مرتب شده اند. وقتي نقطه s_i اضافه مي‌شود، با ديگر رئوس مقايسه مي‌شود؛ اگر s_j اولين نقطه‌اي در ترتيب پادساعت‌گرد باشد که زاويه آن بلافاصله بزرگتر از s_i است، چندضلعي ستاره‌شکل به صورت $s_0, s_1, \dots, s_{j-1}, s_i, s_j, \dots, s_{i-1}$ در مي‌آيد. اگر چنين s_j اي وجود نداشت، s_i به آخر ليست اضافه مي‌شود. در اين صورت چندضلعي ستاره‌شکل به صورت $s_0, s_1, \dots, s_{i-1}, s_i$ در مي‌آيد.



شکل ۶.۳: اجراي مراحل الگوريتم ساخت چندضلعي ستاره‌اي شکل

صحت الگوريتم

اثبات: مراحل ساختن چندضلعي ستاره‌شکل با روش افزايشي بر روي نقاط اضافه‌شده صورت مي‌گيرد. در هر مرحله، نقطه جديد به رئوس قبلي چندضلعي اضافه مي‌شود. نحوه افزودن نقطه جديد به نقاط پوش چندضلعي، به صورت پرتو مي‌باشد. نقاط جديد به گونه‌اي به نقاط پوش اضافه مي‌شوند و اطراف هسته قرار مي‌گيرند؛ که به راحتي مي‌شود ثابت کرد همواره حداقل s_0 تمام رئوس از جمله رأس جديد s_i را مي‌بينند.

آناليز الگوريتم

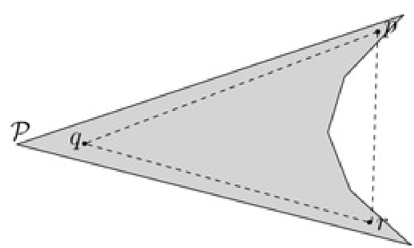
يافتن چندضلعي ستاره‌شکل با کمک الگوريتم افزايشي درجي همانند مرتب‌سازي درجي مي‌باشد. تمام نقاط يکي يکي در محل مناسب به نقاط قبلي پوش محدب اضافه مي‌شوند. انتخاب محلي مناسب براي نقطه جديد با مقايسه زاويه پادساعت‌گرد آن نقطه نسبت به s_0 با نقاط قبلي چندضلعي ساخته‌شده صورت مي‌گيرد. اين مقايسه در مرحله iام داراي پيچيدگي برابر $O(i)$ خواهد بود؛ بنابراين زمان کلي آلگوريتم برابر خواهد بود با:

$$T(n) = \sum_i O(i) = O(n^2)$$

تمرين

مسئله ۱ (ساده): نشان دهيد رويت‌پذيري داراي خاصيت تقارن [20] است. بدين معني که اگر نقطه‌ي p براي نقطه‌ي q قابل رؤيت باشد، آنگاه نقطه‌ي q نيز براي نقطه‌ي p قابل رويت است.

مسئله۲ (ساده): نشان دهيد رويت‌پذيري داراي خاصيت تعدي نمي‌باشد. بدين معني که اگر نقطه‌ي p نقطه‌ي q و نقطه‌ي q نقطه‌ي r را در p رويت کند، لزوما نقطه‌ي p نقطه‌ي r را رويت نمي‌کند. شکل ۶ را ملاحظه نماييد.



شکل ۷.۳: رويت‌پذيري خاصيت تعدي ندارد.

مسئله۳ : اثبات و يا رد کنيد: براي رؤيت‌پذيري چند ضلعي P کافي است ∂P قابل رويت باشد.

اگر پاسخ شما به سوال منفي است، چندضلعي P و يک مجموعه از نگهبان‌ها را طوري بسازيد که همه‌ي مرز چندضلعي قابل رويت باشد؛ ولي حداقل يکي از نقاط دروني آن با هيچ نگهباني رويت نشود. در غير اين صورت اگر پاسخ شما مثبت است، بايد آن را ثابت کنيد.

مسئله ۴: (اثبات و يا رد کنيد) براي رؤيت‌پذيري چندضلعي P کافي است درون P قابل رويت باشد.

[20] Symmetry

مسئله ۵: (اثبات و يا رد كنيد) براي رؤيت‌پذيري چندضلعي P كافي است رئوس P قابل رويت باشد.

مسئله۶: (اثبات و يا رد كنيد) هسته يک چندضلعي كراندار، درون چندضلعي و همبند است.

مسئله۷ (مسئله سخت) آلگوريتمي ارائه دهيد كه هسته يک چند ضلعي را به گونه‌اي محاسبه نمايد كه مساحت آن بيشينه باشد. (بدست آوردن هسته بيشينه)

مسئله۸: فرض كنيد مي‌خواهيم يک چندضلعي ستاره‌شكل را با كمک نقاط داده شده بسازيم. در دو حالت براي مسئله آلگوريتمي با زمان بهينه ارائه دهيد: حالت اول، زماني كه نقاط برخط [۲۱] داده مي‌شود و حالت دوم، زماني كه نقاط همگي از قبل داده شده اند.

مسئله ۹: (مسئله سخت) آلگوريتمي ارائه دهيد كه هسته يک چندضلعي را در حالتي كه شعاع ديد بي‌نهايت نيست، بلكه به ميزان d مي‌باشد، محاسبه نمايد.(هسته با شعاع ديد محدود)

مسئله ۱۰: (مسئله سخت)آلگوريتمي ارائه دهيد كه هسته يک چندضلعي را در حالتي كه زاويه ديد به جاي 2π مقدار π است محاسبه نمايد. (هسته با زاويه ديد محدود)

پوش محدب

دومين مسئله هندسي كه با روش افزايشي حل مي كنيم مسئله يافتن معروف پوش محدب مجموعه اي از نقاط داده شده مي باشد.
فرض كنيد مجموعه $P = \{P_0, \ldots, P_n\}$ شامل n نقطه در صفحه داده شده است. همچنين فرض كنيد هيچ سه نقطه‌اي در يک امتداد نيستند. مي خواهيم پوش محدب P با استفاده از روش افزايشي.

مقدمه و كاربرد:

يافتن پوش محدب مجموعه اي از نقاط داده شده با روش‌هاي متفاوتي امكان پذير است، كه يكي از ساده ترين آنها روش افزايشي مي‌باشد.در اين روش يافتن، پوش محدب بطور طبيعي با افزودن يک به يک نقاط صورت مي‌پذيرد. همچون مسئله قبل ابتدا به تعاريف و مقدمات لازم پرداخته مي شود و سپس مسئله پوش محدب را تعريف و با روش افزايشي حل مي كنيم.
در يک دسته‌بندي اوليه مي‌توان چندضلعي‌هاي ساده را به دو دسته‌ي چندضلعي‌هاي محدب [۲۲] و چندضلعي‌هاي غيرمحدب [۲۳] دسته بندي كرد. چندضلعي‌هاي محدب و اساسا مسئله تحدب و پوش محدب در رياضيات و به خصوص هندسه، به دليل كاربردهاي گسترده‌اي كه دارد، بسيار مورد توجه است.
براي يک مجموعه محدب تعاريف متعددي وجود دارد و در مورد روش ساختن پوش محدب مجموعه‌اي از نقاط و يا پوش محدب يک چندضلعي، شيوه‌هاي مختلفي ارائه شده است. در اين بخش، ابتدا پوش محدب را تعريف مي‌كنيم و سپس آلگوريتم يافتن پوش محدب براي مجموعه‌اي از نقاط با روش افزايشي درجي را توضيح مي‌دهيم. به دليل اهميت و كاربردهاي پوش محدب، در فصل‌هاي آينده اين مسئله با روش‌هاي متعددي حل خواهد شد. در ادامه اين فصل، پوش محدب مجموعه‌اي از نقاط به روش افزايشي مطرح خواهد شد. اگر چه در فصول بعدي، روش‌هاي ديگر كه بعضا سريع‌تر هم هستند، براي ساختن پوش محدب ارائه مي‌شود، لاكن الگوريتم افزايشي پوش محدب، هم بسيار زيبا و هم بسيار الهام‌بخش است. ابتدا به تعريف پوش محدب مجموعه‌اي براي يک مجموعه محدب تعاريف متعددي وجود دارد و در مورد روش ساختن پوش

[۲۱] Online
[۲۲] Convex
[۲۳] Non-Convex

محدب مجموعه‌اي از نقاط و يا پوش محدب يک چندضلعي، شيوه‌هاي مختلفي ارائه شده است. در اين بخش، ابتدا پوش محدب را تعريف مي‌کنيم و سپس آلگوريتم يافتن پوش محدب براي مجموعه‌اي از نقاط با روش افزايشي درجي را توضيح مي‌دهيم. به دليل اهميت و کاربردهاي پوش محدب، در فصل‌هاي آينده اين مسئله با روش‌هاي متعددي حل خواهد شد. در ادامه اين فصل، پوش محدب مجموعه‌اي از نقاط به روش افزايشي مطرح خواهد شد. اگر چه در فصول بعدي، روش‌هاي ديگر که بعضا سريع‌تر هم هستند، براي ساختن پوش محدب ارائه مي‌شود، لاکن الگوريتم افزايشي پوش محدب، هم بسيار زيبا و هم بسيار الهام‌بخش است. ابتدا به تعريف پوش محدب مجموعه‌اي از نقاط يا يک چندضلعي غير محدب خواهيم پرداخت و تعاريف متعددي را ــ که همه آنها معادل هستند ــ براي پوش محدب ارائه خواهيم کرد. (کاربرد پوش محدب)