طراحی هندسی کامپیوتری صفحات ۱ ـ ۳۰

پویا آقاحسینی

۲ دی ۱۳۹۷

فهرست مطالب

فهرست تصاویر

فهرست جداول

پيش‌گفتار

هندسه محاسباتي يا به تعبير فرانسوی‌ها هندسه آلگوريتمي [۱] 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 [۲] كه بعداً با نام مشترک خود وي و استادش پاراپرتا [۳] به صورت اولين كتاب در زمينه هندسه محاسباتي منتشر شد [۴] 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.

[۱] Geometrie algorithmique
[۲] Michael Shamos
[۳] Franco P.Preparata
[۴] Computational Geometry - An Introduction

در کتاب پيش روي به بهانه آموزش طراحي و تحليل آلگوريتم ها مثال هاي جذاب و زيبائي از هندسه محاسباتي طرح و حل مي‌شود. بعضي از مسائل از مقالات و آخرين تحقيقات روز انتخاب شده است و دانشجويان را با مرزهاي دانش آشنا مي شوند. و بعضي ديگر از کتب رايج هندسه محاسباتي.

واقعيت اين است که اين کتاب بيشتر يک کتاب تحليل الگوريتم و روش‌هاي حل مسئله است تا بررسي ساختمان داده‌ها. دليل آن اين است که در حل مسائل عمدتا به تحليل آلگوريتم پرداخته شده است و از مباحث ساختمان داده‌ها نسبتا سطحي عبور کرده‌ايم.

در حقيقت پرداختن به تحليل و الگوريتم و بررسي ساختمان داده‌ها بطور هم زمان مقداري موجب خلط مباحث مي‌شود، و خواننده نه به درستي با روش ها و آلگوريتم‌هاي حل مسئله آشنا مي‌شود و نه با شيوه‌هاي انتخاب و تحليل ساختمان داده‌ها.

به همين دليل در اين کتاب ترجيح داده ام که بررسي آلگوريتم‌ها را انتخاب کنم و تحليل ساختمان داده‌ها را به فرصتي ديگر احاله دهيم.

فصل ۱

مقدمه

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

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

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

نوع داده مجرد

يک نوع داده مجرد [۱] يك مدل رياضي براي داده‌ها است كه تعدادي عمليات روي آن تعريف مي‌شود. بنابراين هر نوع داده مجرد متشكل است از مجموعه‌اي از مقادير (داده‌ها) و مجموعه‌اي از عمليات بر روي آن‌ها. اين مجموعه داده‌ها و عمليات روي آن، يك ساختار رياضي تشكيل مي‌دهد كه با كمك آن يك ساختمان داده‌ها [۲] پياده‌سازي مي‌شود.
يک نوع داده مجرد شامل تعدادي عملگر است. اين عملگرها توسط آلگوريتم بكار گرفته مي‌شوند. بنابراين براي هر يک 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 يك‌ـ بعدي از يک نوع داده مجرد مي‌باشد كه 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

[۱] Abstract Data Type- ADT

[۲] 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 اين كلمه استفاده مي كنيم.

به عنوان مثال‌هايي براي يک نوع داده مجرد مي توان: صف، ليست، صف اولويت، مجموعه، درخت، دنباله نام برد.
ساختمان داده‌ها يک مدل براي سازماندهي داده‌ها است. ساختمان داده‌ها عبارت است از ساختن تعدادي بلوک که بخش‌هاي يک آلگوريتم را بهم متصل مي‌کند، برکارايي آلگوريتم تاثير مي‌گذارد و براي اجراي يک آلکوريتم سهولت ايجاد مي‌کند. به همين دليل يادگيري ساختمان داده‌ها بسيار اساسي و مهم است.
ساختمان داده‌ها يک روش براي ساخت، تغيير و دسترسي به داده‌ها است. به تعبيري رسمي‌تر، ساختمان داده‌ها عبارت است از سه مؤلفه:

۱. يک ساختار پايدار که داده‌هاي مجرد در آن نگهداري مي‌شوند.

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

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

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

۱. ساختارهای خطی [۳]

(آ) ليست های فشرده [۴] :

i. رشته‌ها [۵]

ii. آرايه‌ها [۶]

iii. پشته‌ها [۷]

iv. صف‌ها [۸]

(ب) ليست‌های پيوندی [۹]

i. انواع ليستهاي يك طرفه و دو طرفه و مدور

ii. پشته و صف به روش ليستهاي پيوندي

۲. ساختارهای غيرخطی [۱۰]

(آ) گراف‌ها [۱۱]

(ب) درخت‌ها [۱۲]

مسئله محاسباتی

يک مسئله محاسباتي مسئله‌اي است که مي‌توان آن را با کمک يک مجموعه از دستورات پي در پي رياضي نوشت و سپس توسط کامپيوتر حل نمود. البته توجه داشته باشيد که بعضي مسائل محاسباتي هستند که توسط کامپيوتر قابل حل نيستند؛ مانند يافتن يک دور بسته با حداقل طول که از تمام نقاط يک مجموعه داده شده دقيقا يک 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 کامپيوتری

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 يک 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 كه پاسخ به آنها به تدوين يک مدل مناسب براي مسئله منجر خواهد شد.

- براي مدل‌سازي يک مسئله، راه حل اول كه ساده‌تر به نظر مي‌رسد، اين است كه مسئله ديگري مشابه آن بيابيم و با اعمال تغييراتي كه لازم است، آن را تبديل به مدلي براي حل مسئله خودمان كنيم. براي استفاده از اين روش بايد اطلاعات خود را در مورد مدل‌هاي مختلف حل مسئله و نمونه هاي مسائلي كه مدل و حل شده است افزايش دهيم.
- روش ديگر اين است كه مستقيما تا حد ممكن پارامترهاي مسئله را ساده و سپس براي آن يک مدل طراحي كنيم. بخش اول اين كار 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 اينكار 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 كه مراحل حل يك مسئله توسط كامپيوتر را بيان مي‌كند.
براي انجام هر نوع پردازش در كامپيوتر آلگوريتم لازم است؛ لذا از اين ديدگاه، علم كامپيوتر يعني طراحي و تحليل آلگوريتم.
يک آلگوريتم را مي‌توان به روش‌هاي مختلف بيان و دستورات آن را با كمک كامپيوتر محاسبه كرد. آلگوريتم و ساختمان داده‌ها با يكديگر ارتباط متقابل دارند. آلگوريتم، ساختمان داده‌ها را انتخاب مي‌كند و عملگرهاي آن را بكار مي‌گيرد. ساختمان داده‌ها مانند بلوک‌هايي هستند كه مراحل آلگوريتم با كمک آن‌ها ساخته مي‌شود. يك آلگوريتم بايد از سادگي و قطعيت برخوردار باشد تا بتواند توسط كامپيوتر اجرا شود. يك آلگوريتم به زباني كه 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 كلي و سريع به نظر رسيد و نتوانست شما را متقاعد كند بايد به كتب طراحي الگوريتم مراجعه نماييد.

چند سؤال و جواب اوليه

- يك آلگوريتم را چگونه طراحي كنيم؟
  طراحي آلگوريتم يک علم و در عين حال يک هنر است كه با حل مسئله و نيز تجربه بدست مي‌آيد. در اين كتاب 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 كه آلگوريتم بايد بتواند تمام نمونه‌هاي مسئله را حل كند. بعضي مواقع براي اثبات درستي يک آلگوريتم اثبات لازم است. بعضي مواقع نيز آلگوريتم بازگوكننده 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 تجزيه و تحليل يک آلگوريتم، مسئله مهمي است که نتيجه آن ارزشيابي آلگوريتم‌ها مي‌باشد. با تجريه و تحليل يک الگوريتم، کارايي آن مشخص مي‌شود.

- عملي‌بودن يك آلگوريتم را چگونه بررسي كنيم؟
عملي‌بودن يک آلگوريتم با پياده‌سازي آن قابل بررسي است.اگر آلگوريتمي قابل پياده‌سازي بود و توانست نمونه‌هاي مسئله مربوطه را حل کند، براي حل مسئله يک روش عملي ارائه داده است.

بنابراين براي حل مسئله ابتدا طراحي يک مدل لازم است که مانند يک الگو به بررسي، تحليل و حل مسئله کمک مي کند. سپس طراحي يك آلگوريتم در آن مدل و استفاده از ساختمان داده مناسب براي پياده سازي آلگوريتم. اثبات درستي و صحت آلگوريتم و تجزيه و تحليل آلگورتيم مراحل مهمي است که بدنبال طراحي آلگوريتم بايستي مد نظر قرار گيرد. در بخش هاي زير در مورد اين مفاهيم توضيحات بيشتري خواهيم داد.

طراحی آلگوريتم‌ها

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

۱. مراحل ورودي [۱۵] تعيين ورودي آلگوريتم که بايستي دقيق، معين و قابل بررسي باشد.

مرحله جايگذاري [۱۶] تعريف تعدادي زيرمسئله براي حل مسئله اصلي، چينش دقيق زيرمسائل و تبيين يک مدل و استراتژي براي ورورد به مسئله

۲. مرحله تصميم‌گيري [۱۷] انتخاب يک مسير اصلي براي حل مسئله يا به عبارت ديگر طراحي يک استراتژي از قبيل حريصانه، تقسيم و حل، بازگشت به عقب و .... براي حل مسئله

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

۴. مرحله خروجي [۱۹] لازم است مسئله يک خروجي نهايي 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] آلگوريتم بايستي در زماني قابل قبول تمام شود و کار حل مسئله به انتها برسد.

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

۳. کليت [۲۲] تمام حالات خاص مسائل را در بر بگيرد.

۴. کارايي [۲۳] آلگوريتم بايد 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.

آنالیز 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 ممكن است تعدادي آلگوريتم وجود داشته باشد. سؤال اين است كه "كدام آلگوريتم براي حل مسئله مناسب‌تر است؟" آناليز يک آلگوريتم، عبارت است از اندازه‌گيري كارايي آن؛ بدين معني كه 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 كمک آناليز آلگوريتم، كارآيي ساختمان داده‌هاي مورد نياز تعيين مي‌شود. مي‌توان زمان موثر دو آلگوريتم براي حل يک مسئله را مقايسه كرد 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 مشكلاتي دارد و سؤلات متعددي را به دنبال دارد. از جمله اينكه:

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

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

براي آناليز يک آلگوريتم لازم است كارآيي آن مورد بررسي قرار گيرد. با بررسي كارآيي يک آلگوريتم مشخص مي‌شود كه الگوريتم چه موقع قابل استفاده خواهد بود. از سوي ديگر، آناليز آلگوريتم‌هاي مختلف، امكان مقايسه و درجه‌بندي آنها را در هنگام حل يک مسئله واحد فراهم مي‌آورد.
آناليز يک آلگوريتم بطور كلي به معني بررسي دو نكته در مورد آلگوريتم است: فضا (حافظه) و زمان مورد نياز آلگوريتم در هنگام حل يک مسئله. در اين كتاب، 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 كمک محاسبه مجموع زمان مورد نياز براي انجام عملگرهاي آلگوريتم مشخص مي‌شود.
از آنجا كه زمان اجرا بستگي به مدل محاسبه دارد و نمي‌تواند به كامپيوتر 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

آنالیز آلگوریتم، علاوه بر تعیین کران بالا و کران پائین، زمان حل یک مسئله توسط آن آلگوریتم را نیز تعیین می‌کند. هنگامي که کران بالا و کران پائین زمان مسئله را داشته باشیم، آن‌ها را با هم مقایسه می‌کنیم و میزان مفید بودن آلگوریتم را می‌سنجیم. با محاسبه پیچیدگي حد بالا مي توان به این سوال پاسخ داد که آیا آلگوریتم براي حل تمام نمونه‌هاي مسئله مناسب است یا خیر. همچنین محاسبه پیچیدگي حد بالا نیز نشان می‌دهد که آلگوریتم براي حل کدام نمونه از مسائل می‌تواند ارزشمند باشد.

اعتبارسنجی آلگوریتم

اعتبار سنجي آلگوریتم عبارت است از نشان‌دادن اینکه آلگوریتم براي کلیه ورودي‌هاي داده شده صحیح عمل می‌کند یا خیر. براي سنجش اعتبار آلگوریتم، معمولا نمونه‌هاي خیلي خاص مسئله در نظر گرفته می‌شود و توانایي آلگوریتم در حل این نمونه‌ها مورد تحلیل قرار می‌گیرد.
پس از نهایي‌شدن این مراحل، می‌توان با استفاده از یک زبان برنامه نویسي مناسب، الگوریتم را پیاده‌سازي کرد.

فصل ۲

ساختمان داده‌های هندسی

کوچکترین شيء هندسي، نقطه است. یک رابطه یك به یك بین نقاط در صفحه و زوج‌هاي مرتب (x, y) وجود دارد؛ به این صورت که یك نقطه را در صفحه مي‌توان با زوج مرتب (x, y) نمایش داد. همچنین زوج مرتب (x, y) مي‌تواند به عنوان یك بردار در صفحه نیز در نظر گرفته شود. در حقیقت مبدأ ابتداي این بردار و نقطه (x, y) انتهاي آن خواهد بود. با این تعبیر، یک بردار، یك پاره‌خط جهت‌دار است که از مبدأ به سمت نقطه (x, y) امتداد دارد. مبدأ مختصات یعني نقطه $(0, 0)$ بردار صفر نامیده و با 0 نشان داده مي‌شود.

عملگرهای روی بردارها

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

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

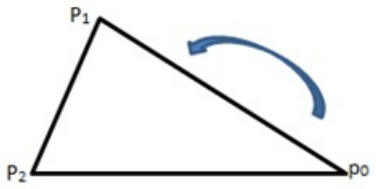
از نظر هندسي، با کمک بردارهاي $\vec{a}$ و $\vec{b}$ مي‌توان یک متوازي‌الاضلاع ساخت که $\vec{a} + \vec{b}$ قطر آن را تشکیل مي‌دهد.

عملگر ضرب عددي: اگر t یك عدد حقیقي باشد، آنگاه تعریف مي‌کنیم: $t\vec{a} = (tx, ty)$

اگر $t > 0$ آنگاه $\vec{ta}$ و $\vec{a}$ هم‌جهت خواهند بود؛ در غیر این صورت، جهت آن‌ها در خلاف جهت یکدیگر مي‌باشد.

عملگر تقاضل : اگر $\vec{a} = (x_1, y_1)$ و $\vec{b} = (x_2, y_2)$ آنگاه تعریف مي‌کنیم: $\vec{a} + \vec{b} = \vec{a} + (\vec{-b})$

پاره‌خط جهت‌دار ab **:** عبارت است از پاره‌خطي که ابتداي آن ثابت و در نقطه a، انتهاي آن در نقطه b و جهت آن نیز از a به سمت b باشد. سه نقطهa و b و c را در نظر مي‌گیریم. گوییم مثلث $\triangle abc$ که توسط این سه نقطه ساخته مي‌شود، چرخش مثبت یا چرخش چپ (چرخش منفي یا چرخش راست) دارد؛ اگر و فقط اگر نقطه c سمت چپ (سمت راست) پاره‌خط جهت‌دار ab باشد. به عبارت دیگر، اگر θ_acb زاویه‌اي 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 یا چرخش مثبت (چرخش منفي) دارد؛ اگر و فقط اگر $(0 < \theta_{acb} < 180)$ $(180 < \theta_{acb} < 360)$



شکل ۱.۲: شکل ۱

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

نمونه‌هايي از اشياء 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.

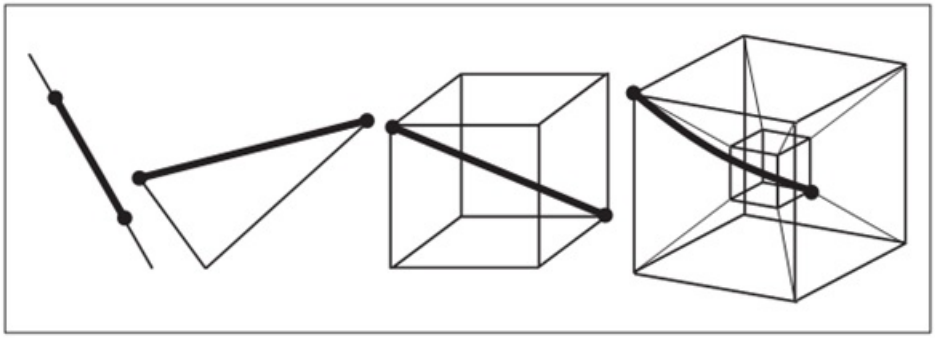
نکاتی راجع‌به فاصله [1]

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.

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

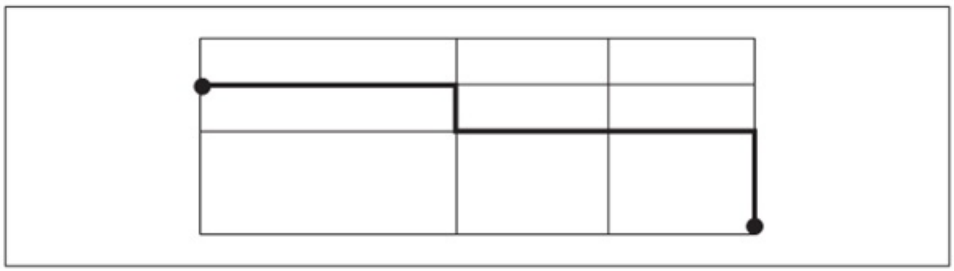
[1] Distance



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

۲. متر Manhattan : خيابان‌هاي محله منهتن نيويورک بدين صورت آرايش يافته است. معمولا تاکسي‌هاي محله منهتن فاصله يک نقطه از نقطه ديگر را با متر منهتن مي‌سنجند. (در فضای nـ بعدی)

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



شکل ۴.۲: متر منهتن

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

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

فصل ۳

روش افزایشی

مقدمه

اجازه دهيد قبل از ورود به مباحث جدي اين فصل، كار را با يک توضيح ساده كه در عين حال مي تواند مفهوم آلگوريتم افزايشي را به خوبي منتقل نمايد شروع كنيم.

فرض كنيد شخصي در حال ديدن يک فيلم پليسي است كه قتلي در آن اتفاق افتاده است. بيننده از ابتدا كه شروع به ديدن فيلم مي‌كند، يک فرضيه در ذهن دارد كه چه كسي مرتكب قتلي شده كه در ابتداي فيلم رخ داده و چگونه و چرا اين امر صورت گرفته است. در طول ديدن فيلم، هر نشانه‌ي جديد ممكن است فرضيه اول را تاييد يا تكميل كند و يا نياز باشد در آن تجديد نظر شود. همچنين ممكن است حتي رها شده و مجددا فرموله شود. تنها در پايان فيلم است كه تمام سرنخ‌ها بدست مي‌آيند و بيننده راز جرم و جنايت را حل كرده است. البته با اين فرض كه بيننده به اندازه كافي باهوش است و نويسنده فيلم نيز داستان را صحيح و منصفانه تعريف كرده است.

الگوريتم‌هاي افزايشي نيز چنين وضعيتي دارند. در بعضي موارد، الگوريتم تنها قادر به بيان و توضيح وضعيت فعلي به عنوان يک راه حل مسئله است. اگر چه ممكن است اين راه حل، مخالف پاسخ نهايي باشد. اين وضعيت مي‌تواند زماني اتفاق افتد كه بخشي از ورودي‌ها خيلي ناقص است و يا يک وضعيت منسجم براي ورودي‌ها وجود ندارد. بنابراين حدس‌زدن پاسخ نهايي در طول حل مسئله، هميشه ما را به پاسخ نهايي رهنمون نمي‌كند.

براي ساختن يک چند ضلعي با كمک تعدادي نقطه داده‌شده، تا زماني كه تمام ورودي‌ها پردازش نشوند، مرز چندضلعي بدست نمي‌آيد. گاهي ممكن است با ورود تعدادي نقطه، ساده‌بودن چندضلعي به هم بخورد و تا زماني كه همه ورودي پردازش نشده است، چند ضلعي مشخص نمي‌شود. همان‌طور كه در مثال ديدن فيلم جنايي، فرضيه‌هاي بدست‌آمده در مراحل مختلف، ممكن است بارها با اشتباه روبرو شوند و يا سرنخ‌هاي اوليه و بينش‌هاي سازمان‌يافته نتوانند به فرآيند بدست‌آوردن پاسخ نهايي كمک كنند. در اين مثال نيز هرگونه حدسي براي جواب ممكن است غلط از كار درآيد.

مثال ۱

يک مثال ساده محاسباتي ديگر از روش افزايشي درجي، مي‌تواند پيداكردن كوچكترين عدد، در يک آرايه از اعداد صحيح باشد. در ابتدا اولين عدد را براي پاسخ مسئله كانديد مي‌نماييم و سپس جلو مي‌رويم. با هر عدد جديد از آرايه، پاسخ قبلي را به روز مي‌كنيم و هرگاه به عددي كوچكتر برخورد كرديم، آن را جايگزين پاسخ قبلي مي‌نماييم. كوچكترين عدد صحيح در ميان اعداد ديده‌شده، پاسخ مسئله تا كنون خواهد بود؛ بنابراين تا آخرين عدد، هر حدسي براي پاسخ مسئله ممكن است غلط باشد و پاسخ مسئله وقتي بدست مي‌آيد كه تمام اعداد آرايه 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 که 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

پيچيدگي زماني مرتب‌سازي درجي

بهترين حالت [1] : 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 يک 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 $A[1 \dots j]$ 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.

[1] 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 كاربرد: 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 [۳] P ، 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، شكلي هندسي است كه از يک مجموعه¬ي مرتب از نقاط در صفحه تشكيل شده است. فرض كنيد $V = v_0, \ldots, v_{n-1}$ ، مجموعه اي از n نقطه در صفحه باشد. همچنين فرض كنيد:

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

n پاره خط باشند كه نقاط را به هم متصل مي‌نمايند؛ آنگاه ناحيه محصورشده توسط مجموعه $C = e_1, e_2, ..., e_n$ تشكيل يك چندضلعي ساده مي‌دهد؛ اگر و تنها اگر شرايط زير برقرار باشند.

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

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

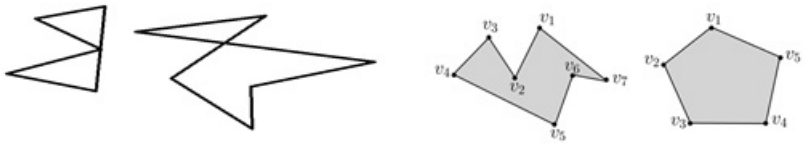
۲. هيچ دو پاره‌خط غيرمجاور يكديگر را قطع نكنند.

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

شرط اول تضمين مي‌كند كه مرز چندضلعي بسته باشد. به عبارت ديگر، انتهاي هر ضلع به ابتداي ضلع بعدي متصل شده باشد. شرط دوم نيز تضمين مي‌كند كه بين هيچ دو ضلع غير مجاور، تقاطع وجود نداشته باشد. بدين ترتيب چنانچه دو شرط فوق برقرار باشد، يک چندضلعي ساده ايجاد مي¬شود كه 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 تشكيل¬شده يک چندضلعي غير¬ساده مي¬باشد. مسائل هندسه¬ي محاسباتي اغلب بر چند¬ضلعي¬ها¬ي ساده تمركز مي¬كنند. در نتيجه ما نيز اين رويكرد را دنبال ميكنيم. شكل زير را ملاحظه نماييد. چنانچه پاره‌خط اول و پاره‌خط انتهايي اشتراكي نداشته باشند، يک زنجير باز [۶] خواهيم داشت. در حقيقت زنجير باز به صورت زير تعريف مي‌شود.
فرض كنيد

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

[۴] Edge
[۵] Vertex
[۶] Open Chain



شکل ۲.۳: شکل سمت چپ، دسته¬بندي چندضلعي¬ها به چندضلعي¬هاي 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 مي¬کند: ناحيه¬ي کران¬دار [۷] که داخل چندضلعي قرار دارد و ناحيه¬ي بي‌کران [۸] که بيرون چندضلعي واقع شده است. معمولاً در مسائل هندسهي محاسباتي، مرز [۹] چندضلعي نيز جزء ناحيه¬ي داخل چندضلعي محسوب مي¬شود. 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 دسته¬بندي مي¬توان چندضلعي¬هاي ساده را به دو دسته¬ي چندضلعي¬هاي محدب [۱۰] و چندضلعي¬هاي غيرمحدب [۱۱] طبقه‌بندي کرد. اگر چه در اين مورد بطور مفصل در بخش هاي بعدي صحبت خواهد شد. جهت آشنائي ابتدائي مي توان گفت چند¬ضلعي ساده¬ي P يک چندضلعي محدب است؛ اگر و تنها اگر هر پاره¬خطي که دو نقطه¬ي دلخواه از داخل P را به هم وصل مي¬کند، کاملاً داخل P قرار گيرد. در غير اين صورت P يک چندضلعي نامحدب است. از نگاهي ديگر، در چندضلعي محدب P، همه¬ي زواياي داخلي چندضلعي کوچکتر از π راديان مي¬باشند؛ در حالي که در يک چندضلعي نامحدب، حداقل يک زاويه¬ي 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.

رويت پذيری

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 مي‌كنند. در اين مسئله، رؤيت‌پذيري در هدايت ربات‌ها كاربرد پيدا مي‌كند.
مسئله مسيريابي در سيستم‌هاي اطلاعات جغرافيايي كه با يک شبكه گراف شبيه‌سازي مي‌شود نمونه ديگري از كاربردهاي رويت‌پذيري مي‌باشد. در اين مسئله، مسيريابي يک متحرک با كمک رويت‌پذيري صورت مي‌گيرد.
رويت‌پذيري در هندسه معماري نيز كاربرد دارد. طراحي پلان و ابعاد يک ساختمان، چشم انداز، جهت و نماي ساختمان و تقسيم‌بندي بخش‌هاي درون يک ساختمان با كمک 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 شبكه سنسور و طراحي سيستم امنيت براي ساختمان‌ها و نيز نورپردازي سالن‌ها و نماي ساختمان‌ها كاربرد دارد.
يک كاربرد 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 كاهش تعداد نگهباناني كه از نگارخانه مراقبت مي‌كنند، يكي از معروف‌ترين كاربردهاي رويت‌پذيري مي‌باشد. در بخش بعدي به اين مسئله خواهيم پرداخت. ادامه اين بخش ما را از مسيرمان مقداري دور مي كند 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 كار ما را مقداري از مسيرمان منحرف مي كند 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 كنيد يک نگارخانه با سالن‌ها و راهروهاي پيچ در پيچ وجود دارد كه محل نگهداري تابلوها و آثار هنري گران‌قيمت است. به دليل اهميت و ارزش آثار هنري، نگارخانه بايد به طور دائم و كامل توسط نگهبانان يا دوربين‌هاي مداربسته تحت كنترل قرار گيرد. همچنين نكات زير در مورد نگارخانه و نگهبانان آن مطرح است:
دوربين‌ها بايد به نحوي در مكان‌هاي مختلف نگارخانه نصب شوند كه تعدادشان كمينه شود؛ بدين وسيله، در هزينه‌هاي مراقبت از نگارخانه صرفه‌جويي مي‌شود. در عين حال، بايد تمام نقاط مراقبت شوند. بنابراين نبايد تعداد دوربين‌ها به اندازه‌اي كم باشد كه بخش‌هايي از نگارخانه توسط آن‌ها پوشش داده نشود. علاوه بر آن، هرچه تصاوير ارسالي از دوربين‌ها كمتر باشد، نظارت و كنترل آن‌ها راحت‌تر خواهد بود. در واقع در بعضي مسائل، كاهش 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 كرد.
مسئله‌ي نگارخانه هنر ابتدا در سال ۱۹۷۳ توسط كلي [۱۲] مطرح شد. او سؤال زير را مطرح كرد:
"چه تعداد نگهبان براي كنترل و حفاظت از يک نگارخانه مورد نياز است؟"
در مسئله كلاسيک 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 كه قادر است تمامي جهات را ببيند و داراي دامنه‌ي ديد ۳۶۰ درجه و با شعاع ديد بي‌نهايت مي‌باشد.
- نگهبان‌ها نمي‌توانند پشت ديوارها را ببينند.

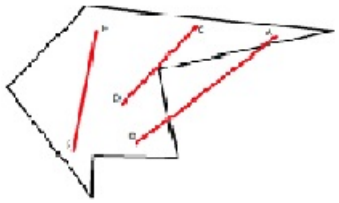
مسائل جذاب، ارزشمند و در عين حال كاربردي در زمينه نگارخانه هنر وجود دارد كه بعضي شرايط فوق را ناديده مي‌گيرند. به عنوان مثال، نگارخانه مي‌تواند ساده نباشد، بلكه تعدادي ستون در ميانه سالن‌ها و راهروها وجود داشته باشد. در اين حالت چندضلعي ساده نخواهد بود؛

[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.

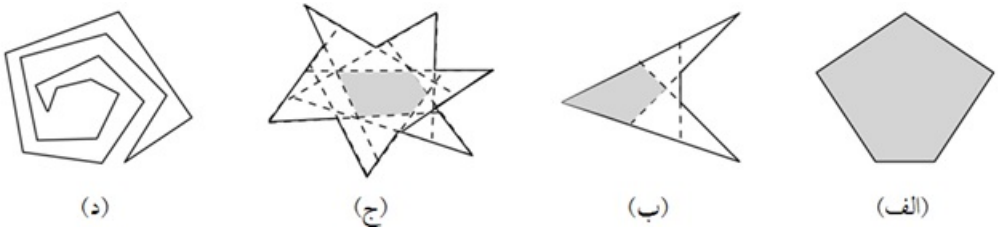
$$x\ can\ see\ y\ iff\ \overline{xy} \subseteq P$$

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 یك 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.

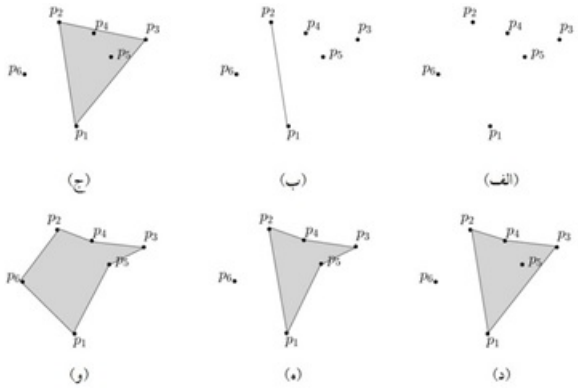
چندضلعي ستاره‌شكل: يك چندضلعي را ستاره‌شكل [۱۷] گوييم اگر هسته آن ناتهي باشد.

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 آلگوريتم به صورت زير است.

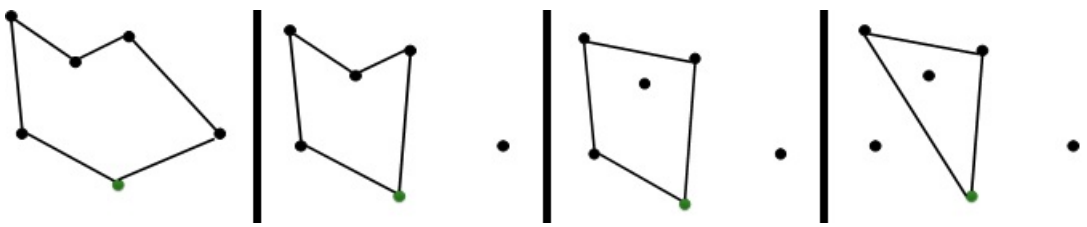
- ابتدا با نقطه $s = s_0$ شروع مي‌كنيم.
- سپس با كمک نقاط ورودي دوم و سوم، يک مثلث مي‌سازيم كه ستاره‌اي شكل است و شامل نقطه s_0 نيز مي‌باشد.
- فرض كنيد تا نقطه i ام اضافه و با كمک اين نقاط تا كنون چندضلعي ستاره‌شكل ساخته شده است. اكنون در مرحله i ام هستيم و نقطه i ام وارد شده است. نقطه s_i به گونه‌اي به چندضلعي كه تا كنون ساخته شده است، اضافه مي‌شود كه چندضلعي جديد ستاره‌شكل باقي بماند. براي اين كار، مرز چندضلعي موجود را از s_1 بصورت پادساعت‌گرد حول s_0 دور مي‌زنيم تا به رأسي مانند s_j برسيم كه زاويه پادساعت‌گرد نقطه s_i نسبت به s_0 و خط افق بلافاصله كمتر 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 كه دور به پايان رسيد و چنين رأسي يافت نشد، در اين حالت زوايه 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 كوچكتر از s_q باشد.

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

گوئيم نقطه $s_p = (r_p, \theta_p)$ كوچكتر 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 كوچک به بزرگ مرتب مي‌شوند.

فرض كنيد در مرحله iام چندضلعي ستاره‌شكل به صورت $\{s_0, s_1, \ldots, s_{i-1}\}$ باشد كه 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 كه زاويه آن بلافاصله بزرگتر از s_i است، چندضلعي ستاره‌شكل به صورت $s_0, s_1, \ldots, s_{j-1}, s_i, s_j, \ldots, s_{i-1}$ در مي‌آيد. اگر چنين s_j اي وجود نداشت، s_i به آخر ليست اضافه مي‌شود. در اين صورت چندضلعي ستاره‌شكل به صورت $s_0, s_1, \ldots, s_{i-1}, s_i$ در مي‌آيد.



شكل ۶.۳: اجراي مراحل الگوريتم ساخت چندضلعي ستاره‌اي شكل

صحت الگوريتم

اثبات: مراحل ساختن چندضلعي ستاره‌شكل 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؛ كه به راحتي مي‌شود ثابت كرد همواره حداقل s_0 تمام رئوس از جمله رأس جديد s_i را مي‌بينند.

آناليز الگوريتم

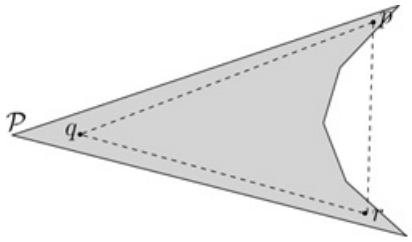
يافتن چندضلعي ستاره‌شكل با كمک الگوريتم افزايشي درجي همانند مرتب‌سازي درجي مي‌باشد. تمام نقاط يكي يكي در محل مناسب به نقاط قبلي پوش محدب اضافه مي‌شوند. انتخاب محلي مناسب براي نقطه جديد با مقايسه زاويه پادساعت‌گرد آن نقطه نسبت به s_0 با نقاط قبلي چندضلعي ساخته‌شده صورت مي‌گيرد. اين مقايسه در مرحله iام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

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

تمرين

مسئله ۱ (ساده): نشان دهيد رويت‌پذيري داراي خاصيت تقارن [۲۰] است. بدين معني كه 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 كه اگر نقطه‌ي p نقطه‌ي q و نقطه‌ي q نقطه‌ي r را در p رويت كند، لزوما نقطه‌ي p نقطه‌ي r را رويت نمي‌كند. شكل ۶ را ملاحظه نماييد.



شكل ۷.۳: رويت‌پذيري خاصيت تعدي ندارد.

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

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

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

---

[۲۰] Symmetry

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

مسئله۶: (اثبات و یا رد کنید) هسته یک 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 بهيته 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 دسته¬بندي اوليه مي¬توان چندضلعي¬هاي ساده را به دو دسته¬ي چندضلعي¬هاي 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)