بسم الله الرَّحمن الرَّحیم

مکان یابی نقاط

زهره معظمی

دانشگاه یزد

مساله مکان یابی

- تهیه یک نقشه از محل
- استفاده از ستارگان و خورشید برای مکان یابی
- استفاده از دستگاه های مکان یابی که موقعیت یک نقطه را با استفاده از اطلاعات ماهواره ها و با دقت لازم تعیین می کنند.
- مکان یابی خودکار توسط دستگاه های الکترونیکی. نقشه به صورت الکترونیکی ذخیره می شود و مکان یابی به طور خودکار انجام می شود.
- اگر هدف رصد مسیر حرکت باشد، باید مکان یابی چندین بار در فواصل زمانی معین تکرار شود.

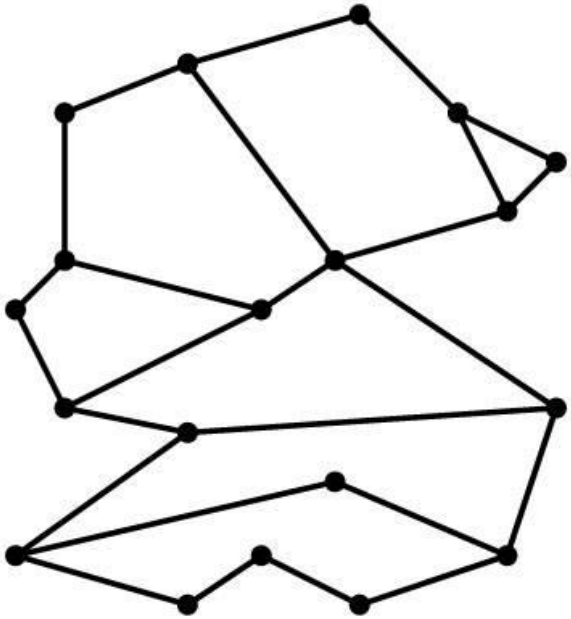
صورت مساله

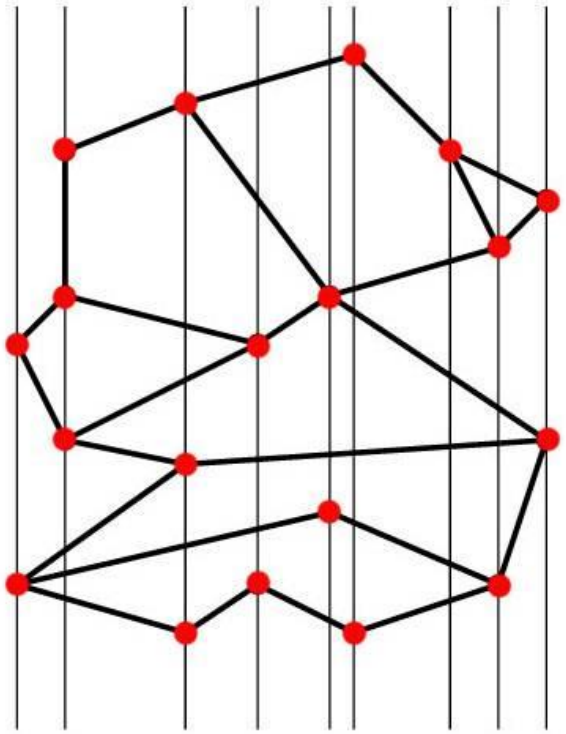
- ورودی: یک نقشه و یک نقطه q است.
- خروجی: تعیین ناحیه ای از نقشه است که q در آن قرار دارد.

توجه: نقشه یک زیر تقسیم از صفحه است.

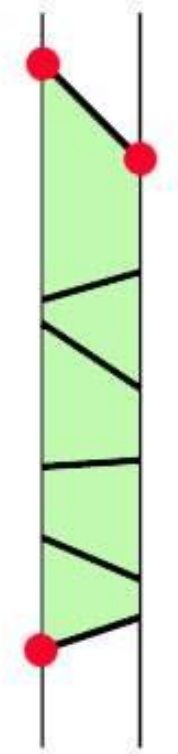
مکان یابی با نقشه ذوزنقه ای

- در هر راس از زیر تقسیم یک خط عمودی رسم می کنیم
- در هر نوار عمودی یال هایی وجود دارند که یکدیگر را قطع نمی کنند.





مکان یابی با نقشه ذوزنقه ای



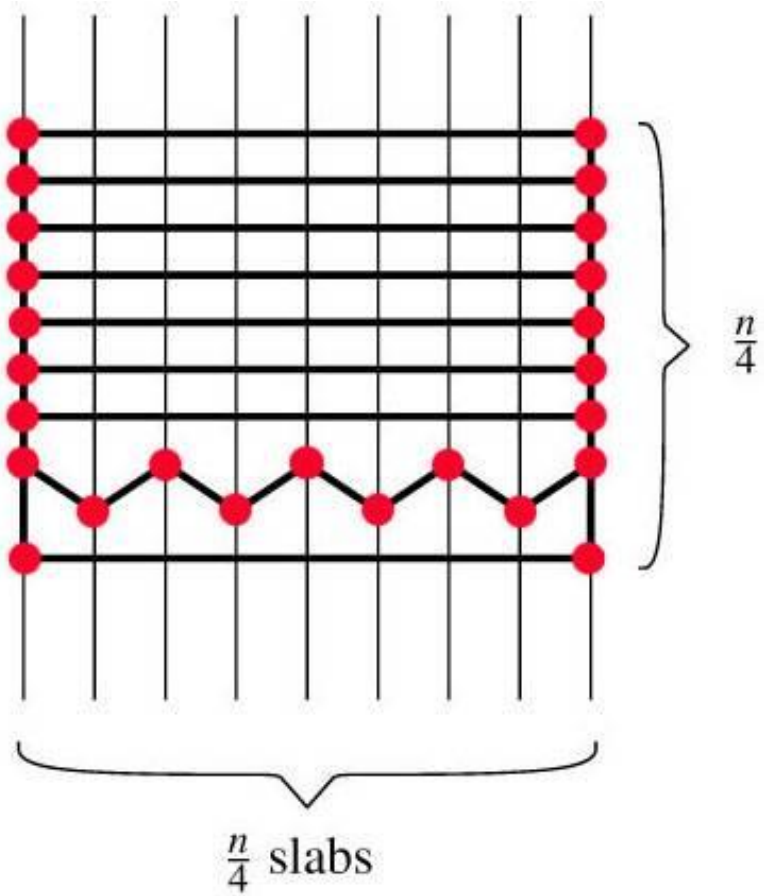
- در هر راس از زیر تقسیم یک خط عمودی رسم می کنیم
- در هر نوار عمودی یال هایی وجود دارند که یکدیگر را قطع نمی کنند.
- هر ناحیه از نوار در دقیقا یک وجه از زیر تقسیم قرار دارد.
- دو ناحیه بیکران در بالا و پایین قرار دارند.
- یال های یک نوار را به ترتیب در یک آرایه مرتب می کنیم.
- هر یال نوار را با وجهی از زیر تقسیم که بالای آن قرار دارد برچسب می دهیم.

الگوریتم

- جستجوی دودویی با مختص x برای تعیین نواری که نقطه در آن است. O(log n)
- جستجوی دودویی با مختص y برای تعیین پاره خطی که نقطه در بالای آن قرار دارد. O(log n)
- تعیین ناحیه با استفاده از برچسبی که در این پاره خط ذخیره شده است. O(1)
- زمان جستجو: O(log n)
- حافظه لازم:
- O(n) نوار و هر نوار O(n) پاره خط دارد. $O(n^2)$

تحلیل

- این کران بالا ممکن است اتفاق بیفتد.
  - زیر تقسیمی با n راس n/4 نوار که هر کدا n/4 ناحیه دارند.

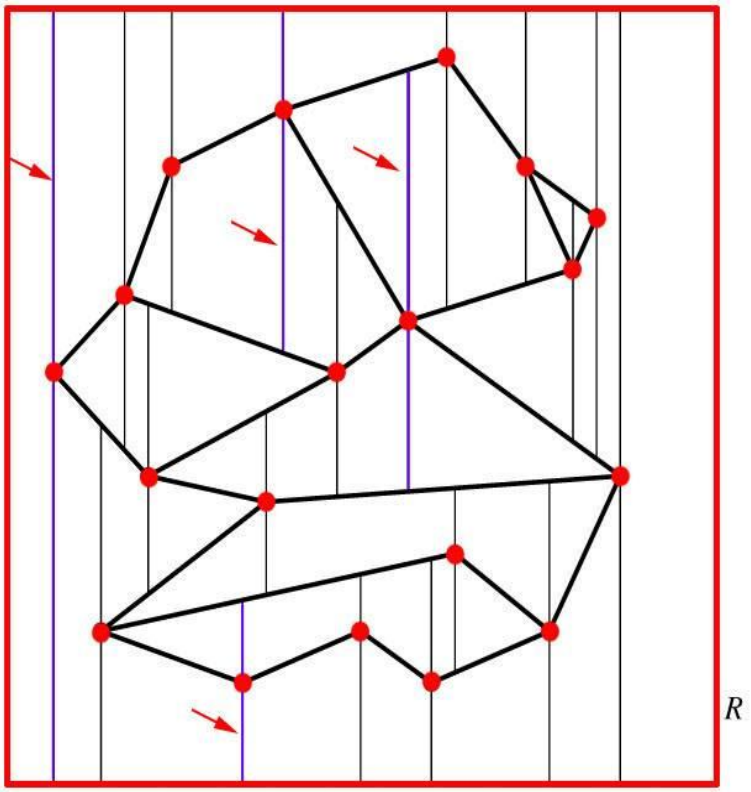


- در این روش زیر تقسیم را به یک زیر تقسیم دیگر تبدیل کردیم که جستجو در آن ساده تر است اما این زیر تقسیم دارای $O(n^2)$ وجه است.

راه حل بهتر

- استفاده از روش ذوزنقه ای برای کم کردن تعداد وجه ها.
- دو پاره خط را غیر متقاطع می نامیم هرگاه تقاطع نداشته باشند و یا در یک راس انتهایی مشترک متقاطع باشند.
  - یال های زیر تقسیم تقاطع ندارند.
- فرض می کنیم هیچ دونقطه انتهایی از پاره خط ها دارای مختص X یکسان نباشند. (پاره خط ها در موقعیت کلی هستند)
- برای زیر تقسیم S، زیر تقسیمی که از افراز ناحیه های S به ناحیه های کوچکتری به دست می آید، یک افراز S نام دارد.

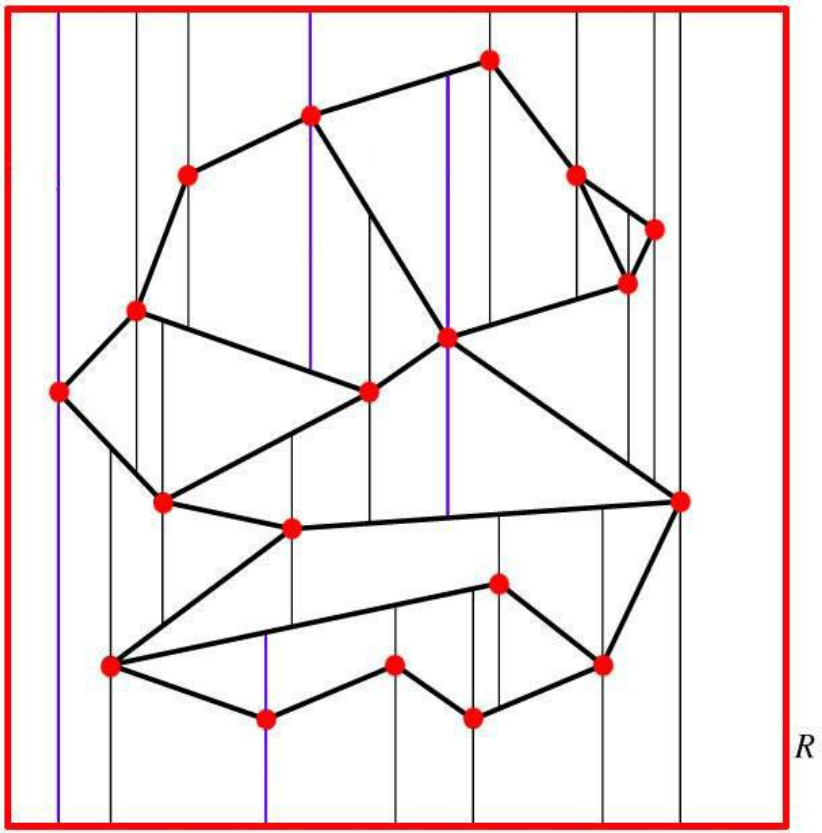
نقشه ذوزنقه ای

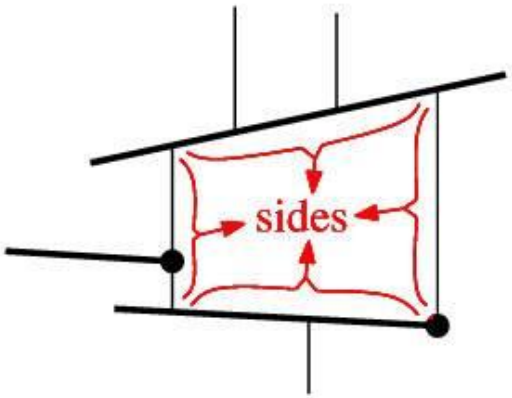


- زیر تقسیم را با یک مستطیل R محدود می کنیم.
- از راس زیر تقسیم دو پاره خط عمودی یکی به طرف بالا و یکی به طرف پایین رسم می شود.
- پاره خط ها تا جایی رسم می شوند که به یک یال S و یا مستطیل R برسند.

افراز ذوزنقه ای

- ممکن است چند پاره خط هم راستا و مجاور یک وجه از T(S) را محدود کنند اجتماع چنین پاره خط هایی را یک کناره (side) وجه می نامیم.

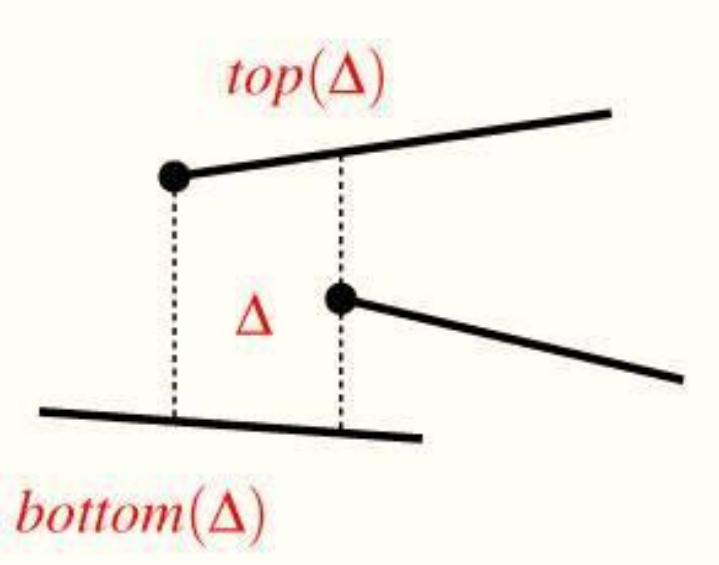




- قضیه: هر وجه از یک افراز ذوزنقه ای از یک مجموعه S از پاره خط های در موقعیت کلی دارای یک یا دو کناره عمودی و دقیقا دو کناره غیر عمودی است.
  - اثبات:
  - اگر f وجهی از T(S) باشد، f محدب است زیرا همه زوایه های داخلی کمتر از ۱۸۰ درجه است.
  - پس f حداکثر دو کناره عمودی دارد.
  - اگر f بیش از دو کناره غیر عمودی داشته باشد دوتای آنها در بالا یا پایین قرار دارند. یال های زیر تقسیم تقاطع ندارند، پس این دو پاره خط در یک راس انتهایی مشترک به هم می رسند. که در آن یک گسترش عمودی رسم شده است و در نتیجه هر دو پاره خط در یک وجه نیستند.
  - f کراندار است، پس حداقل دو کناره غیر عمودی دارد.

افراز ذوزنقه ای

- اگر Δ یک وجه از افراز ذوزنقه ای باشد کناره های بالا و پایین آن را به ترتیب $top(\Delta)$ و $bottom(\Delta)$ می نامیم.



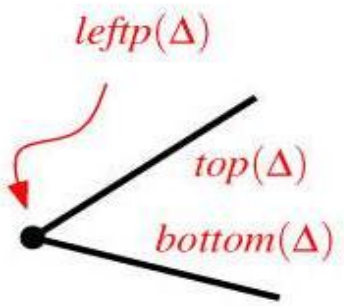
افراز ذوزنقه ای

- برای کناره های سمت راست و چپ غیر عمودی می توان حالتهای Δ مختلفی در نظر گرفت. یک راس روی مرز سمت چپ (راست) Δ را به عنوان راس سمت چپ (راست) تعیین می کنیم.

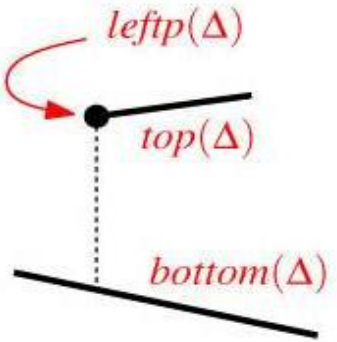
- هر وجه Δ با داشتن $top(\Delta)$، $botton(\Delta)$، $rightp(\Delta)$ و $leftp(\Delta)$ به طور منحصر به فرد تعیین می شود.

افراز ذوزنقه ای

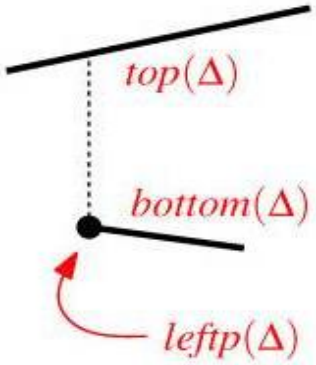
هر وجه با داشتن top(Δ) ، botton(Δ)،rightp(Δ) و leftp(Δ) به طور منحصربه فرد تعیین می شود.



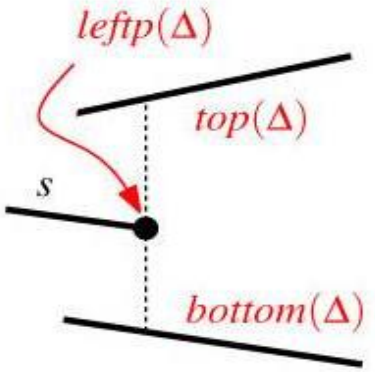
(a)



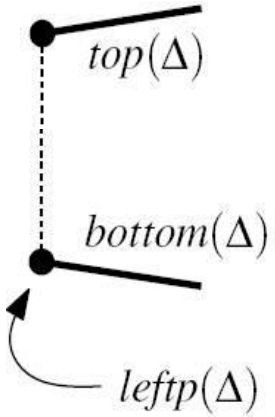
(b)



(c)



(d)



افراز ذوزنقه ای

قضیه: افراز ذوزنقه ای $T(S)$ از مجموعه S با n پاره خط در موقعیت کلی حداکثر $6n+4$راس و حداکثر $3n+1$ ذوزنقه دارد.

اثبات:

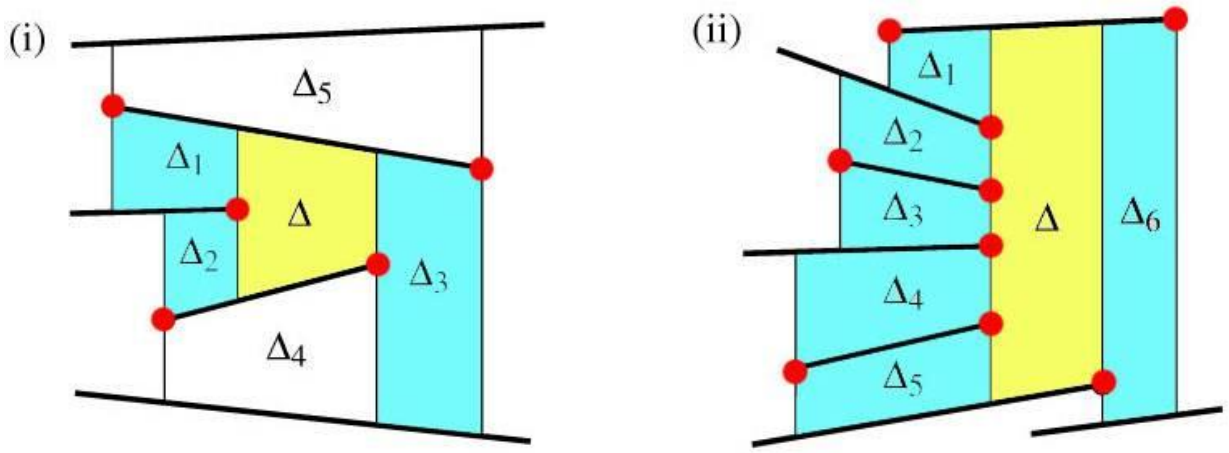
هر راس افراز ذوزنقه ای راسی از S، راسی از R یا راسی در محل تقاطع یک گسترش عمودی و یک پاره خط است. پس تعداد راس ها برابر است با:

$$4+2n+2(2n)=6n+4$$

تعداد وجه ها با تعداد راس های سمت چپ آنها برابر است. راس انتهای سمت چپ هر پاره خط از S برای حداکثر ۲ وجه و راس سمت راست آن برای حداکثر یک وجه به عنوان leftp انتخاب می شود . تنها یک راس از R به عنوان leftp انتخاب می شود. پس تعداد وجه ها از $3n+1$ بیشتر نمی شود.

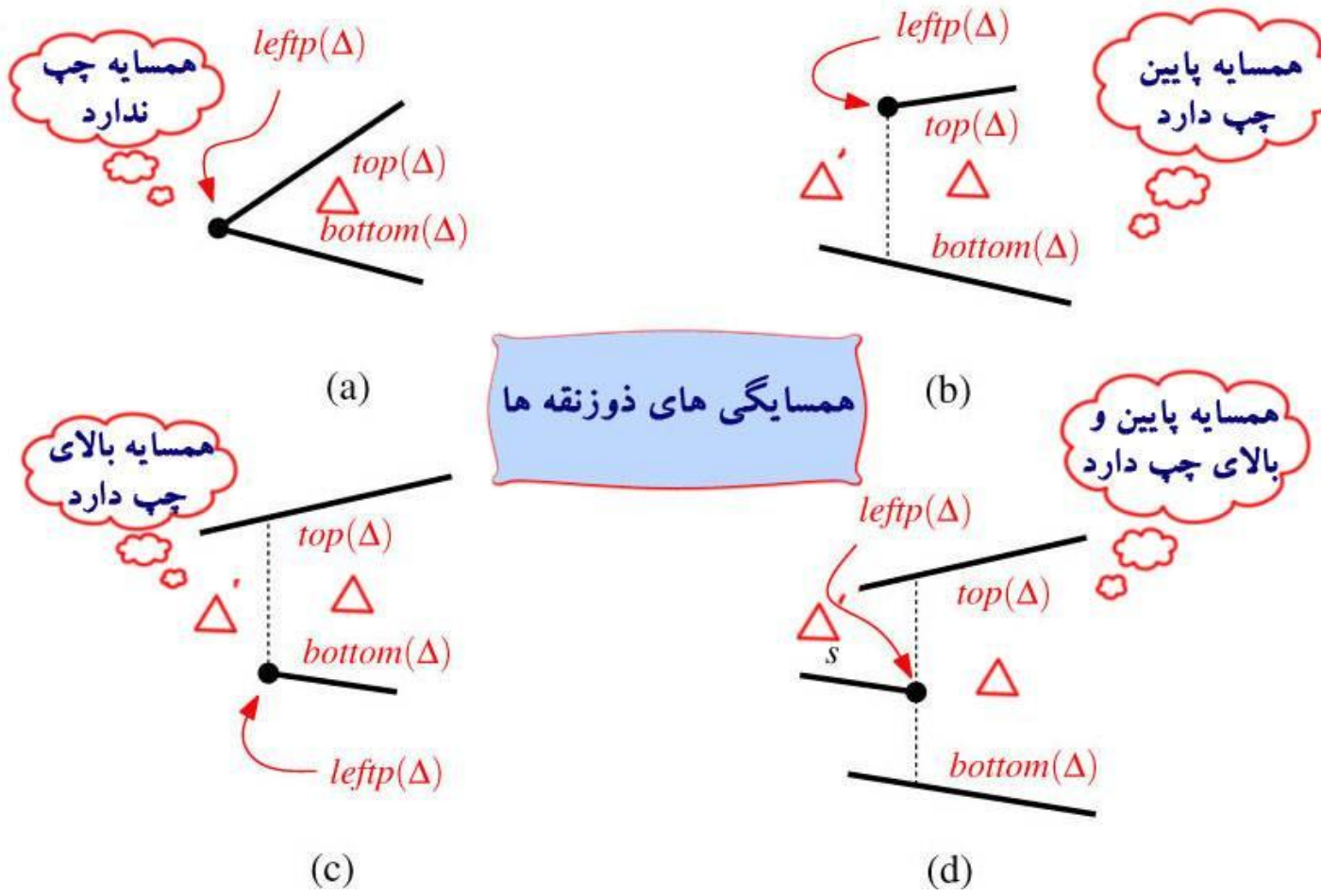
افراز ذوزنقه ای

- اگر Δ یک وجه از افراز ذوزنقه ای باشد، وجه Δ' را مجاور با Δ می نامیم هرگاه Δ و Δ' یک یال عمودی مشترک داشته باشند.



- با توجه به این که پاره خط ها در موقعیت کلی هستند هر ذوزنقه با حداکثر ۴ ذوزنقه دیگر مجاور است.

افراز ذوزنقه ای



افراز ذوزنقه ای

- روش اول:
  - یک افراز ذوزنقه ای یک زیر تقسیم است پس می توان آن را با یک لیست دو همبند یالی ذخیره کرد.
- روش دیگر:
  - با توجه به اینکه هر ذوزنقه Δ به طور منحصربه فردی با top، bottom، leftp و rightp مشخص می شود، برای هر ذوزنقه Δ یک لیست متشکل از top، bottom، leftp و rightp و اشاره گر هایی به ۴ همسایه آن در نظر می گیریم.
  - در این ساختار راس های ذوزنقه را می توان در زمان ثابت تعیین کرد.

ساختن افراز ذوزنقه ای

- الگوریتم تصادفی افزایشی:
  - ورودی: یک مجموعه S از پاره خط ها که در موقعیت کلی قرار دارند.
  - خروجی: یک افراز ذوزنقه ای T(S) با ساختمان داده D که برای مکان یابی در افراز ذوزنقه ای به کار می رود.

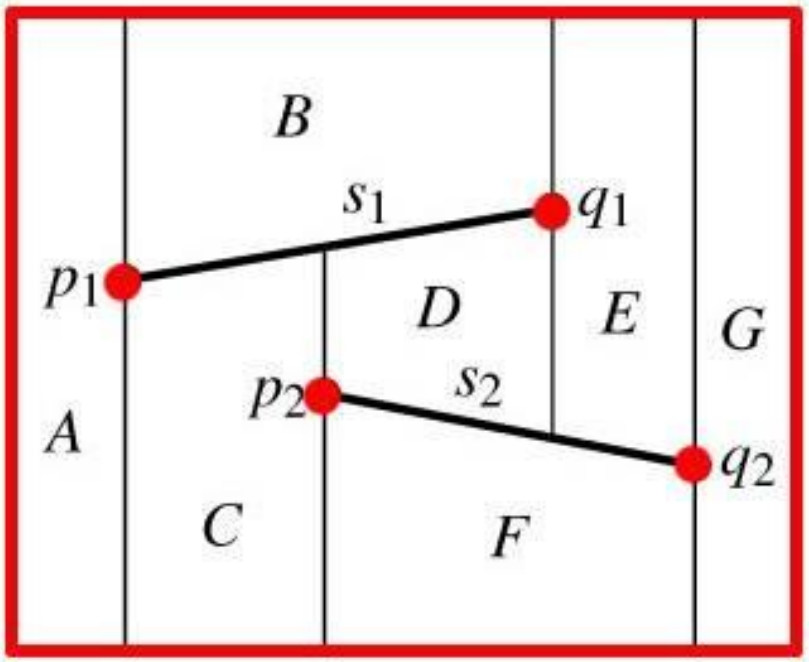
ساختن افراز ذوزنقه ای

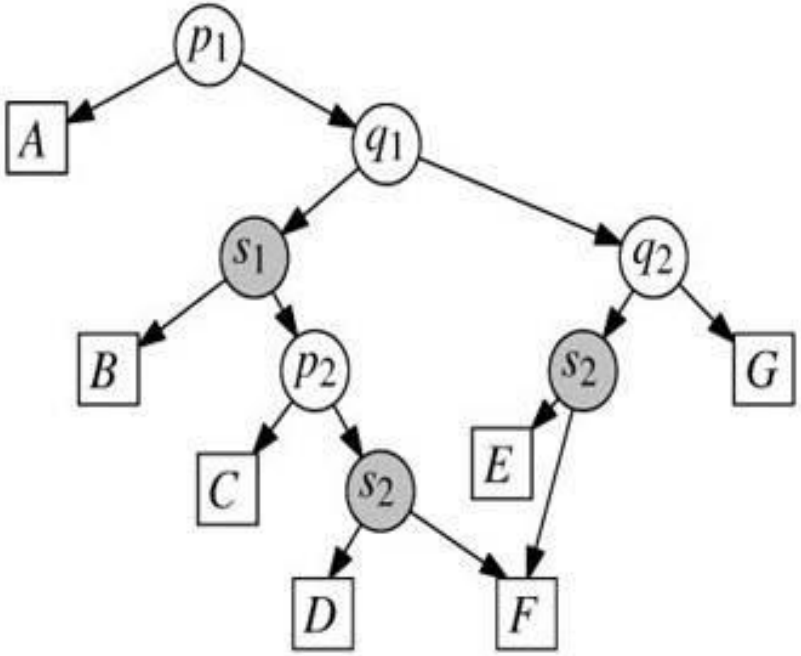
- ساختمان داده D یک گراف جهتدار بدون دور، با یک راس یکتای ریشه و یک برگ به ازای هر ذوزنقه از افراز ذوزنقه ای S است.

- دو نوع راس داخلی:
  - x –راس ها: با یک راس انتهایی پاره خطی از S برچسب دارد.
  - y –راس ها: با یک پاره خط برچسب گذاری شده اند.

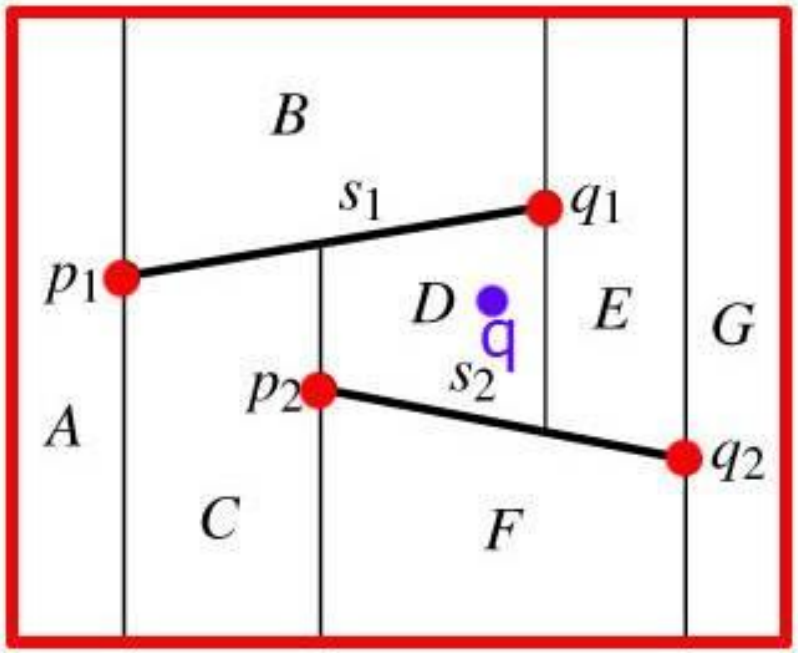
- هر راس داخلی درجه خروجی ۲ دارد.

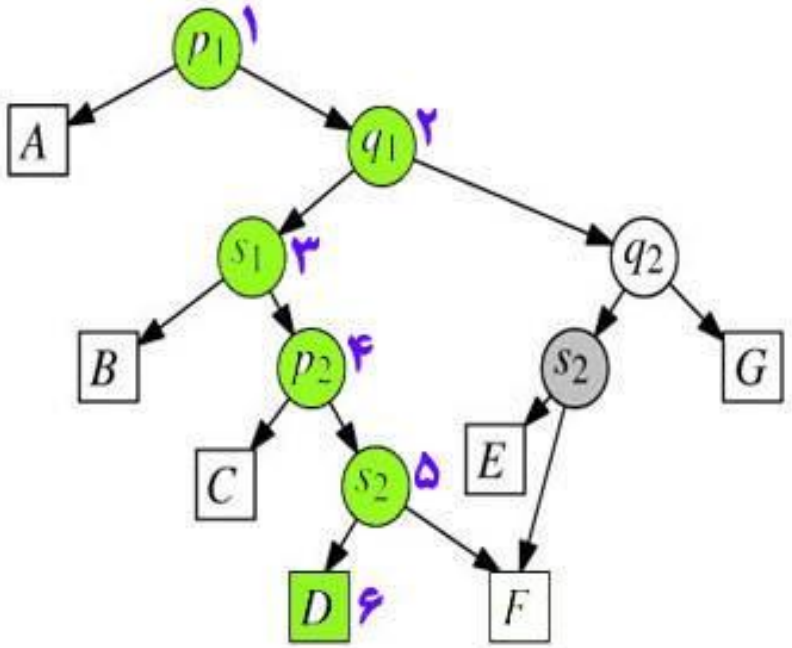
جستجوی یک نقطه در D





جستجوی یک نقطه در D





Algorithm TRAPEZOIDALMAP(S)

Input. A set S of n non-crossing line segments.

Output. The trapezoidal map $\mathcal{T}(S)$ and a search structure $\mathcal{D}$ for $\mathcal{T}(S)$ in a bounding box.

1. Determine a bounding box R that contains all segments of S, and initialize the trapezoidal map structure $\mathcal{T}$ and search structure $\mathcal{D}$ for it.
2. Compute a random permutation $s_1, s_2, \ldots, s_n$ of the elements of S.
3. **for** $i \leftarrow 1$ **to** n
4. **do** Find the set $\Delta_0, \Delta_1, \ldots, \Delta_k$ of trapezoids in $\mathcal{T}$ properly intersected by s_i.
5. Remove $\Delta_0, \Delta_1, \ldots, \Delta_k$ from $\mathcal{T}$ and replace them by the new trapezoids that appear because of the insertion of s_i.
6. Remove the leaves for $\Delta_0, \Delta_1, \ldots, \Delta_k$ from $\mathcal{D}$, and create leaves for the new trapezoids. Link the new leaves to the existing inner nodes by adding some new inner nodes, as explained below.

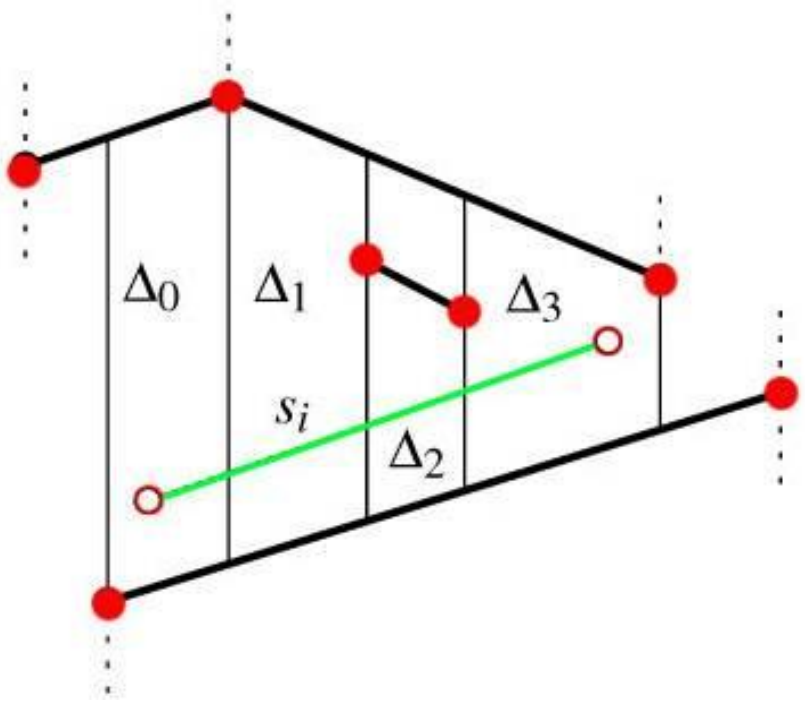
بررسی الگوریتم

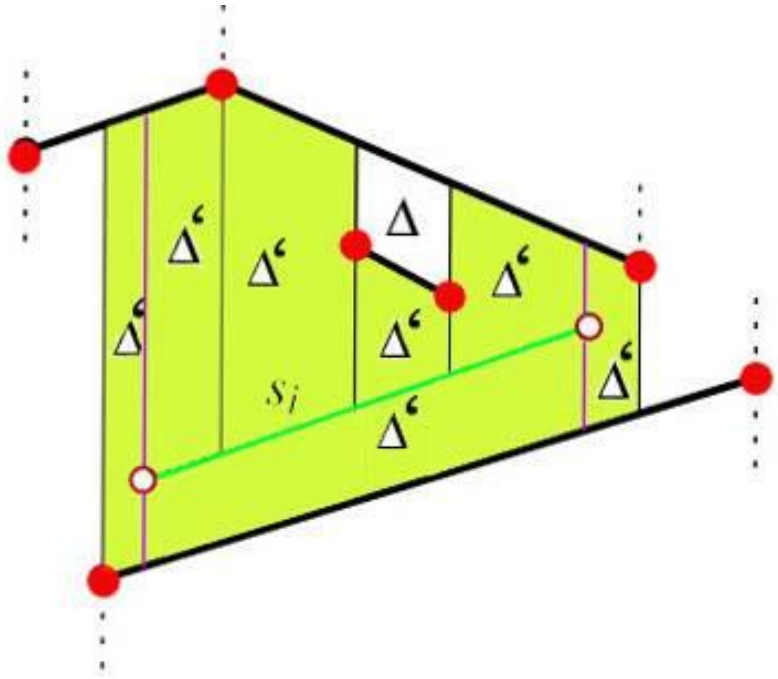
- فرض کنید $S_i:=\{s_1, s_2, \ldots, s_i\}$
- در ابتدا $T=T(S_0)=T(\emptyset)$ شامل تنها یک ذوزنقه است که همان مستطیل R است.
- بعد از اضافه کردن پاره خط s_i افراز $T(S_{i-1})$ به $T(S_i)$ تبدیل می شود.
- $\Delta_0, \Delta_1, \ldots, \Delta_k$ ذوزنقه هایی از $T(S_{i-1})$ که s_i آنها را قطع می کند (به ترتیب از چپ به راست)
  - هر ذوزنقه Δ_{j+1} همسایه سمت راست Δ_j است.
  - با بررسی این که $\text{rightp}(\Delta_j)$ بالا یا پایین s_i قرار دارد، مشخص می شود Δ_{j+1} همسایه بالا یا پایین است.

بررسی الگوریتم

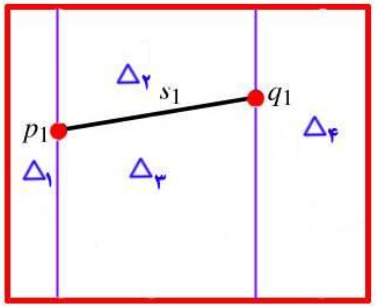
- پس کافی است ذوزنقه Δ_0 که راس انتهای چپ s_i (که آن را p می نامیم) در آن قرار دارد تعیین شود. بقیه ذوزنقه ها با بررسی ساختار افراز ذوزنقه ای تعیین می شوند.
- اگر p داخل یک ذوزنقه باشد با جستجوی D این ذوزنقه را پیدا می کنیم.
- اگر p روی یک راس و یا یال از ذوزنقه باشد قرارداد می کنیم
  - اگر p روی پاره خط عمودی در یک x–راس باشد می گوییم p در سمت راست آن قرار دارد.
  - اگر p روی یک ضلع s باشد شیب s و s_i را مقایسه می کنیم. اگر شیب s_i بزرگتر باشد می گوییم p بالای s قرار دارد و برعکس.

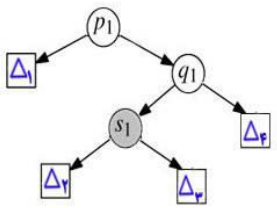
هر ذوزنقه Δ_{j+1} همسایه سمت راست Δ_j است.
با بررسی این که rightp(Δ_j) بالا یا پایین s_i قرار دارد، مشخص می شود Δ_{j+1} همسایه بالا یا پایین است.

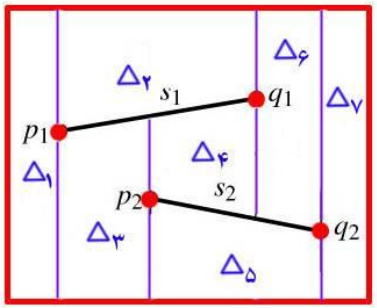


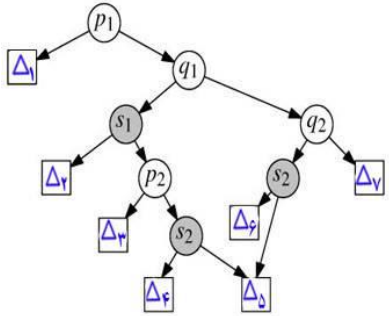



R


Δ_2
s_1
q_1
p_1
Δ_4
Δ_1
Δ_3


p_1
Δ_1
q_1
s_1
Δ_4
Δ_2
Δ_3


Δ_6
Δ_2
s_1
q_1
Δ_7
p_1
Δ_4
Δ_1
p_2
s_2
Δ_3
q_2
Δ_5


p_1
Δ_1
q_1
s_1
q_2
Δ_2
p_2
s_2
Δ_7
Δ_3
s_2
Δ_6
Δ_4
Δ_5

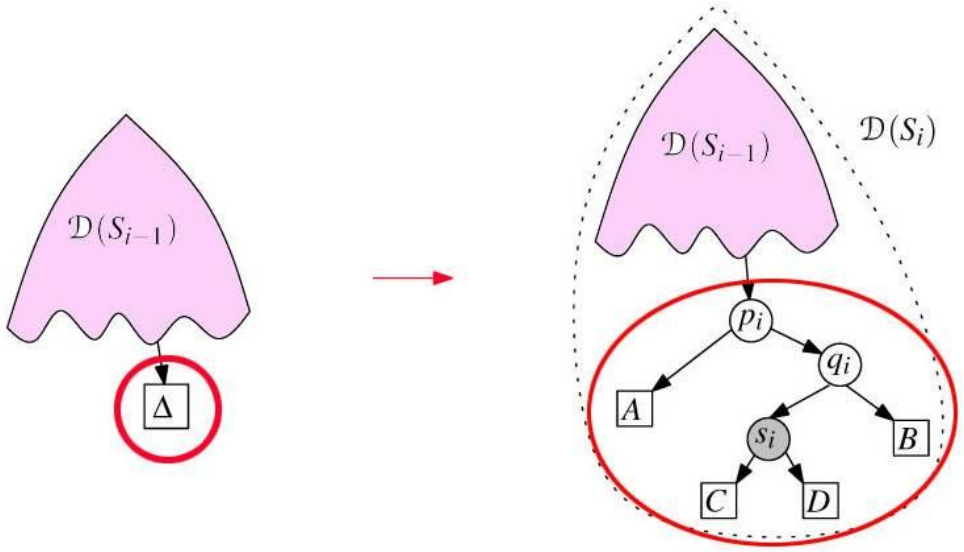
الگوریتم پیدا کردن $\Delta_0, \Delta_1, \ldots, \Delta_k$

Algorithm FOLLOWSEGMENT($\mathcal{T}, \mathcal{D}, s_i$)
Input. A trapezoidal map $\mathcal{T}$, a search structure $\mathcal{D}$ for $\mathcal{T}$, and a new segment s_i.
Output. The sequence $\Delta_0, \ldots, \Delta_k$ of trapezoids intersected by s_i.

1. Let p and q be the left and right endpoint of s_i.
2. Search with p in the search structure $\mathcal{D}$ to find Δ_0.
3. $j \leftarrow 0$;
4. **while** q lies to the right of $rightp(\Delta_j)$
5. **do if** $rightp(\Delta_j)$ lies above s_i
6. **then** Let Δ_{j+1} be the lower right neighbor of Δ_j.
7. **else** Let Δ_{j+1} be the upper right neighbor of Δ_j.
8. $j \leftarrow j + 1$
9. **return** $\Delta_0, \Delta_1, \ldots, \Delta_j$

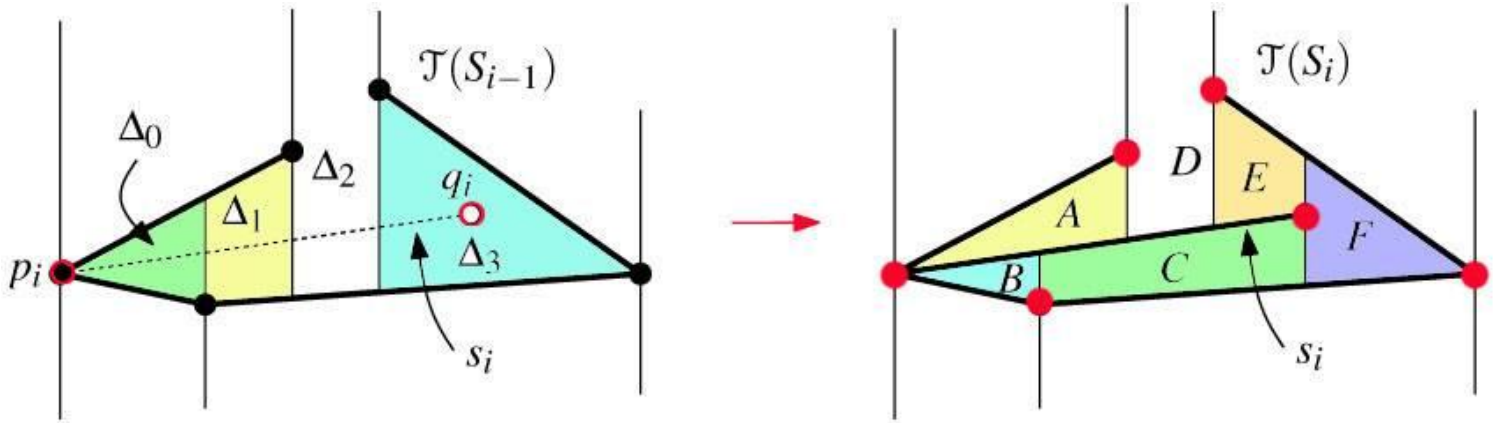
- در مرحله بعد باید D و T را به روز کنیم.
- T را می توان در زمان ثابت به روز کرد.

- برگ Δ از D حذف می شود و به جای آن یک درخت جایگزین می شود.



به روز کردن ساختار جستجوی D در حالت کلی

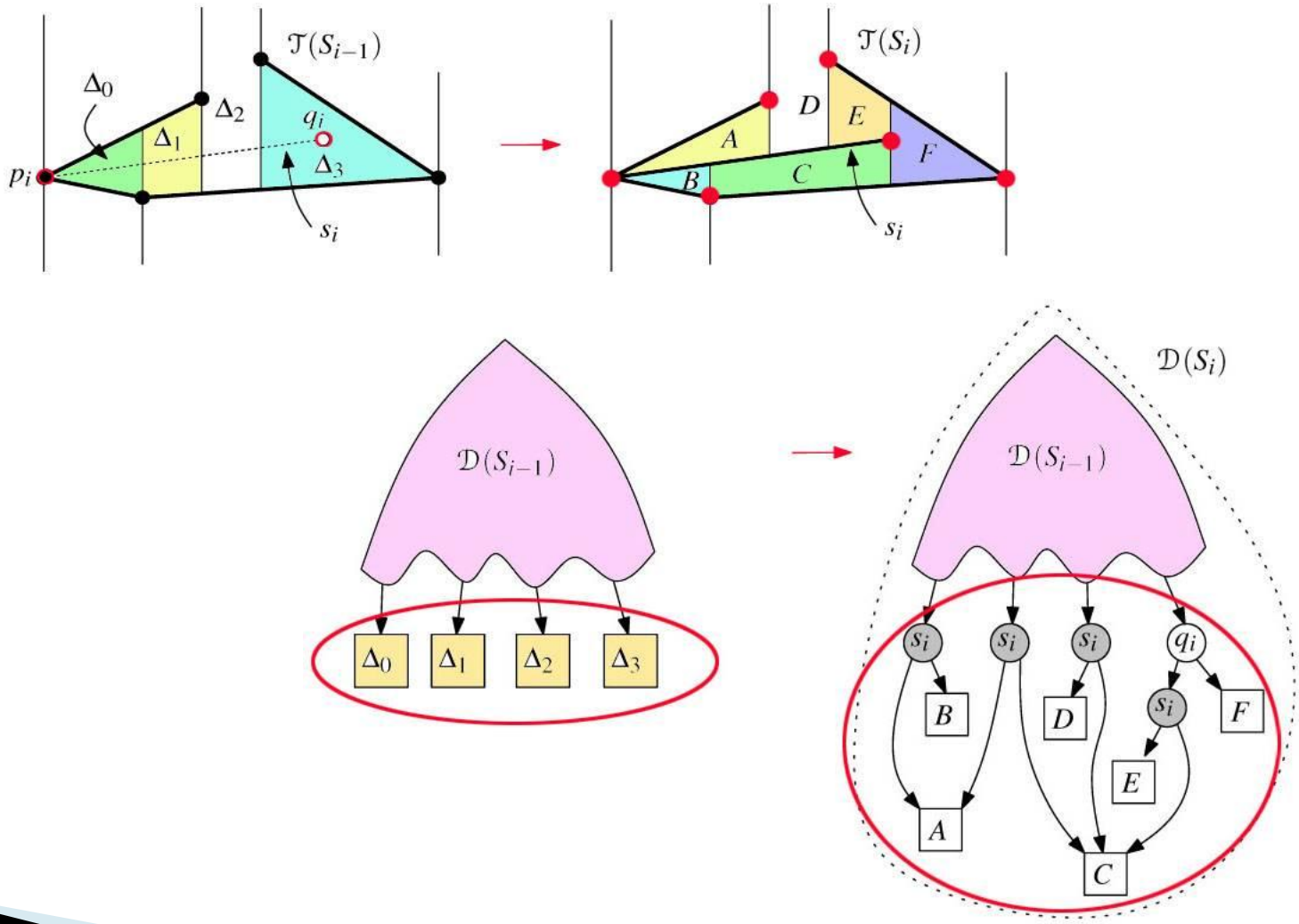
- در حالتی که یکی یا هر دو انتهای s_i بر leftp با rightp منطبق باشند، تنها سه ذوزنقه جدید اضافه می شود.
- در حالتی که s_i بیش از یک ذوزنقه را قطع کند
  - دو گسترش عمودی از دو انتهای s_i رسم میکنیم.
  - پاره خط های عمودی که s_i آنها را قطع می کند کوتاه می کنیم.



به روز کردن ساختار جستجوی D در حالت کلی

- برگهای $\Delta_0, \Delta_1, \ldots, \Delta_k$ را حذف می کنیم.
- اگر راس انتهای سمت چپ s_i در داخل Δ_0 باشد، به جای آن یک x–راس به عنوان انتهای چپ s_i و یک y–راس برای s_i می دهیم.
- اگر راس انتهای سمت راست s_i در داخل Δ_k باشد، به جای آن یک x–راس به عنوان انتهای راست s_i و یک y–راس برای s_i می دهیم.
- برگ های $\Delta_0, \Delta_1, \ldots, \Delta_k$ را با یک y–راس برای s_i جایگزین می کنیم و برگ های این راس ها را با توجه به افراز ذوزنقه ای می سازیم.
- ممکن است یک برگ بیش از یک یال ورودی داشته باشد.

به روز کردن ساختار جستجوی D در حالت کلی



قضیه:

الگوریتم افراز ذوزنقه ای برای یک مجموعه S از n پاره خط در موقعیت کلی $T(S)$ و ساختار جستجوی D را در زمان مورد انتظار $O(n \log n)$ تولید می کند. اندازه مورد انتظار ساختار جستجو $O(n)$ است و زمان مورد انتظار برای جستجوی هر نقطه q برابر با $O(\log n)$ است.

اثبات:

زمان جستجو در ساختار D

فرض q، نقطه ثابت جستجو باشد. زمان جستجوی، q در طول مسیر D خطی است. (تعداد ذوزنقه ها حداکثر 3n+1 است). در بدترین حالت عمق گراف در هر تکرار ۳ نود اضافه می شود.

در D هر نود در مسیر جستجوی q در برخی تکرارهای الگوریتم ایجاد می شود.

X_i که $0<=i<=n$ است تعداد نودهای ایجاد شده روی مسیر است که در تکرار iام ایجاد شده اند.

s و q ثابت هستند.

X_i: متغیر تصادفی است که به ترتیب تصادفی پاره خط ها بستگی دارد.

ادامه اثبات:

طول مسیر مورد انتظار:

$$E(\sum_{i=1}^{n} X_i) = \sum_{i=1}^{n} E(X_i)$$

$$E(X) = \sum f(X)\, X$$

$$f(X_i) = \begin{cases} 1 & if\ \ X_i\ created\ in\ itration\ i \\ 0 & O.W \end{cases}$$

در هر تکرار حداکثر ۳ نود به مسیر جستجو اضافه می شود. یعنی $X_i \leq 3$

P_i: احتمال وجود نود روی مسیر جستجوی q که در تکرار iام ایجاد شده است.

$$E(X_i) \leq 3\, P_i$$

احتمال ایجاد تغییر در ساختار D:

$$P_i = \Pr[\Delta_q(S_i) \neq \Delta_q(S_{i-1})]$$

ادامه اثبات:

توجه: تمام ذوزنقه هایی که در تکرار iام ایجاد می شوند همسایه s_i هستند.

فرض $S_i \subset S$ است.

نقشه ذوزنقه ای $T(S_i)$ و $\Delta_q(S_i)$ بطور یکتا تابعی از S_i را تعریف میکند.

پس با درج s_i ممکن است محدوده ذوزنقه شامل q تغییر کند.

حال با استفاده از تحلیل backward زمان مورد انتظار را بدست می آوریم.

ادامه اثبات:

ابتدا $T(S_i)$ را در نظر گرفته و احتمال اینکه $\Delta_q(S_i)$ از نقشه حذف شود را می یابیم.

با حذف پاره خط s_i، $\Delta_q(S_i)$ نیز حذف می شود اگر و تنها اگر یکی از همسایگی های $\Delta_q(S_i)$ با حذف s_i حذف شوند.

پاره خط های S_i بصورت تصادفی درج می شوند بنابراین احتمال هر پاره خط در S_i مساوی است.

احتمال رخ دادن s_i در $top(\Delta_q(s_i))$ برابر $1/i$ است و اگر $top(\Delta_q(s_i))$ یال بالایی R باشد احتمالش صفر است. در نتیجه:

$$P_i = \Pr[\Delta_q(S_i) \neq \Delta_q(S_{i-1})] = \Pr[\Delta_q(S_i) \notin T(S_{i-1})] \leq \frac{4}{i}$$

ادامه اثبات:

با استفاده از روابط قبل

$$E(\sum_{i=1}^{n} X_i) \leq \sum_{i=1}^{n} 3P_i \leq \sum_{i=1}^{n} \frac{12}{i} \leq 12 \sum_{i=1}^{n} \frac{1}{i} = 12\, Hn$$

که

$$Hn = \sum_{i=1}^{n} \frac{1}{i}$$

برای $n>1$

$$\ln n < Hn < \ln n + 1$$

که

$$\int_1^n \frac{1}{x}\, dx = \ln n$$

پس زمان مورد انتظار جستجو برای هر نقطه q، $O(\log n)$ است.

اندازه ساختار D:

محدوده اندازه D به اندازه نودهای D است. برگها تناظر یک به یک با ذوزنقه دارد از طرفی تعداد ذوزنقه ها O(n) است. یعنی:

$$O(n) + \sum_{i=1}^{n} (\ number\ of\ inner\ nodes\ created\ in\ itreation)$$

K_i: تعداد ذوزنقه های جدید ایجاد شده در تکرار iام با درج پاره خط s_i.

بعبارتی k_i تعداد برگهای جدید در D است، تعداد نودهای داخلی ایجاد شده دقیقا برابر $k_i - 1$ است.

$$O(n) + \sum_{i=1}^{n} O(i) = O(n^2)$$

در صورتی که خیلی بدشانس باشیم اندازه D می تواند مربعی باشد.

اندازه مورد انتظار:

$$O(n) + E[\sum_{i=1}^{n}(k_i - 1)] = O(n) + \sum_{i=1}^{n} E(k_i)$$

و

$$\delta(\Delta, s) = \begin{cases} 1 & if\ \Delta\ disappears\ from\ T(S_i)\ when\ s_i\ is\ removed\ from\ S_i \\ 0 & O.W \end{cases}$$

برای $\Delta \in T(S_i)$ ، $s \in S_i$ است.

حداکثر۴ بخش است که باعث حذف ذوزنقه می شوند. از اینرو

$$\sum_{s \in S_i} \sum_{\Delta \in T(S_i)} \delta(\Delta, s) \leq 4|T(S_i)| = O(i)$$

از آنجاییکه s_i عضو تصادفی S_i است می توانیم ارزش مورد انتظار k_i را با یافتن میانگین همه $s \in S_i$ پیدا کرد.

ادامه:

$$E(k_i) = \frac{1}{i}\sum_{s\in S_i}\sum_{\Delta\in T(S_i)}\delta(\Delta, s) \le \frac{O(i)}{i} = O(1)$$

تعداد مورد انتظار از ذوزنقه های ایجاد شده جدید $O(1)$ است. در هر تکرار الگوریتم $O(n)$ حد بالای مورد انتظار از ذخیره سازی است.

زمان اجرای مورد انتظار ساختمان داده الگوریتم:

تنها نیاز به محاسبه زمان درج پاره خط s_i داریم.

$$O(k_i) + the\ time\ needed\ to\ locate\ the\ left\ end\ point\ of\ s_i\ in\ T(S_i)$$

با استفاده از زمان جستجو و محدوده k_i داریم:

$$O(1) + \sum_{i=1}^{n}[(O(\log i) + O\big(E(k_i)\big)] = O(n \log n)$$

اتمام اثبات.

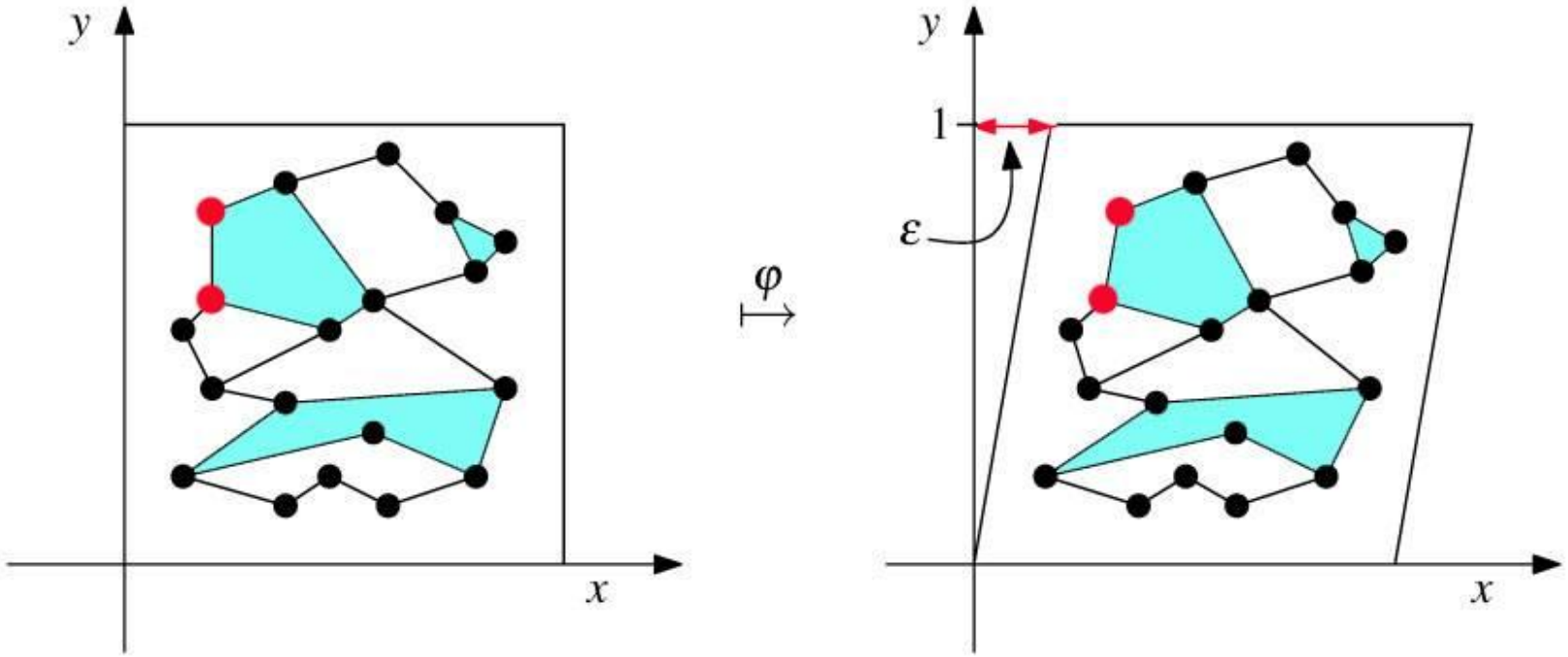
نتیجه:

- اگر S یک زیر تقسیم از صفحه با n یال باشد آنگاه در زمان مورد انتظار O(n log n) می توان ساختار داده ای از فضای O(n) ساخت بطوریکه برای هر نقطه جستجوی q، زمان مورد انتظار برای مکان یابی نقطه O(log n) باشد.

حالات خاص

- ۲ فرض در بخش های قبل در نظر گرفتیم:
  - هیچ ۲ نقطه ای X یکسان نداشته باشد.
  - نقطه جستجو هرگز روی خط عمودی یا پاره خط قرار نداشته باشد.
- یک روش چرخاندن سیستم محور ها با زاویه چرخش به اندازه کافی کوچک است. (به محاسبات دقیق با دقت بسیار بالا نیاز دارد.)
- روش دیگر چرخش نمادین است که از تبدیل زیر استفاده می کنیم:

$$\varphi = \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x + \varepsilon y \\ y \end{pmatrix}$$


y
x
φ
y
1
ε
x