فصل ششم:
Point Location
Knowing Where You Are

مینا حسین نیا شوکی

آبان 93

ساخت نقشه ذوزنقه ای:

با استفاده از یک الگوریتم افزایشی سعی می کنیم نقشه ذوزنقه ای، T(S)را بسازیم.

در طول ساخت نقشه ذوزنقه ای، ساختار داده ای D نیز جهت جستجوی مکان نقاط ساخته میشود.

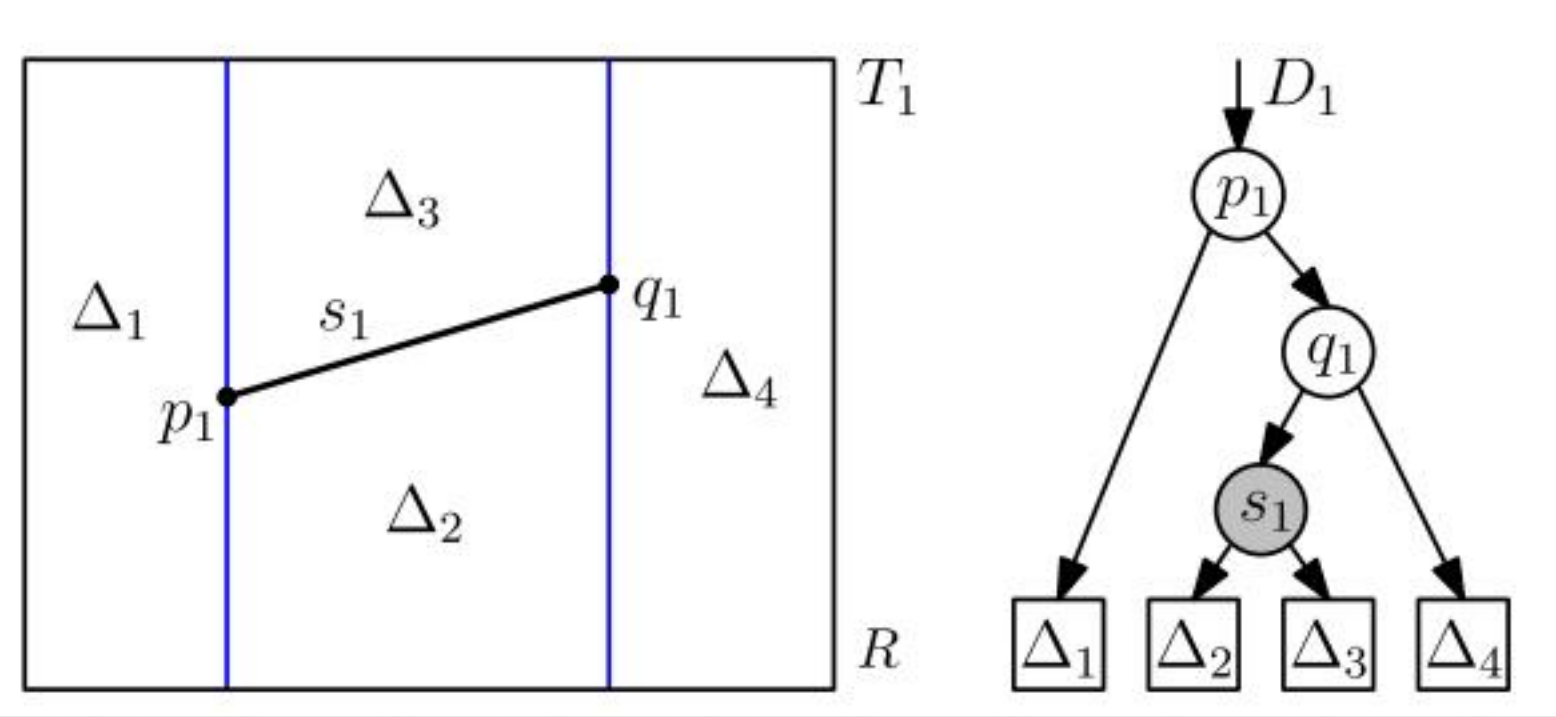
این ساختار جستجو یک گراف جهت دار بدون دور است.

این گراف دارای یک ریشه است و دقیقا یک برگ به ازای هر ذوزنقه در آن وجود دارد.

در x-node بررسی می کنیم که آیا نقطه q سمت چپ یا راست خط واصل نقاط انتهایی قرار دارد؟

در y-node بررسی می کنیم که آیا نقطه q بالا یا پایین آن segment قرار دارد؟

نمایش نودها در گراف

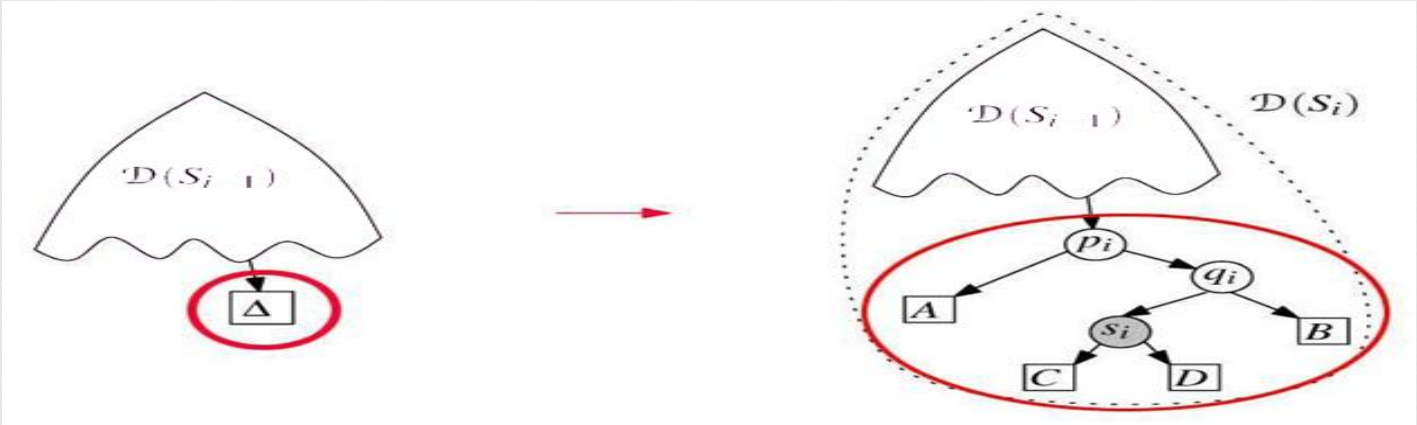


Algorithm TRAPEZOIDALMAP(S)
Input. A set S of n non-crossing line segments.
Output. The trapezoidal map $\mathcal{T}(S)$ and a search structure $\mathcal{D}$ for $\mathcal{T}(S)$ in a bounding box.

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.

sample


$\mathcal{T}(S_{i-1})$
Δ
s_i
q_i
p_i
$\mathcal{T}(S_i)$
A
C
B
D


$\mathcal{D}(S_{i-1})$
Δ
$\mathcal{D}(S_{i-1})$
$\mathcal{D}(S_i)$
p_i
q_i
A
s_i
B
C
D

نحوه تشخیص همسایه ها:

هرذوزنقه Δ_{j+1} همسایه سمت راست Δ_j هست:

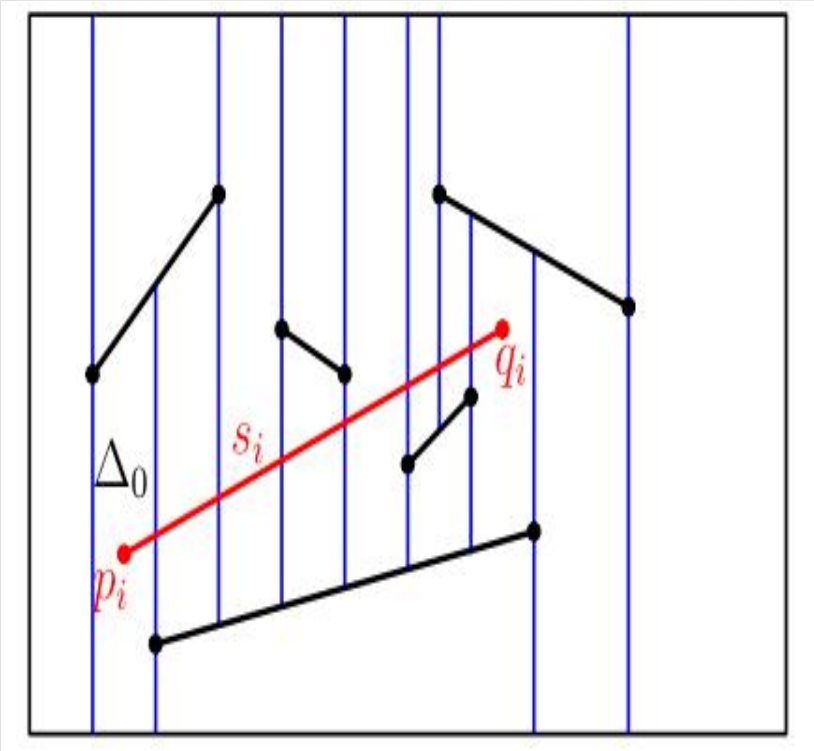
- اگر rightp(Δ_j) بالای s_i قرار بگیرد Δ_{j+1} همسایه پایین راست Δ_j می باشد.
- اگر rightp(Δ_j) پایین s_i قرار بگیرد Δ_{j+1} همسایه بالا راست Δ_j می باشد.

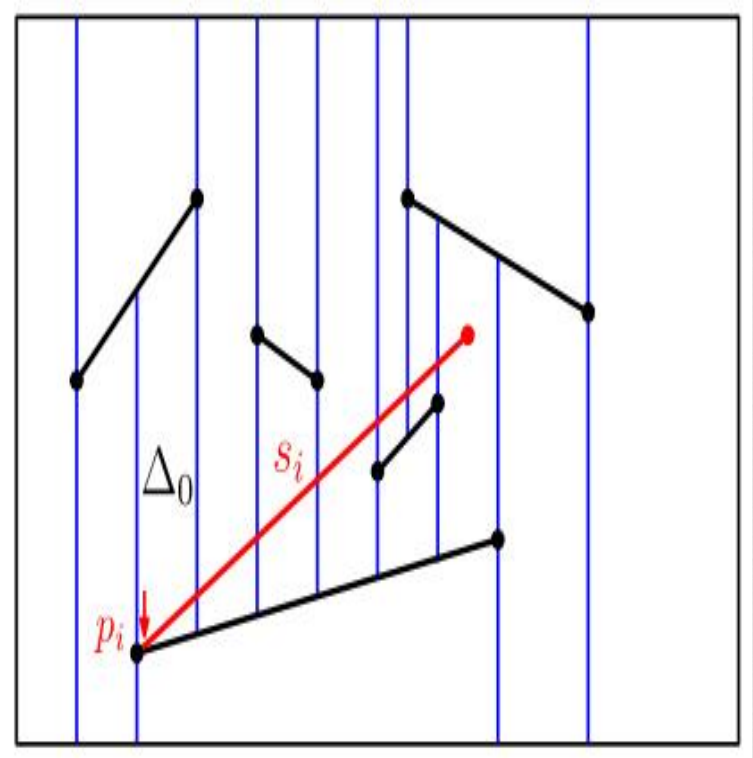
در نتیجه ما با داشتن Δ_0 و پیمودن نقشه دوزنقه ای میتوانیم Δ_k... Δ_1 را بدست آوریم.

است ذوزنقه $\Delta 0$ که راس انتهایی چپ si (p) در آن قرار دارد مشخص شود و بقیه ذوزنقه ها با بررسی ساختار افراز ذوزنقه ای تعیین می شوند.

- اگر pداخل ذوزنقه باشد با جستجوی Dآن را پیدا میکنیم.
- اگر pروی یک یال یا راس از ذوزنقه باشد قرار داد میکنیم:
- اگر pروی پاره خط عمودی در یک x- nodeباشد، p در سمت راست آن قرار دارد.
- اگر pروی یک ضلع sاز یک y- node باشد، شیب sو siرا مقایسه میکنیم:

  اگر شیب siبزرگتر بود پس pبالای sقرار دارد و بالعکس.


q_i
s_i
Δ_0
p_i


s_i
Δ_0
p_i

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.

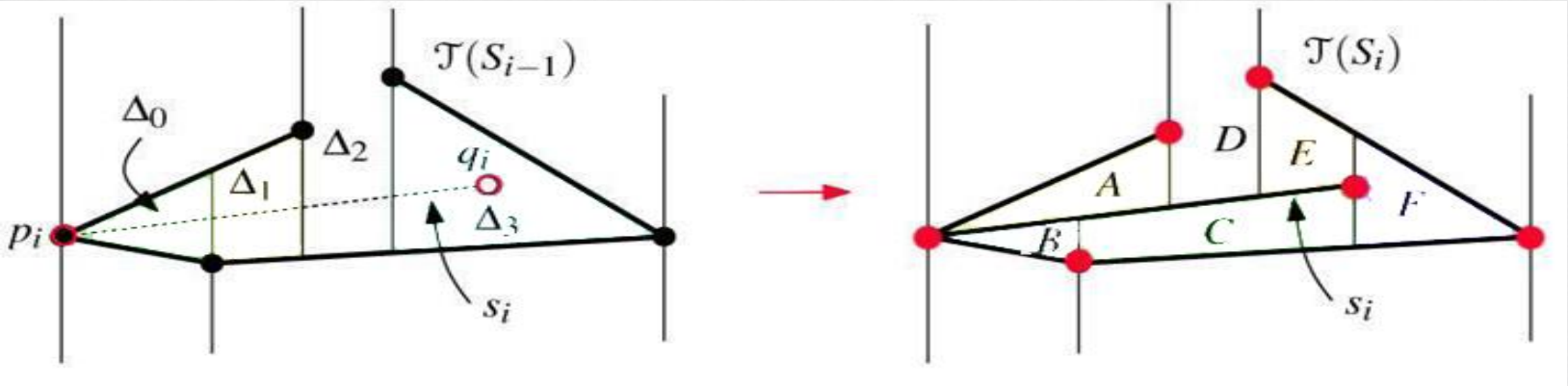
1. Let p and q be the left and right endpoint of s_i.
2. Search with p in the search structure $\mathcal{D}$ to find Δ_0.
3. $j \leftarrow 0$;
4. **while** q lies to the right of $rightp(\Delta_j)$
5. **do if** $rightp(\Delta_j)$ lies above s_i
6. **then** Let Δ_{j+1} be the lower right neighbor of Δ_j.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

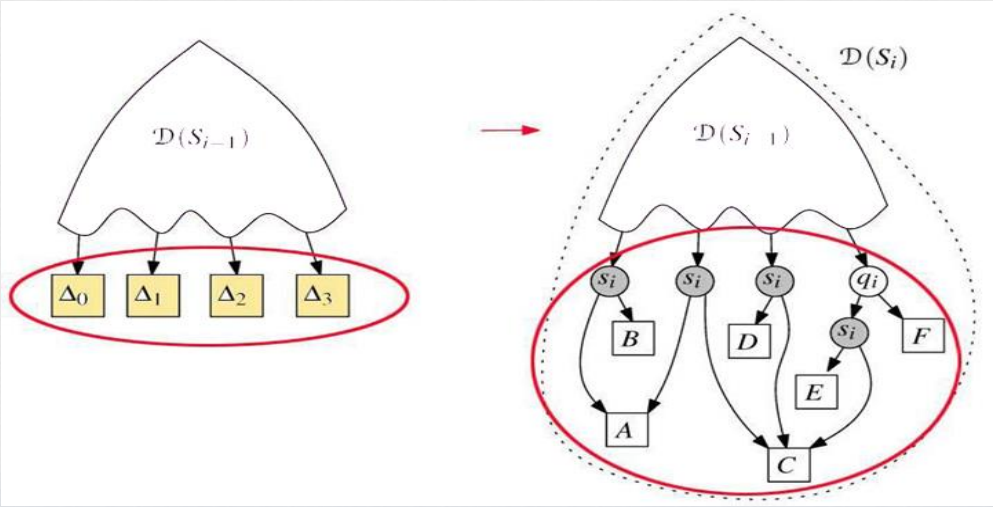
بروز رسانی TوD

برای update کردن T، Δ را از T حذف میکنیم و آن را با 4 ذوزنقه جدید A,B,C,D جایگزین میکنیم.

برای update کردن D، برگ Δ را از D حذف میکنیم و آن را با یک درخت کوچیک با 4 برگ جایگزین میکنیم.

این زیر درخت شامل نودهای داخلی جدید نیز میشود.


$\mathcal{T}(S_{i-1})$
Δ_0
Δ_1
Δ_2
Δ_3
q_i
p_i
s_i
$\mathcal{T}(S_i)$
A
B
C
D
E
F
s_i


$\mathcal{D}(S_{i-1})$
Δ_0
Δ_1
Δ_2
Δ_3
$\mathcal{D}(S_i)$
$\mathcal{D}(S_{i-1})$
s_i
q_i
A
B
C
D
E
F

بروز رسانی T

اگر $\Delta_1,\Delta_2,\ldots,\Delta_k$ دنباله ای از ذوزنقه های برخورد کرده باشند،ابتدا از نقاط انتهایی s_i، خط های عمودی را بسط میدهیم.

خط های عمودی اطراف را کوتاه میکنیم.

ذوزنقه های جدید را با هم merg می کنیم.

این مراحل در زمان خطی انجام میشود.

بروز رسانی D

اگر راس انتهای چپ siداخل Δ0 باشد:

به جای آن یک - xراس به عنوان انتهای چپ و یک -yراس برای siدر Dقرار میدهیم.

اگر راس انتهای راست siداخل Δkباشد:

به جای آن یک x -node به عنوان انتهای راست و یک -yراس برای siدر Dقرار میدهیم.

برگهای Δ1,…,, Δ k-1 را با یک y-node برای siجایگزین میکنیم و برگهای این رئوس را می سازیم.

قضیه 6.3:

الگوریتم TrapezoidalMap، نقشه ذوزنقه ای T(S)، از مجموعه S با n پاره خط در موقعیت کلی و ساختار جستجو D برای T(S)را، در زمان مورد انتظار O(nlgn)محاسبه می کند.

فضای مورد انتظار برابر O(n) است.

برای جستجو هر نقطه q زمان مورد انتظار O(lgn) می باشد.

اثبات ←زمان جستجوی ساختار جستجوی D

q نقطه ثابت جستجو است.

Xiمتغیر تصادفی و برابر تعداد نود هایی است که در تکرار iاضافه میشود.

در بدترین حالت در هر تکرار عمق گراف 3 نود اضافه میشود.

یعنی: $x_i \leq 3$

طول مسیر مورد انتظار:

$$E(\Sigma x_i) = \Sigma E(x_i)$$

Pi: احتمال وجود نود روی مسیر جستجوکه در تکرار iام ایجاد شده است

پس:

$$E(xi) \leq 3\ pi$$

Δq (Si-1) :ذوزنقه حاوی qدر T(Si−1)

احتمال ایجاد تغییر در ساختار D:

$$Pi = Pr[\Delta q(Si) \neq \Delta q(Si-1)]$$

تمام ذوزنقه های ایجاد شده در تکرار i همسایه های si هستند درواقع:

یا si خود Top(Δ) ویا Bottom(Δ) است .

یا نقاط انتهایی آن rightp (Δ) و یا leftp (Δ) هستند .

نقشه ذوزنقه ای را در نظر میگیریم،

در صورتی (Δq (Si حذف Δq (Si میشود که یکی ازچهار مورد , (((Δq (Si)Top

Bottom((Δq (Si))یا leftp (Δq (Si)) ,((Δq (Si)) rightpحذف شود.

پاره خط های Si به صورت تصادفی درج میشوند پس احتمال هر بخش برابر Si است.

احتمال آنکه ((Δq (Si)top با حذف پاره خط si، شود حذف 1/i است.

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

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

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

و در نتیجه زمان مورد انتظار جستجو نقطه q ، O(lgn) می باشد.

اثبات ←فضای مورد انتظار

برای بدست آوردن اندازه این ساختار کافی است تعداد نودهای D را محاسبه کنیم

تعداد نودها برابر است با : تعداد برگ ها + تعداد نودهای داخلی

که برگ ها در تناظر یک به یک با ذوزنقه ها هستند و از مرتبه O(n) (طبق لم ۶.۲)

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

اگر k_i تعداد ذوزنقه های جدید ایجاد شده (تعداد برگ ها) در تکرار i ام باشد.

k_{i-1}تعداد نودهای داخلی ایجاد شده در تکرار i ام است.

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

و تعریف می کنیم:

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

که $s \in S_i$, $\Delta \in T(S_i)$

در آنالیز زمان جستجو دیدیم حداکثر 4 بخش است که باعث حذف ذوزنقه می شود:

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

از طرفی s_i یک عضو تصادفی S_i می باشد، پس می توانیم مقدار مورد انتظار k_i را با گرفتن میانگین روی تمام s_i ها بدست آورد:

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

تعداد مورد انتظار ذوزنقه های ایجاد شده در هر تکرار $O(1)$هست و در بدترین حالت nتکرار داریم پس:

حد بالای مورد انتظاربرای ذخیره سازی $O(n)$ می باشد.

زمان اجرای مورد انتظار ساختمان داده الگوریتم:

تنها نیاز به محاسبه زمان درج پاره خط s_i داریم:

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

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

to search in $D(S_{i-1})$
for the left end point of segment s_i

to replace the affected
trapezoids

نتیجه 6.4

فرض کنید Sیک زیر تقسیم مسطح با nیال باشد. در زمان مورد انتظار O(nlgn) می توان یک ساختار داده از فضای مورد انتظار O(n)، ساخت. به طوری که برای هر نقطه جستجوی q، O(lgn)زمان مورد انتظار برای مکان یابی نقطه جستجو باشد.

حالات خاص:

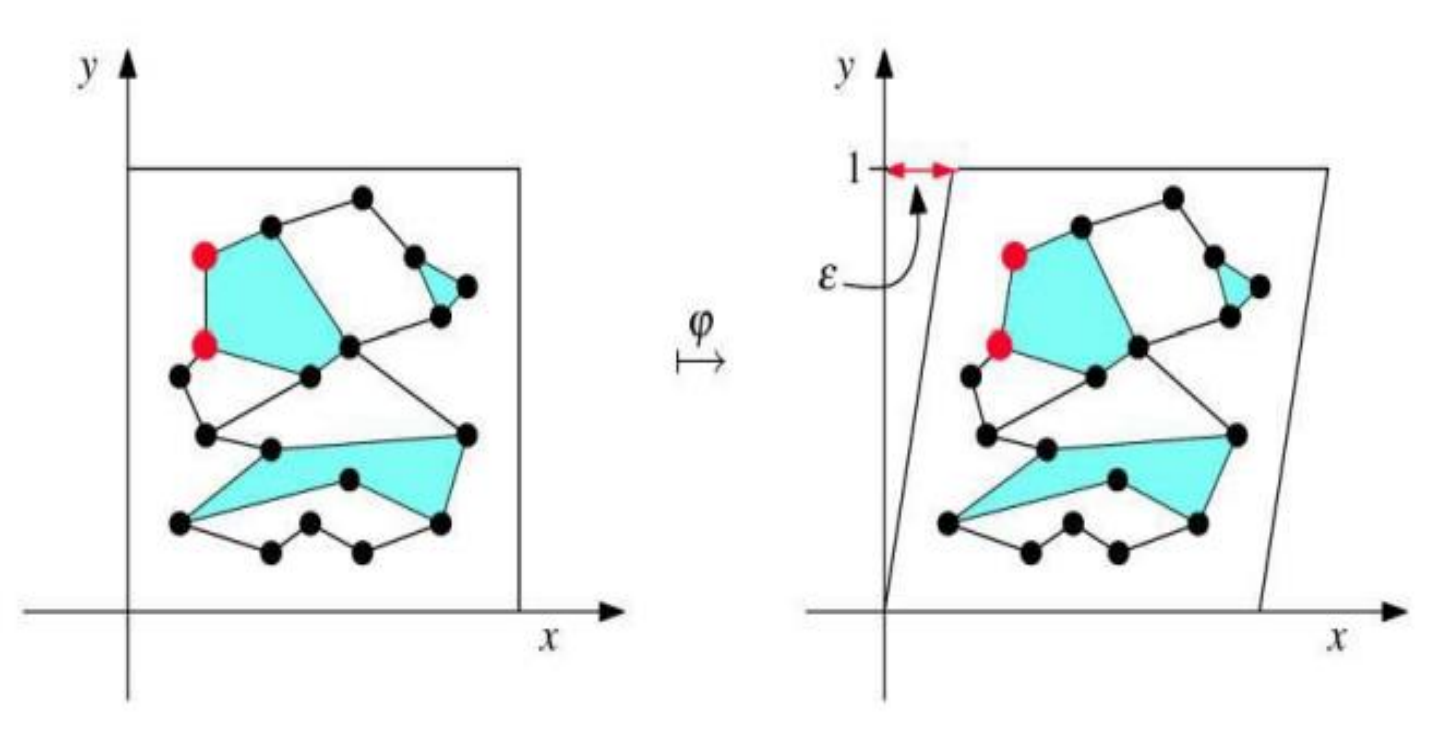
در بخش قبل فرضیاتی برای مسئله در نظر گرفتیم:

مجموعه پاره خط ها در موقعیت کلی هستند یعنی هیچ دو نقطه مجزایی دارای مختصه x یکسان نیستند.

یک نقطه جستجو هیچ گاه روی پاره خط عمودی از یک x-node یا روی پاره خطی از یک y-node نمی باشد.

حال این فرضیات را نادیده میگیریم و از الگوریتم shear transformation استفاده میکنیم.

Shear transformation



براي يک $\varepsilon>0$ داريم $\varphi(x , y) \rightarrow (x+\varepsilon y , y)$

برای مشخص کردن ساختار D باید

- در x-node :دو نقطه q و pرا می گیرد و با داشتن مختصات($(xp+\varepsilon yp,yp)$, $(xq+\varepsilon yq,yq)$ مشخص میکند qسمت «چپ»، «راست» یا «روی خط عمودی» است که pقرار دارد.
- در y-node :نقطه q و پاره خطی با نقاط ابتدایی،انتهایی p1 , p2 را می گیرد و مشخص میکند qدر «بالا», «پایین» و یا «روی پاره خط» است. این عملگر وقتی اجرا میشود که خط عمودی از میان q، پاره خط را قطع کند:

$$x1+\varepsilon y1 \leq xq+\varepsilon yq \leq x2+\varepsilon y2$$

If $x1= xq = x2$ and $y1 \leq yq \leq y2$ Then q lies on s