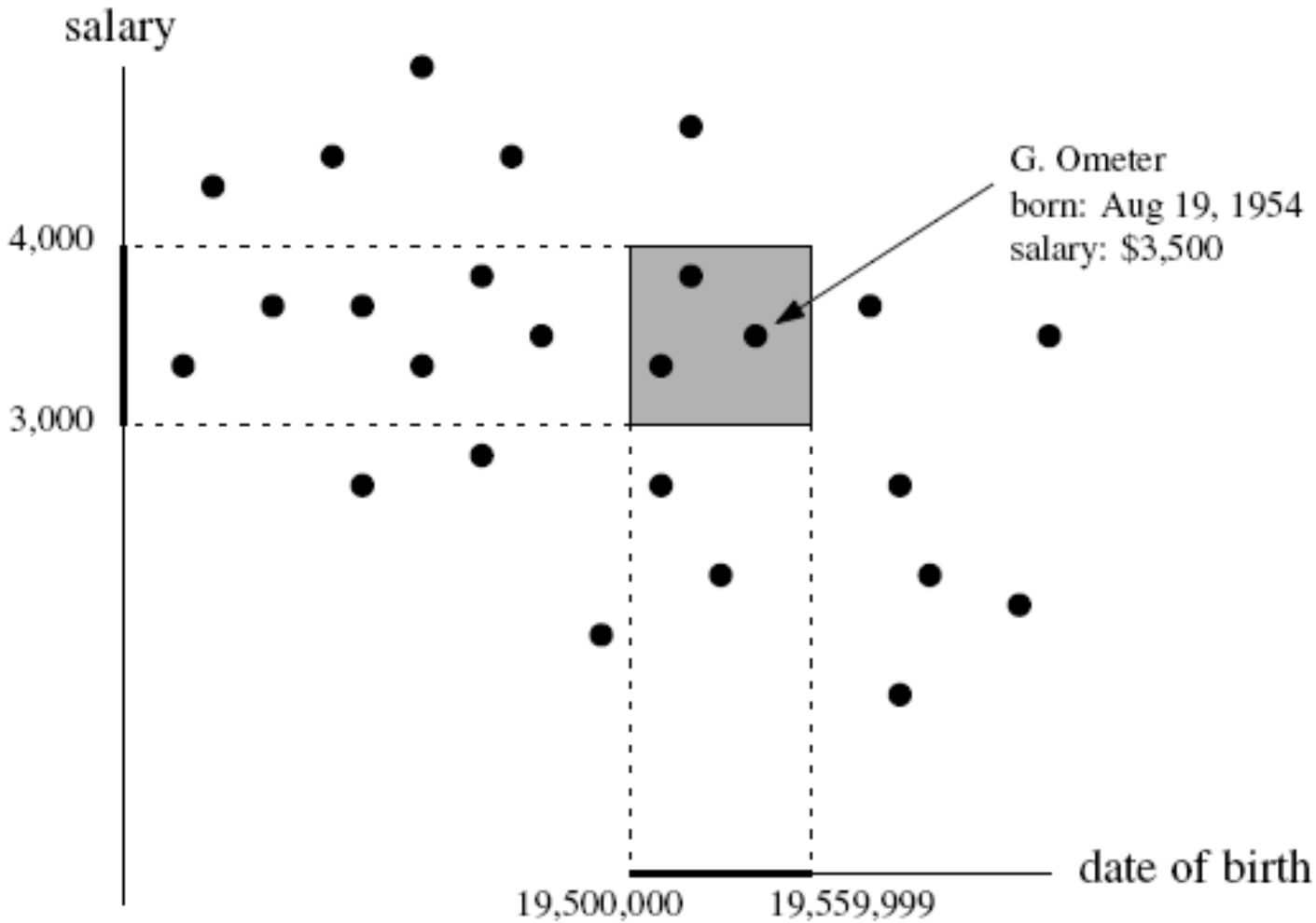
Orthogonal Range Searching

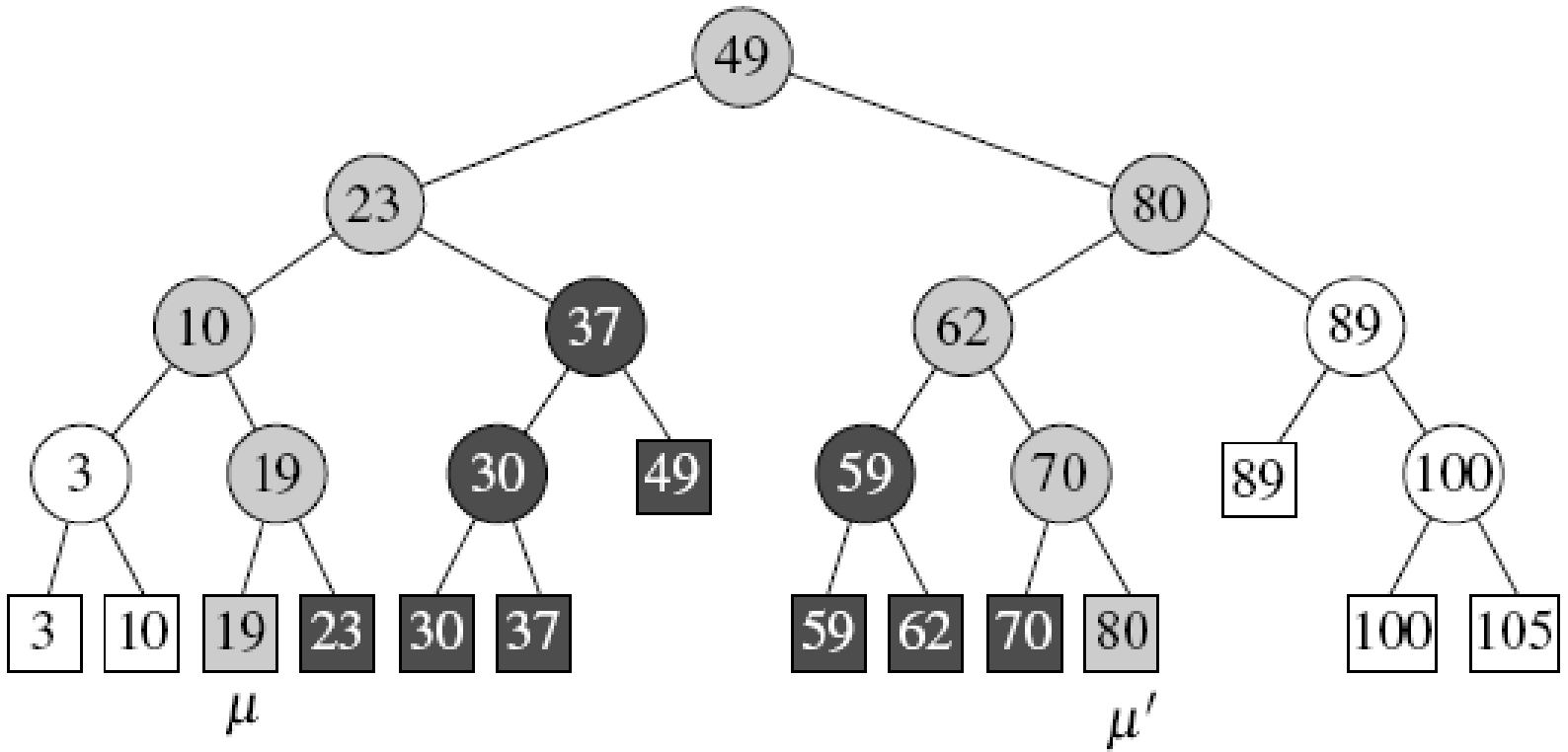
Querying a Database

زینب محمدی-عاطفه اکبری ابان ۹۲

- بسیاری از انواع پرسش هایی که روی یک پایگاه داده پرسیده میشود قابل تفسیر به صورت هندسی است.

برای این منظور رکوردهای پایگاه داده را به نقاطی در فضای چند بعدی انتقال می دهیم و پرسش ها را روی این نقاط اجرا می‌کنیم.


salary
4,000
3,000
G. Ometer
born: Aug 19, 1954
salary: $3,500
19,500,000
19,559,999
date of birth



در هر مسیری که به چپ حرکت میکنیم برگهای زیر درخت راست را گزارش میکنیم و در حرکت به سمت راست برگ های زیر درخت چپ را.

Algorithm 1DRANGEQUERY($\mathcal{T}, [x : x']$)
Input. A binary search tree $\mathcal{T}$ and a range $[x : x']$.
Output. All points stored in $\mathcal{T}$ that lie in the range.

```
1.  v_split ← FINDSPLITNODE(T,x,x')
2.  if v_split is a leaf
3.     then Check if the point stored at v_split must be reported.
4.     else (* Follow the path to x and report the points in subtrees right of the
            path. *)
5.          v ← lc(v_split)
6.          while v is not a leaf
7.             do if x ≤ x_v
8.                   then REPORTSUBTREE(rc(v))
9.                        v ← lc(v)
10.                  else  v ← rc(v)
11.         Check if the point stored at the leaf v must be reported.
12.         Similarly, follow the path to x', report the points in subtrees left of
            the path, and check if the point stored at the leaf where the path
            ends must be reported.
```

لم ۵-۱: الگوریتم 1DRangeQuery دقیقا نقاطی را که در بازه قرار دارد گزارش میکند.

اثبات:

۱- هر نقطه ای که گزارش شده در بازه بوده

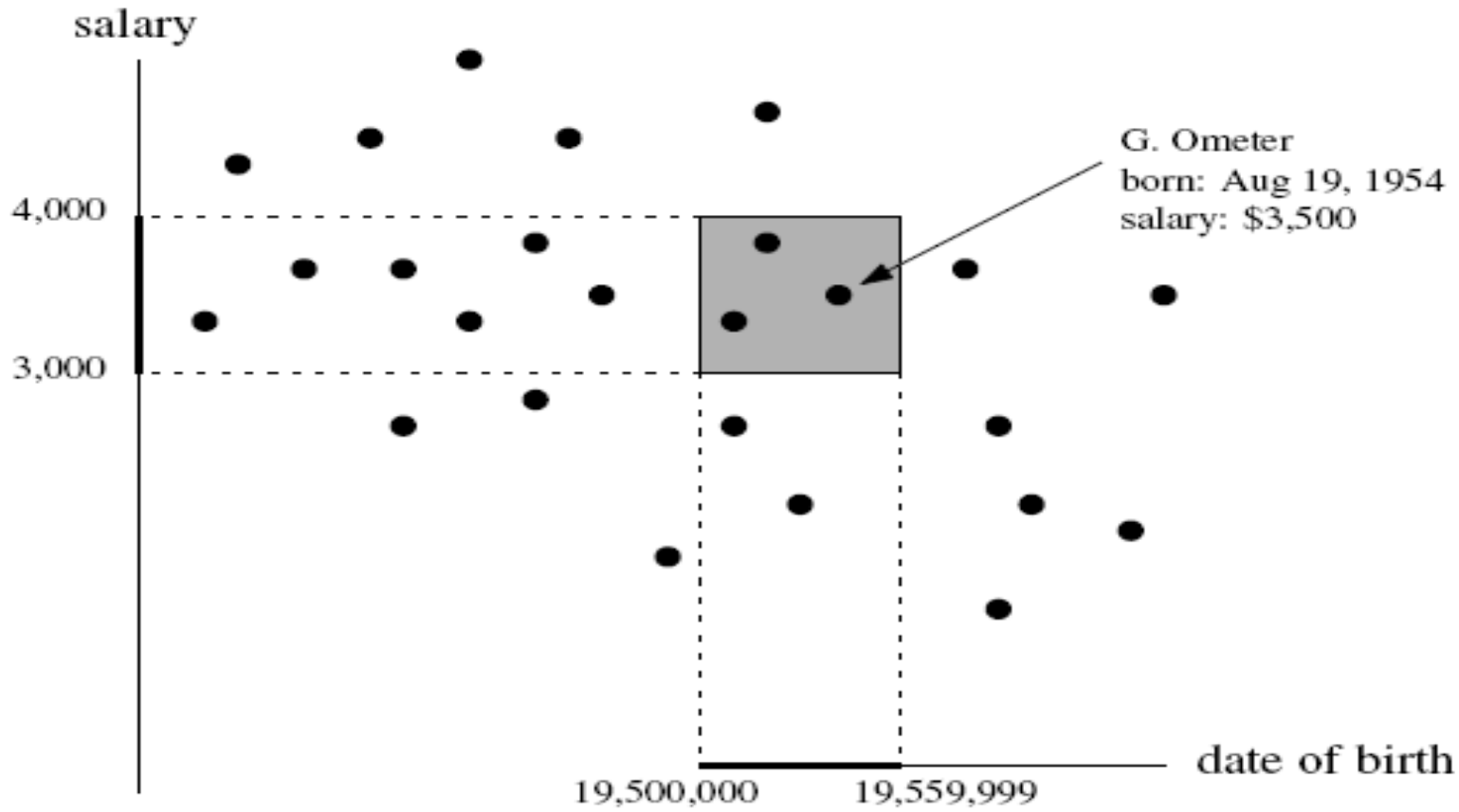
۲- هر نقطه ای که در بازه هست گزارش میشود.

Complexities:

Query Time	O(K + log n)	$\mathcal{T}$,[x,x'] $\rightarrow$ output
Construction Time	O(n log n)	P $\rightarrow$ $\mathcal{T}$
Space	O(n)	store $\mathcal{T}$

5.2 Kd-Trees

فرض های اولیه: هیچ دو نقطه ای مختصات x,y یکسان ندارند



- یک فضای ۲ بعدی را می‌توان ترکیبی از ۲ تا فضای ۱ بعدی در نظر گرفت.

ساختمان داده ی مناسب : bstمتوازن؟

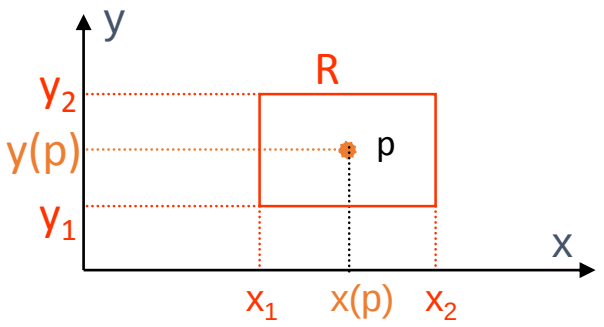
با توجه به تعریف بازگشتی bst متوازن قابل تعمیم به فضای با بعد بالاتر هست.

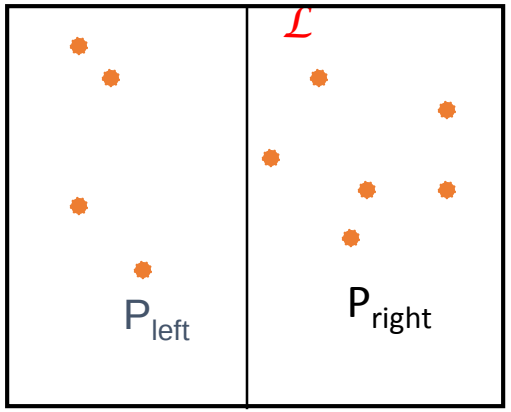
تعریف بازگشتی: مجموعه ای از نقاط که به دو زیرمجموعه ی تقریبا مساوی تقسیم میشوند یک زیر مجموعه شامل نقاط کوچکتر یا مساوی با راس جداساز و دیگری شامل نقاط بزرگتر از راس جداساز

dimension d=2:

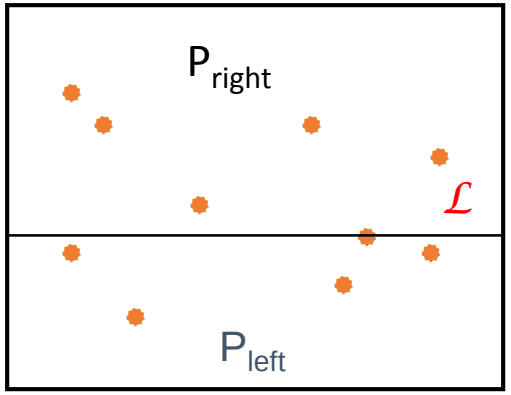
point p=(x(p) , y(p)) , range
$R = [x_1 : x_2] \times [y_1 : y_2]$

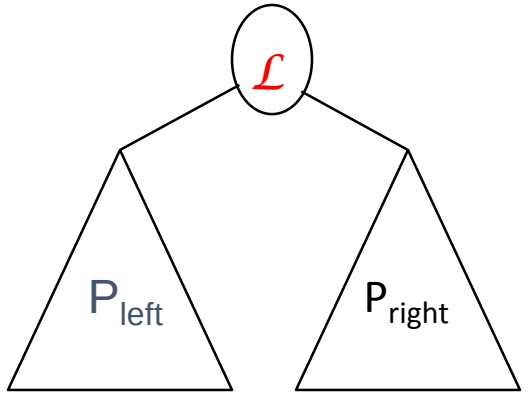
$p \in R \Leftrightarrow x(p) \in [x_1 : x_2]$ and $y(p) \in [y_1 : y_2]$.





OR

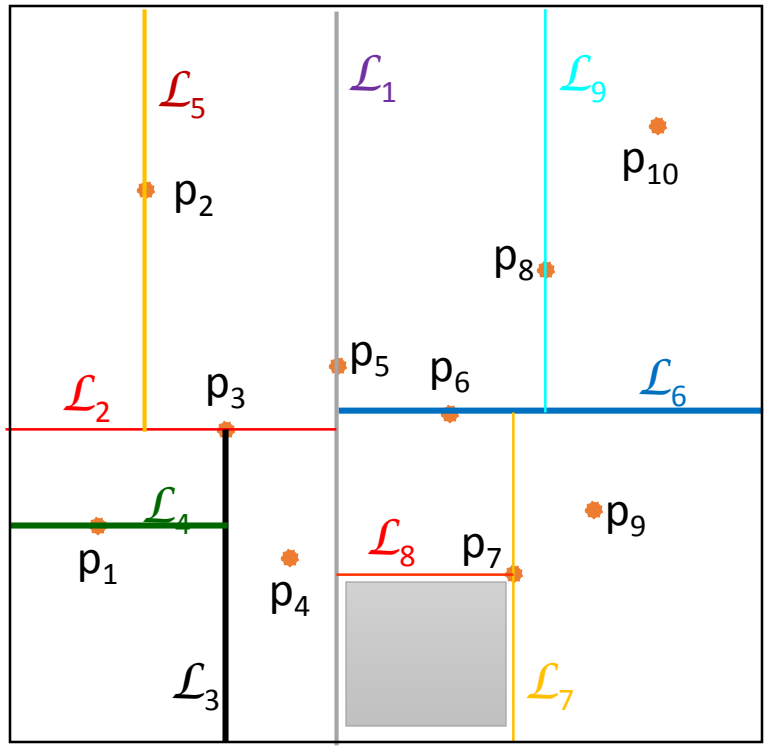


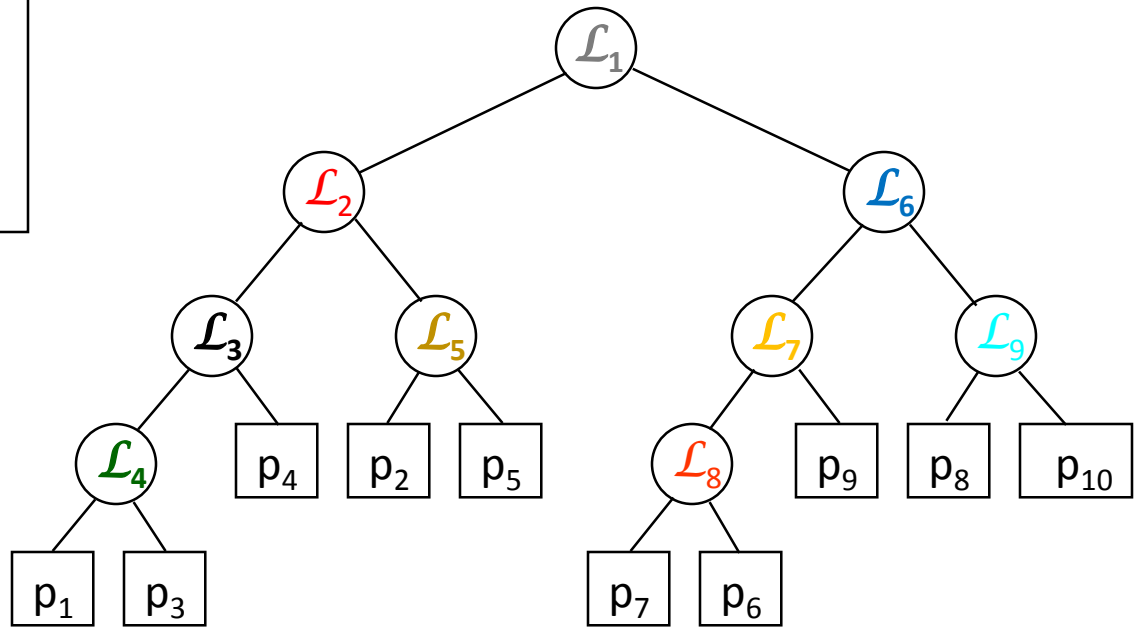


2D-tree

L:خط جداساز

Buildkdtree(p,depth)





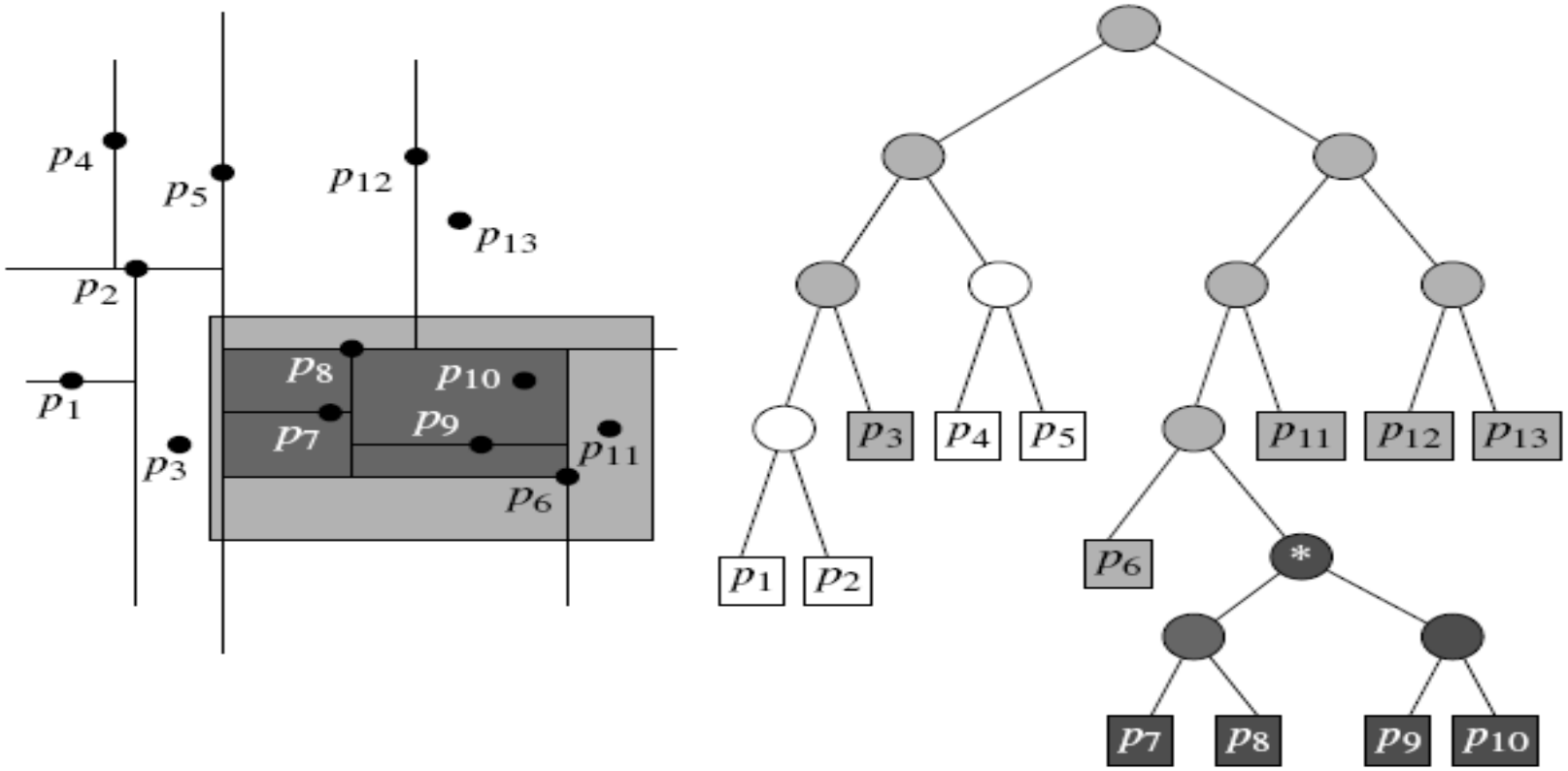
زمان ساخت:

$T(n)=O(n)+2T(n/2)$ ⟶ $O(n\log n)$

فضای ذخیره سازی:$O(n)$

لم ۵-۳: یک kdtree روی مجموعه ای از nنقطه فضایی برابر با $o(n)$ خواهد برد و در زمان $o(n\log n)$ ساخته میشود.

نتیجه: اگر ناحیه ی پرسش ما با region(v)اشتراک داشته باشد باید به دنبال زیردرختی باشیم که ریشه ی ان vهست.



Algorithm SEARCHKDTREE(v, R)

Input. The root of (a subtree of) a kd-tree, and a range R.

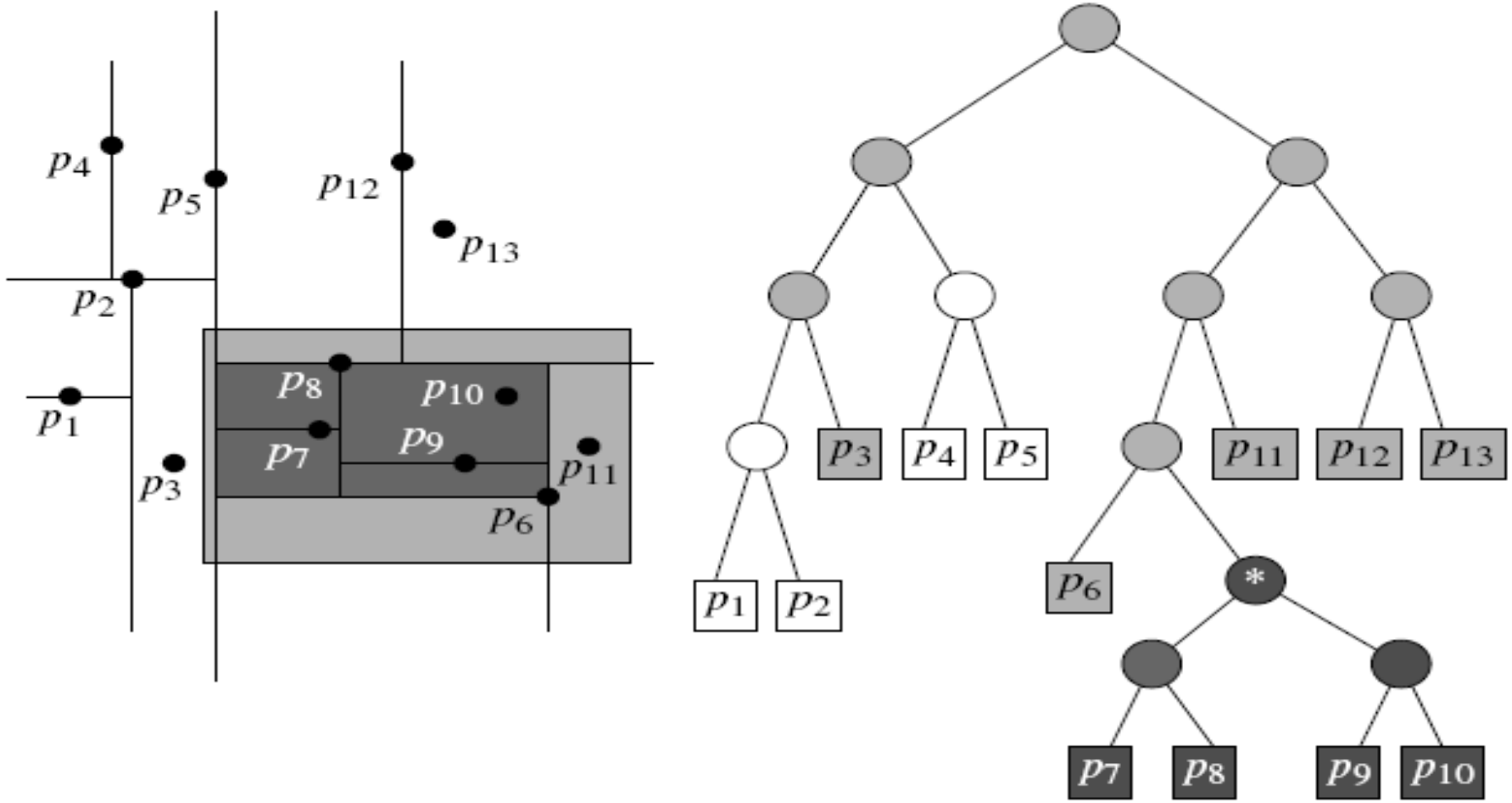
Output. All points at leaves below v that lie in the range.

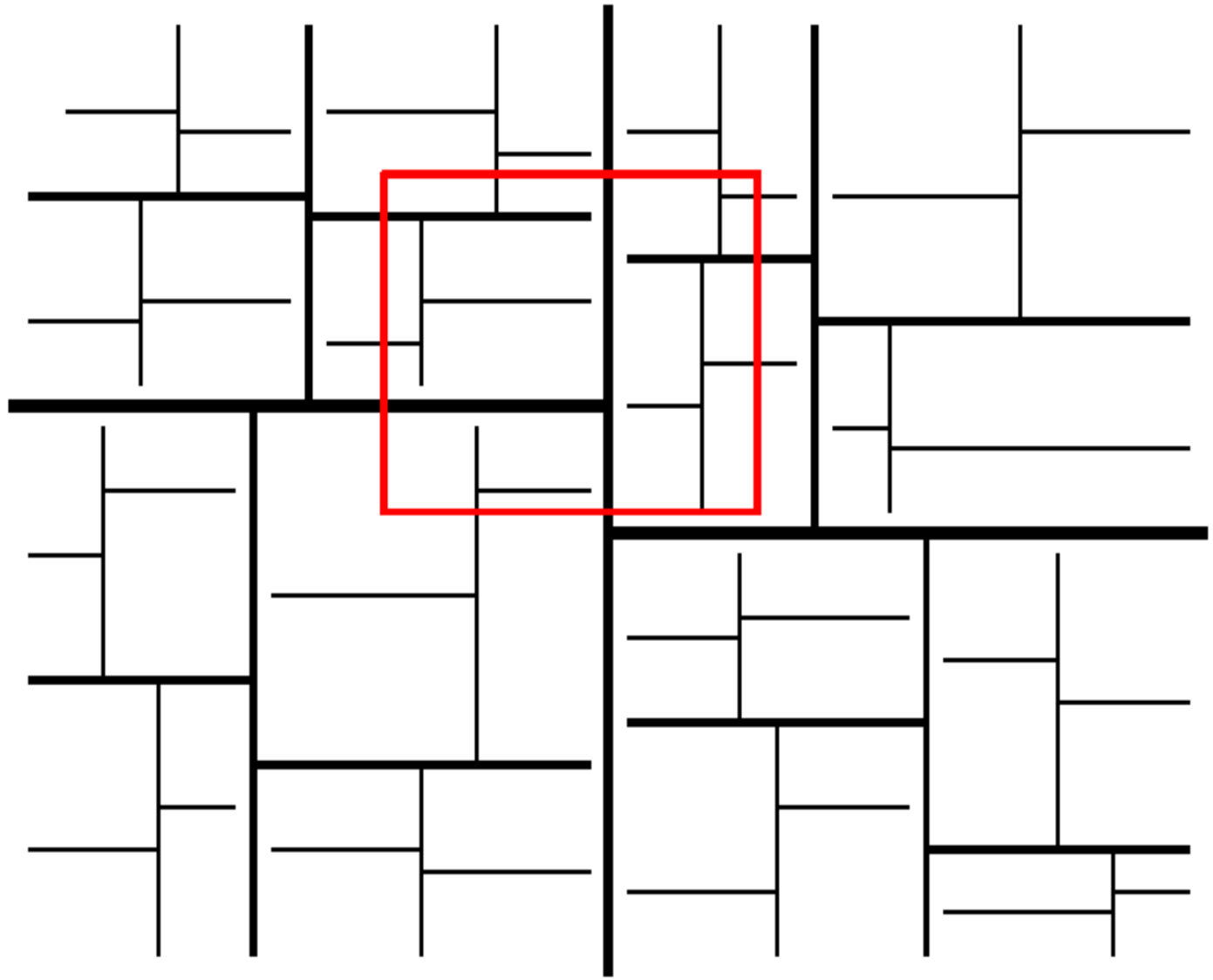
```
1.  if v is a leaf
2.      then Report the point stored at v if it lies in R.
3.      else if region(lc(v)) is fully contained in R
4.              then REPORTSUBTREE(lc(v))
5.              else if region(lc(v)) intersects R
6.                      then SEARCHKDTREE(lc(v), R)
7.           if region(rc(v)) is fully contained in R
8.              then REPORTSUBTREE(rc(v))
9.              else if region(rc(v)) intersects R
10.                     then SEARCHKDTREE(rc(v), R)
```

لم ۵-۴: پرس و جو برای یک دامنه ی مستطیل شکل در یک kdtreeبا nنقطه در زمان O(k+√n) انجام میشود.

قضیه ۵-۵:

یک درخت kdبرای مجموعه ای از nنقطه در صفحه در زمان O(nlogn) ساخته میشود و فضایی برابر با O(n) استفاده خواهد کرد.و پرس و جو برای داده در فضای ۲ بعدی O(K+√n)میباشد.

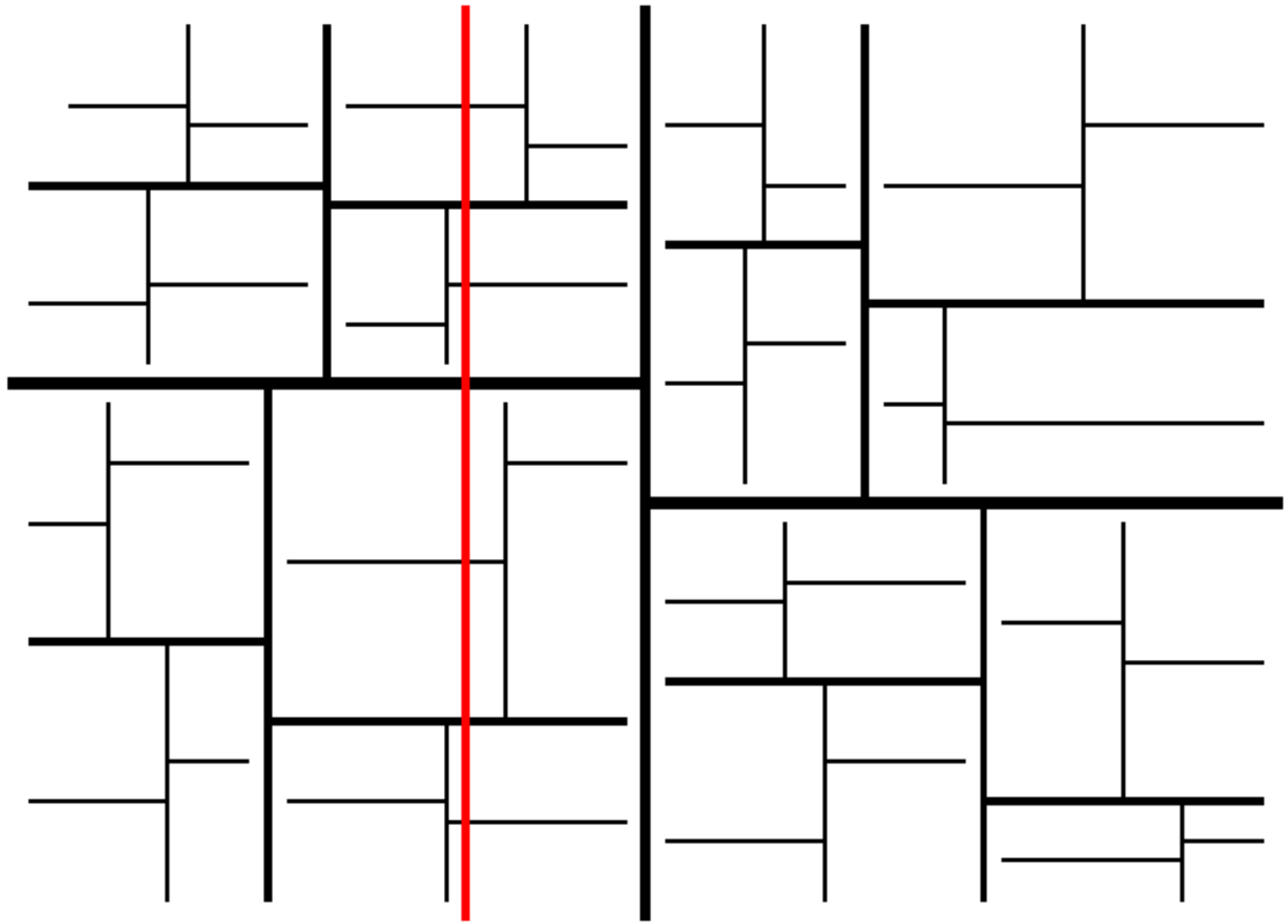

p_4
p_5
p_{12}
p_{13}
p_2
p_8
p_{10}
p_1
p_7
p_9
p_{11}
p_3
p_6
p_1
p_2
p_3
p_4
p_5
p_{11}
p_{12}
p_{13}
p_6
*
p_7
p_8
p_9
p_{10}



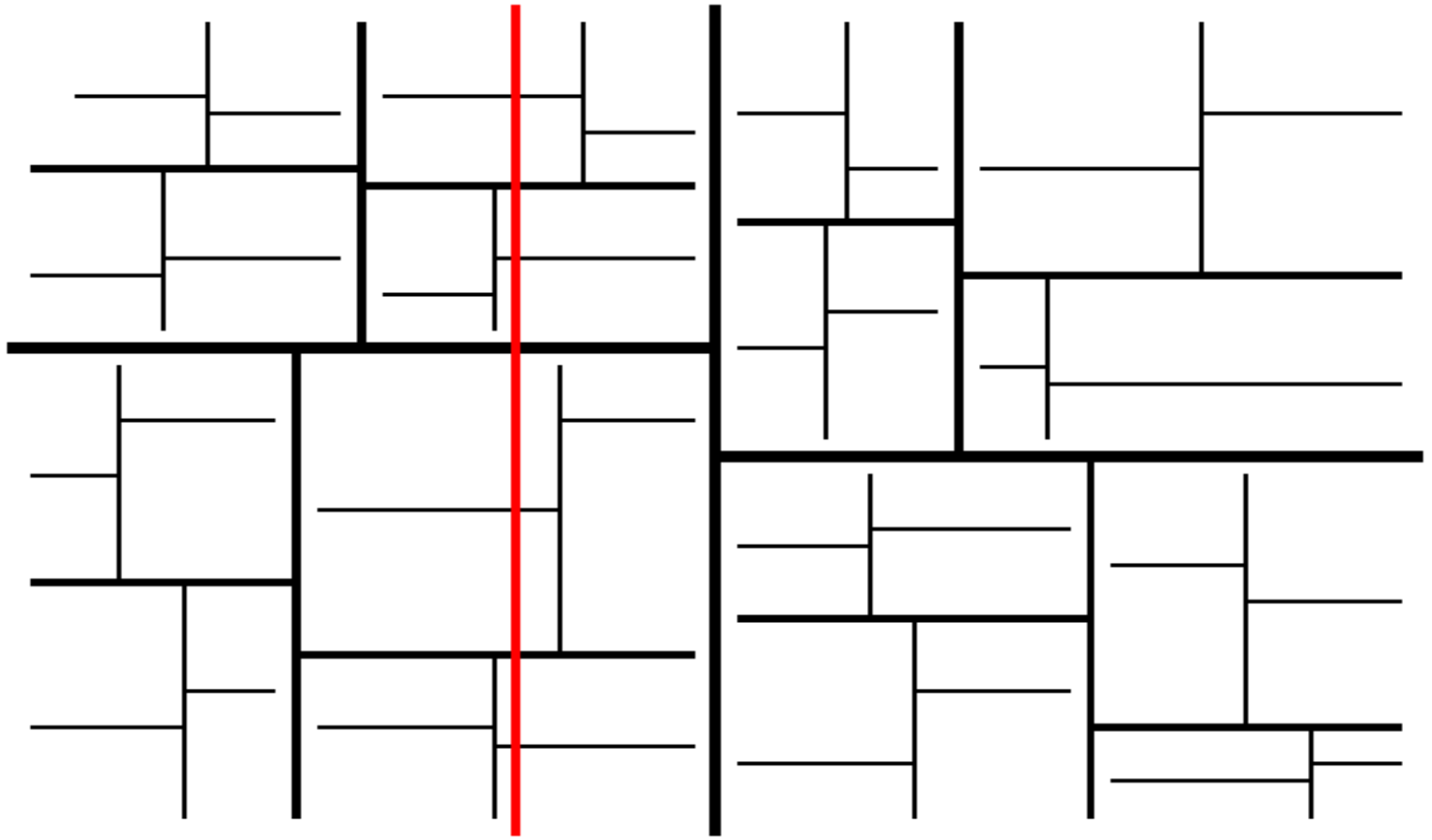
Question: How many grey and how many black *leaves*?

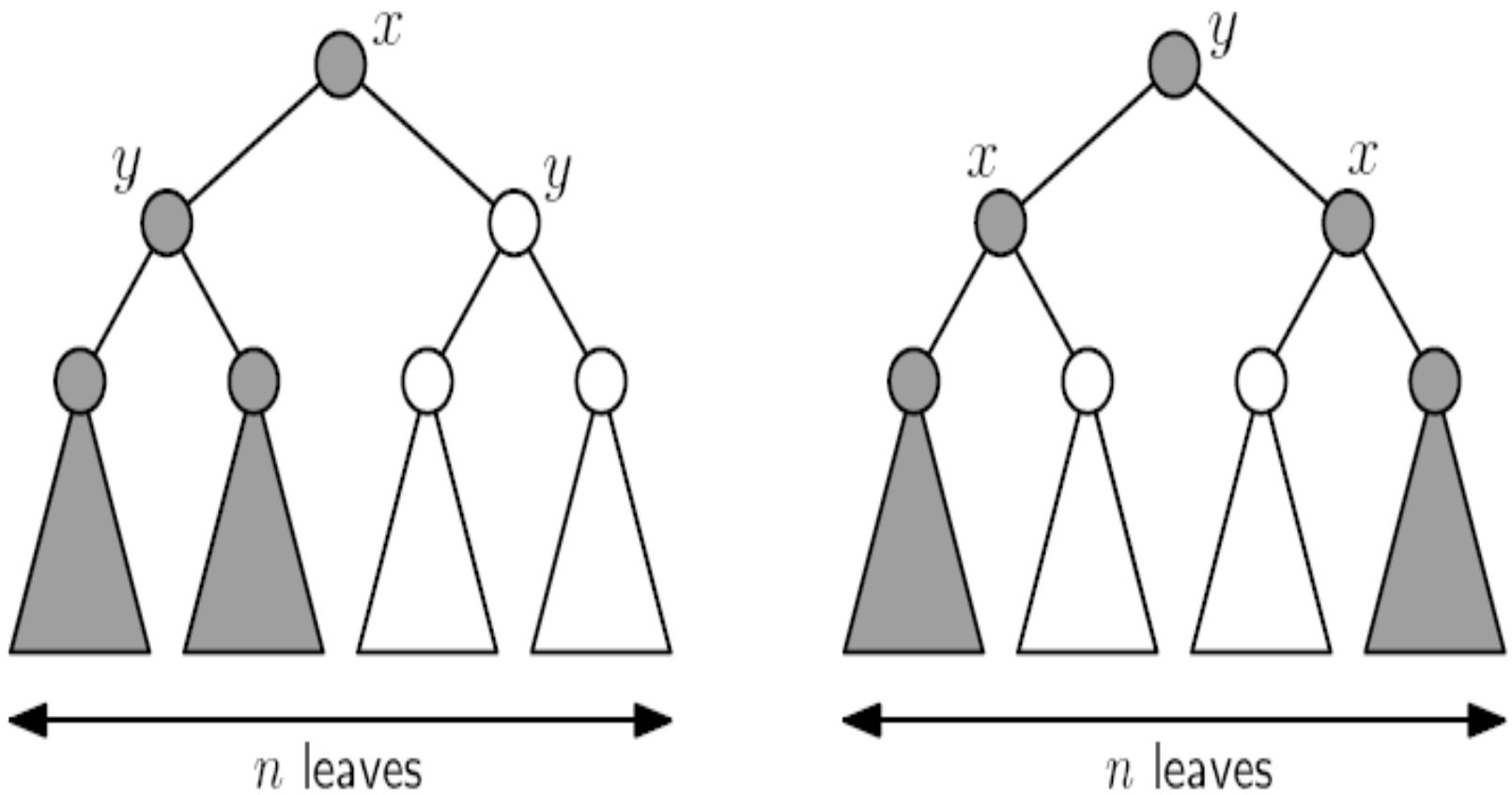
- برای راحتی کار و پیدا کردن حداکثر ناحیه هایی که با مرز های ما برخورد دارند یکی از ۴ یال (چون فضا ۲ بعدی است) رو در نظر گرفته و مساله را برای ان حل می‌کنیم و چون برای همه ی یال ها روال کار مشابه است نتیجه را در ۴ ضرب می‌کنیم.

- برای اینکه کران بالا برای نواحی دارای اشتراک با ناحیه ی جستجو بدست بیاید پاره خط ها را خط در نظر می‌گیریم.



توجه:یال مورد نظر(خط قرمز)فقط یک سمت هر خط جداساز عمودی قرار داشته و دو طرف هر خط جداساز افقی است.




x
y
y
n leaves
y
x
x
n leaves

اگر $Q(n)$را تعداد نود های دارای ناحیه ی مشترک در یک kdtree با n برگ در نظر بگیریم میتوان روابط زیر را نتیجه گرفت:

اگر در عمق زوج درخت باشیم:

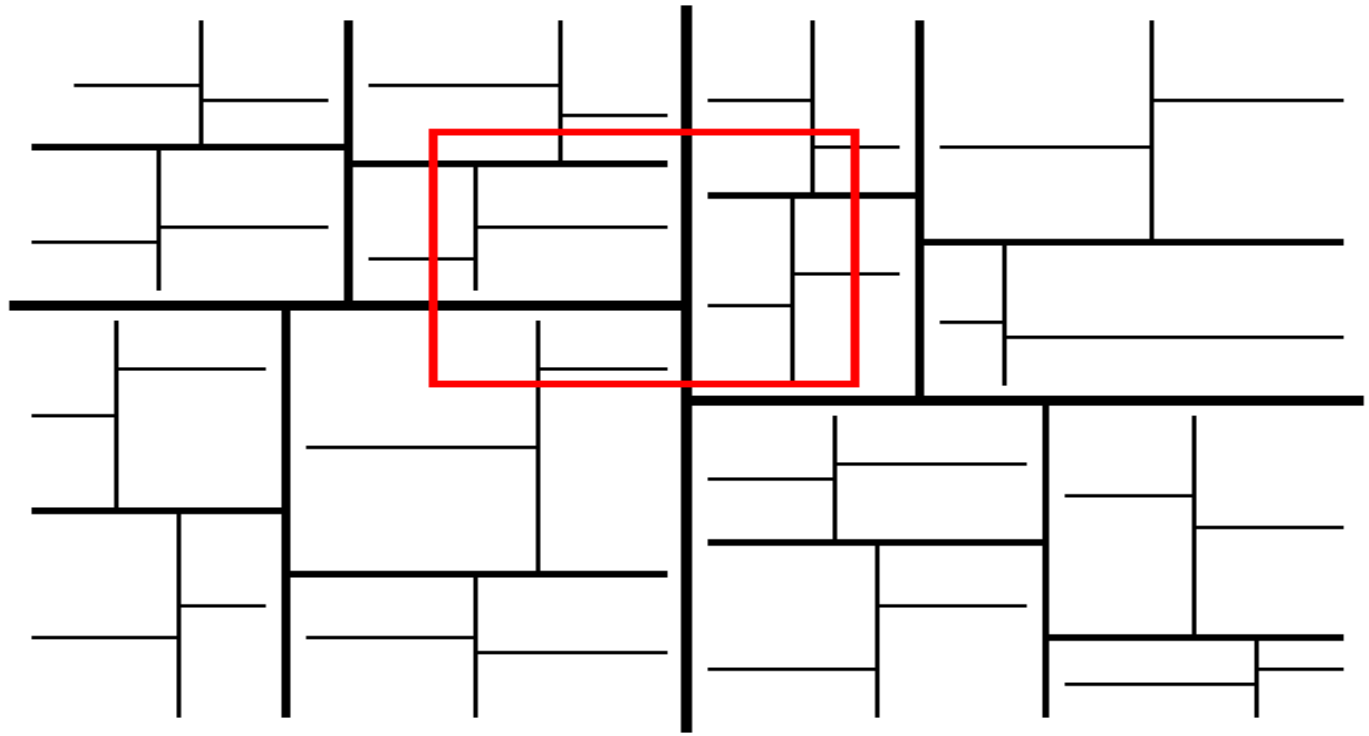
$$Q(n)=1+Q(n/2) \qquad (1)$$

و اگر در عمق فرد باشیم

$$Q(n)=1+2Q(n/2) \qquad (2)$$

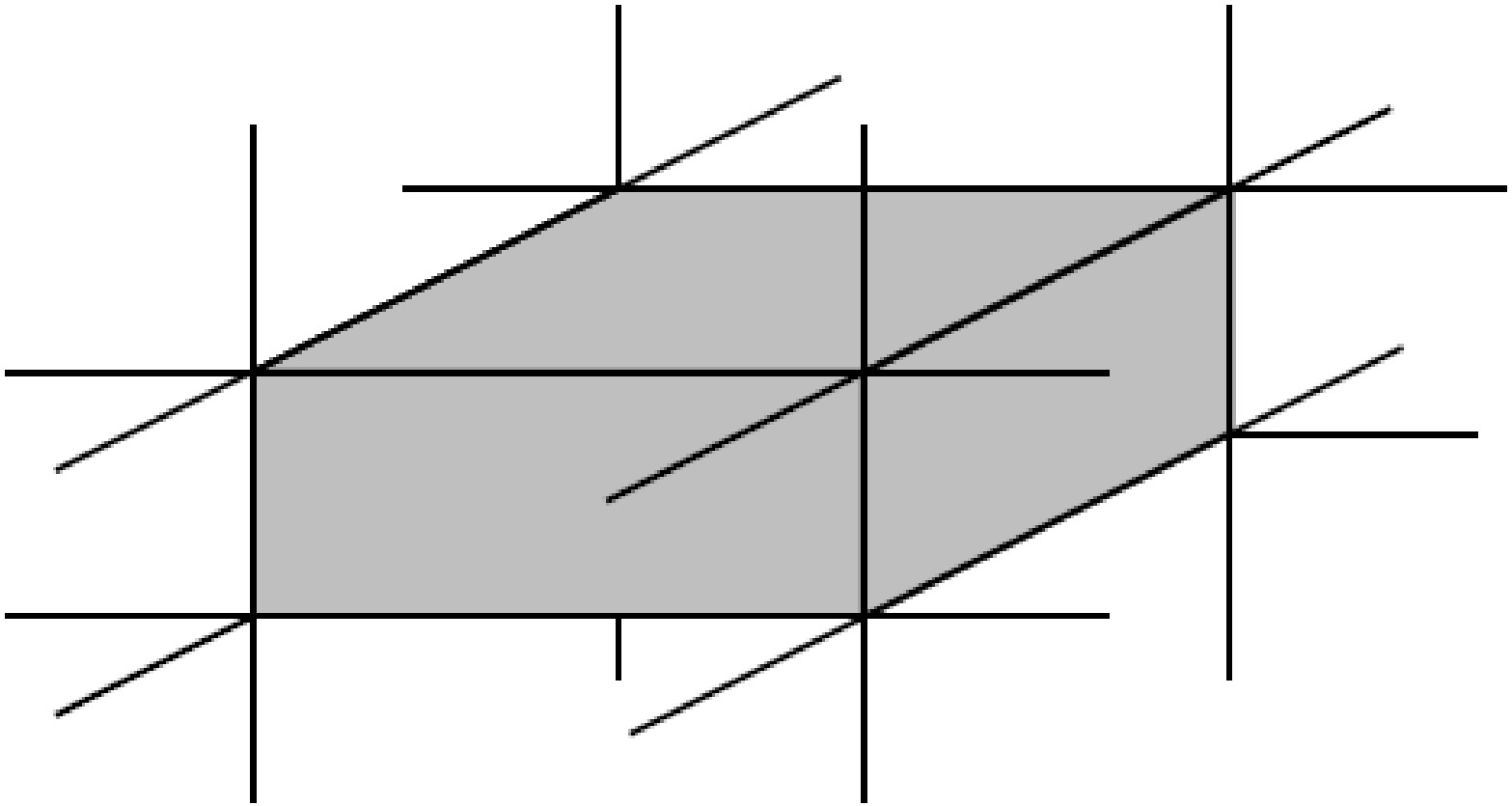
از (1) و (2) نتیجه میشود:

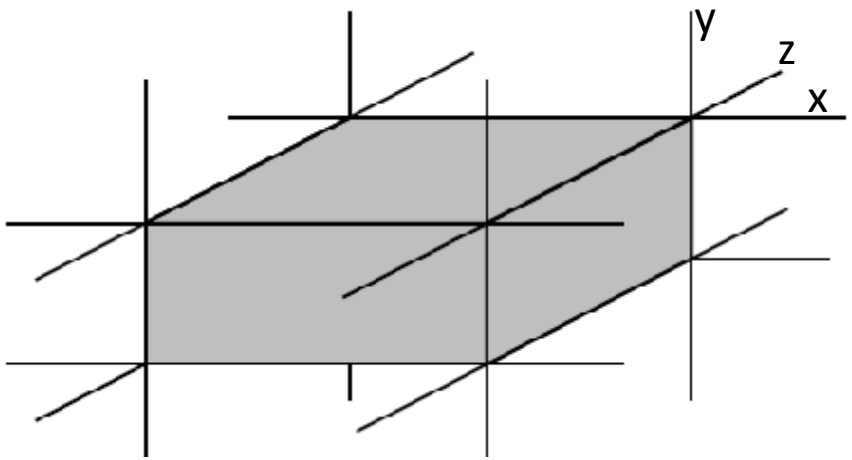
$$Q(n)=O(k)+2Q(n/4) \longrightarrow Q(n)\epsilon O(\sqrt{n})$$

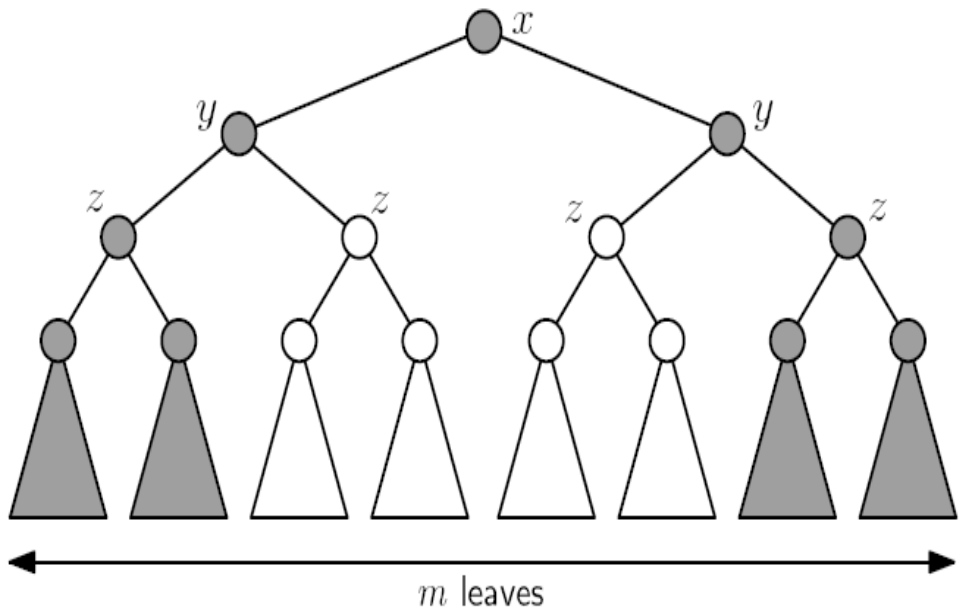


۲تا خط عمودی و ۲ تا خط افقی داریم که حداکثر نواحی مشترک با هر یک از انها o(√n)است

فضای ۳ بعدی:




y
z
x


x
y
y
z
z
z
z
m leaves

مشابه حالت قبل:

x: Q(n)=1+Q(n/2)

y: Q(n)=1+2Q(n/2)

z: Q(n)=1+2Q(n/2)

با ترکیب ۳ رابطه ی بالا:

Q(n)=O(1)+4Q(n/8) ⟶ Q(n)ϵ $O(n^{1-1/d}+k)$

هر مکعب ۶ وجه دارد که برای هر کدام رابطه ی بالا صادق است.

Range Tree

Range tree

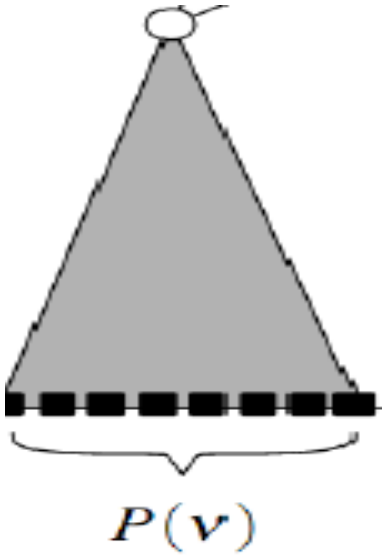
نتیجه:پرس و جو در یک kdtreeدرزمان $O(n^{1-1/d}+k)$ جراست.

ایا میتوان زمان را بهبود داد؟ **بله**

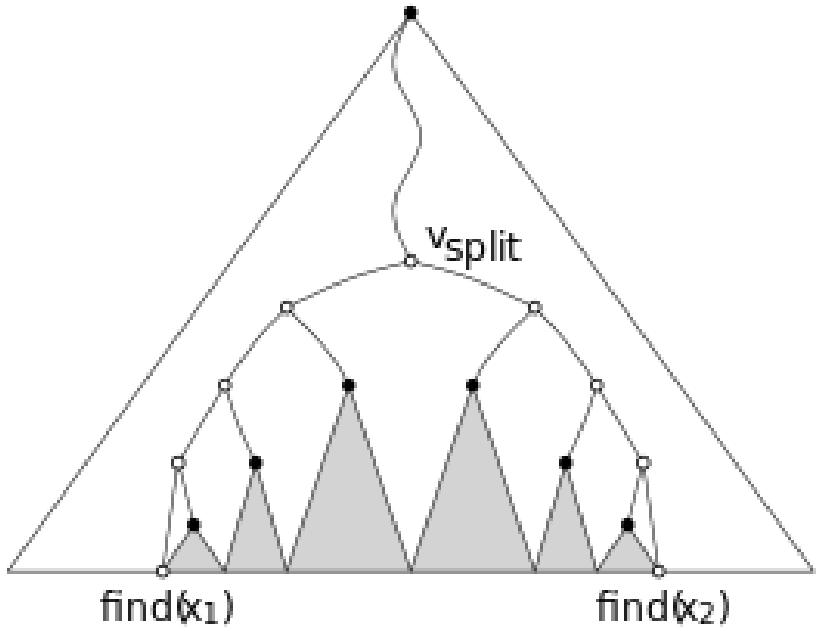
Range tree

- زمان ساخت k_dtree از مرتبه‌ی $O(n)$ ⟵ $O(n \log n)$
- زمان جستجو در k_dtree ازمرتبه‌ی $O(\sqrt{n} + k)$ ⟵ $O((\log n)2 + k)$

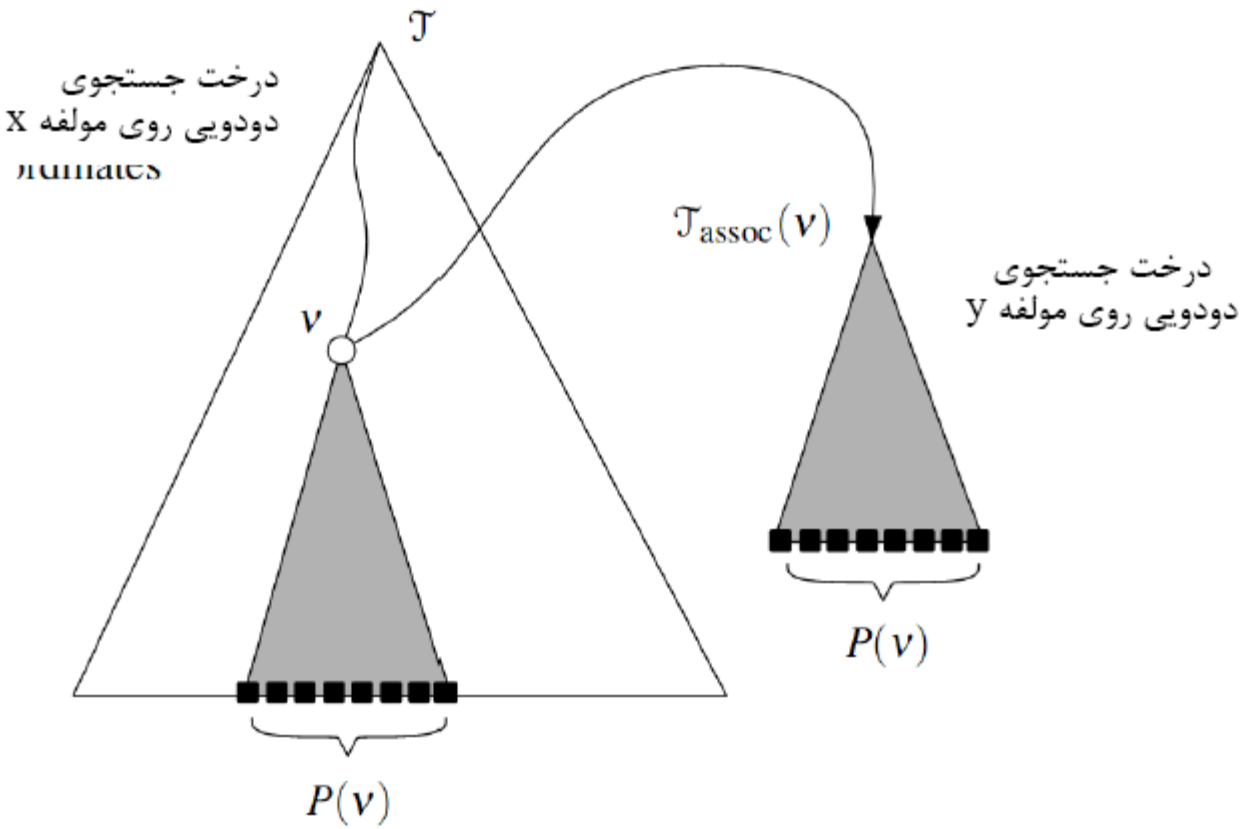
زیرمجموعه ی کانونی $p(v)$: مجموعه‌ی نقاطی که در برگ‌های زیر درخت با ریشه ی **v** ذخیره شده اند.



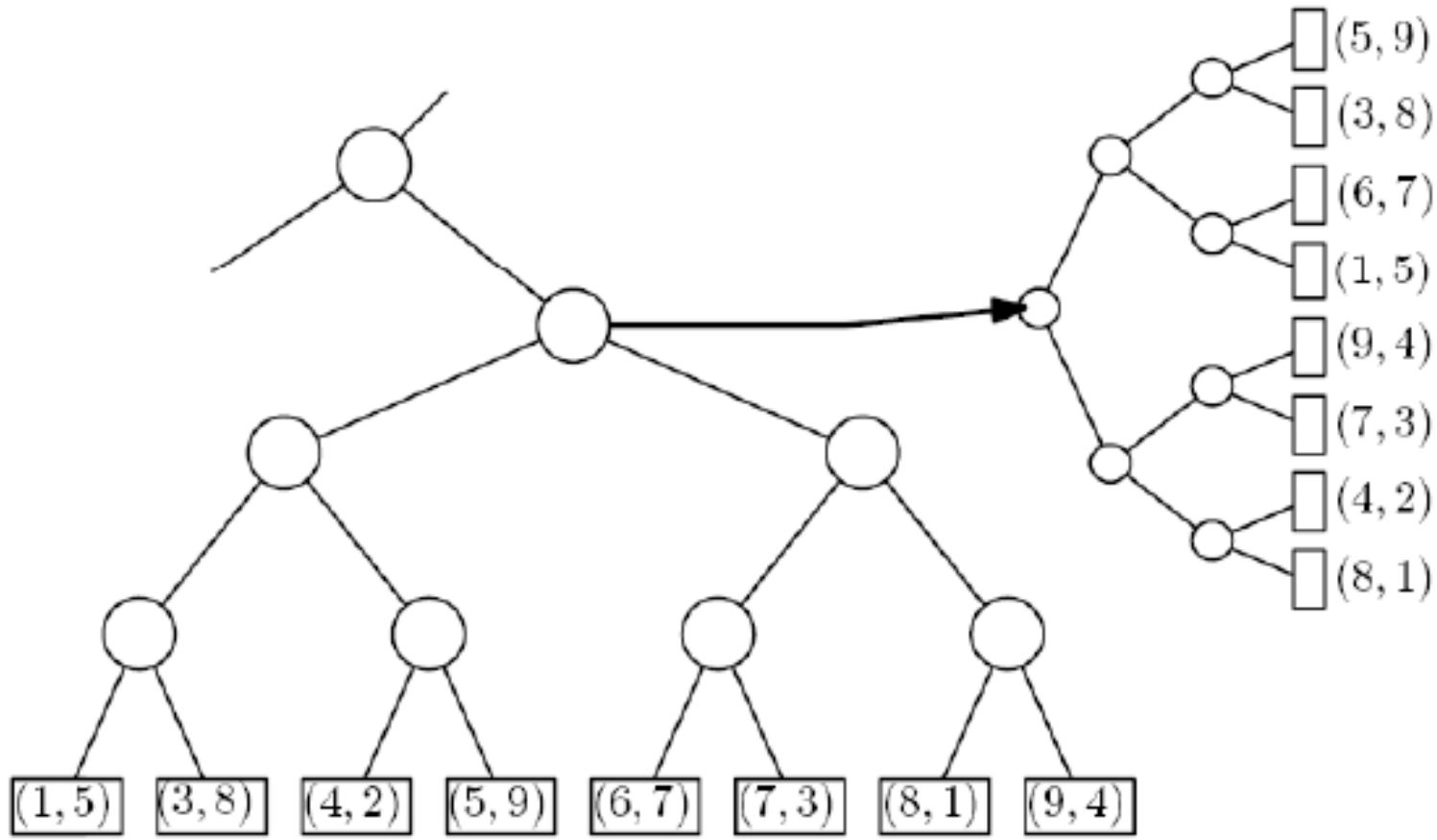
- اجتماع مجزای $O(\log n)$ زیرمجموعه‌ی کانونی را برای هر درخت در لایه‌ی اول تعریف می‌کنیم.



- درخت اصلی ⟵ بر اساس مؤلفه x
- درخت وابسته ⟵ بر اساس مؤلفه y



A example of 2-dimensional range tree



Algorithm BUILD2DRANGETREE(P)

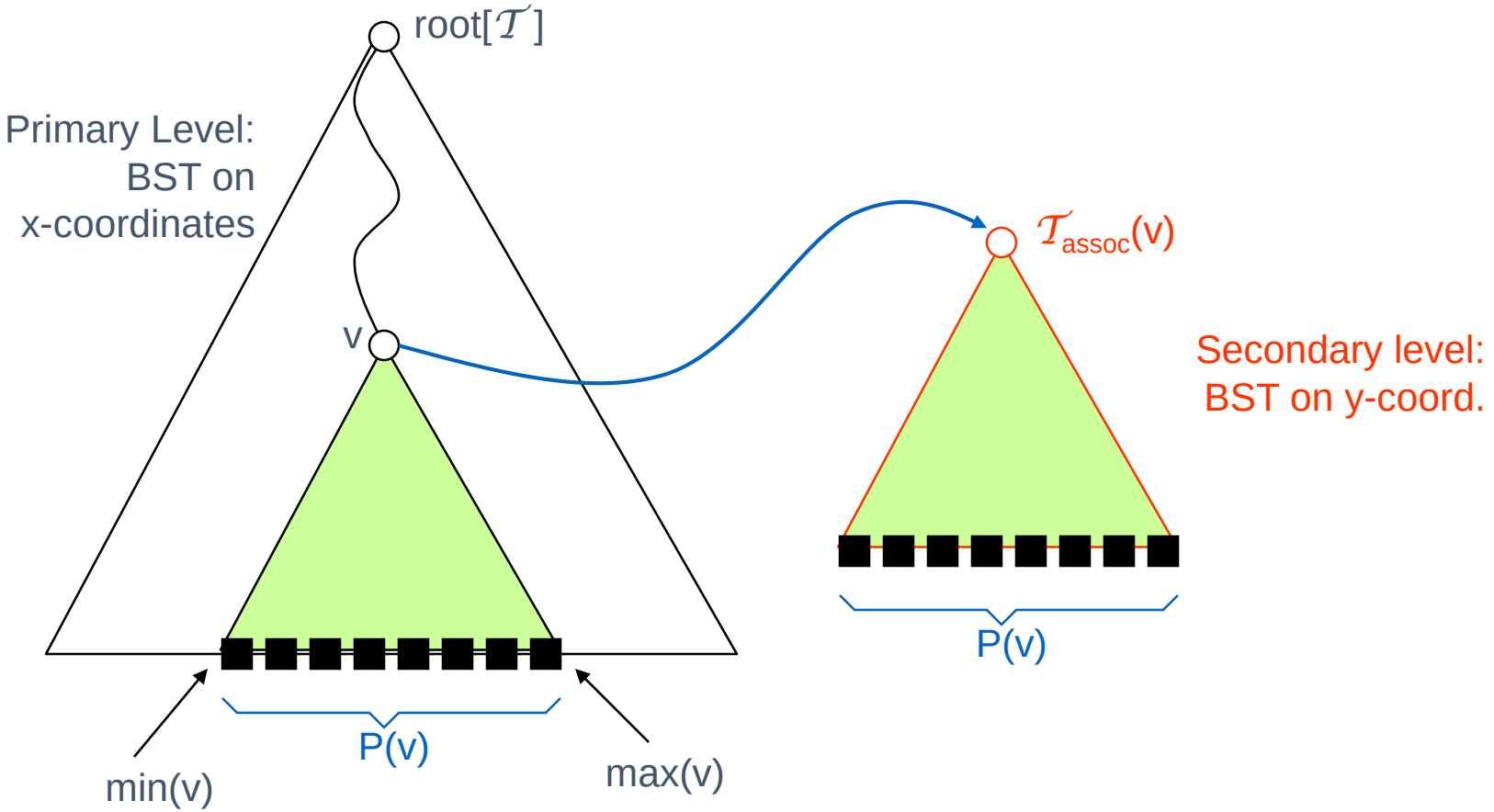
Input. A set P of points in the plane.

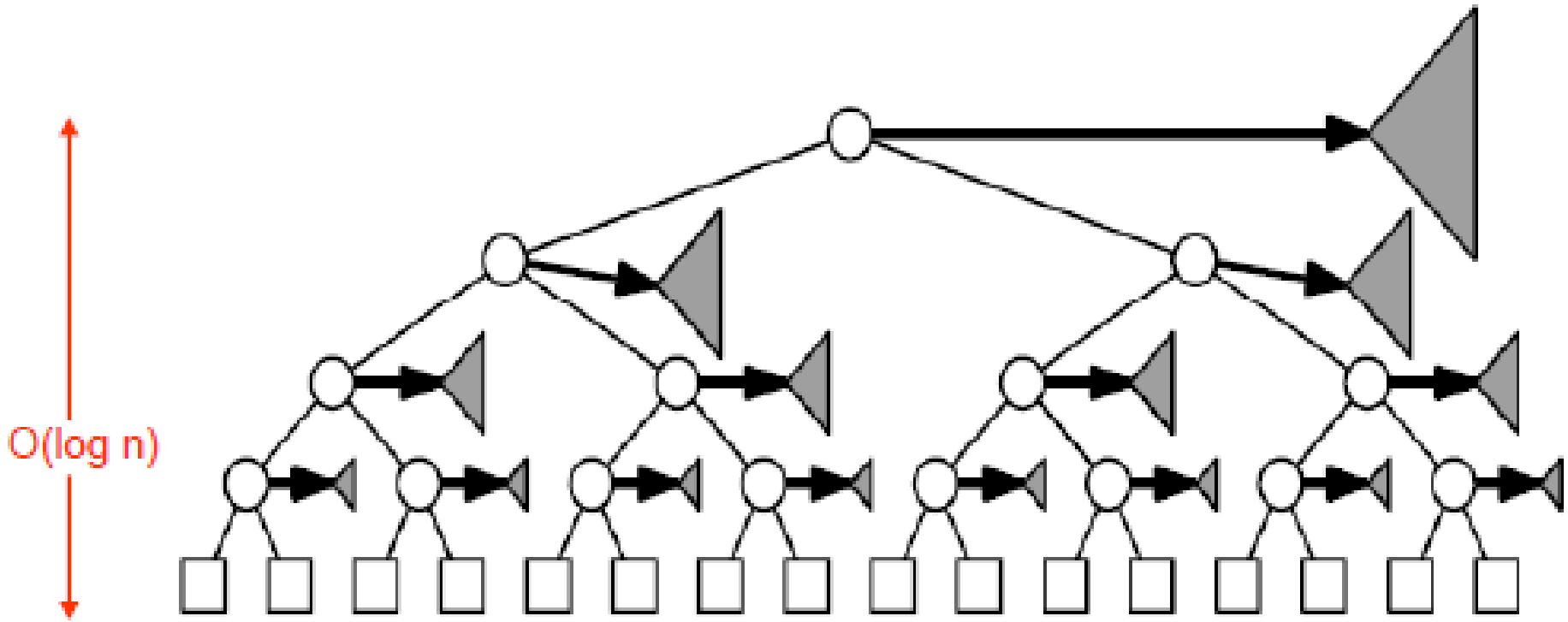
Output. The root of a 2-dimensional range tree.

1. Construct the associated structure: Build a binary search tree $\mathcal{T}_{\text{assoc}}$ on the set P_y of y-coordinates of the points in P. Store at the leaves of $\mathcal{T}_{\text{assoc}}$ not just the y-coordinate of the points in P_y, but the points themselves.
2. **if** P contains only one point
3. **then** Create a leaf v storing this point, and make $\mathcal{T}_{\text{assoc}}$ the associated structure of v.
4. **else** Split P into two subsets; one subset P_{left} contains the points with x-coordinate less than or equal to x_{mid}, the median x-coordinate, and the other subset P_{right} contains the points with x-coordinate larger than x_{mid}.
5. $v_{\text{left}} \leftarrow$ BUILD2DRANGETREE(P_{left})
6. $v_{\text{right}} \leftarrow$ BUILD2DRANGETREE(P_{right})
7. Create a node v storing x_{mid}, make v_{left} the left child of v, make v_{right} the right child of v, and make $\mathcal{T}_{\text{assoc}}$ the associated structure of v.
8. **return** v

Range Trees

2-level data structure:




O(log n)

- در این جا به بررسی سه مورد می‌پردازیم:
- زمان ساخت
- فضای ساخت
- زمان جستجو

زمان ساخت:

ساخت درخت وابسته: O(nlogn)

- یافتن میانه: O(1)
- رابطه‌ی بازگشتی: 2T(n/2)
- T(n)=O(nlogn)+O(1)+2T(n/2)

$T(n)=O(n\log^2 n)$

این یک زمان **بهینه** برای ساخت درخت نیست!!

با فرض مرتب بودن نقاط به طور **پیش فرض** براساس y داریم:

ساخت درخت وابسته: $O(n)$

یافتن میانه: $O(1)$

روابط بازگشتی: $2T(n/2)$

در نهایت داریم: $T(n)=O(n)+2T(n/2)$

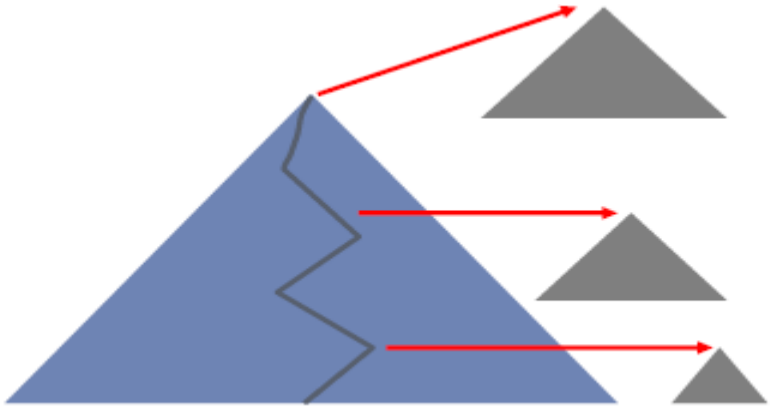
با استفاده از فرمول اصلی داریم: $T(n)=O(n\log n)$

لم ۵-۶: یک range treeروی یک مجموعه از nنقطه فضای O(nlogn) نیاز دارد.

اثبات:به تعداد نود های هر سطح درخت اول ,درخت وابسته داریم.مثلا در سطح اول یک نود وجود دارد و یک درخت وابسته با n نود,در سطح دوم ۲ نود با n/2 برگ وجود دارد که برای هر کدام درخت وابسته با n/2برگ وجود داردو....پس میتوان رابطه ی زیر را نوشت:

$$O(n)+2O(n/2)+4O(n/4)+...+nO(1)$$

عبارت بالا به O(n)تعلق دارد.تعداد سطوح درخت اولیهnlogn و هر سطح O(n)فضا نیاز دارد.پس فضای ذخیره سازی:O(nlogn)

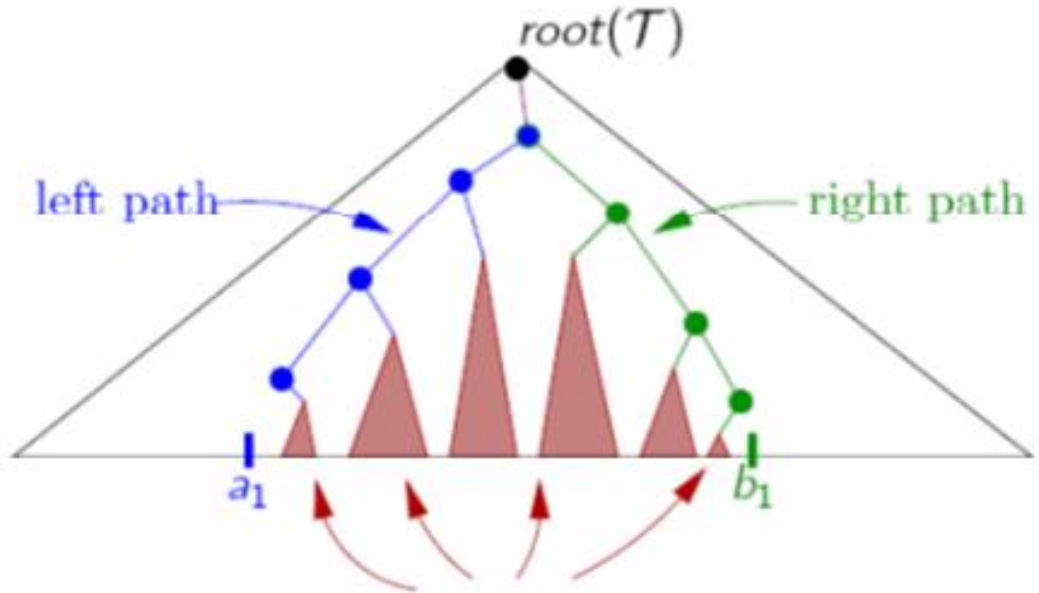


الگوریتم جستجو برای Range Tree دوبعدی:

۱-این الگوریتم در ابتدا در زمان O(logn) (برای هر گره میانی) زیرمجموعه‌ی کانونی را انتخاب می کند که همه‌ی آن‌ها با هم شامل نقاطی هستند که مؤلفه‌ی x آن‌ها در بازه‌ی [x,x’] قرار دارد.(که می‌تواند توسط یک جستجوی یک بعدی انجام شود.)

۲-نقاطی از مجموعه‌ی فوق مد نظر است که مؤلفه‌ی y آن‌ها نیز در بازه‌ی [y,y’] قرار داشته باشند.(این کار نیز توسط یک الگوریتم جستجوی یک بعدی قابل انجام است، که در این الگوریتم همان ساختارهای وابسته که قبلا در مجموعه‌ی کانونی ذخیره شده اند بکار می رود.)

این الگوریتم یعنی 2-DRANGEQUERY شبیه همان الگوریتم 1-DRANGEQUERY است که تفاوت این دو در فراخوانی REPORTSUBTREE است، که در این جا به جای آن از 1-DRANGEQUERY استفاده می‌شود.


root(T)
left path
right path
a_1
b_1

Algorithm 2DRANGEQUERY(*T.*[x:x']×[y:y'])

Input. A 2-dimensional range tree *T* and a range [x:x']×[y:y'].

Output. All points in *T* that lie in the range.

1. v_{split} ← FINDSLPITNODE(*T,*x,x')
2. **If** v_{split} is a leaf
3. **then** check if the point stored at v_{split} must be reported.
4. **else** (*follow the path to x and call 1DRANGEQUERY on the subtrees right of the path.*)
5. v ←lc(v_{split})
6. **While** v is not a leaf
7. **do if** x ≤ x_v
8. **then** 1DRANGEQUERY(T_{assoc}(rc(v)), [y:y'])
9. v ←lc(v)
10. **else** v ←rc(v)
11. Check if the point stored at v must be reported.
12. Similarly, follow the path from rc(v_{split}) to x', call 1DRANGEQUERY with the range [y:y'] on the associated structures of subtrees left of the path, and check if the point stored at the leaf where the path ends must be reported.

لم ۷-۵:

نشان دهید یک جستجوی مستطیلی موازی محور xها و yها در یک Range Tree که n نقطه ذخیره کرده است زمان $O((\log n)^2 + k)$ نیاز دارد که k تعداد نقاط گزارش شده است.

•اثبات:

❖ 1- دردرخت اصلیv در مسیر چپ و راست کلا برای این که تصمیم بگیریم آیا به فراخوانی DRANGEQUERY نیاز است یا نه به زمان O(logn) نیاز است.

❖ اگر تمام گره‌های میانی واقع شده در مسیر[x,x'] که طول logn دارند را بخواهیم، باید الگوریتم DRANGEQUERY را فراخوانی کنیم که طبق لم ۲-۵ از مرتبه‌ی

$$O(K_V + \log |Tassoc(v)|)=O(K_V + \log n_v)$$

است.

K_V تعداد نقاط گزارش شده است.

اگر v را تعداد نقاطی که به ازای آن‌ها الگوریتم فراخوانی می شود در نظر بگیریم زمان کل مصرف شده برای الگوریتم برابر است با:

$$\sum_{v} O(lognv + kv)$$

و داریم:

$$\sum_{v}(kv) + \sum_{v}(lognv)$$

که اگر K را تعداد تمام نقاط گزارش شده در نظر بگیریم، جمع اول در این رابطه برابر است با:

$$K=\sum_{v}(kv)$$

و چون الگوریتم 1-DRANGEQUERY حداکثر log n بار انجام می‌شود، داریم:

$$\sum_{v}(logn_v) = logn_v + \cdots + lognv$$

و می دانیم :

$$n_v <= n$$

پس حالت کلی n در نظر می‌گیریم و داریم:

$$\sum_{v}(logn) = logn + \cdots + log = (\log n)^2$$

و زمان کلی الگوریتم برابر است با:

$$O(\log^2 n+k)$$

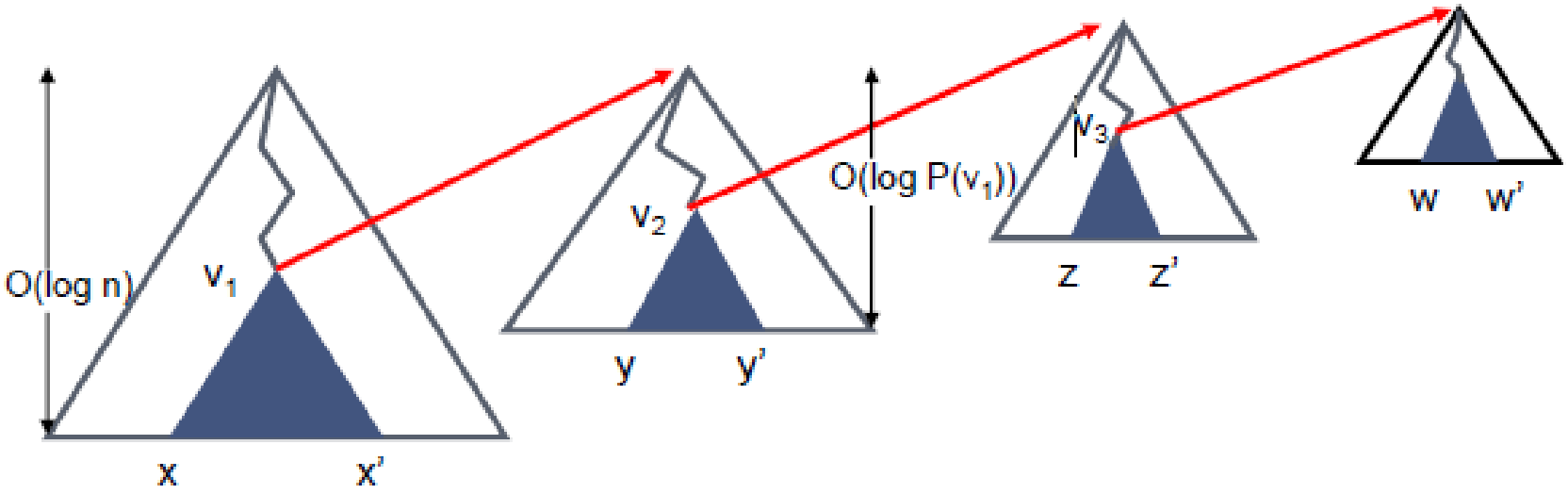
- **نکته:**

اگر تعداد نقاطی که در بازه‌ی [x.x’]*[y,y’] خواسته شود، در این صورت نیازی به گزارش تک تک نقاط نداریم و کافی است گره‌ای که برگهایش در محدوده است شمارش شود، که در این صورت مقدار k از زمان بدست آمده حذف می‌شود.

قضیه ۸-۵:

فرض کنید p یک مجموعه از n نقطه در صفحه باشد که یک Range Tree برای ذخیره سازی این p نقطه به زمان O(n logn) و برای ساخت هم به زمان O(n logn) نیاز دارد.جستجو بر روی Range Tree می‌تواند نقاطی از مجموعه‌ی p را که در مستطیلی موازی محور xها و y ها است را در زمان $O(\log n^2+k)$ می‌دهد. K تعداد نقاط گزارش شده است.

Higher_Dimensional Range Tree:



❖ برای ساختن Range Tree هایی در ابعاد بالاتر باید چندین سطح از درخت‌های وابسته داشته باشیم.هر درخت توسط اشاره گرهایی از درخت سطح قبل، قابل دسترسی است.

ساخت درخت‌هایی در ابعاد بالاتر:

(فرض کنید p مجموعه نقطه‌هایی در فضای d بُعدی باشد.)

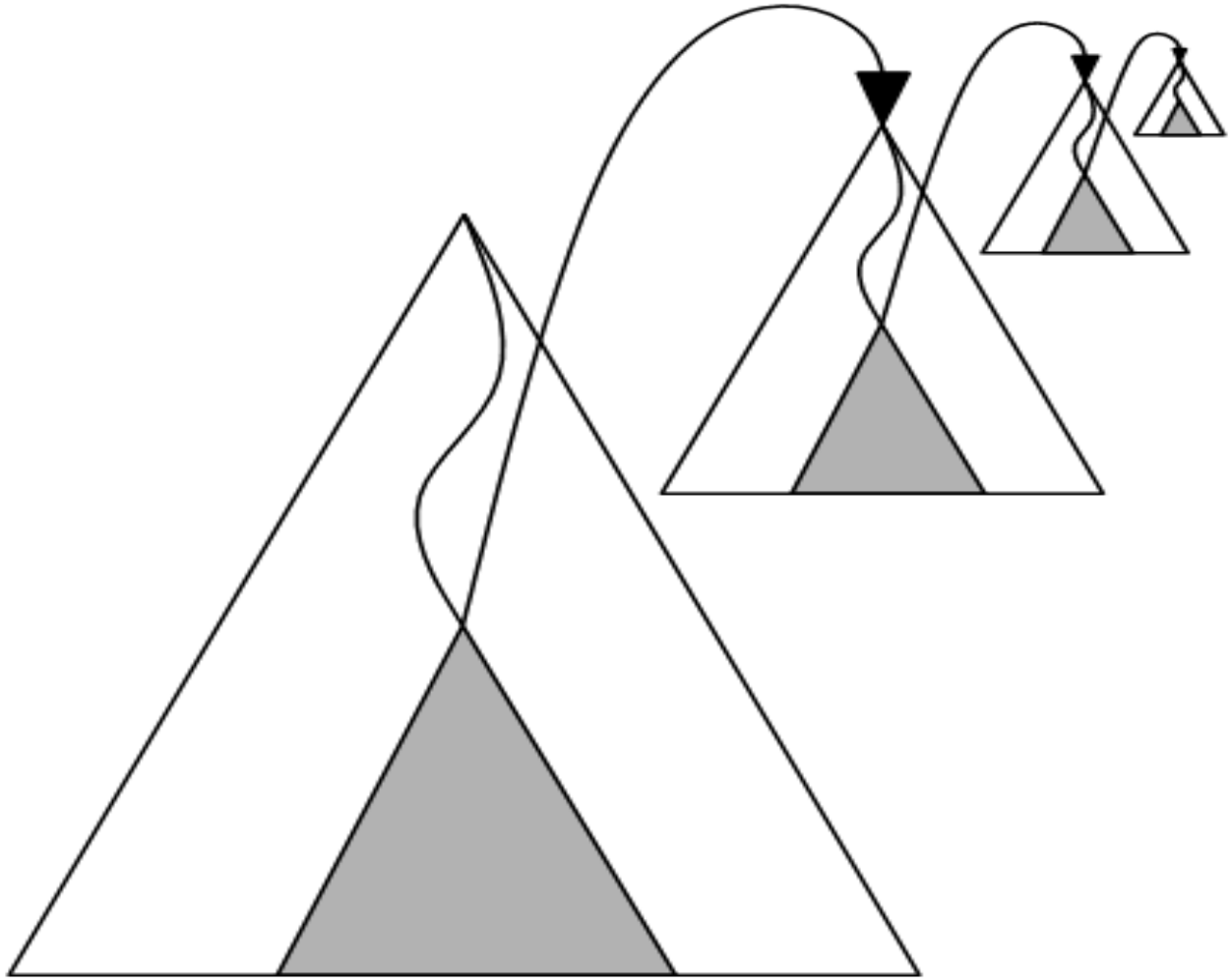
❖ ساختن یک درخت جستجوی دودویی متعادل براساس مؤلفه‌ی اول

❖در هر راس v در درخت سطح اول زیرمجموعه‌ی کانونی p(v)مجموعه نقاطی است که در برگ‌های زیردرخت با ریشه v قرار دارند.

❖برای هر راس v یک درخت وابسته Tassocc(v) می‌سازیم که بر اساس مؤلفه‌ی دوم ساخته شده است. این درخت d-1 بُعدی است.

❖ درخت به صورت بازگشتی ساخته می‌شود.

❖توقف بازگشت لحظه‌ای است که به یک درخت با عمق ۱ برسیم.



قضیه۹-۵:

فرض کنید p یک مجموعه از نقاط درفضای d بُعدی است. جایی که d>=2 ، یک Range Tree برای p زمان $O(n\log^{d-1}n)$ برای ذخیره سازی مصرف می‌کند و می‌تواند در زمان $O(n\log^{d-1}n)$ ساخته شود. زمان گزارش نقاطی که در محدوده‌ی مستطیلی خواسته شده هستند، $O(\log^{d}n+k)$ است، k تعداد نقاط گزارش شده است.

اثبات:

فرض کنید Td(n) زمان ساختن یک Range Tree بر روی مجموعه‌ی n نقطه در فضای d بُعدی باشد.

طبق قضیه‌ی ۸-۵ داریم:

$$T_2(n)=O(n\ \log n)$$

ساختن این درخت دو مرحله دارد:

ساخت درخت سطح اول(درخت اصلی)=> O(nlogn)

ساخت درخت های سطح‌های بعد (وابسته)=> کلا log n عمق گره‌های میانی و در صورتی که زمان ساخت درخت‌ها حداقل خطی باشدرابطه‌ی زیر برقرار است:

$T_d(n)=O(n\log n)+ O(\log n)*T_{d-1}(n)$

$O(n\log n)$ زمان ساخت درخت اصلی

$O(\log n)$ عمق درخت اصلی

$T_{d-1}(n)$ زمان ساخت درخت‌های وابسته

این رابطه با توجه به $T_2(n)=O(n\log n)$ و بر اساس نتیجه‌ی فرمول اصلی در زمان $O(n\log^{d-1} n)$ حل می‌شود. زمان ذخیره سازی نیز همین گونه حساب می‌شود.

زمان جستجو:

فرض کنید $Q_d(n)$ زمان جستجو روی یک Range Tree با d بُعد بر روی n نقطه را نمایش می‌دهد. (شامل زمان گزارش نقاط نمی‌شود.)

جستجو در ابتدا در درخت اصلی است که زمان جستجو در این سطح ازمرتبه‌ی $O(\log n)$ است.

همچنین شامل تعداد $\log n$ تا جستجو روی Range Tree های d-1 بُعدی می‌باشد.

$$Q_d(n)=O(\log n)+O(\log n)*Q_{d-1}(n)$$

که می‌دانیم:

$$Q_d(2)= O(\log^2 n)$$

با حل رابطه‌ی بازگشتی فوق داریم:

$$Q_d(n)= O(\log^d n)$$

حال با اضافه کردن زمان مورد نیازبرای گزارش نقاط، که برابر است با $O(k)$ به مقدار فوق داریم:

زمان کل جستجو=$O(\log^d n+k)$

General Sets Of Point:

تا کنون فرض بر این بود که در مجموعه‌ی p هیچ دو نقطه‌ای دارای مؤلفه‌ی x یکسان و y یکسان نیستند.

راه حل:

❖استفاده از اعداد ترکیبی قاموسی بجای اعداد حقیقی **Composite Number Space:**

❖نمایش اعداد ترکیبی که از هر دو عدد عدد a,b ساخته می‌شود به فرم زیر است:

$(a,b) \rightarrow (a \mid b)$

❖حال یک رابطه کلی روی فضای ترکیبی توسط ترتیب قاموسی تعریف می‌کنیم. پس برای دو عدد $(a \mid b)$ و $(a' \mid b')$ داریم:

$(a \mid b) < (a' \mid b') \Leftrightarrow a<a' \text{ or } (a=a' \ \& \ b<b')$

❖نمایش هر نقطه در مجموعه p بصورت $p = (px , py)$ اکنون بصورت:

$p' = ((px \mid py) , (py \mid px))$

در این روش مجموعه‌ی جدید p’ از n نقطه به وجود می‌آید.

- هدف گزارش نقاطی از مجموعه‌ی p است که در ناحیه‌ی R هستند. R=[x,x’]*[y,y’]
- ساختن درخت برروی p’

انتقال R به فضای ترکیبی جدید R’

R=[x:x’]×[y:y’] → R’ = [(x | -∞) : (x’ | +∞)] × [(y | -∞) : (y’|+∞)]

درستی روش با لم زیر اثبات می‌شود.

لم ۱۰-۵:

فرض کنید p یک نقطه باشد و Rectangular Rangeباشد. آنگاه:

$$p \in R \Leftrightarrow p' \in R'$$

اثبات:

فرض کنید p در ناحیه‌ی R قرار دارد، طبق تعریف p درR قرار دارد اگر وتنهااگر

$$x \le px \le x' \text{ و } y \le py \le y'$$

وبه راحتی می‌توان نشان داد حالت فوق برقرار است اگر و تنها اگر:

$$(x \mid -\infty) \le (px \mid py) \le (x' \mid +\infty) \text{و} (y \mid -\infty) \le (py \mid px) \le (y' \mid +\infty)$$