Orthogonal Range Searching

Querying a Database

عاطفه اکبری

Range Tree

Range tree

نتیجه:پرس و جو در یک kd-tree درزمان $O(n^{1-1/d}+k)$ قابل اجراست.

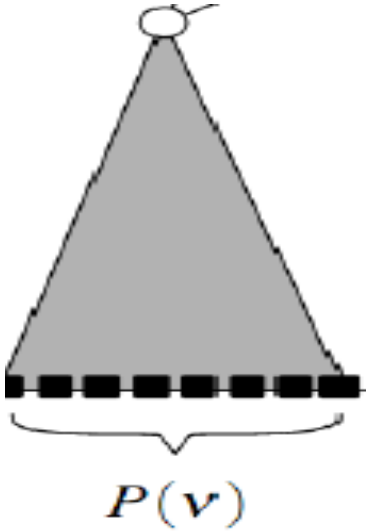
آیا میتوان زمان را بهبود داد؟ **بله**

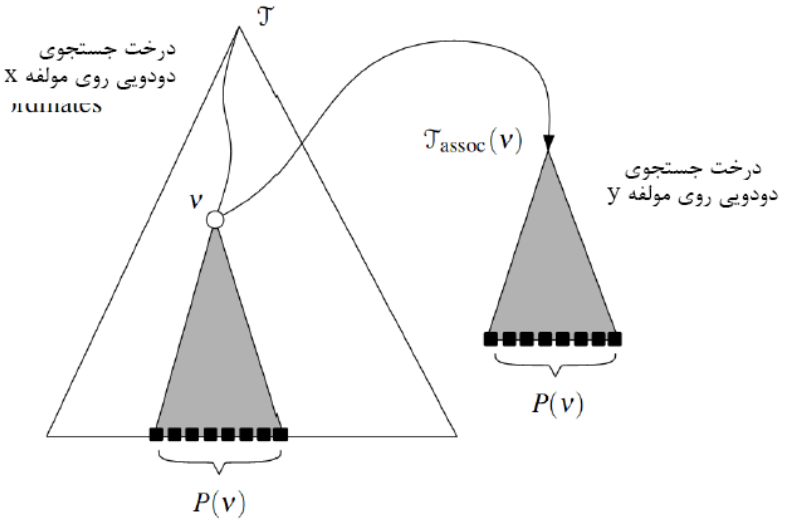
Range tree

زمان ساخت k_dtree از مرتبه‌ی $O(n)$ ⟸ $O(n \log n)$

زمان جستجو در k_dtree ازمرتبه‌ی $O(\sqrt{n} + k)$ ⟸ $O((\log n)^2 + k)$

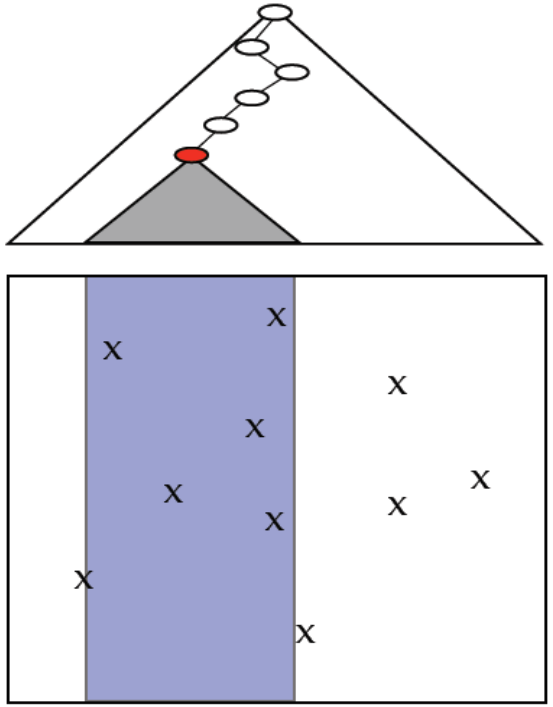
زیرمجموعه ی کانونی $p(v)$: مجموعه‌ی نقاطی که در برگ‌های زیر درخت با ریشه‌ی v ذخیره شده اند.

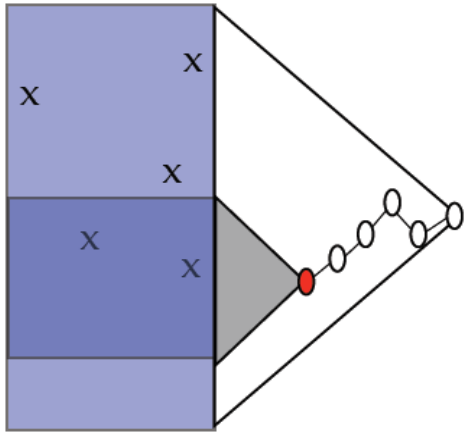




درخت اصلی ← بر اساس مؤلفه x

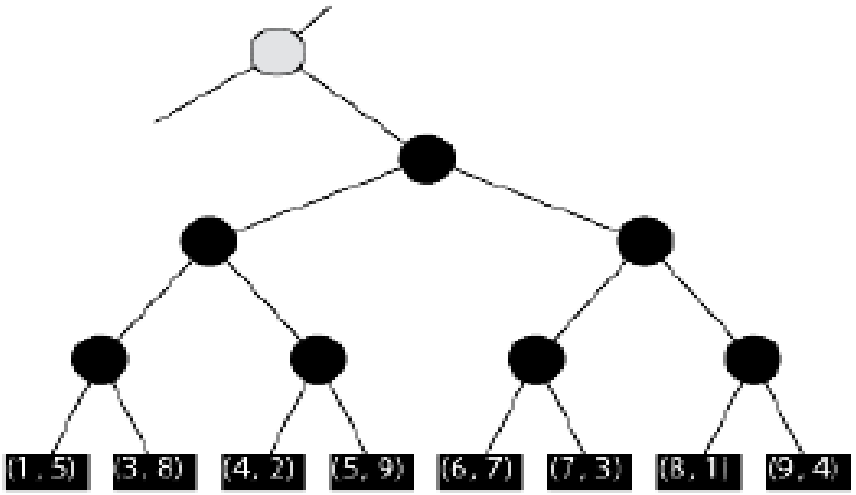
درخت وابسته ← بر اساس مؤلفه y

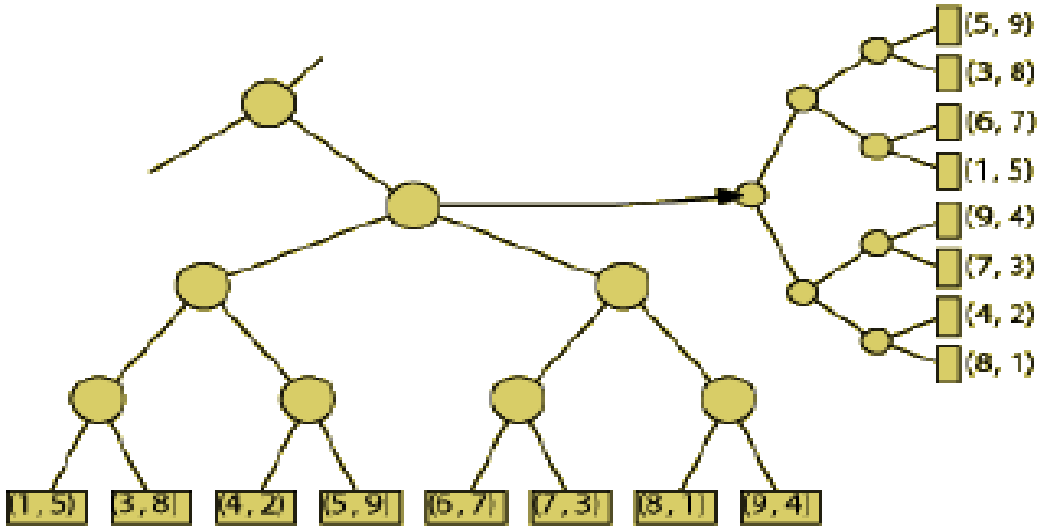




Points of P that lie in $[x: x']$ = U disjoint $P(v)$

Those points laying in $[y: y']$ are important.



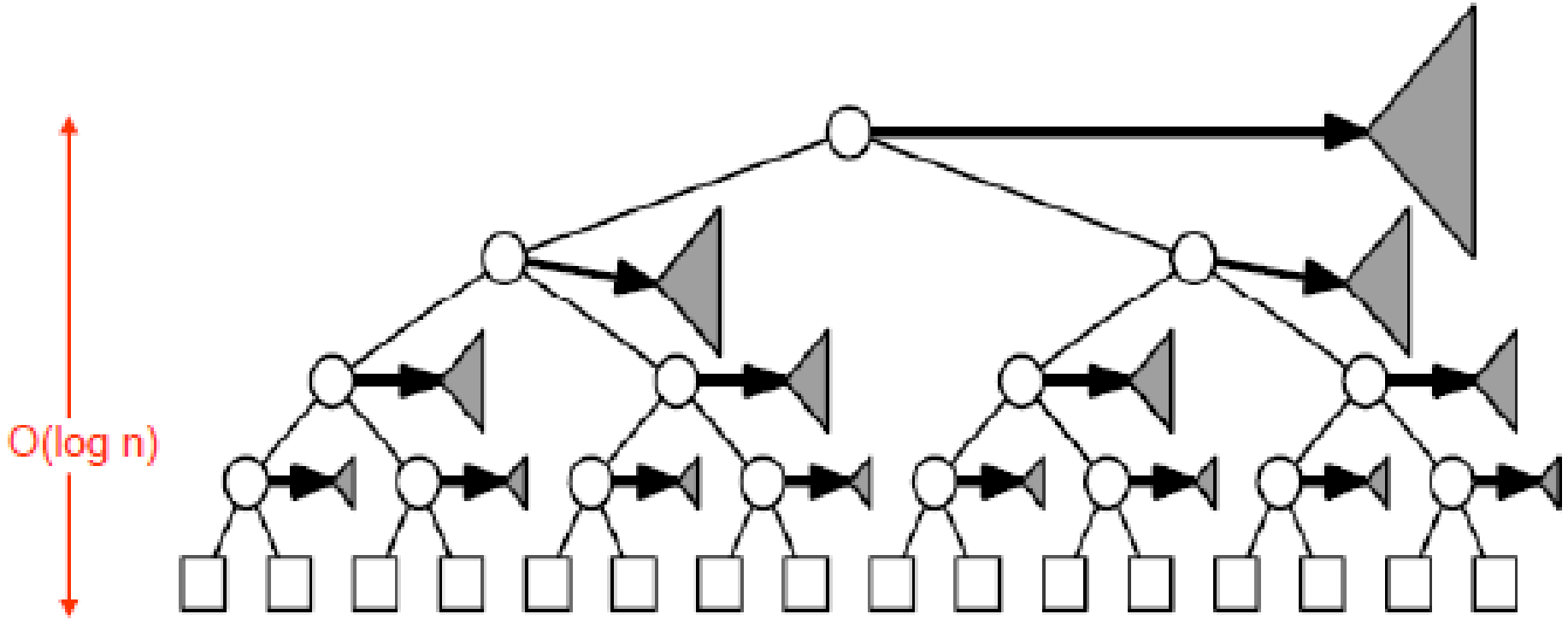


Algorithm BUILD2DRANGETREE(P)

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 $\mathcal{T}_{assoc}$ on the set P_y of y-coordinates of the points in P. Store at the leaves of $\mathcal{T}_{assoc}$ 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 $\mathcal{T}_{assoc}$ 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. $v_{left} \leftarrow$ BUILD2DRANGETREE(P_{left})
6. $v_{right} \leftarrow$ 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 $\mathcal{T}_{assoc}$ the associated structure of v.
8. **return** v


O(log n)

در این جا به بررسی سه مورد می‌پردازیم:

زمان ساخت

فضای ساخت

زمان جستجو

زمان ساخت:

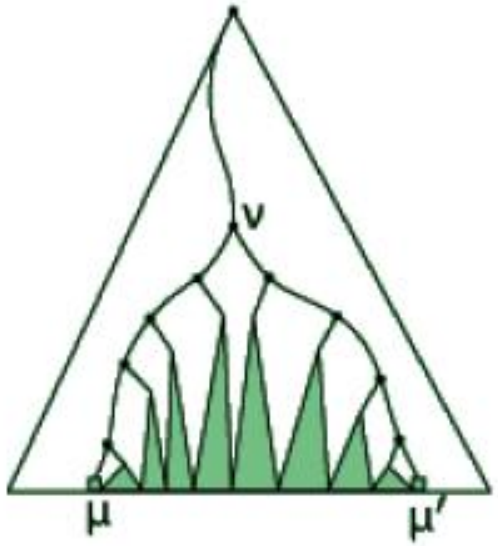
ساخت درخت وابسته: $O(n\log n)$

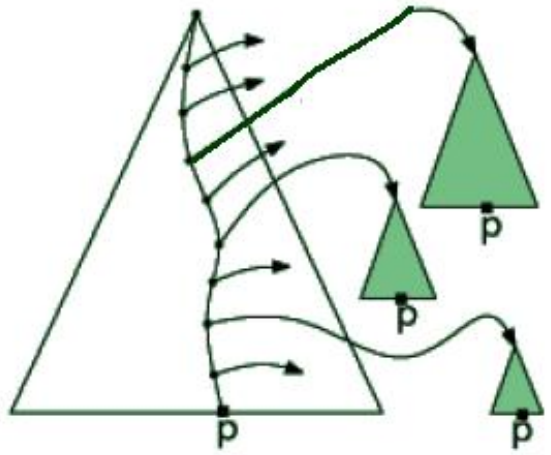
یافتن میانه: $O(1)$

رابطه‌ی بازگشتی: $2T(n/2)$

$$T(n)=O(n\log n)+O(1)+2T(n/2)$$

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





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(nlog n)

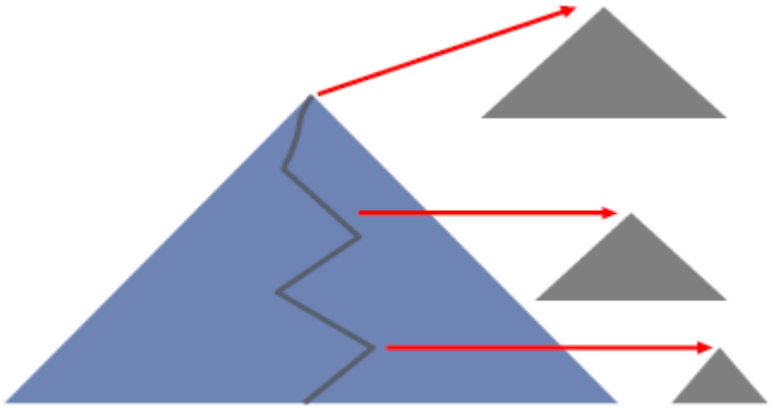
لم ۵-۶: یک range tree روی یک مجموعه از n نقطه فضای O(nlogn) نیاز دارد.

اثبات:

به تعداد نودهای 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/2 برگ 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 اولیه logn و هر سطح O(n)فضا نیاز دارد.پس فضای ذخیره سازی: O(nlogn)

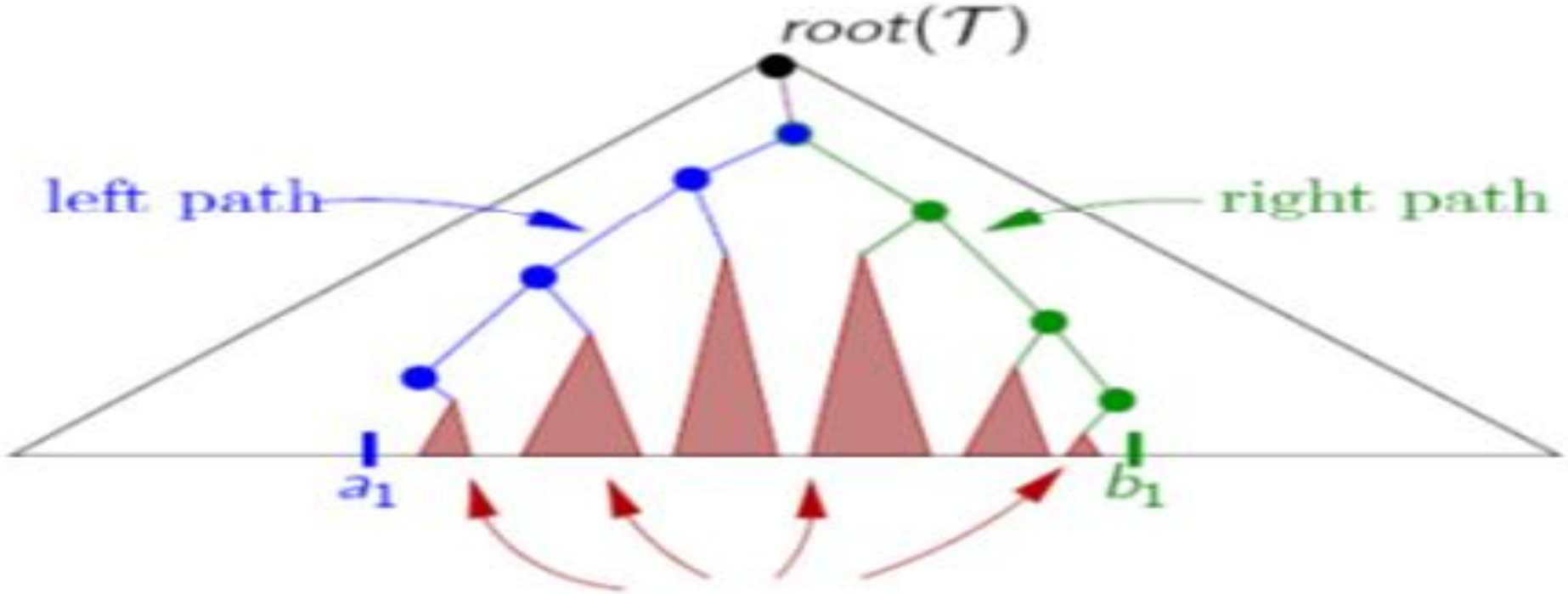


الگوریتم جستجو برای Range Tree دوبعدی:

۱-این الگوریتم در ابتدا در زمان O(log) (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:

- دردرخت اصلی v در مسیر چپ و راست کلا برای این که تصمیم بگیریم آیا به فراخوانی 1-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:

$$\sum_v O(logn_v + k_v)$$

و داریم:

$$\sum_{v}(k_v) + \sum_{v}(logn_v)$$

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

و چون الگوریتم 1-DRANGEQUERY حداکثر log n بار انجام می‌شود، داریم: $K=\sum_v (k_v)$

و می دانیم :

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

$n_V <= n$

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.

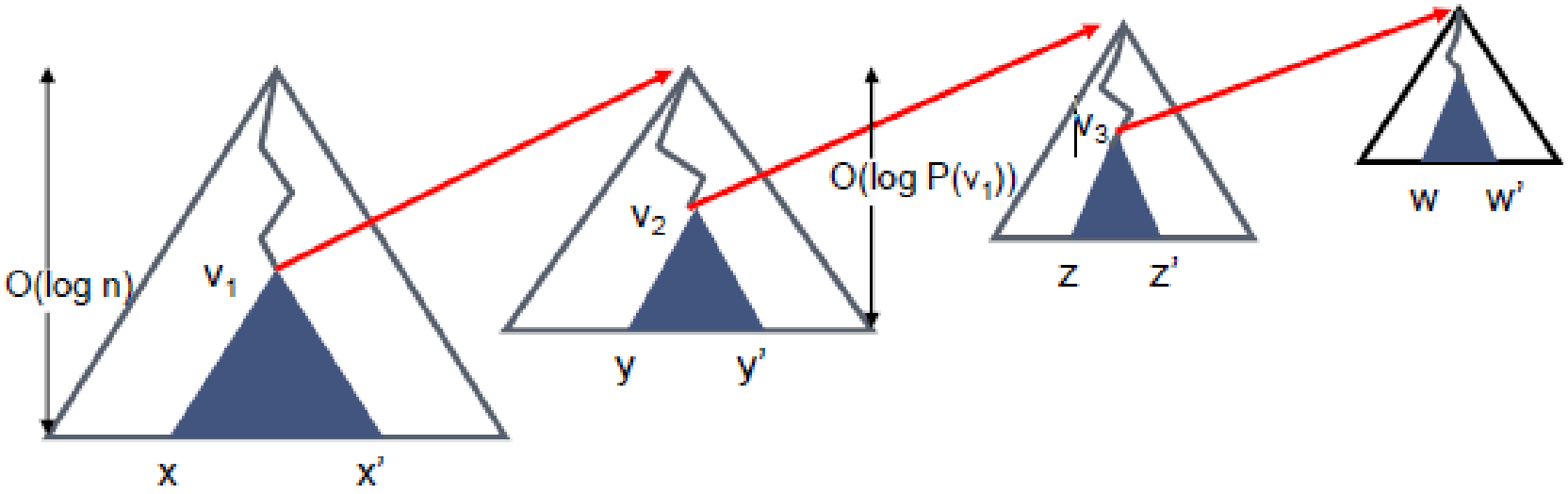
قضیه ۸-۵:

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 $O(logn^2+k)$ می‌دهد. K تعداد نقاط گزارش شده است.

مقایسه دو الگوریتم:

n	$\log n$	$\log^2 n$	$\sqrt{n}$
4	2	4	2
16	4	16	4
64	6	36	8
256	8	64	16
1024	10	100	32
4096	12	144	64
16384	14	196	128
65536	16	256	256
1M	20	400	1K

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:

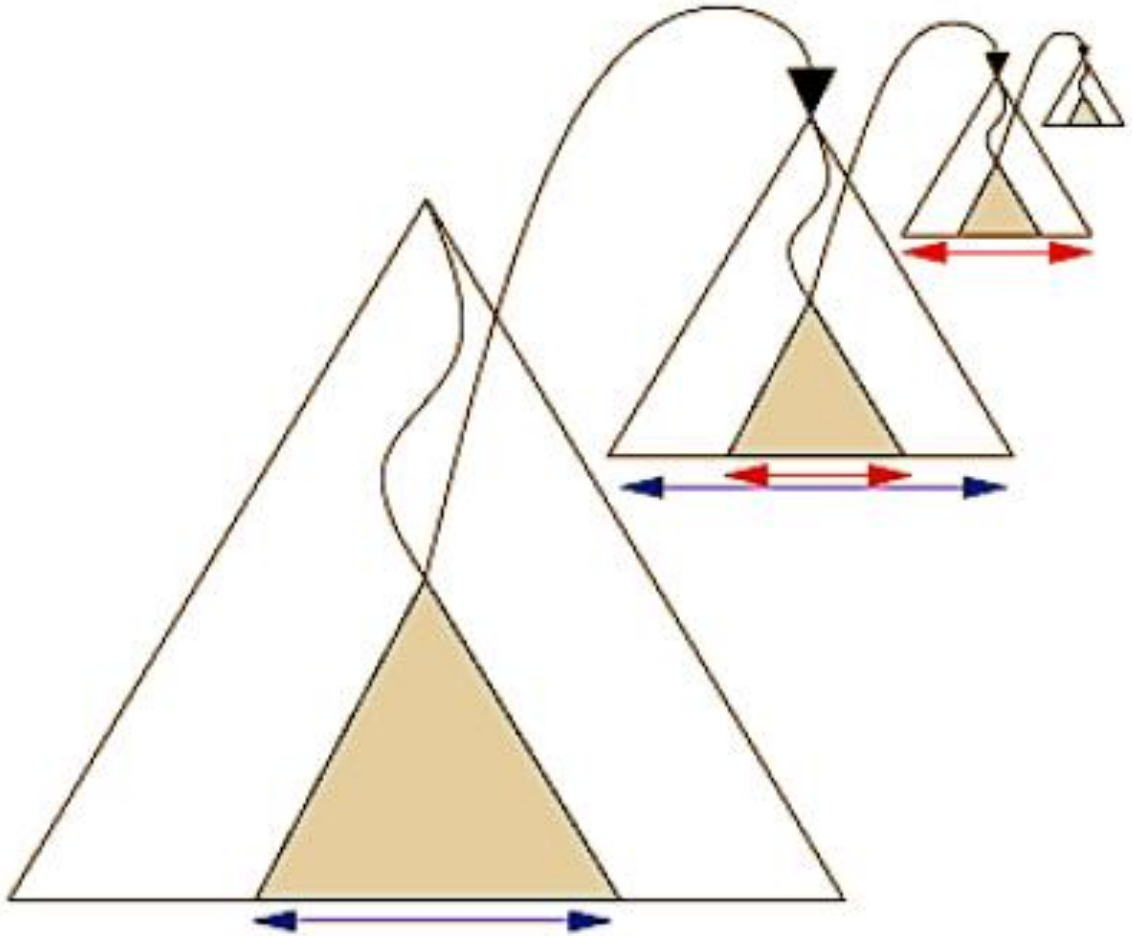
(فرض کنید p مجموعه نقطه‌هایی در فضای d بُعدی باشد.)

❖ 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 T̲assocc(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:

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

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 $p = (p_x , p_y)$ اکنون بصورت:

$p' = ((p_x \mid p_y) , (p_y \mid p_x))$

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

- هدف گزارش نقاطی از مجموعه‌ی p است که در ناحیه‌ی R هستند. $R=[x,x'] \times [y,y']$
- ساختن درخت برروی p'

انتقال R به فضای ترکیبی جدید R'

$$R=[x:x']\times[y:y'] \rightarrow R' = [\ (x \mid -\infty) : (x' \mid +\infty)\ ] \times [\ (y \mid -\infty) : (y' \mid +\infty)\ ]$$

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

$$x \le p_x \le x' \quad \text{و} \quad y \le p_y \le y'$$

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

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

با تشکر

؟؟؟