Vícerozměrné struktury

(1)

Typický problém: Máme množinu $X \subseteq \mathbb{R}^d$ → např. částečné dotazy v databázích
Chceme vyjmenovat/spočítat $X \cap [x_1, y_1] \times \_\_ \times [x_d, y_d]$ "obdélníkový" dotaz

Alternativy: • místo obdélníku mnohoúhelník/stěn)
• místo bodů složitější objekty, hledáme, které protnou dotaz

Jednoduché řešení: kd-stromy – konstrukce v $O(n \log n)$ → prostor $O(n)$ [konstanty závisí na d]
- dotazy v $O(n^{1-1/d})$ pro $d>1$
- pro lineární prostor to lépe nejde

Intervalové stromy (range trees)

1D – už známe, staticky by stačilo setříděné pole + bin. vyhledávání -- dotaz: $O(\log n + k)$ # hlášených bodů

2D – 1D strom podle x, vrcholy odpovídají pásům roviny
↳ v každém 1D strom podle y obsahující všechny body v pásu

• každý bod leží v 1 y-stromu na každé hladině
→ $O(\log n)$ kopií → $O(n \log n)$ prostoru celkem

• konstrukce (statická): body setřídíme zvlášť podle x a y, konstruujeme x-ový strom rekurzivně, dolů předáváme setříděné podseznamy: x-ový použijeme níže, y-ový pro konstrukci y-ového stromu v $O(n)$ → celkem $O(n \log n)$

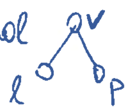
• dotaz: $[x, x'] \times [y, y']$ – interval $[x, x']$ rozložím na intervaly $O(\log n)$ vrcholů x-ového stromu, příslušným y-ovým položím dotaz $[y, y']$ → celkem $O(\log^2 n + k)$

d-D – 1D strom podle x_1, na jeho vrcholech visí $(d-1)$-D stromy
• $O(n \log^{d-1} n)$ prostoru
• konstrukce v čase $O(n \log^{d-1} n)$
• dotaz v čase $O(\log^d n)$, příp. $O(\log^d n + k)$ pro hlášení k bodů

Dynamické intervalové stromy

- problém: rotace v x-ovém stromu vyžaduje přebudovat y-ové stromy!
→ dynamizace částečnou přestavbou

1D – Pro vrchol



definujeme váhu $w(v) :=$ # vrcholů v podstromu
Vyžadujeme: $w(l) \geq \alpha \cdot w(v)$
$w(p) \geq \alpha \cdot w(v)$
} pro $\alpha = \frac{1}{2}$ dokonalá vyváženost
jinak zaručuje hloubku $O(\log_{\frac{1}{1-\alpha}} n)$
↳ $0 < \alpha < 1/2$

$\alpha = 1/3$ odpovídá $\frac{w(l)}{w(p)} \in [\frac{1}{2}, 2]$ míň problémy s $w=0$

Vyvažování: • vrchol si pamatuje váhu
• Ins/Del přepočítává váhy na cestě do kořene & kontroluje vyváženost
• pokud někde podmínka neplatí, rozebereme podstrom a postavíme nový, dokonale vyvážený
→ to stojí $O(w(v))$, ale zamortizujeme to
- optimalizace: stačí nejvyšší vrchol porušující podmínku

Amortizace: $\Phi(v) := \begin{cases} |w(l(v)) - w(p(v))| & \text{pokud je to} \geq 1 \\ 0 & \text{jinak} \end{cases}$

$\Phi := \sum_v \Phi(v)$

Insert/Delete zvýší Φ o $O(\log n)$
Přestavby zaplatíme z Φ

2D: přestavba v x-ovém stromu přestaví i y-ové stromy → trvá $O(w(v) \log w(v))$

- do Φ musíme spočítat $\log n$-krát více → $\log^2 n$
- též musíme spočítat na samostatné přestavby y-ových stromů → $\log^2 n$

} Insert/Delete stojí $O(\log^2 n)$ amort.

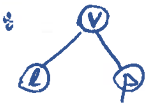
d-D: opět iterujeme → $O(\log^d n)$ amort.

Rychlejší 2D struktura (statická) → ~~ale~~ bude odpovídat v $O(\log n + k)$ → obecný případ zlepšíme na $O(\log^{d-1} n + k)$

- y-ové struktury budou setříděná pole

dotazy na jednotlivé y-ové struktury nejsou nezávislé

x:



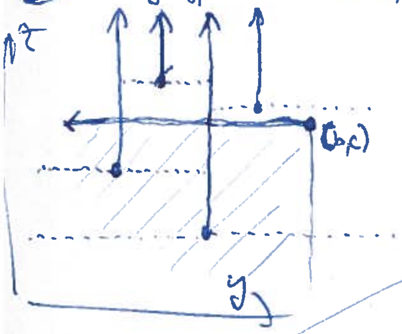
$Y(l), Y(p) \subseteq Y(v)$ ... stačí si pro každý prvek $Y(v)$ pamatovat jeho následníka v $Y(l)$ a $Y(p)$ (nebo $+\infty$)

→ znám-li výsledek dotazu na $Y(v)$, umím v $O(1)$ odpovědět na tentýž dotaz v $Y(l)$ a $Y(p)$

Důsledky:
- konstrukci nezpomalíme, prostor vzroste jen konst.-krát
- dotaz na y provádíme $\underbrace{1\times \text{ poctivě (v } Y(root)) + O(\log n) \text{ kroků do podmnožiny}}_{\text{celkem } O(\log n)}$
- rozbili jsme dynamizaci

Rychlejší 3D struktura – též $O(\log n)$, též statické

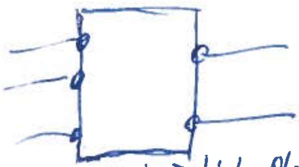
(1) dotazy typu $\mathbb{R} \times (-\infty, b) \times (-\infty, c)$



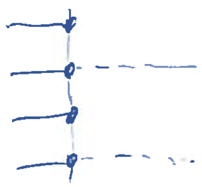
- z každého bodu vedu polopřímku nahoru
  které protne polopřímka z (b,c) doleva? "1st", kterou střílím do ostatních polopřímek
- přidáme úsečky z bodů doleva/doprava až k nejbližší polopřímce
  ↓
  rozklad roviny na obdélníkové oblasti
  ↓
  zapamatuji si graf sousednosti oblastí
  + posloupnost oblastí "úplně vlevo" (u přímky $x = -\infty$)
- dotaz: najdu v $O(\log n)$ oblast vlevo, kam padne dotazová polopřímka, pak pokračuji po oblastech doprava podél polopřímky, každá další oblast odpovídá dalšímu nalezenému bodu
  ↓
  složitost $O(\log n + k)$

zametením roviny v $O(n \log n)$

problém: oblast může být hodně složitá



nahoře/dole OK, vlevo/vpravo může být příliš hran
↓
nechám každou druhou pokračovat → na L/P straně bude max. 1 vnitřní bod
↓
to sestrojím dvojím zametením, ukážeme, že jsme přidali jen $O(n)$ dalších čar



každá druhá končí, každá druhá pokračuje
↓
pokračování účtuji koncům
+1 na zaokrouhlení
↓
$O(n)$ úsečkám, $O(n)$ oblastem

Celkově:

Konstrukce v $O(n \log n)$

Prostor $O(n)$

Dotaz v $O(\log n + k)$

↑

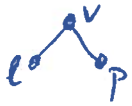
bin. vyhledávání dle z v seznamu levých oblastí

2) $[a_1,a_2]\times(-\infty,b)\times(-\infty,c)$

- intervalový strom podle x, v každém vrcholu struktura 1
- Konstrukce v $O(n\log^2 n)$, prostor $O(n\log n)$
- dotaz $\rightarrow$ $O(\log n)$ dotazů na 1 $\rightarrow$ $O(\log^2 n+k)$

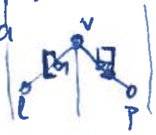
3) $[a_1,a_2]\times[b_1,b_2]\times(-\infty,c)$

- intervalový strom podle y



... v ~~p~~ struktura 2
v l její **obrácená** verze pro $[a_1,a_2]\times[b,\infty)\times(-\infty,c)$
(vlastně se liší jen tím, že v 1 uvažuje oblasti úplně vpravo a pak chodí doleva)

- konstrukce v $O(n\log^3 n)$, prostor $O(n\log^2 n)$
- dotaz: jdeme dolů, než narazíme na



$\rightarrow$ dotaz na 2 v p a otočenou 2 v l

$[a_1,a_2]\times(b_1,\infty)\times(-\infty,c)$ $\quad$ $[a_1,a_2]\times(-\infty,b_2)\times(-\infty,c)$

... $O(\underbrace{\log n}_{\text{zde}}+\underbrace{\log^2 n}_{2}+k)$

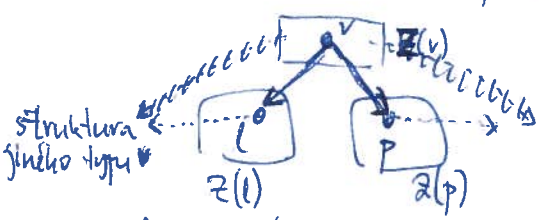
4) $[a_1,a_2]\times[b_1,b_2]\times[c_1,c_2]$

- uděláme z 3 podobně, jako jsme udělali 3 z 2
- intervalový strom ~~podle~~ podle z, používá 3 a obrácenou verzi 3 (tentokrát s otočením z)
- konstrukce v $O(n\log^4 n)$, prostor $O(n\log^3 n)$
- dotaz ... $O(\log n+\log^3 n+k)$

Mnohem pomalejší ... co z toho?

! Všechny dotazy na 1 jsou korelované – jsou to dotazy na různé seznamy podle z, všechny je lze získat jako podseznamy seznamu všech bodů postupným vyškrtáváním (rozmyslete si, že otáčení znamének neuškodí)

– každý seznam má $O(1)$ podseznamů, na které odkazuje



- levý + pravý podseznam
- podseznam ~~pro~~ pro pomocnou strukturu (typ struktury víme z toho, zda je v levém či pravém synu)

– tedy každý krok do podseznamu stojí $O(1)$

$\rightarrow$ celkem $O(\underbrace{\log n}_{\text{hledání v prvním seznamu}}+\underbrace{\log n}_{\text{všechny kroky do podseznamů}}+k)=O(\log n+k)$

ale konstrukce + paměť jsou $\log n$-krát horší a neumíme dynamicky upravovat

$\rightarrow$ $O(\log^{d-2} n)$ pro ~~R~~ d

Fractional Cascading

- seznamy L_1 — L_t o n ~~prvk~~ prvcích (nemusí mít nic společného)
- umíme najít polohu x ve všech seznamech v čase $O(\log n+t)$
- sestrojíme L'_1 – L'_t: $L'_i := L_i\cup$ (každý druhý prvek L'_{i+1})

  přidali jsme nejvýš $O(n)$ prvků (i nepřímo)
- prvek v $L'_i\setminus L_i$ si pamatuje ukazatel do L'_{i+1}

  prvek v L'_i si pamatuje nejbližší větší/menší opačného typu (původní/"stín" L'_{i+1})

~~po~~ v L_1 hledáme pačtivě, pak $O(1)$ na každý další seznam.