11. cvičení

Úloha 1 (Nejhorší případ pro k-d strom)
Mějme 2-d strom, tedy binární strom, kde rozdělujeme rovinu střídavě podle os x a y. Ukažte, že dvourozměrný intervalový dotaz (na počet bodů v obdélníku) může trvat až $\Omega(\sqrt{n})$, konkrétně najděte množinu bodů uloženou ve stromě a dotaz, který bude trvat takto dlouho.
Bonus: zkuste toto rozšířit na obecné k-d stromy s časovou složitostí $\Omega(n^{1-\frac{1}{k}})$.

Řešení
Vezmeme množinu $\{(i,i) : 1 \leq i \leq n\}$ a budeme se ptát na dotaz $\{0\} \times \mathbb{R}$. Pro jednoduchost mějme $n = 2^t - 1$. V každém x-vrcholu jdeme doleva, ale v každém y-vrcholu musíme jít oběma směry. Strom má hloubku $t \approx \log n$, a v každém druhém kroku zdvojnásobíme počet podstromů, na které se musíme podívat. Celková složitost tedy je $\Omega(2^{(\log n)/2}) = \Omega(\sqrt{n})$.
Rozšíření je analogické, jenom budeme mít k-tice $(i, \ldots, i)$ a dotaz bude $\{0\} \times \mathbb{R}^{k-1}$.

Úloha 2 (Hledání minima a mazání v k-d stromě)
Máme k-d strom. Navrhněte, jak na něm implementovat operaci nalezení nejmenšího prvku v dané dimenzi. (Tedy funkci FindMin(d), která nalezne prvek, jehož d-tá souřadnice je minimální.)
Dále navrhněte implementaci operace Delete. Pro jednoduchost můžete předpokládat, že žádné dva body spolu nesdílejí žádnou souřadnici.
(V obou případech neřešte časovou složitost.)

Řešení
Pro FindMin: Ve vrcholu, který řeže dimenzi rozdílnou od d-té vezmeme minimum z obou podstromů a sebe. Ve vrcholu, který řeže d-tou dimenzi, vezmeme minimum z podstromu odpovídajícímu nižší hodnotě souřadnic (a sebe).
Pro Delete: Najdeme vrchol, který mažeme, a nalezneme minimum v pravém podstromě pro souřadnici, podle které vrchol řeže. Z něj přesuneme vrchol výše na místo, které chceme smazat, a nyní chceme smazat právě vyprázdněné místo. Tím můžeme potenciálně přejít na mazání dalšího vnitřního vrcholu, ale ten bude ve stromě níže, takže po konečném počtu iterací se dostaneme do listu, který můžeme beztrestně smazat.

Úloha 3 (Nejbližší soused a k-NN)
Máme k-d strom. Navrhněte, jak na něm implementovat operaci nalezení nejbližšího souseda, a operaci nalezení m nejbližších sousedů.
Hint: pro m-NN se může hodit mít k dispozici haldu.

Řešení
Máme-li bod x, začneme jako bychom hledali x. Jakmile dorazíme do listu, zapamatujeme si poslední vrchol jako nejbližší. Budeme se v rekurzi vracet, a v každém vrcholu se podíváme, jestli vrchol není bližší, a jestli není šance, že v druhém podstromě může být bližší vrchol. (Vzdálenost nejbližšího bodu druhého obdélníka musí být menší než vzdálenost k nejbližšímu vrcholu.) Pokud šance je, podíváme se i do druhého podstromu.
Zlepšení: to samé, jenom si nejbližší vrcholy ukládám do haldy a beru vrcholy shora, abych měl aspoň nějaké omezení.

Úloha 4 (Zrychlení intervalových stromů)
Mějme zjednodušený dvourozměrný intervalový strom, tedy binární strom podle souřadnice x, který má v každém vrcholu pole bodů z daného x-ového intervalu seřazené podle y. Dvourozměrné intervalové dotazy (na počet bodů v obdélníku) zde trvají $\mathcal{O}(\log^2 n)$, protože potřebujeme binárně vyhledávat v $\mathcal{O}(\log n)$ polích. Ukažte, že přidáním (lineárně mnoha) pointerů z každého pole na správná místa v obou polích v synech můžeme složitost snížit na $\mathcal{O}(\log n)$.
(Pro d-rozměrný strom pak dostaneme $\mathcal{O}(\log^{d-1} n)$.)

Řešení
Pro každý vrchol v a jeho syna w povede z každého prvku pole ve vrcholu pointer na stejný prvek nebo největší menší v poli w. Binární vyhledávání poté provedeme jen v největším poli v kořeni. Pro nalezení odpovídajících hranic v poli v synovi pak stačí jen použít pointery a příp. se posunout o jeden prvek vedle. Během vyhledávaní

ve stromě řazeném podle x-ové souřadnice tak jen s konstantním zpomalením udržujeme hranice v poli v aktuálním vrcholu. Této metodě se říká Fractional cascading a obecně umožňuje i propojování polí se stejným řazením ale různými prvky.

Bonusy

Úloha 5 (Něco „praktičtějšího“)
Při hashování často počítáme modulo, ale to se přeloží do strojového kódu jako operace `idiv`, která je hodně drahá (latence pro 64 bitová čísla může být, dle procesoru, klidně i řádově desítky cyklů). Existují ale speciální prvočísla, tzv. Mersennova, pro která to jde rychleji. Ta mají tvar $p = 2^s - 1$.
Zkuste naimplementovat modulo Mersennovo prvočíslo jen pomocí bitových operací (bitshift, bitwise and/or), sčítání, odčítání a porovnávání (obecně rychlých operací).

Řešení
Chceme spočítat $k \mod p, p = 2^s - 1$. Stačí vzít `i = (k & p) + (k >> s)`, a vrátit `if i >= p then i-p else i`. Tohle předpokládá $k \leq 2^{2s} - 1$, jinak bude potřeba počítat rekurzivně.
Proč? Rozdělíme $k = x \cdot 2^s + r \equiv_{2^s-1} x + r$, kde r získáme bitmaskou, a k jsme bitshiftem vydělili 2^s.