kd-stromy (kd-trees)

k čemu to je:

- ukládání vícerozměrných dat (k-dimenzionální data)
- **vstup:** Množina bodů (nebo složitějších geometrických objektů) v k-dimenzionálním prostoru.
- **problém:** Zkonstruovat strom, který rozděluje prostor polorovinami t.ž. každý objekt je obsažený ve svém vlastním kvádru.
- **záměr:** rozložit plochu (prostor, ...) na malý počet buňěk tak, aby žádná neobsahovala velký počet vstupních objektů. Snadný přístup k objektům na základě jejich pozice.

Zdroje: H. Samet. Multidimensional spatial data structures. In D. Mehta and S. Sahni, editors, Handbook of Data Structures and Applications, pages 16:1– 16:29. Chapman and Hall / CRC, 2005.

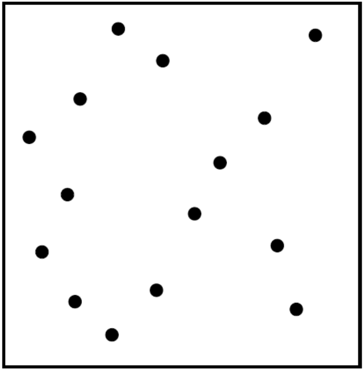
H. Samet. Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann, 2006.

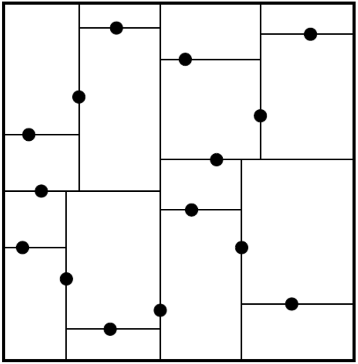
Skiena S. S. The Algorithms Design Manual. Springer, New York, 1998. ISBN 0-387-94860-0.

pozn.: přestože u jednotlivých instancí by měl být uváděn název podle rozměru (tj. 2d-stromy, 3d-stromy, atd.), používá se název kd-strom.

*k*d-stromy (*k*d-trees)

- **vstup:** Množina bodů (nebo složitějších geometrických objektů) v *k*-dimenzionálním prostoru.
- **problém:** Zkonstruovat strom, který rozděluje prostor polorovinami t.ž. každý objekt je obsažený ve svém vlastním kvádru.





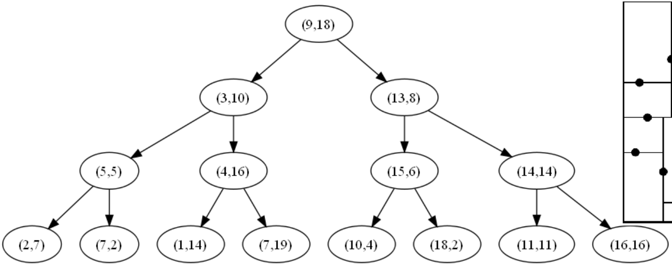
kd-stromy (kd-trees)

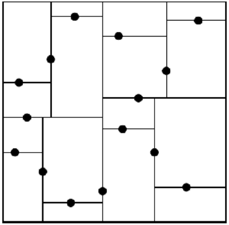
Typický algoritmus konstrukce (statický kd-strom):

- konstrukce kd-stromu rozkladem množiny bodů.
- každý uzel ve stromě je určen rovinou skrze jeden ze svých rozměrů.
- $=$ rozklad bodů na ty nalevo a napravo (resp. nad a pod)
- tak, aby jich bylo stejně.
- ty jsou pak děleny stejným způsobem (přes jiné rozměry)
- po $\log(n)$ krocích se rozklad zastaví, každý bod je v jedné vlastní buňce (v listu).

Existují alternativní konstrukce kd-stromu, které vkládají body postupně a dělí příslušné buňky, přestože takové stromy se snadno stanou velmi nevyváženými.

kd-stromy (kd-trees)





Prostor: $\mathcal{O}(n)$ – protože každý uzel $=$ jeden bod,
Výška: $\log(n)$ – kvůli dělení

Typický algoritmus konstrukce – podrobněji:
jak provádět dělení?

- setřídit + rozseknout $\mathcal{O}(n \cdot \log^2(n))$
- random $\mathcal{O}(n^2)$ (kvůli možnosti degenerace, nestává se často)
- lineární výběr mediánu $\mathcal{O}(n \cdot \log(n))$

lineární výběr mediánu:

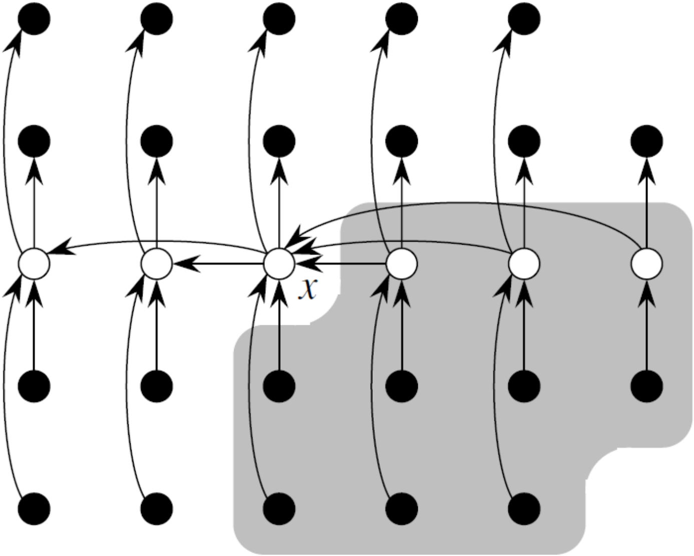
- známá záležitost z QuickSortu
- Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms. Second Edition. MIT Press, 2001. ISBN 0-262-53196-8.

Intermezzo: lineární výběr mediánu

Algoritmus SELECT:
(výběr i-tého nejmenšího prvku)

1. Rozděl n prvků vstupního pole na $\lceil n/5 \rceil$ skupin o 5 prvcích (popř. s jednou se zbývajícími ($n \mod 5$) prvky).
2. Najdi medián každé z $\lceil n/5 \rceil$ skupin (setřídění insert-sortem + vybrání prostředního)
3. Použij rekurzivně algoritmus SELECT na nalezení mediánu x z $\lceil n/5 \rceil$ mediánů nalezených v kroku 2.
4. Rozděl pole (PARTITION z QuickSortu) podle x. $k = 1+$počet prvků v dolním segmentu, takže x je k-tý nejmenší prvek a $n - k$ prvků je v horním segmentu.
5. Porovnej i a k:
   - Pokud $i = k$, vrať x.
   - Pokud $i < k$ rekurzivně použij SELECT na nalezení i-tého nejmenšího prvku v dolním segmentu.
   - Pokud $i > k$ rekurzivně použij SELECT na nalezení $(i - k)$-tého nejmenšího prvku v horním segmentu.

Analýza algoritmu Select



Intermezzo: lineární výběr mediánu

Počet prvků větších než x:

$$3\left(\left\lceil\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil\right\rceil - 2\right) \geq \frac{3n}{10} - 6$$

to znamená, že $\textsc{Select}$ se rekurzivně zavolá nejvýše na $7n/10+6$ prvků v kroku 5.

Funkce $T(n)$ reprezentující časovou složitost algoritmu $\textsc{Select}$:

- Kroky 1,2 a 4 zaberou $\mathcal{O}(n)$ času.
- Krok 3 zabere $T(\lceil n/5\rceil)$ času
- Krok 5 zabere $T(7n/10+6)$ času

Záhadný předpoklad: Na 140 a méně prvku stačí konstantní čas.

$$T(n) \leq \begin{cases} \mathcal{O}(1) & \text{pokud } n \leq 140, \\ T(\lceil n/5\rceil) + T(7n/10+6) + \mathcal{O}(n) & \text{pokud } n > 140 \end{cases}$$

Intermezzo: lineární výběr mediánu

$$T(n) \leq \begin{cases} \mathcal{O}(1) & \text{pokud } n \leq 140, \\ T(\lceil n/5 \rceil) + T(7n/10 + 6) + \mathcal{O}(n) & \text{pokud } n > 140 \end{cases}$$

Ukážeme, že čas je lineární (substitucí):
$T(n) \leq cn$ pro dostatečně velkou konstantu c a všechna $n > 0$.

pro $n \leq 140$ – jasné;

předpokladejme $\mathcal{O}(n)$ (nerekurzivní složka $T(n)$) je ohraničená $a \cdot n$:
$\mathcal{O}(n) \leq an$ pro všechna $n > 0$.
Použitím substituce na pravé straně rekurence:

$$\begin{aligned} T(n) &\leq c\lceil n/5 \rceil + c(7n/10 + 6) + an \\ &\leq cn/5 + 7cn/10 + 6c + an \\ &= 9cn/10 + 7c + an \\ &= cn + (-cn/10 + 7c + an). \end{aligned}$$

což je nejvýše cn pokud $-cn/10 + 7c + an \leq 0$.

(...pokračování z předchozího slajdu) což je nejvýše cn pokud $-cn/10 + 7c + an \leq 0$.

Ta nerovnost je ekvivalentní s $c \geq 10a(n/(n-70))$. Protože předpokládáme, že $n \geq 140$, dostaváme $(n/(n-70)) \leq 2$... A tedy: výběr a a c, t.ž. $c \geq 20a$ splňuje tu nerovnost.

Pozn.: na „záhadné konstantě“ 140 vlastně není nic záhadného, klidně bychom mohli zvolit cokoli většího než 70 a podle toho vybrat c.

konec intermezza

kd-stromy (kd-trees)

Point location: nalézt buňku, ve které se nalézá dotazovaný bod q ($\mathcal{O}(\log(n))$)

Nearest neighbor search: Nalést bod ve stromu S, který je nejblíže k dotazovanému bodu q.

- provede se **Point location** a najde se buňka c obsahující q.
- q je ohraničená nějakým bodem p (to ale nemusí být nejbližší bod)
- zjistí se všechny buňky c' ve vzdálenosti $d(p, q)$.
- testne se jestli c' neobsahují bližší bod

Range search: nalézt všechny body, které se nalézají v daném čtverci / v dané oblasti.

Partial key search: Hledání na základě neceléo klíče.

Např.: v 3d-stromu s dimenzemi věk, výška, váha, hledáme někoho s věkem 35 a výškou 174cm, ale neznáme váhu.

*k*d-stromy (*k*d-trees)

Varianty *k*d-stromů: liší se v tom, jak je volen řez:

- **cyklení přes dimenze:** $d_1, d_2, \ldots, d_k, d_1, \ldots$
- **řezání největší dimenze:** vybírá se tak, aby výsledné buňky byly, co nejbližší čtvercům / krychlím / ...
- **Quadtrees a Octtrees:** namísto jednoho řezu se použijí řezy přes všechny dimenze $\rightarrow$ 4 potomci ve 2d, 8 potomků ve 3d.
- **BSP-trees:** (Binary space partitions) používají obecné řezy (nejen paralelní k osám), například oddělělení polygonů.
- **R-trees:** . . . to už je jiná struktura.

Modifikování *k*d-stromu:
Inkrementální vkládání Najde se vhodný list a rozřízne se

Mazání Najde se mazaný uzel, odstraní se, jeho uzly z potomků se tam znovu naskládají (inkrementálním vkládáním)

(tyto operace ale dělají strom nevyvážený)

k-D-B-stromy

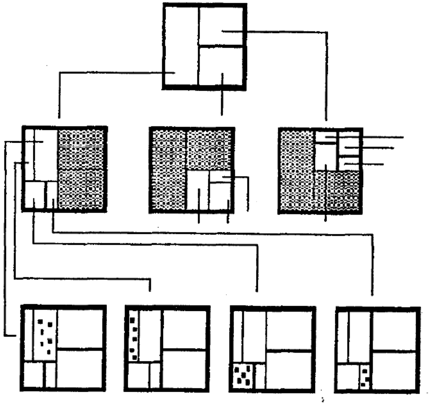
(vyvažovací varianta kd-stromů; kombinace kd-stromů a B-stromů)
Dva druhy stránek:

1. *Region pages* – obsahuje kolekci párů $(\text{region}, \text{page id})$ (max M)
2. *Point pages* – obsahuje kolekci párů $(\text{bod}, \text{lokace})$; lokace je umístění záznamu v databázi (max M).

Vlastnosti k-D-B-stromu

- každou stránku považujeme za uzel a každé page id v region page za ukazatel na ukazatel na uzel.
  Dále: žádná region page neobsahuje null ukazatel, a žádná region page není prázdná
- délka cesty z kořene k listu je stejná
- v každé region page, regiony tvoří rozklad
- pokud kořen je region page, sjednocení regionu je celá oblast $(d_1 \times d_2 \times \cdots \times d_k)$
- pokud máme $(\text{region}, \text{page id})$ a stránka na kterou se odkazuje page id je region page, pak sjednocení regionů v ní je region
- pokud máme $(\text{region}, \text{page id})$ a stránka na kterou se odkazuje page id je point page, všechny body v ní leží v regionu.

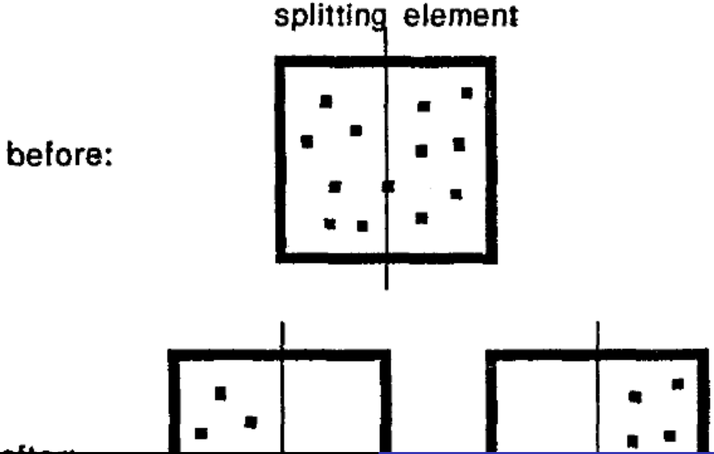
Příklad: 2-D-B strom



John T. Robinson: The K-D-B-tree: a search structure for large multidimensional dynamic indexes, International Conference on Management of Data archive Proceedings of the 1981 ACM SIGMOD international conference on Management of data, pp 10 – 18.

Vyhledávání: v podstatě známe;
Vkládání: v podstatě známe, jenom potřebujem split

Split podle bodu x_i: point page body se rozdělí na ty „napravo" od x_i a na ty „nalevo" – vznikají dvě nové point page.



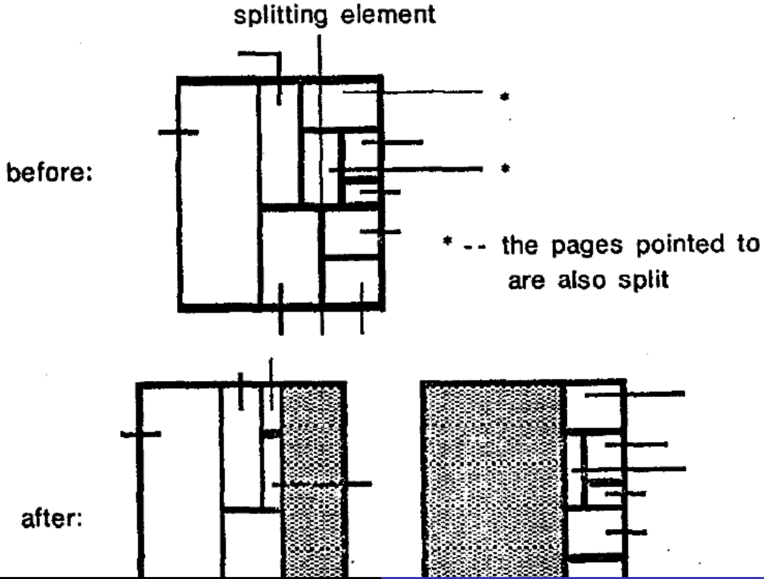
Split podle bodu x_i: region page

Pro každý pár (region, page id) ve staré region page:

1. Pokud region leží nalevo od řezu, dá se do levé stránky,
2. Pokud region leží napravo od řezu, dá se do levé stránky,
3. Jine
   - Splitni stránku, na kterou se odkazuje page id podle x_i – vznikají stránky s id: left id, right id
   - Rozděl region podle x_i na levý a pravý.
   - Dej do levé stránky (levý, left id) a do pravé (levý, left id);

pozn.: Split je tedy rekurzivní

Split podle bodu x_i: region page



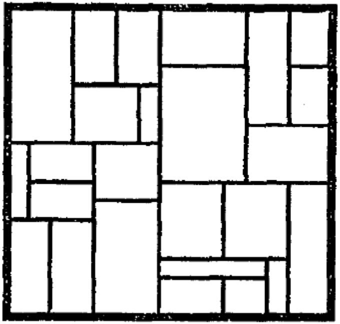
Vkládání záznamu (bod, lokace)

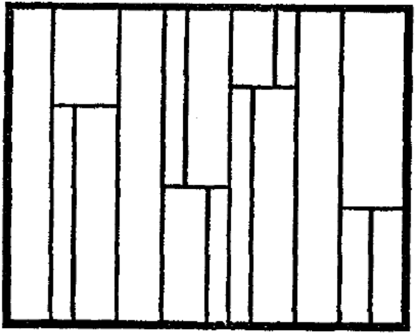
- Pokud je strom prázdný, vytvoř kořen s (bod, lokace), skonči.
- Pokus se vyhledat bod, pokud už tam je, skonči.
- Přidej (bod, lokace) do point page nalezené v předchozím bodu, pokud není přeplněná, skonči.
- Vyber rozměr a bod, podle kterého se provede split. Proveď split.
- Pokud nešlo o kořen, uprav předka (protože potomek byl splitnutý), pokud to povede k přeplnění, pokračuj rekurzivně předchozím bodem.
- Pokud šlo o kořen, vytvoř nový kořen, skoči.

Varianty algoritmu: různé druhy výběru rozměru, podle kterého se provádí split (čtvrtý bod)

Vkládání záznamu (bod, lokace)

Varianty algoritmu: různé druhy výběru rozměru, podle kterého se provádí split (čtvrtý bod)





Mazání záznamu (bod, lokace)
Základ: najdi záznam a smaž ho.

Narozdíl od B-stromů, uzly k-D-B-stromu stačí, když jsou neprázdné; merge je potřeba provádět méně často.

Merge je podobný jako u B-stromů, občas nastává problém s tím, že 2 regiony se nedají spojit tak, aby tvořily region: