

Prostorová indexace.

Hilbert Curve. Z-Curve. Quad Tree. k-D Tree. R-Tree. Clustering.

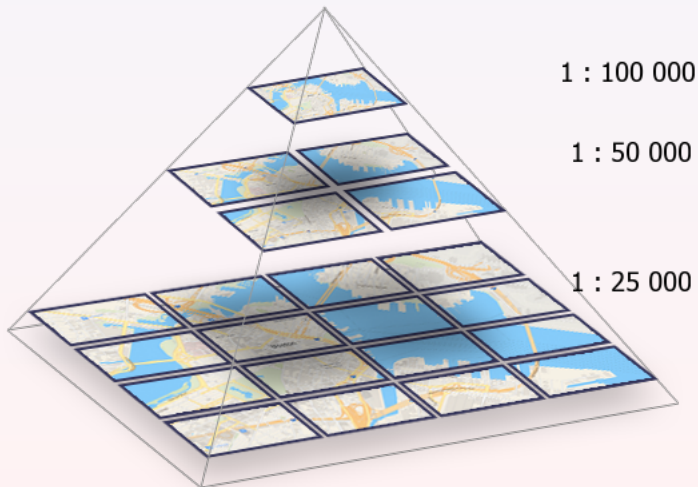
Tomáš Bayer | bayertom@natur.cuni.cz

Katedra aplikované geoinformatiky a kartografie, Přírodovědecká fakulta UK.

1. Motivace

Multiměřítková reprezentace kartografických dat.

Jakou dlaždici načíst při posunu mapy v aktuálním zvětšení?



2. Prostorová data

Prostorová data reprezentují objekty na zemském povrchu.
Mají geometrickou a atributovou složku.

Prostorové dotazy: práce s podmnožinou prvků splňujících kritérium.
Využívají vzájemné prostorové vztahy mezi jednotlivými entitami.
Prostorové vazby budovány v rámci předzpracování.

Nejčastější prostorové dotazy:

- Nalezení k –nejbližších sousedů (KNN Search).
- Nalezení objektů ležících v intervalu (Interval Search).
- Nalezení objektů ležících uvnitř jiného objektu (Location Search).
- Nalezení objektů ležících v určité vzdálenosti od jiných (Distance Search).

Datové soubory rozsáhlé, efektivní implementace prostorových operací.

Prostorové dotazy lze urychlit s využitím *prostorové indexace*.

Zásadní při zpracování Big-Data.

Metody vhodné pro 0D-2D entity, lze rozšířit i pro 3D.

Topologická korektnost výstupů.

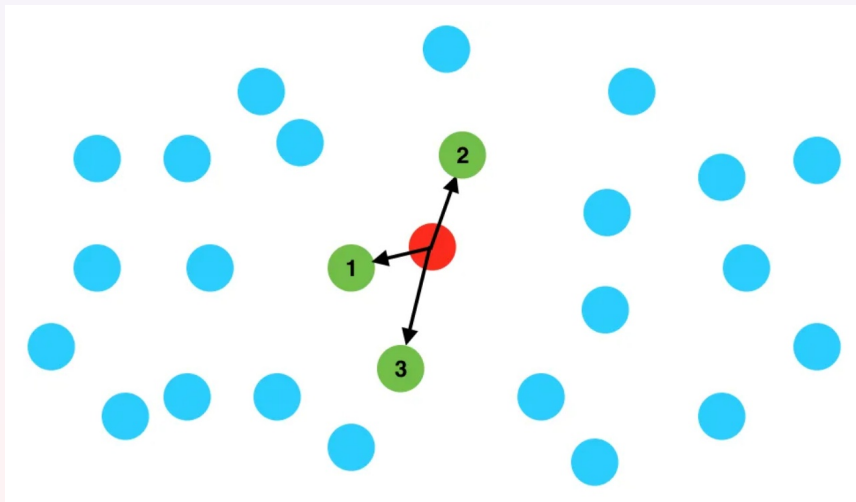
Indexace & GIS:

Zrychlení operací s prostorovými daty, prostorové dotazy a analýzy.

Web GIS, rastrové / vektorové dlaždice, rychlé načítání dat.

Jakou dlaždici načítst v aktuálním zvětšení?

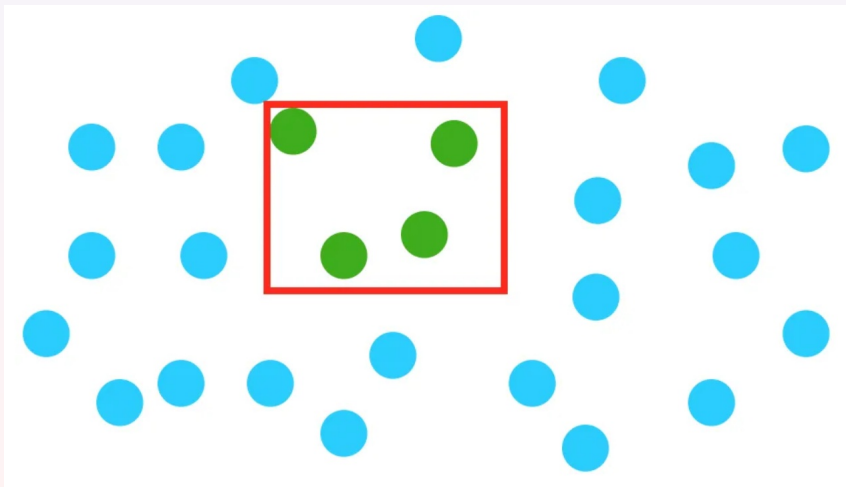
3. KNN Search



KNN search výpočetně drahý.

ANN (Approximate NN) search: body v dlaždicích stejným prostorovým indexem.

4. Interval Search



5. Metody prostorové indexace

Prostorová indexace založeny na transformačních metodách.

Dvě základní strategie:

Mapování do prostoru s nižší dimenzí:

tzv. Geo-Hashing.

Zpravidla konverze na jednodimenzionální problém, výsledkem 1D prostorový index.

Využívá Space Ordering (seřazení dle hodnoty hashe).

Indexovány buňky v 2D/3D gridu.

+ Extrémně rychlá, $O(c)$.

- Problematické definice sousedství.

- Vznik kolizí.

Zástupci: Snake Curve, Spiral Curve, Hilbert Curve, Z-Curve.

Dynamické dělení prostoru:

tzv. Space Partitioning.

Prostor opakovaně dělen na menší části stejné dimenze (+- pravidelně).

Části se mohou / nemusejí překrývat.

Snaha minimalizovat jejich překryt.

+ Rychlost.

+ Částečné zachování sousedství.

Zástupci: Grid, Quad-Tree, k-D Tree, R-Tree, Hierarchical Clustering.

6. Geo-Hashing

Transformace úlohy na jednodimenzionální indexaci, vhodné pro bodové prvky.

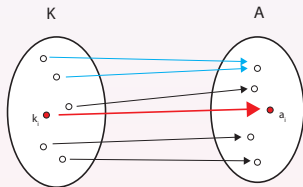
Hashovací funkce (jednosměrná)

$$h : K \rightarrow A, \quad h(k) = a, \quad \|A\| \ll \|K\|, \quad k \in K, a \in A.$$

Prostor klíčů K zobrazen do prostoru adres A .

Klíč k posloupnost bitů libovolné délky, adresa a posloupnost bitů pevné délky.

Pro $\forall a \in A$ snadný výpočet $a = h(k)$, klíčem poloha bodu $p = [x, y, z]$



Protože $\|A\| \ll \|K\|$, dochází ke kolizím

$$h(k_1) = h(k_2), \quad k_1 \neq k_2.$$

Blízké objekty mají podobnou adresu, lze je uspořádat dle polohy.

Využití *Space Filling Curves* pro výpočet hashovací funkce:

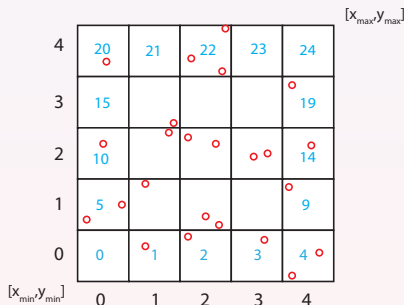
- Snake Curve.
- Spiral Curve.
- Hilbert Curve.
- Z-Curve.

7. Pokrytí prostoru gridem

Prostor lze pokrýt 2D / 3D gridem.

Počet buněk (buckets) n_b funkcí velikosti datasetu n a dimenze k

$$n_b = n^{1/k}.$$



Řádkový a sloupcový index bodu $p = [x, y]$

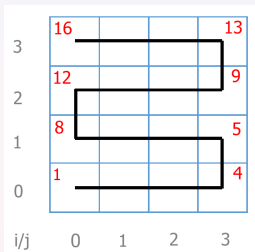
$$i = (\text{int}) \frac{y - y_{\min}}{y_{\max} - y_{\min}}, \quad j = (\text{int}) \frac{x - x_{\min}}{x_{\max} - x_{\min}}.$$

Blízké objekty leží ve stejné buňce.

8. Snake Curve

Jednoduchá vyplňující křivka.

Každá buňka gridu získá 1D index.



Postup po řádcích resp. sloupcích, dle x resp. y souřadnic vzestupně.

$$p.addr = \begin{cases} i \cdot n + j + 1, & i \text{ liché,} \\ (i + 1)n - j, & i \text{ sudé.} \end{cases}$$

Objekty v jedné buňce \rightarrow stejný index.

Alternativně lze použít i bitovou reprezentaci.

Získání adresy v $O(c)$.

+ Jednoduchá implementace.

- Blízké objekty nemusejí mít podobnou adresu.

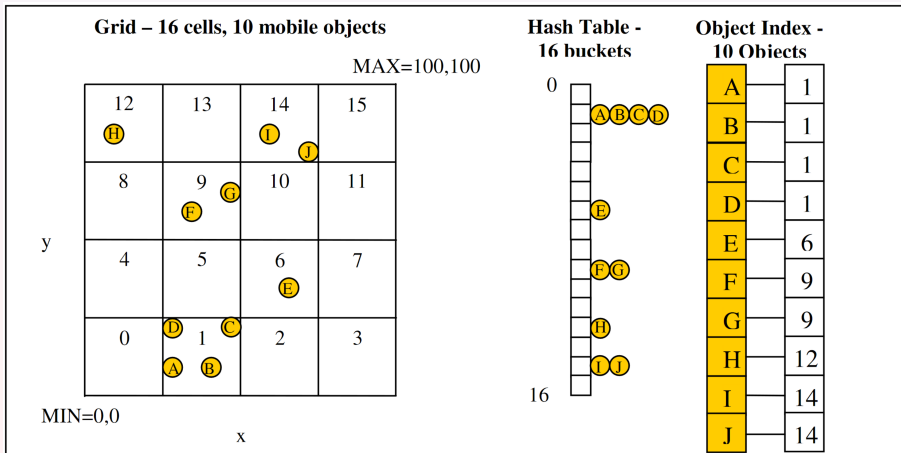
9. Geo-hashing, Snake Curve

ID buňky použito jako hash, 2 seznamy:

- Seznam bodů v buňce

{ID : [points]} #Využití Dictionary, konstantní čas

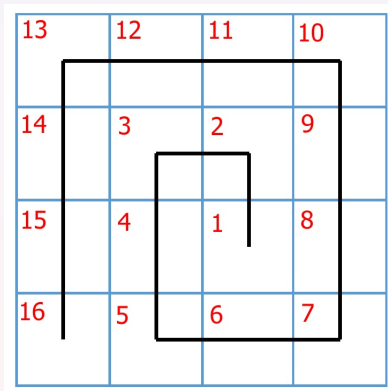
- Příslušnost bodu vzhledem k buňce.
Implementace polem.



10. Spiral Curve

Jednoduchá vyplňující křivka.

Buňky uspořádány ve směru od středu k okrajům do spirály.



Preference vnitřních objektů, časté v počítačové grafice/geometrii.

Získání adresy v $O(c)$.

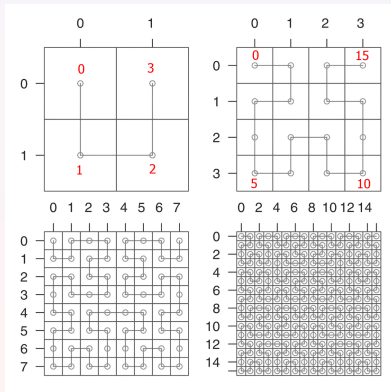
+ Jednoduchá implementace.

- Blízké objekty nemusejí mít podobnou adresu.

11. Hilbert Curve

Rekurzivně definovaná křivka vyplňující prostor.

Patří mezi 1D fraktály.



Posloupnost U-shaped segmentů.

+ Lépe zachovává vzdálenost.

- Složitější implementace (rekurzivní).

12. Indexace s využitím Hilbert Curve

Pořadová čísla buněk možno reprezentovat binárně.

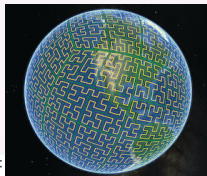
Level 1 (0-3): 00, 01, 10, 11.

Level 2: (0-15): 0000, 0001, 0010, ..., 1111.

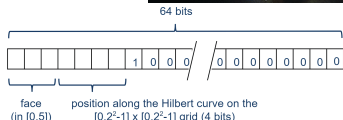
Dvojice binárních čísel představují pozice buněk v Quad-Tree.

00	11
01	10

0000	0001	1110	1111
0011	0010	1101	1100
0100	0111	1000	1011
0101	0110	1001	1010



S2 Cell ID of a **level-2** cell:



Prostorový index S2 (64 bitů):

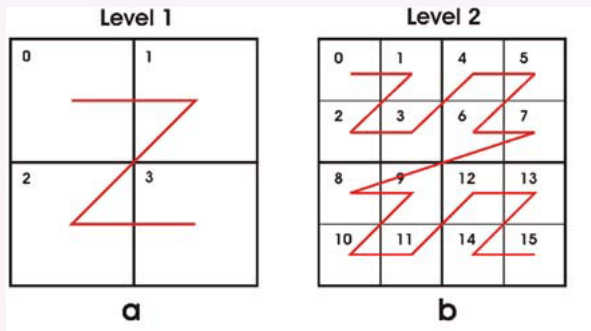
- Face ID: identifikátor na zemském povrchu.
- Pořadové číslo v Hilbertově křivce (odpovídá Level).

Indexace s využitím HC využívá např. Google.

13. Z-Curve

Rekurzivně definovaná křivka vyplňující prostor, 1D fraktál.

Často používána v geoinformatice (MySQL 7+).



Posloupnost Z-shaped segmentů.

+ Zachovává vzdálenost, avšak hůře než HC.

- Jednodušší implementace než HC.

14. Indexace s využitím Z-Curve

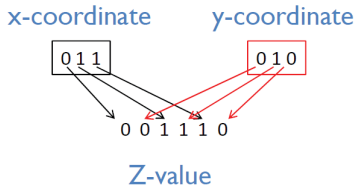
Pro indexaci využívány souřadnice buněk ve směru x, y zapsané binárně:

Level 1: 0, 1.

Level 2: 00, 01, 10, 11.

1	01	11
0	00	10
	0	1

11	0100	0111	1101	1111
10	0101	0110	1100	1110
01	0001	0011	1001	1011
00	0000	0010	1000	1010
	00	01	10	11



Prostorový index:

Pozice buňky získána prolínáním bitů (Bit Interleaving).

Střídají se bity ve směru x a y

$(0, 0) \rightarrow (00, 00) \rightarrow 0000 = 0$

$(1, 1) \rightarrow (01, 01) \rightarrow 0011 = 3$

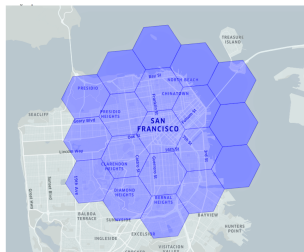
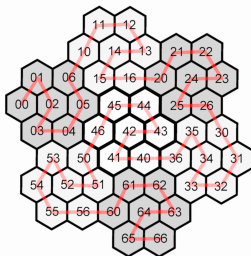
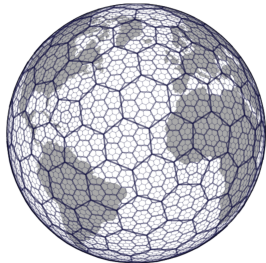
$(2, 3) \rightarrow (10, 11) \rightarrow 1101 = 13$

15. Hexagonal Indexation

Prostorový index, zemský povrch rozdelen do sférických šestiúhelníků.

Indexace realizována prostřednictvím Peano-Gosper Curve (Space Filling Curve).

Využito v H3 Spatial index.



Princip indexace:

Prováděna po skupinách tvořených 7 šestiúhelníky.

Buňky ($n = 67$) číslovány koeficienty $k_i = [0, 66]$.

Indexační funkce

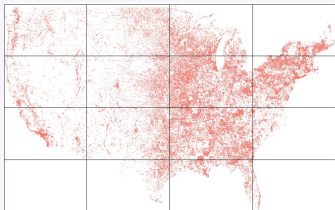
$$idx = k_1 2^{3(n-1)} + k_2 2^{3(n-2)} + \dots + k_{n-1} 2^{3(1)} + k_n 2^{3(0)}.$$

Prostorový index:

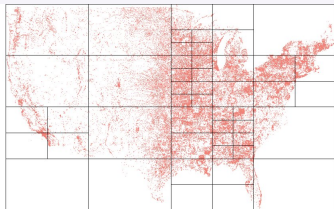
Číslo hexagonální dlaždice v rámci světa + poloha s využitím idx.

Reprezentace 64-bitovým integerem.

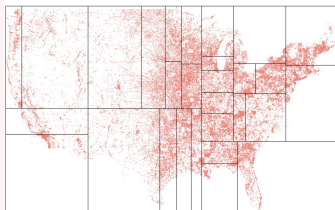
16. Přehled prostorových indexačních struktur



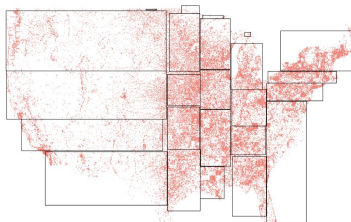
Uniform grids



Quad-Tree



KD-Tree



R-Tree

17. Quad-Tree

Datová reprezentující rekurzivní dělení prostoru do kvadrantů.

Každý kvadrant obsahuje nejvýše zadaný počet objektů.

Strom stupně 4, každý uzel má právě 4 potomky.

Kvadranty čtvercové nebo obdélníkové.

Lze zobecnit do prostoru dimenze 3 \Rightarrow Oct-Tree (8 potomků).

Space / Data driven decomposition:

- Regional Quad-Tree: pravidelné dělení (častější).
- Point Quad-Tree: dělicí hranice závisí na vstupních bodech (v mediánu).

+ Adaptabilní dělení prostoru (zohledňuje hustotu).

+ Uchovává sousednost.

+ Lze použít pro libovolné objekty (nejen body).

+ Snadná implementace.

Nutnost předzpracování dat (vytvoření Quad-Tree, $O(n \log n)$).

Paměťové nároky $O(n)$.

- Méně efektivní než ostatní stromy.

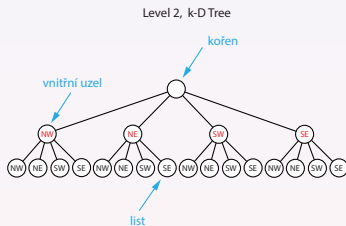
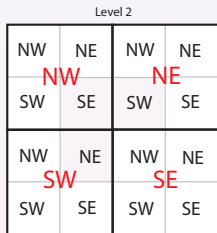
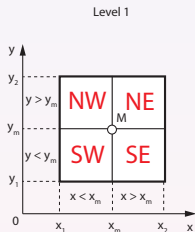
- Nevhodný pro data s proměnlivou prostorovou hustotou.

- Nejsou vyvážené.

18. Princip Quad-Tree

Opakované dělení prostoru do 4 kvadrantů: NW, NE, SW, SE.

Kvadrant kartézský součinem dvojice intervalů $[x_1, x_2] \times [y_1, y_2]$.



Souřadnice středu kvadrantu

$$x_m = (x_1 + x_2)/2, y_m = (y_1 + y_2)/2$$

Přidávaný bod $p = [x, y]$.

Hledáme kvadrant σ , ve kterém leží.

$$p \in \begin{cases} NE & x > x_m, y > y_m, \\ NW & x \leq x_m, y > y_m, \\ SW & x \leq x_m, y \leq y_m, \\ SE & x > x_m, y \leq y_m. \end{cases}$$

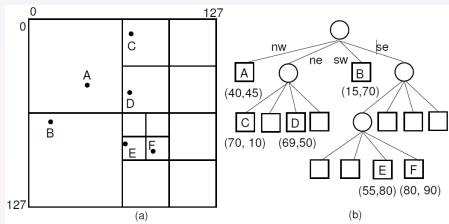
Vnitřní uzly pouze pro orientaci, data uchovávána v listech.

19. Quad-Tree, reprezentace

Dělení do kvadrantů prováděno opakovaně.

Podmínka: kvadrant obsahuje alespoň nějaký bod.

Výsledkem adaptivní dělení prostoru.



Stromová reprezentace, 2 typy uzlů:

- interní (4 potomci),
- list: představuje kvadrant (prázdný/obsahuje body).

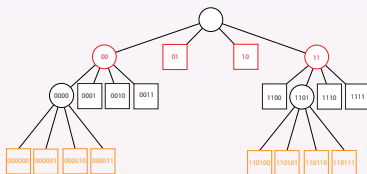
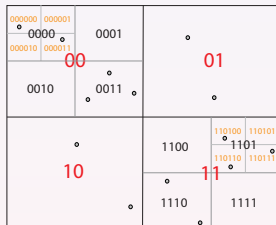
Jednoduchá implementace v Pythonu:

```
class QTreeNode():
    def __init__(self, xm, ym, w, h, points):
        self.xm = xm
        self.ym = ym
        self.width = w
        self.height = h
        self.points = points
        self.children = []
```


20. Quad-Tree, prostorová indexace

Dvě základní metody indexace:

- Využití Quad-Tree indexu (analogie s HC).
- Využití Space Filling Curves (viz Z-Curve, efektivnější).



Quad-Tree index:

Pořadová čísla kvadrantů (identifikátory) možno reprezentovat binárně.

Level 1 (0-3): 00, 01, 10, 11.

Level 2 (0-3): 00, 01, 10, 11.

Index kvadrantu: kombinace identifikátorů kvadrantů z úrovně Level 1 až Level k.

Level 1 + Level 2: 0000-0011, 0100-0111, 1000-1011, 1100-1111.

- Pro vysoké stromy Quad-Tree indexy dlouhé.

Využití: databázové systémy Oracle, MySQL.

21. k-D Tree

Nejčastěji používaná struktura pro prostorovou indexaci.

Dělení střídavě probíhá v k směrech.

Reprezentace stromem stupně k , každý uzel má právě k potomků.

2D varianta: střídá se dělení ve směru x, y , strom stupně 2.

Dělicí nadrovina prochází mediánem množiny.

Výsledkem obdélníkové oblasti, každá obsahuje nejvýše zadaný počet objektů.

Konkrétní tvar závislý na vstupním datasetu.

Efektivnější dělení než Quad-Tree.

Nutnost předzpracování dat (vytvoření k-D Tree, $O(n \log n)$).

Paměťové nároky $O(n)$.

- + Nejpoužívanější prostorová indexační struktura.
- + Schopnost adaptace na proměnnou prostorovou hustotu množin.
- + Vysoká efektivita zejména pro malá k .
- Výpočetně složitější konstrukce.
- Nejsou vyvážené.

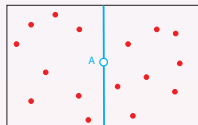
22. Princip k-D Tree

Modifikace Binary Search Trees, podpora multidimenzionálních klíčů.

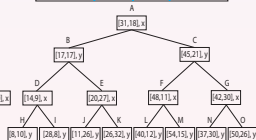
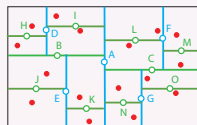
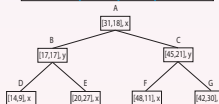
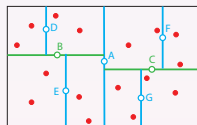
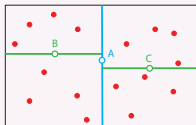
Vnitřní uzly pouze pro orientaci, data uchovávána v listech.

Discriminator δ : Pomocný klíč, ukládá informaci o tom, ve kterém směru dělení probíhá

$$\delta = i \% k, \quad i = 0, \dots, k - 1.$$



A
[31,188, x]



Dvoudimenzionální strom ($k = 2, \delta = 0, 1$):

Oblast σ k-D Tree obdélníková, dělena na levou část σ_l a pravou část σ_r .

Určen medián M souřadnic všech bodů σ .

Dělicí nadrovina prochází M , pro $\delta = 0$ resp. $\delta = 1$ rovnoběžná s osou y resp. s x .

Body v σ_l resp. v σ_r vlevo resp. vpravo od M vzhledem k x resp. y .

Třidimenzionální strom ($k = 3, \delta = 0, 1, 2$):

3D varianta stromu důležitá při práci s bodovými mračky.

23. Implementace k-D Tree

Implementace uzlu v programovacím jazyce Python:

```
class kDNode:
    def __init__(self, discr, med, points>):
        self.discr = discr
        self.med = med
        self.left = None
        self.right = None
        self.points = points
```

Konstrukce k-D Tree nad P (2D):

Max. počet bodů n_{max} v σ , $k = 2$.

- 1] Pokud $\|P\| < n_{max}$, ukonči proceduru.
- 2] Urči discriminator $\delta = i \% k$.
- 3] Spočti medián

$$\delta = 0 : x_m = \text{med}(x_i), \quad \delta = 1 : y_m = \text{med}(y_i).$$

- 4] Rozděl P na P_l a P_r , pro každý $p = [x, y]$

$$P_l = \{p, \delta = 0 \wedge x < x_m \vee \delta = 1 \wedge x < y_m\}, \quad P_r = \{p, \delta = 0 \wedge x > x_m \vee \delta = 1 \wedge x > y_m\}.$$

Podmnožina P_l levým podstromem P , podmnožina P_r pravým podstromem P .

- 5] Inkrementuj $i = i + 1$.
- 6] Opakuj rekurzivně 1-6 nad P_l a P_r .

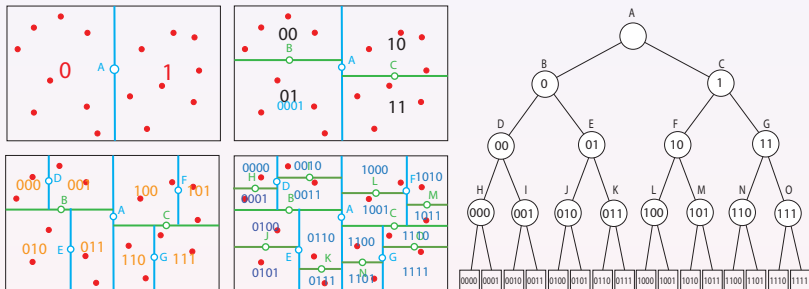
Hledání oblasti σ obsahující p :

Procházení k-D Tree od kořenu k listů.

Dle hodnot x resp. y přecházíme do levého resp. pravého podstromu.

24. k-D Tree, prostorová indexace

Podobný princip jako u Quad-Tree, využívá binárního indexu



k-D Tree index:

Identifikátory kvadrantů v binárním tvaru:

Level 1: 0, 1.

Level 2: 00, 01, 10, 11.

Index kvadrantu: kombinace identifikátorů kvadrantů z úrovně Level 1 až Level k.

Level 1 + Level 2: 00, 01, 10, 11.

Level 1 + Level 2 + Level 3: 000, 001, 010, 011, 100, 101, 110, 111.

Velmi používaný prostorový index.

- Pro vysoké stromy k-D-Tree indexy dlouhé.

25. R-Tree

Nejčastěji používaná struktura pro indexaci prostorových dat.

Jiná koncepce, oblasti σ nejsou disjunktní (mohou se vzájemně překrývat, avšak jen málo).

Objekty, které jsou vzájemně blízké, jsou seskupeny do uzlů.

Lze použít pro libovolné entity, generalizace do vyšších dimenzí.

Objekty uchovávány v listech, vnitřní uzly pro orientaci.

Jsou vyvážené (na rozdíl od předchozích), nižší výška stromu.

Výška nezávisí na vstupních datech.

Vnitřní uzly reprezentovány min-max boxy.

Několik praktických problémů:

- Vnitřní uzly se by měly co nejméně překrývat.
- Neměly by obsahovat příliš volného místa.
- Blízké objekty by měly ležet v blízkých vnitřních uzlech.

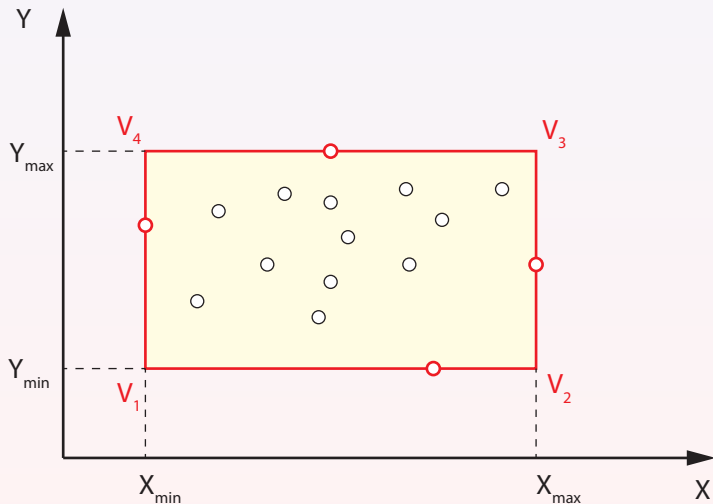
+ Výborné výkonnostní parametry: NN Search.

- Horší parametry: Interval Search.

- Objekt se může nacházet ve více uzlech (překrývají se).

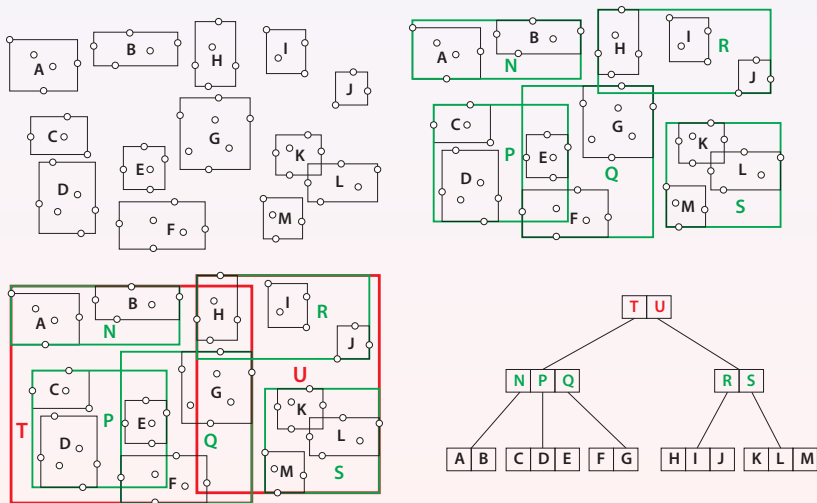
- Obtížnější implementace.

26. Min-max box



27. Princip konstrukce R-Tree

Hierarchická dekompozice scény zdola nahoru.



28. Vlastnosti R-Tree

Délkově vyvážený strom.

Uzel má nejvýše k potomků (počet proměnlivý).

Interní uzly obsahují min-max boxy.

Seznam objektů (0D-3D) uchováván v listech.

Každý uzel obsahuje $[m, M]$ prvků, M maximální počet objektů uvnitř min-max boxu

$$m \leq \frac{M}{2}.$$

Výška R-Tree s n záznamy (nezávisí na struktuře vstupních dat)

$$h \leq \log_m n.$$

Implementace třídy R-Tree Node:

```
class RNode:
    def __init__(self, left, right, children = None, data = None):
        self.low_left = rect
        self.high_right = rect
        self.children = children
        self.data = data
```

Efektivní struktura pro **Range Search**.

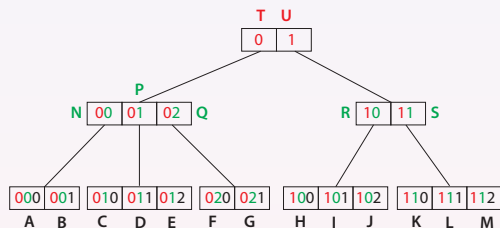
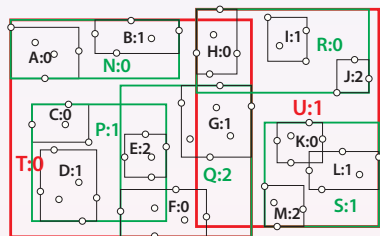
Vyhledání všech objektů min-max boxů kolidujících se vstupním obdélníkem.

R-Tree procházen od kořene, hledáme kolidující min-max boxy.

V každém listu vybereme objekty uvnitř obdélníku, výsledek může ležet v různých podstromech.

29. Prostorová indexace s R-Tree

Využívá celočíselného indexu min-max boxů



R-Tree index:

Identifikátory min-max boxů v celočíselném tvaru:

Level 1: 0, 1, ..., k_1 .

Level 2: 0, 1, ..., k_2 .

Index kvadrantu: kombinace identifikátorů min-max boxů od úrovně 1:

Level 1 + Level 2: 0.0, 0.1, 0.2, 1.0, 1.1.

Level 1 + Level 2 + Level 3: 0.0.0, 0.0.1, 0.1.0, 0.1.1, 0.1.2, 0.2.0, ...

Velmi používaný prostorový index (PostGIS, Oracle).

Efektivní pro prostorové operace.

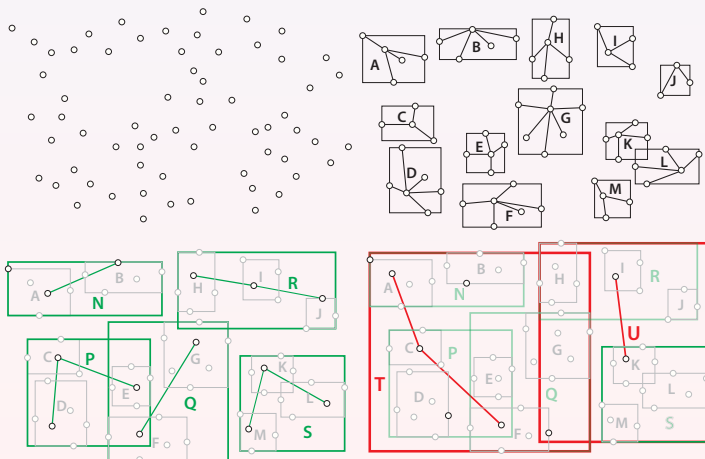
30. Hierarchická dekompozice vstupních dat

Specializované metoda, využívající pokročilé heuristiky.

Minimalizace překrytů a hluchých míst.

Netriviální implementace.

Alternativně, hierarchická clusterizace: opakovaná clusterizace datasetu.

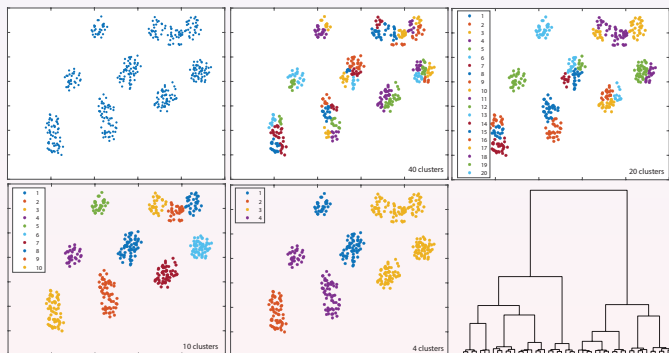


31. Hierarchická clusterizace

Shlukování blízkých objektů na základě geometrických parametrů do clusterů.

Každý cluster má centroid.

Clusterizace dle různých metrik / pseudometrik, nejčastěji používána $L2$ norma.



Hierarchická clusterizace:

Opakování postupu v několika iteracích.

V následujícím kroku clusterizovány centroidy shluků.

Hierarchie clusterizace znázorněna dendrogramem.

+ Efektivní metoda, adaptivní vzhledem ke vstupním datům.

- Konstrukce clusterizace časově náročná.

32. Prostorová indexace, hierarchická clusterizace

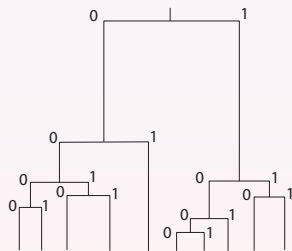
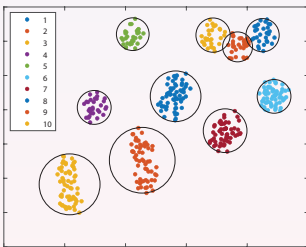
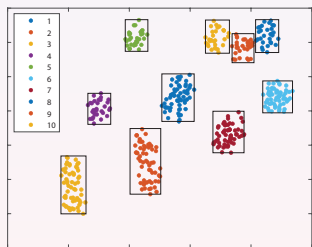
Dendrogram strom stupně 2 (binární).

Využívá binárního indexu.

Oblasti vymezené clusterly nekonvexní.

Aproximace min-max boxy nebo kružnicemi, částečně se překrývají.

Indexovaný bod padne do nejbližšího min-max boxu, kružnice.



Spatial index:

Identifikátory clusterů v binárním tvaru:

Level 1: 0, 1.

Level 2: 0, 1.

Index clusteru: kombinace identifikátorů kvadrantů z úrovně Level 1 až Level k.

Level 1 + Level 2: 0, 1

Level 2 + Level 2: 00, 01, 10, 11

Level 1 + Level 2 + Level 3: 000, 001, 010, 011, 100, 101, 110, 111.

Varianta s min-max boxy využívá R-Tree.

33. Porovnání metod prostorové indexace

Index Type	Data Structure	Best For	Efficiency	Supported Data Types	Main Advantages
R-tree	Hierarchical (Tree-based)	Datasets with varying object sizes, shapes, and dimensions	Good	Points, Lines, Polygons	Good for overlapping regions, adaptable, handles various data types
Quadtree	Hierarchical (Tree-based)	2D datasets with varying density, adaptability to Octree for 3D	Good	Points, Polygons	Space partitioning, efficient for sparse datasets
Geohash	Grid-based (string)	Approximate nearest-neighbor queries, simple grid indexing	Moderate	Points	Easy to implement, compact representation
KD-tree	Binary tree	Point data in multi-dimensional spaces	High	Points	Efficient for point data, handles high-dimensional data

34. Aplikace prostorové indexace

Bing Maps Tile System: aplikace Quad Tree.

