Metricky prostor

M = (D, d)
D – datova domena
d – funkce vzdalenosti/metricka funkce/metrika: D x D -> R
-funkce muze byt:
- -diskretni (vraci hodnoty pouze z male, predem definovane mnoziny)
- -spojita (kardinalita mnoziny je velmi velka az nekonecna)

Metricky prostor musi splnovat:
(a) nezaporna funkce vzdalenosti:

$$\forall x, y \in \mathcal{D}, d(x, y) \geq 0$$

(b) symetrie:

$$\forall x, y \in \mathcal{D}, d(x, y) = d(y, x)$$

(c) identita:

$$\forall x, y \in \mathcal{D}, x = y \Leftrightarrow d(x, y) = 0$$

(d) trojuhelnikova nerovnost:

$$\forall x, y, z \in \mathcal{D}, d(x, z) \leq d(x, y) + d(y, z)$$

Alternativni definice:
(p1) nezaporna funkce vzdalenosti:

$$\forall x, y \in \mathcal{D}, d(x, y) \geq 0$$

(p2) symetrie:

$$\forall x, y \in \mathcal{D}, d(x, y) = d(y, x)$$

(p3) reflexivita:

$$\forall x \in \mathcal{D}, d(x, x) = 0$$

(p4) kladnost (positiveness):

$$\forall x, y \in \mathcal{D}, x \neq y \Rightarrow d(x, y) > 0$$

(p5) trojuhelnikova nerovnost:

$$\forall x, y, z \in \mathcal{D}, d(x, z) \leq d(x, y) + d(y, z)$$

Pseudo-metrika
kdyz vlastnost (p4) neplati?

Quasi-metrika

kdyz vlastnost (p2) neplati?

Super-metrika
kdyz je na vlastnost (p5) kladen silnejsi pozadavek:

$$\forall x, y, z \in \mathcal{D} : d(x,z) \leq \max\{d(x,y), d(y,z)\}$$

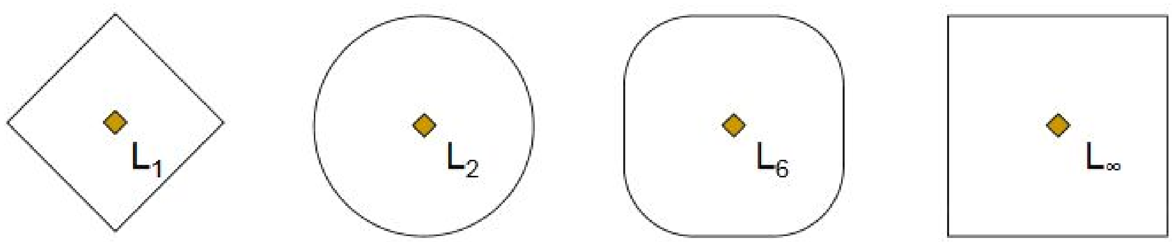
Minkowskeho vzorec/vzdalenosti/metriky

-tez nazyvane jako Lp metriky
-definovane na n dimenzionalnich vektorech:

$$L_p[(x_1, \cdots, x_n), (y_1, \cdots, y_n)] = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}$$

specialni pripady:
L1 – Manhattanska metrika (p=1)
L2 – Euklidovska metrika (p=2)
Lnekonecno – Maximalni (nekonecna) metrika:

$$L_\infty = \max_{i=1}^{n} |x_i - y_i|$$



na obrazku je znazornena mnozina bodu ktere maji v dane metrice stejnou vzdalenost od pivota

Jaccarduv koeficient

-index podobnosti
-mereni "vzdalenosti" mezi dvemi mnozinami:

$$d(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

"Tanimoto similarity" definovana na vektorech:

$$d_{TS}(\vec{x},\vec{y}) = 1 - \frac{\vec{x}\cdot\vec{y}}{||\vec{x}||^2 + ||\vec{y}||^2 - \vec{x}\cdot\vec{y}}$$

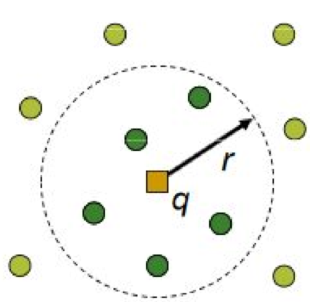
x.y je skalarni soucin
||x|| je velikost vektoru (eukleidovska norma)

Vyjmenovat 3 typy dotazu

-range query
-nearest neighbor query
-k-nearest neighbor query
-reveresed k-nearest neighbor query
-similarity join
-kombinovane dotazy (combined query)

Range query

"vsechny muzea az do vzdalenosti 2km od meho hotelu"
R(q,r) = { x€X | d(q,x) < r}
q – pivot
r – radius



Nearest neighbor (NN) query

"nejblizsi muzeum od meho hotelu"

$$NN(q) = x$$
$$x \in X,\ \forall y \in X,\ d(q,x) \leq d(q,y)$$

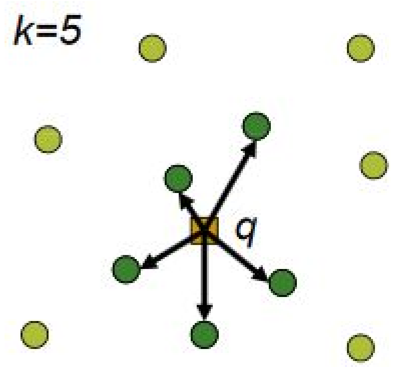
x ma mensi vzdalenost k pivotovi nez kdokoli jiny

k-NN query

"pet nejblizsich muzei od meho hotelu"

$$k\text{-}NN(q,k) = A$$
$$A \subseteq X,\ |A| = k$$
$$\forall x \in A,\ y \in X - A,\ d(q,x) \leq d(q,y)$$

*vysledek je **A**, coz je podmnozina domeny **X** o velikosti **k**, pricemz kazdy prvek z **A** ma mensi vzdalenost k pivotovi **q** nez kazdy prvek z doplnku*



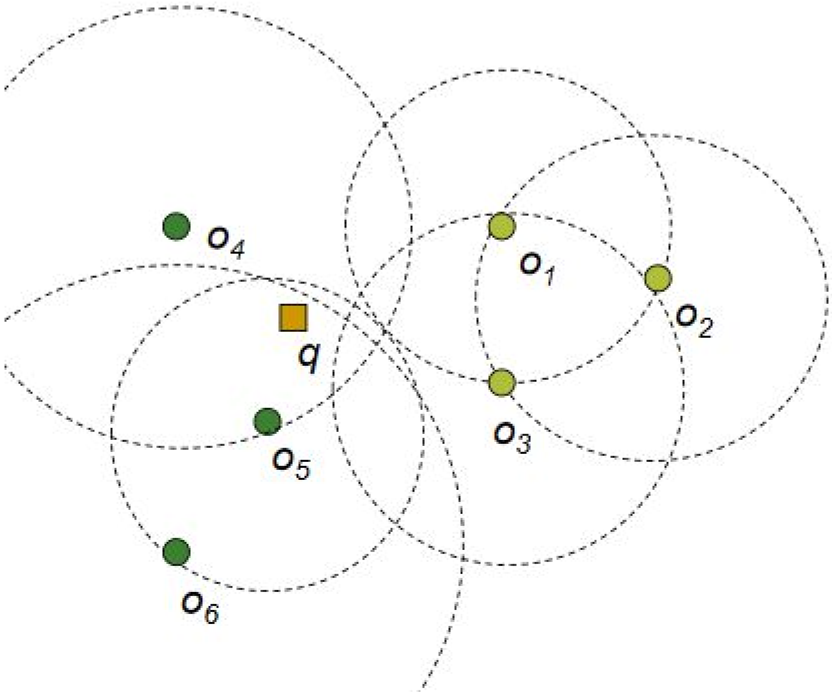
Reversed k-NN query

"vsechny hotely, ktere maji dane muzeum mezi svymi K nejblizsimi muzei"

$$kRNN(q) = \{R \subseteq X, \forall x \in R : q \in kNN(x) \wedge \forall x \in X - R : q \notin kNN(x)\}$$

Kazdy prvek **x** *ktery vybereme, musi mit pivota **q** mezi svymi k-nejlbizsimi sousedy a do vysledku zaradime kazde takove **x**. Tj. pokud jej nevybereme tak jen proto ze pivot **q** mezi k-nejblizsimi sousedy prvku **x** neni.=> **VYSLEDEK NEMUSI BYT VELIKOSTI k!!!***

priklad pro 2-RNN



objekty: o4, o5 a o6, maji pivota q mezi svymy dvema nejblizsimi sousedy => vysledek nema velikost k.

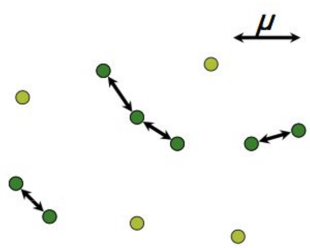
Similarity join

"pary hotelu a muzeji ktere jsou od sebe 5 minut pesky"

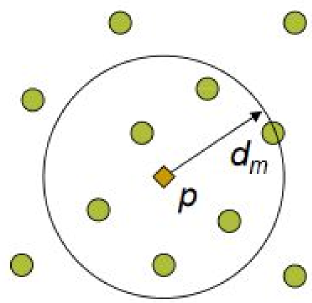
$$X \subseteq \mathcal{D}, Y \subseteq \mathcal{D}, \mu \geq 0$$

$$J(X, Y, \mu) = \{(x, y) \in X \times Y : d(x, y) \leq \mu\}$$

obecne definovane na dvou ruznych mnozinach (X, Y, muzea, hotely..)
pokud X=Y pak se jedna o Similarity Self-join
u je vzdalenost kterou od sebe prvky paru mohou maximalne mit



Ball partitioning
-rozdeluje datovou mnozinu na dve casti podle pivota (p) a radiusu (dm)



Vnitrni mnozina (inner set)
{x€X | d(p,x) <= dm} *vsechny prvky uvnitr a na hranici koule*

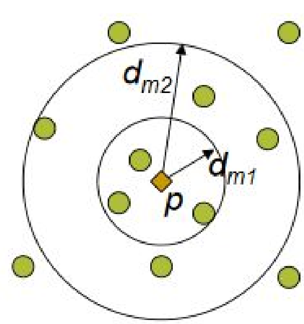
Vnejsi mnozina (outer set)
{x€X | d(p,x) > dm} *vsechny prvky mimo kouli*

Multi-way ball partitioning
-rozsireni ball partitioningu ktery domenu rozdeluje pomoci dvou radiusu (dm1, dm2) na 3 mnoziny:

Vnitrni mnozina (inner set)
{x€X | d(p,x) <= dm1} *vsechny prvky lezici uvnitr*

Vnejsi mnozina (outer set)
{x€X | d(p,x) > dm2 } *vsechny prvky lezici vne*

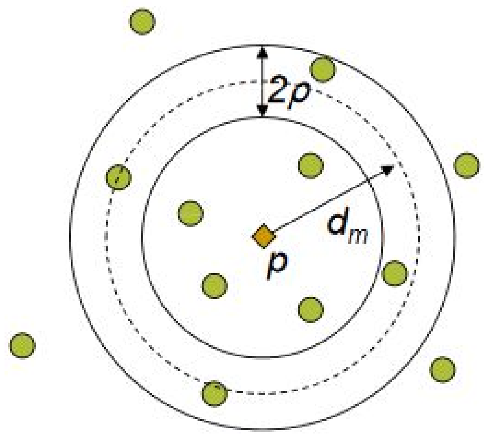
Prostredni mnozina (middle set)
{x€X | d(p,x) > dm1 & d(p,x) <= dm2} *vsechny prvky ve "strednim pasmu"*



prvky na hranici se berou ze patri dovnitr koule, proto je rovnost u vnitrni a prostredni mnoziny

Excluded middle partitioning
-rozsireni bezneho ball partitioningu

motivace:
kvuli objektum ktere lezi blizko hranice muzeme casto byt nuceni pristupovat k obema mnozinam (inner i outer set). Proto datovou mnozinu rozdelime na 3 casti, pricemz minimalne do jedne z nich nebude nutne pristupovat – objekty na hranici zahrneme do teto treti mnoziny.



Vnitrni mnozina (inner set)
{x€X | d(p,x) <= dm – p}

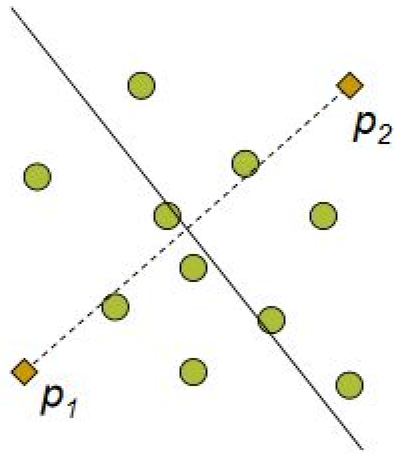
Vnejsi mnozina (outer set)
{x€X | d(p,x) > dm + p}

Vynechana mnozina (excluded set)
vsechny ostatni prvky

Generalized hyperplane partitioning
-rozdelovani datove mnoziny pomoci nadrovin
-zvolime dva pivoty a rozdelime datovou mnozinu na dve, podle toho ke kteremu pivotu ma ktery prvek blize.

Jedna mnozina:
{x€X | d(p1,x) <= d(p2,x) } *(rovnost - musime si zvolit kam padnou objekty na hranici)*

Druha mnozina:
{x€X | d(p1,x) > d(p2,x) }



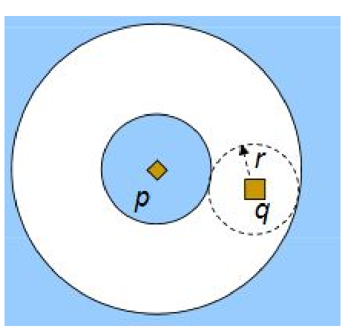
Pivot filtering

-strategie pouzivana k redukci poctu kontrolovanych prvku → rychlost ← to je dostatecna motivace ne?
-pouziva trojuhelnikovou nerovnost pro “odstrel” kontrolovanych prvku
-predpokladame ze mame strukturu ve ktere jsou spocitane vzdalenosti vsech prvku od pivota p

1)priklad filtrovani s jednim pivotem
p – pivot
q,r – range query
o – kontrolovany prvek

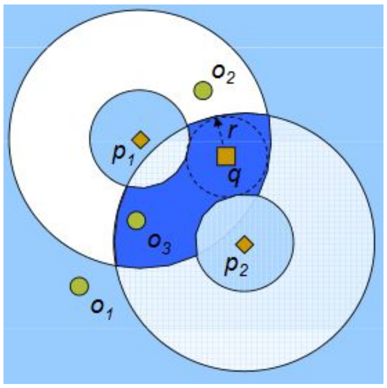
Algoritmus
odstrel prvek **o** pokud neco z tohoto plati:

d(p,o) < d(p,q) – r *(vnitrni modry kruh)*
d(p,o) > d(p,q) + r *(vnejsi modra oblast)*

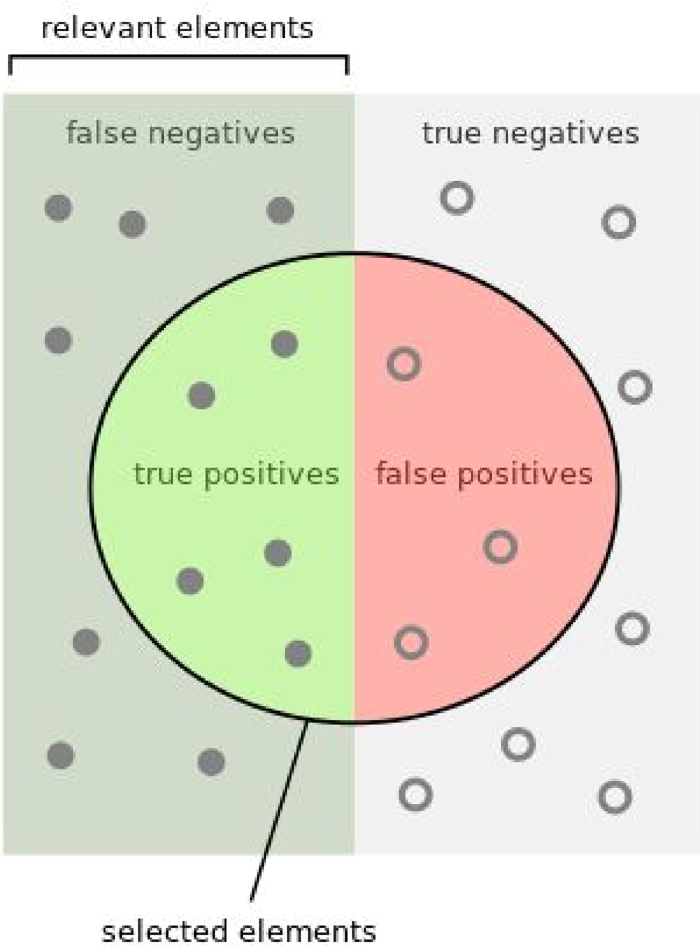


prvky v modre oblasti se nemuzou kvalifikovat pro dotaz q,r a to jen diky znalosti jejich vzdalenosti od p

2)priklad filtrovani se dvema pivoty
-jen tmave modrou oblast ma cenu uvazovat.
-efektivita se zvysuje s pouzitim vice pivotu
(oblast kterou je potreba kontrolovat se zmensuje – delame prunik)



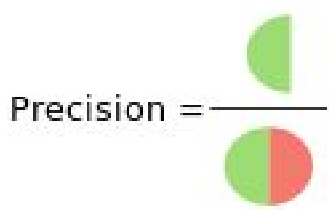
Precision, Recall



Precision (presnost)
pomer mezi poctem vybranych relevantnich prvku k poctu vsech vybranych.

Jinak receno
jaka cast vybranych prvku je relevantni? (jsou spravnou odpovedi na dotaz)

jeste jinak receno:

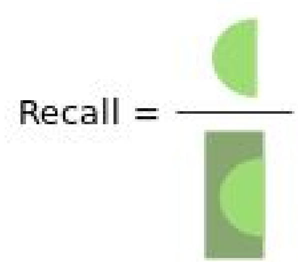


rika nam to chybu jakou jsme udelali.
1 → vsechny prvky, ktere jsme vybrali jsme vybrali opravnene
0.9 → nektere prvky jsme vybrali spatne

Recall (senzitivita)
pomer mezi poctem vybranych relevantnich prvku ku vsem RELEVANTNIM prvkum

jinak receno:
jakou cast z relevantnich prvku jsme vybrali

jeste jinak receno:



rika nam to jak velkou cast spravne odpovedi na dotaz jsme vybrali.
1 → vsechny prvky ktere jsou spravne jsme vybrali
0.7 → nektere prvky jsme nevybrali prestoze jsou spravne

DUSLEDEK
Muzeme mit velkou **precision** (klidne 1) ale pokud vracime v odpovedi na dotaz jen malo prvku, i kdyz jich tam je relevantnich hodne (mame nizky **recall**) tak vysledek nestoji za nic.

Stejne tak kdyz mame kvalitni **recall**(klidne 1), coz znamena ze zadny z relevantnich prvku ve vysledku nevynechame, ale zase mame spatnou **precision** (vracime skoro vsechny prvky – tzn I spoustu spatnych), tak opet vysledek nestoji za nic.

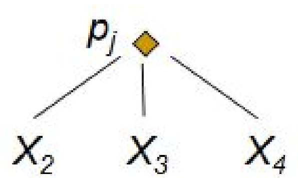
Burkhard-Keller Tree, popsat, nakreslit priklad

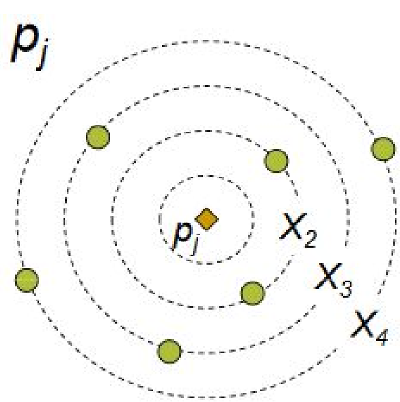
2/5

-pouzitelny pouze při pouziti diskretni distance funkce
-rekurzivne deli datovou mnozinu
-zacneme s vyberem libovolneho prvku pj z datove mnoziny a sestrojime mnoziny:

$$X_i = \{o \in X \mid d(o,p_j) = i\}$$

-tzn ze například mnozina X2 obsahuje vsechny objekty které mají od pj vzdalenost 2
-z prvku pj udelame koren stromu, z kazdeho Xi se stane podstrom. Prazdne mnoziny Xi vynechame (v pripade ze zadny objekt nemá od pj vzdalenost i).



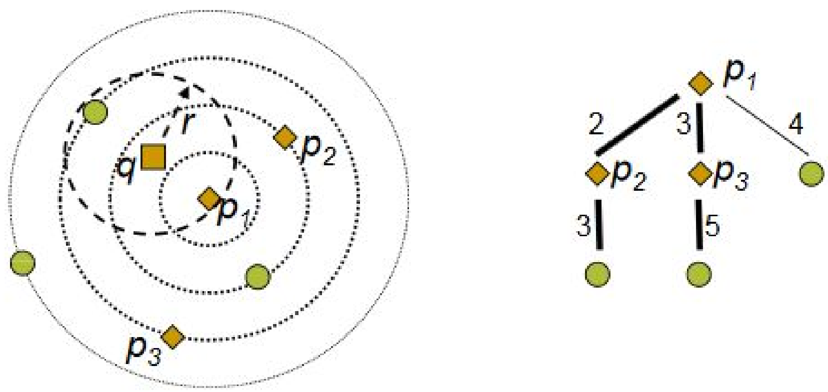


Range query R(q,r)

-zacneme od korene
-v kazdem vnitrnim uzlu pj delame:

-zarad pj do vysledku když: if $d(q, p_j) \le r$

-vejdi do potomka pi když: if $\max\{d(q,p_i) - r, 0\} \le i \le d(q,p_i) + r$



zlute prvky jsou klasifikovane, tluste cary znazornuji uzly které jsme navstivili

Vantage point tree

-Používá ball partitioning
-rekurzivně dělí datovou množinu
-vyvazeny binarni strom

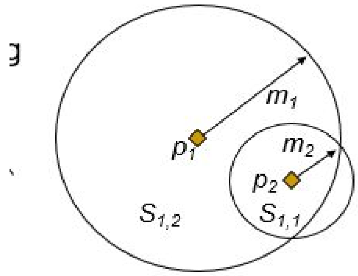
Algoritmus

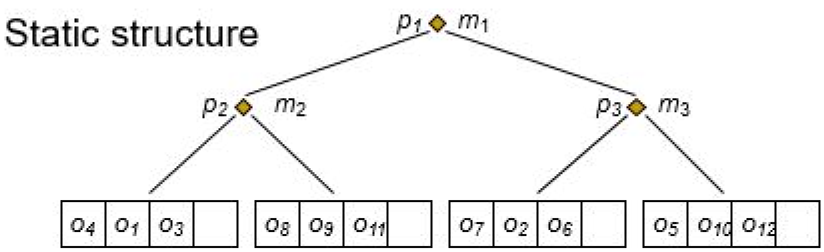
1. Je potřeba vybrat vztyčný bod (vantage point) a poté určit median**(median ze vzdaleností vzhledem k vantage pointu)**
2. Rozdělíme datovou mnozinu na dve podmnoziny podle mediánu

$$S_1 = \{x€X – \{p\} \mid d(x,p) \leq m\}$$

$$S_2 = \{x€X – \{p\} \mid d(x,p) \geq m\}$$

-Rovnitko je pouzito v obou množinách a proto je strom vyvážený
-vantage point (p) je ulozen vzdy v nelistovem uzlu
-Jeden nebo vice objektů jsou umístěny v listech





Range query R(q,r)

-(**uvazujeme pripad ze v kazdem listu je jen jeden objekt**)

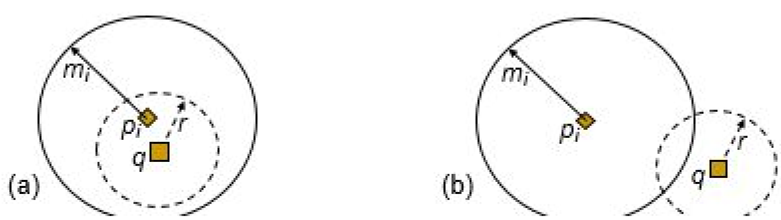
-prochazime strom od korene

-V každém vnitřním uzlu je potřeba udělat:

if $d(q,p_i) \le r$ zaradime pi do vysledku

if $d(q,p_i) - r \le m_i$ prohledame levy podstrom – inner set (a, b)

if $d(q,p_i) + r \ge m_i$ prohledame pravy podstrom – outer set (b)



Jak lze videt v pripade (b), muze se stat ze bude nutne prohledat levy i pravy podstrom

Nearest neighbor NN(q)

-inicializujeme: $d_{NN} = d_{max}$ *NN=nil*

-(dnn=dosavadni nejmensi vzdalenost od q, NN=prvek s touto vzdalenosti)

-prochazime strom od kořene

-v každém interním uzlu (p_i, m_i), delame:

if $d(q,p_i) \le d_{NN}$	nastav $d_{NN} = d(q,p_i)$, $NN=p_i$
if $d(q,p_i) - d_{NN} \le m_i$	prohledame levy podstrom
if $d(q,p_i) + d_{NN} \ge m_i$	prohledame pravy podstrom

k-NN(q)

-jedina zmena je ze potrebujeme pole: $d_{NN}[k]$ a $NN[k]$

-dana pole udrzujeme setridana vzhledem k vzdalenosti od pivota q

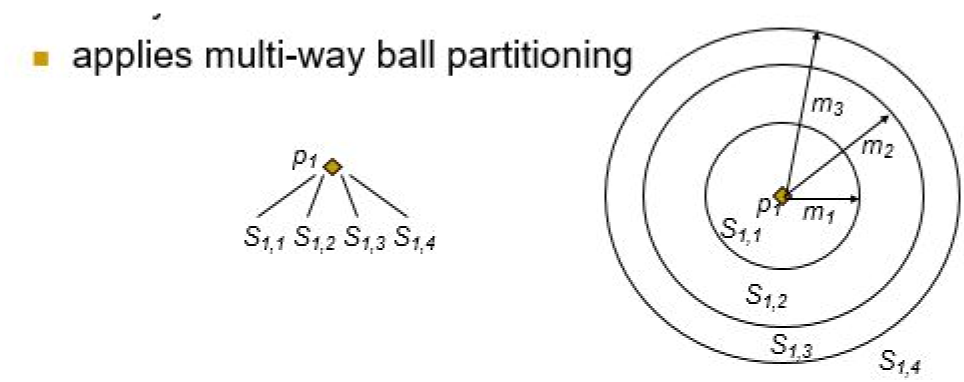
Multi Way VPT

-další typ VPT

-Veškeré principy dědí z klasického VPT

-jedná se o m-ární strom oproti klasickému VPT (ktery je binarni).

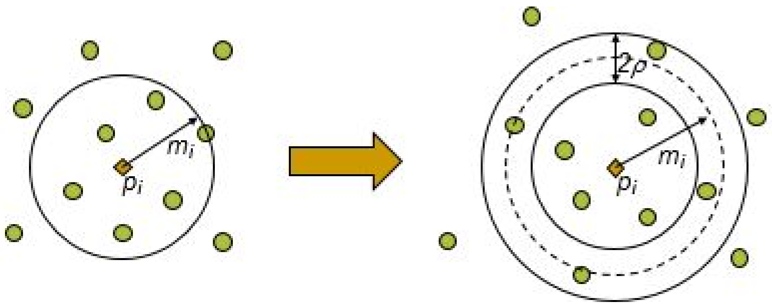
-misto bezneho ball paritioningu pouzivame Multi-way ball partitioning (nemame jen jeden radius, ale mame jich vice => rozdelujeme na vice mnozin, ne jen vnitrek a vnejsek)



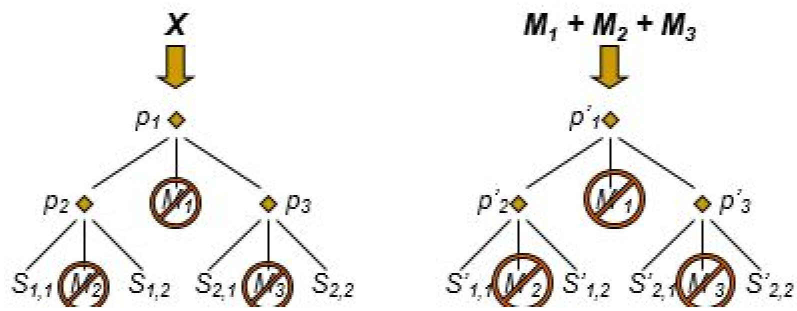
Vantage point forest (VPF)

-,,les binárních stromů”

-oproti beznemu VPT (ktery pouziva ball paritioning) pouziva VPF excluded middle partitioning



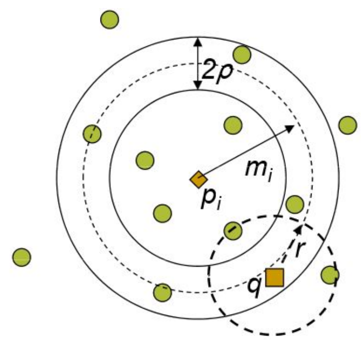
-excluded middle partitioning rozdeluje datovou mnozinu na 3 podmnoziny, pricemz ale my tu vytvarime binarni strom…Jak? Tak ze proste vytvarime strom jen z dvou podmnozin: vnitrku (inner set) a vnejsku (outer set).
-treti podmnozinu (exluded set - prvky na hranici koule) si dame bokem a ze vsech techto vynechanych casti pak zkonstruujeme dalsi (opet binarni) VPT (rekurze).



Range query R(q,r)
-prochazime strom od kořene
-ró(ρ) udava polovicni sirku hranice excluded middle setu
-v kazdem vnitrnim uzlu (p_i,m_i), delame:

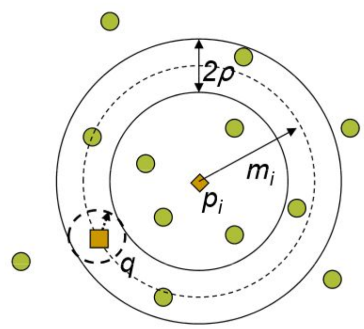
if $d(q,p_i) \leq r$	zaradime p_i do vysledku
if $d(q,p_i) - r \leq m_i - \rho$	prohledame levy podstrom
if $d(q,p_i) + r \geq m_i - \rho$	prohledame nasledujici strom
if $d(q,p_i) + r \geq m_i + \rho$	prohledame pravy podstrom
if $d(q,p_i) - r \leq m_i + \rho$	prohledame nasledujici strom
If d(q,pi) – r ≥ mi – ρ && d(q,pi) + r ≤ mi + ρ	prohledame POUZE nasledujici strom

PRIKLAD1



Dotaz R(q,r) zasahnul vsechny 3 mnoziny => prohledame oba podstromy (levy i pravy) a jeste k tomu jdeme dal hledat v nasledujicim stromu (protoze ten byl vyvoreny z exluded middle)

PRIKLAD2



Dotaz R(q,r) zasahl jen prostredni (excluded middle) mnozinu => levy ani pravy podstrom neprohledavame, jdeme rovnou hledat do nasledujiciho strom (protoze ten byl vyvoreny z exluded middle)

Bisector Tree

-zaklad pro GHT (generalized hyperplane tree)
-vyuzivame deleni pomoci nadrovin (viz generalized hyperplane partitioning)

Algoritmus
-zvolime dva pivoty $p_1, p_2 \in X$
-ty nam rozdeli datovou mnozinu na dve podmnoziny, podle toho, kteremu pivotu ma dany objekt bliz. Delici uzel (nelistovy) ma tedy tvar: <p1, p2>

$S_1 = \{o \in X, d(o,p_1) \leq d(o,p_2)\}$
$S_2 = \{o \in X, d(o,p_1) > d(o,p_2)\}$

-zvolime hodnoty r_1^c a r_2^c tak, aby:

Vsechny prvky z mnoziny S1 byly uvnitr koule se stredem v p1 a polomerem r_1^c

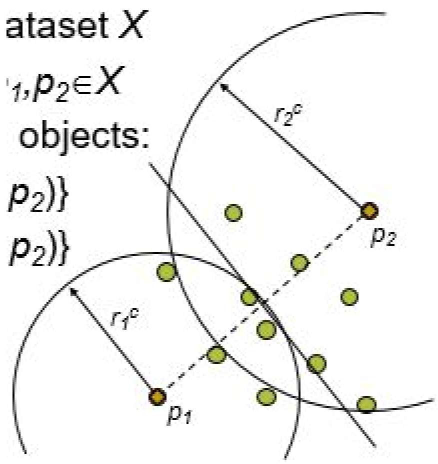
a

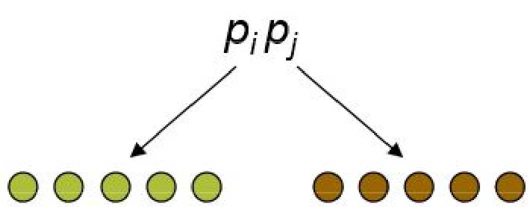
Vsechny prvky z mnoziny S2 byly uvnitr koule se stredem v p2 a polomerem r_2^c

-zvolenym hodnotam r_1^c a r_2^c se rika ‘covering radii’

-koule se mohou prekryvat (no problemo)

-rekurzivne opakujeme na mnozine S1 a S2





Range query R(q,r)

-prochazime strom od korene

-v kazdem internim uzlu <pi, pj> delame tohle (jak pro pi, tak pro pj):

Zaradime px do vysledku $\quad$ if d(q,px) <= r $\quad$ (px je primo v r-okoli q)

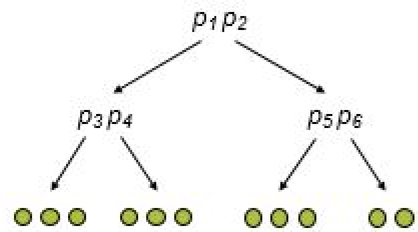
Vlezeme do podstromu px $\quad$ if d(q,px) - r <= r_x^c $\quad$ (range query koule zasahuje do covering radiusu px => musime prohledat)

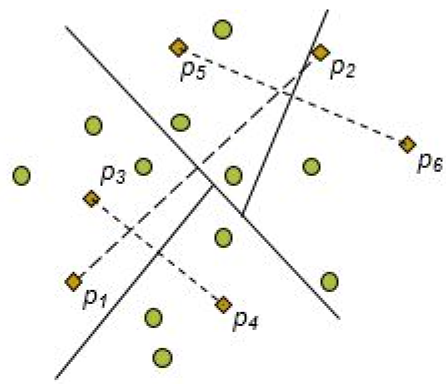
GHT (Generalized hyperplane tree)

-podobny jako bisector tree

-dělení se děje pomocí nadrovin

-je vybrana dvojce objektu a nasledne je rozdelena datova mnozina na dve podmnoziny podle toho ke kteremu objektu ma kdo bliz.

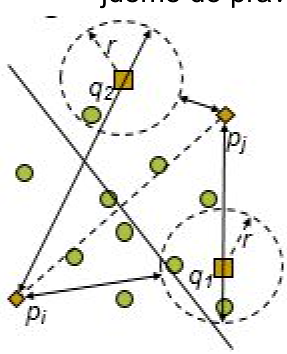




Range query R(q,r)
-prochazime strom od kořene

-V každém vnitřním uzlu $<p_i,p_j>$ dělame:

-zaradime px do vysledku	if $d(q,p_x) \leq r$	*(jsme v dosahu radiusu)*
-jdeme do leveho podstromu	if $d(q,p_i) - r \leq d(q,p_j) + r$	*(jsme blize pi)*
-jdeme do praveho podstromu	if $d(q,p_i) + r \geq d(q,p_j) - r$	*(jsme blize pj)*



M-tree vlastnosti, idea, naco je dobry, jake tam jsou typy uzlu, co je v nich ulozene

2/65
3/6
M-tree
- *převážně dynamická struktura*
- *Disk-oriented (fixní velikost   uzlů)*
- *Vytvořeno pomocí bottom-up architektury*
    - *Inspirováno R-trees a B-trees*
- *Veškerá data jsou uložená v listech*
- *Interní uzly: odkazy na podstromy a další informace*
- *Podobné jako GNAT, ale objekty jsou uloženy v listech.*

M-tree: Vnitřní uzly

- *Vnitřní uzel obsahuje přístup do každého podstromu*
- *Obsah vnitřního uzlu:*
    - *Pivot: p*
    - *Pokrývající poloměrv podstromu: r^c*
    - *Vzdálenost od p do parent pivota p^p: $d(p,p^p)$*
    - *Ukazatel na podstrom: ptr*

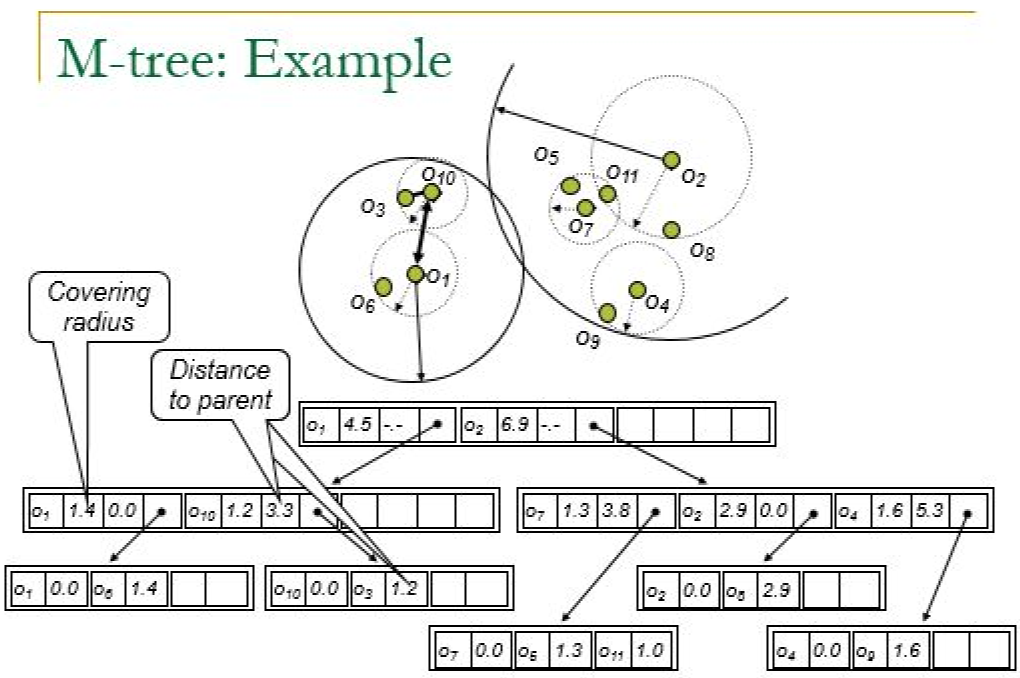
$$\boxed{\langle p_1, r_1^c, d(p_1, p^p), ptr_1\rangle}\boxed{\langle p_2, r_2^c, d(p_2, p^p), ptr_2\rangle}\cdots\boxed{\langle p_m, r_m^c, d(p_m, p^p), ptr_m\rangle}$$

- *Všechny objekty v podstromu ptr jsou v mezích vzdálenosti r^c z p.*

M-tree: Listové uzly

- *Obsahují **data***
- *Každá každý záznam je složený z dvojice:*
    - *object (jeho identifikátor): o*
    - *vzdálenost mezi o a jeho nadřazeným pivotem: $d(o,o^p)$*

$$\boxed{\langle o_1, d(o_1, o^p)\rangle}\boxed{\langle o_2, d(o_2, o^p)\rangle}\cdots\boxed{\langle o_m, d(o_m, o^p)\rangle}$$



M-tree: Vložení prvku

- *vložíme nový objekt o_N:*
- *rekurzivně sestupně projdeme strom a určíme nejlepší list pro uložení o_N*
- *v každém kroku vstoupíme do podstromu s pivotem p pro který:*
    - *není potřeba zvětšit poloměr r^c i.e., $d(o_N,p) \leq r^c$*
        - *v případě, že existuje vice listů, vybereme ten s nejbližším p v o_N*

- *minimaliujeme zvětšení r^c*
- *když dosáhneme listu N pak je třeba rozlišit tyto situace:*
    - *if N není plný then ulož o_N do N*
    - *else **Split(N,o_N).***

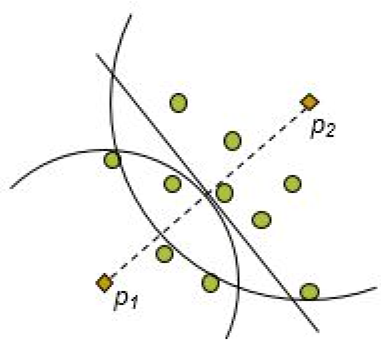
M-tree: Split

Split(N,o_N):

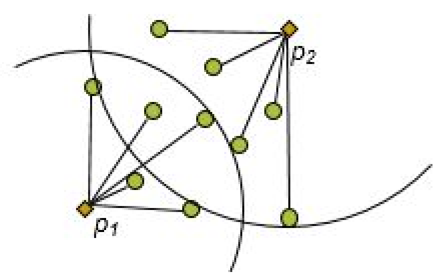
- *Nechť S je množina obsahující všechny prvky z N a o_N*
- *Vybereme dva pivoty p_1 a p_2 z S*
- *Partition S do S_1 a S_2 v závislosti na p_1 a p_2*
- *Ulož S_1 do N a S_2 do nově vytvořeného uzlu N'*
- *If N is root*
    - *Allocate a new root and store entries for p_1, p_2 there*
- *else (let N^p and p^p be the parent node and parent pivot of N)*
    - *Replace entry p^p with p_1*
    - *If N^p is full, then Split(N^p,p_2)*
    - *else store p_2 in node N^p*

M-tree: Split Policy

- Partition S to S_1 and S_2 according to p_1 and p_2
- Unbalanced
    - Generalized hyperplane



- Balanced
    - Larger covering radii
    - Worse than unbalanced one



Leaf node splitting a metody jakymi se to dela, rozdil mezi splitovanim v Slim a M-tree

3/36

Distance-index (D-index)

3/72

-Hybridní struktura pro velke databaze.
-Kombinace dvou veci: pivot filtering a partitioning