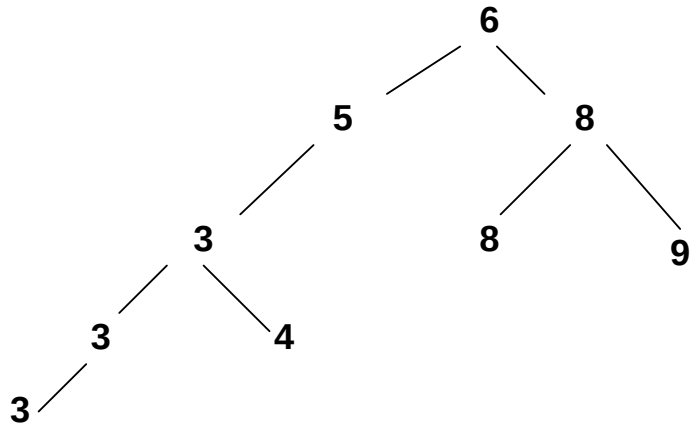
Modifikácie vyhľadávacích stromov

Doteraz známe vyhľadávacie stromy (binárny vyhľadávací strom, ...) je možné „upraviť“ tak, aby podporovali aj viacrozmerné intervalové vyhľadávanie.

A. Modifikácia (jednorozmerných) vyhľadávacích stromov pre prípad nejedinečných hodnôt kľúčov (sekundárnych kľúčov)

- **kľúče v ľavom podstrome ≤**
- **kľúče v pravom podstrome > (alebo symetricky opačne)**

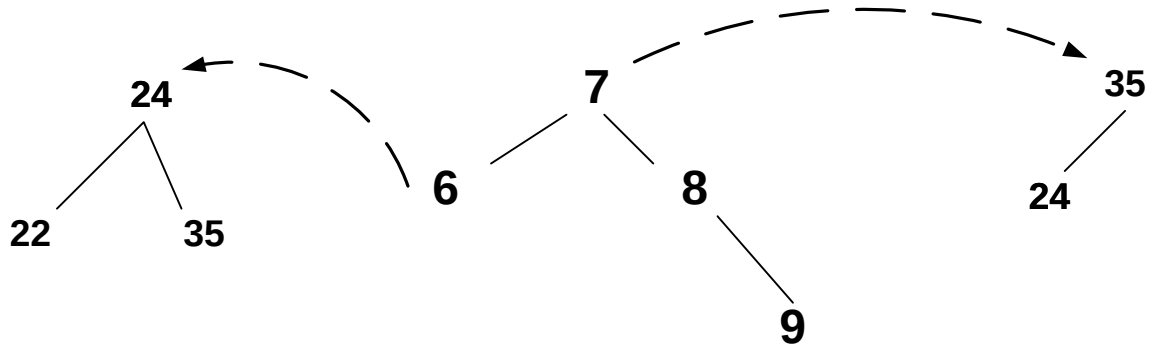


Takto modifikovaný strom umožňuje uchovávať aj sekundárne (nejedinečné kľúče). Je potrebné správne ošetriť operácie vlož (dovoliť duplicitu), vymaž (jednoznačne identifikovať mazaný prvok) a hľadaj (vypísať všetky prvky so sekundárnym kľúčom).

B. Modifikácia (jednorozmerných) vyhľadávacích stromov pre viacrozmerné vyhľadávanie - *viacúrovňové stromy* (stromy stromov).

- **majme množinu dvojíc sekundárnych kľúčov:**
  (7,35) (6,24) (7,24) (6,22) (6,35) (9,15) (8,22)

- **vytvoríme dvojúrovňový strom:**



Najskôr sa vybuduje vyhľadávací strom podľa prvého sekundárneho kľúča. Každý vrchol tohto stromu nesie odkaz na podstrom vybudovaný podľa ďalšieho sekundárneho kľúča. Počet sekundárnych kľúčov nie je obmedzený. Táto modifikácia umožňuje implementovať operáciu viacrozmerného intervalového vyhľadávania pomocou doteraz známych postupov. Toto vyhľadávanie neumožňuje ale efektívne hľadať iba podľa jediného sekundárneho kľúča (v našom príklade podľa súradnice y). Riešením je vybudovanie prístupu aj s iným poradím sekundárnych kľúčov.

- **intervalové vyhľadanie (pri rozumnom vyvážení): $O(m + \log_2 n)$ (m je počet kľúčov v intervale)**

k - d strom
(*k* - dimenzionálny strom)

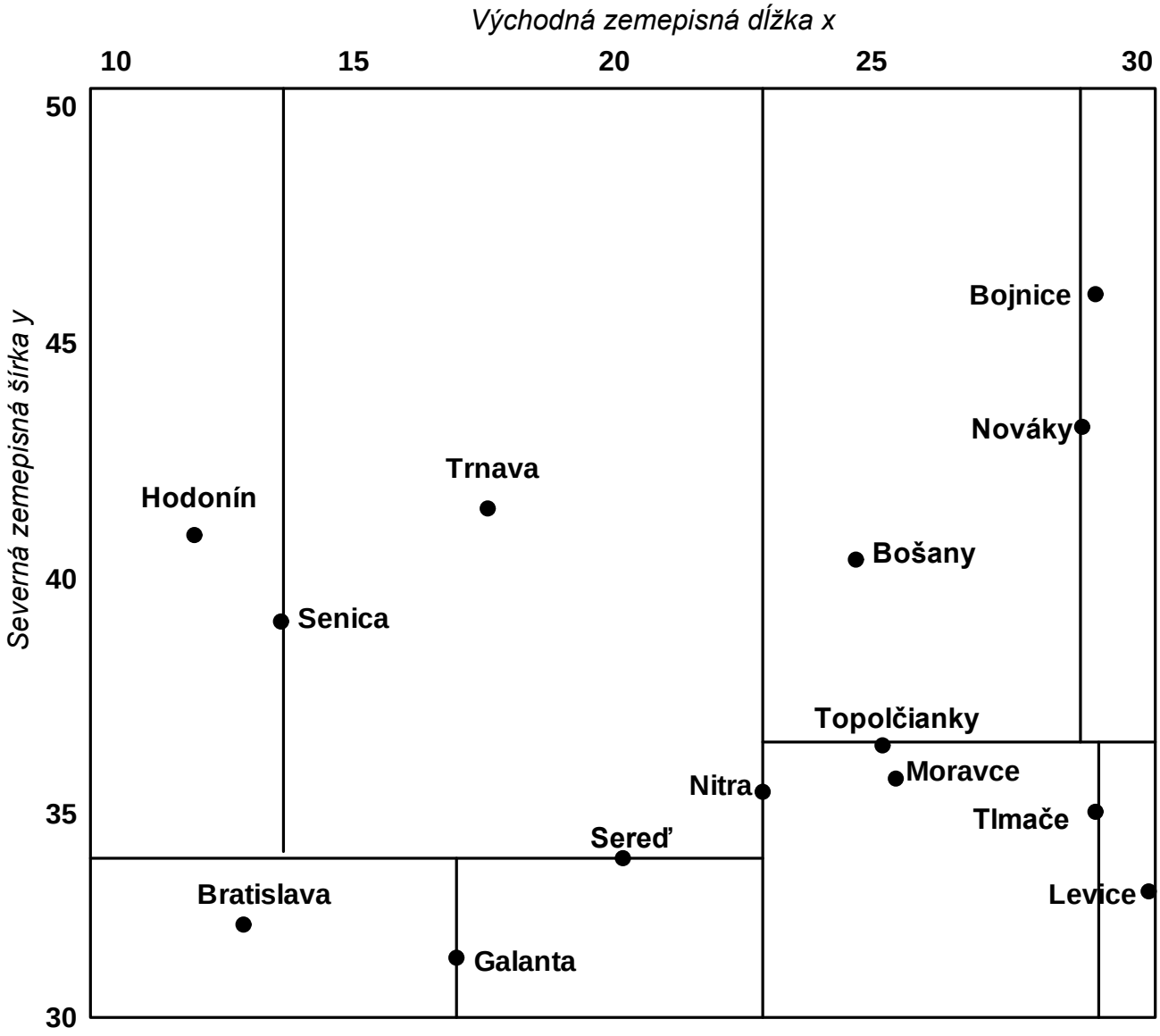
- ***k* – d strom je modifikáciou binárneho vyhľadávacieho stromu pre viacrozmerné intervalové vyhľadávanie**
- **je to teda implementácia tabuľky podporujúca *viacrozmerné intervalové* vyhľadávanie**
- **štruktúra je uložená v internej pamäti (v operačnej pamäti)**
- **vo vrchole je záznam obsahujúci *k* sekundárnych (nejedinečných) kľúčov K_0, ... K_{k-1} a dátovú časť**
- **pre koreň (vrchol úrovne *0*) platí, že v jeho ľavom podstrome sú záznamy s kľúčom K_0 menším alebo rovným ako hodnota K_0 v koreni a v pravom podstrome s kľúčom K_0 väčším**
- **vrcholy na úrovni 1 podobne rozdeľujú záznamy do ľavého a pravého podstromu ale podľa kľúča K_1**
- **vrcholy na úrovni *k* znovu rozdeľujú podľa K_0**
- **všeobecne vrcholy na úrovni *h* rozdeľujú podľa kľúča K_h mod *k***
- **kľúč môže rovnako ako v ostatných stromových štruktúrach predstavovať akýkoľvek dátový typ, je len potrebné aby bolo umožnené porovnanie kľúčov**
- **štruktúra nie je vhodná pre dáta, ktoré sa majú nachádzať v súbore**

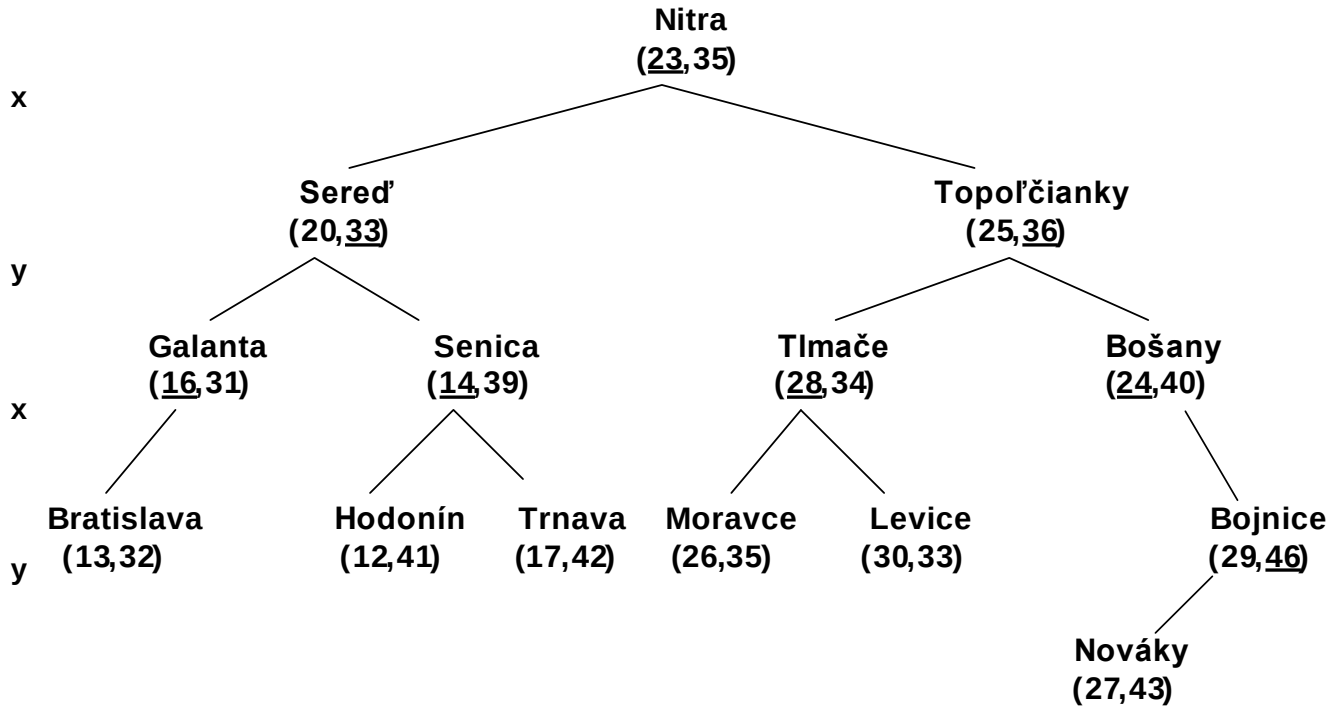
Príklad *2* - d stromu pre nasledujúce dáta:

Mesto	x	y
Bojnice	29	46
Bošany	24	40
Bratislava	13	32
Galanta	16	31
Hodonín	12	41
Levice	30	33
Moravce	26	35
Nitra	23	35
Nováky	27	43
Senica	14	39
Sereď	20	33
Tlmače	28	34

Súradnice a polohy sú fiktívne

Topoľčianky	**25**	**36**
Trnava	**17**	**42**





- **rozdelenie podľa súradnice x : zvislé čiary,**
- **rozdelenie podľa súradnice y : vodorovné čiary**

Find (vyhľadanie) :
Bodové i intervalové vyhľadávanie analogické bodovému a intervalovému vyhľadávaniu na jednorozmernom binárnom vyhľadávacom strome.

Príklad: $12 \leq x \leq 26, 35 \leq y \leq 48$
Poradie prehliadnutých (pri prehliadke inorder): Hodonín, Senica, Trnava, Sereď, Nitra, Moravce, Tlmače, Topoľčianky, Bošany, Nováky, Bojnice

Zložitosť vyhľadávania môže byť horšia ako $O(\log_2 n)$ i pri vyváženom strome.

Insert (vkladanie):

a) **Jednotlivé (bez znalosti množiny kľúčov) - nie je známa verzia, ktorá by udržovala strom vyvážený. Výsledná podoba stromu bude teda závisieť od poradia vkladaných prvkov (rovnako ak v binárnom vyhľadávacom strome). Je tu teda vysoká**

pravdepodobnosť degenerácie stromu. Delenie priestoru sa robí na súradniciach vloženého vrcholu.

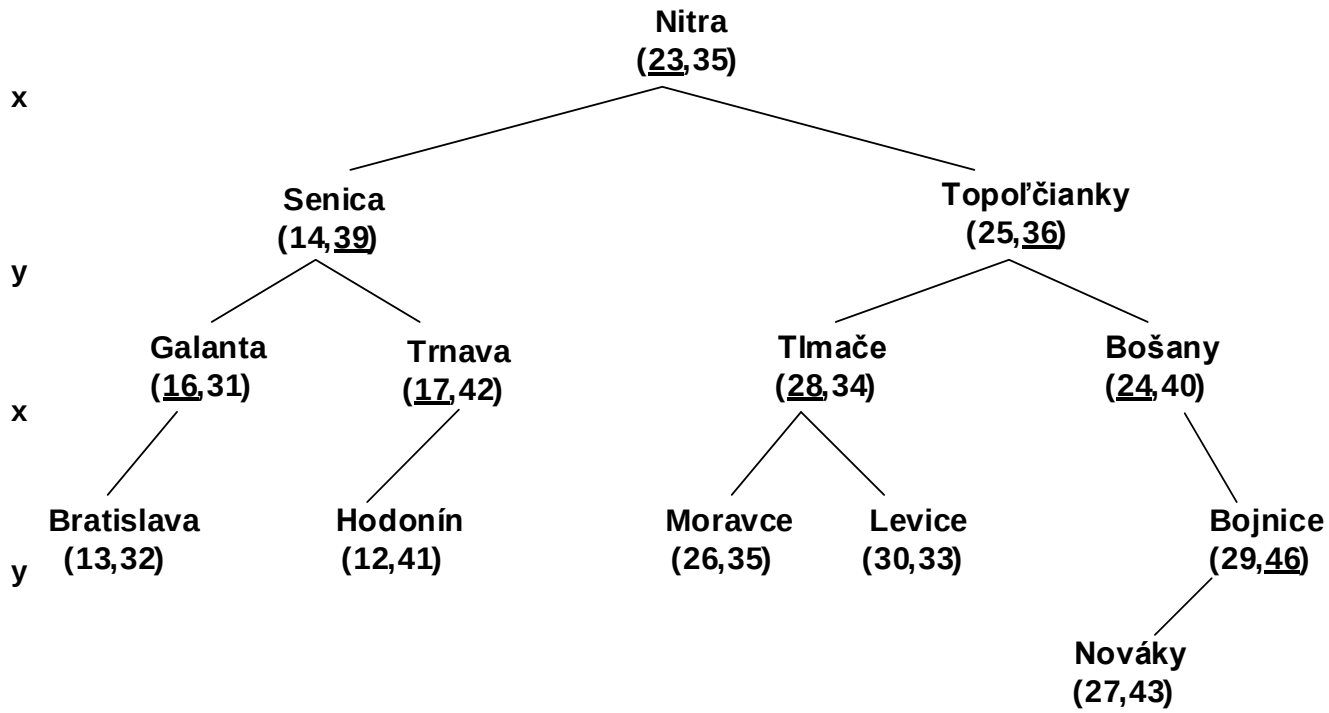
b) Máme vopred množinu kľúčov: možno rýchlo skonštruovať úplne vyvážený strom – zložitosť $O(n\log_2 n)$.

- **rezy robíme v záznamoch, ktorých kľúče sú mediánmi (najprv podľa K_0, potom podľa K_1 atď.)**

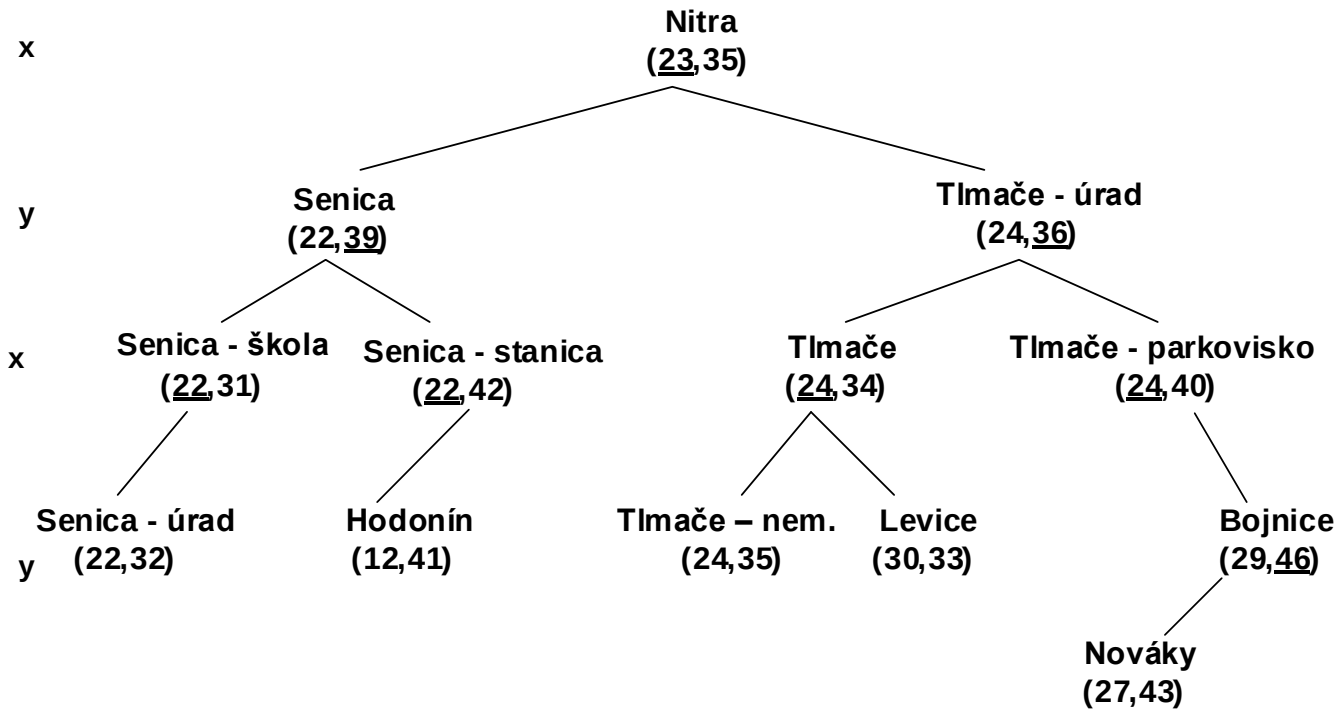
<u>Delete (mazanie):</u>

- **Nájdi záznam**
- **Ak je vrchol listom, tak sa odoberie.**
- **Ak nie je vrchol listom, nahraď ho najväčším v ľavom alebo najmenším v pravom podstrome podľa príslušného kľúča K_i, ktorý je použitý na tejto úrovni stromu. Ten však môže (na rozdiel od binárneho vyhľadávacieho stromu) mať aj dvoch synov – musíme pokračovať cyklicky ďalej.**

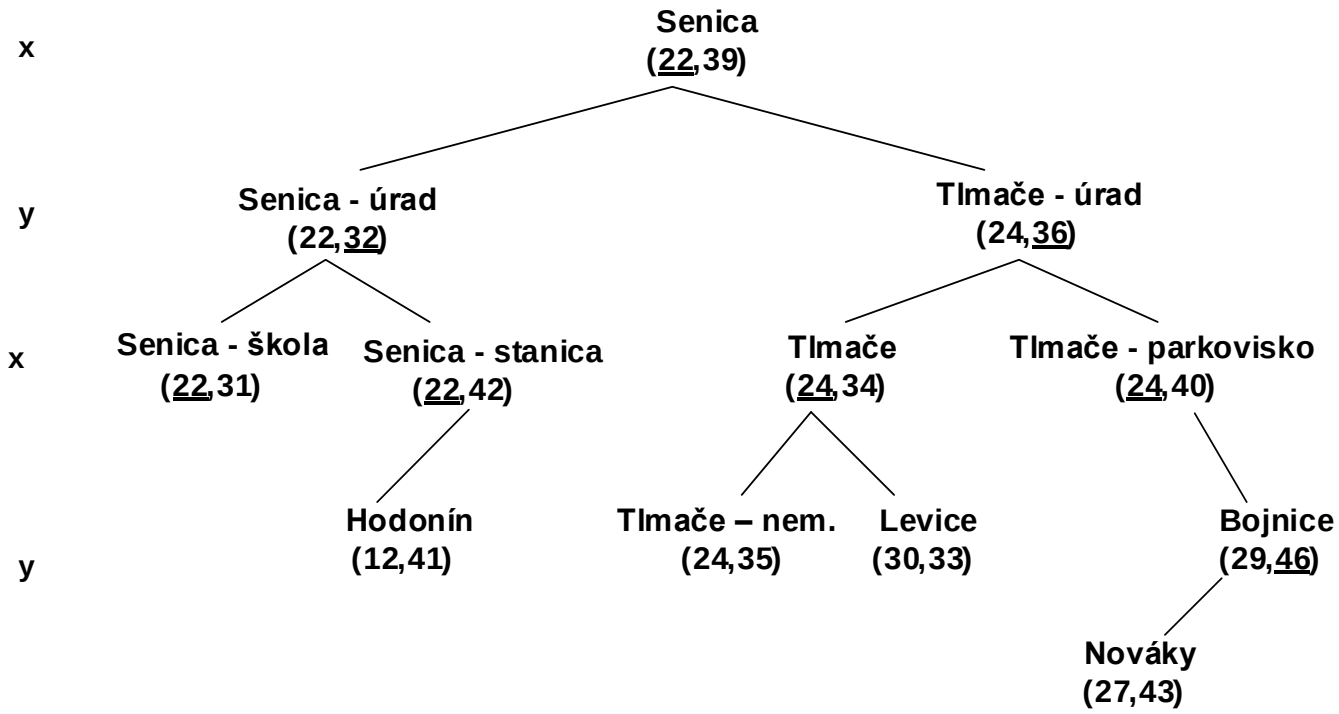
Príklad mazania Serede: V úrovni, kde sa nachádza Sereď sa vetví podľa kľúča y. V pravom podstrome teda prehľadáme všetky prvky tak, aby sme našli prvok s minimálnou súradnicou y. Našli sme Senicu. Sereď odstránime a nahradíme Senicou. Keďže Senica nebola listom stromu opakujeme hľadanie tento krát podľa kľúča x. Na uvoľnené miesto po Senici presunieme Trnavu. Keďže Trnava bola listom stromu, tak mazanie končí. Výsledok mazania je zobrazený na obrázku nižšie.



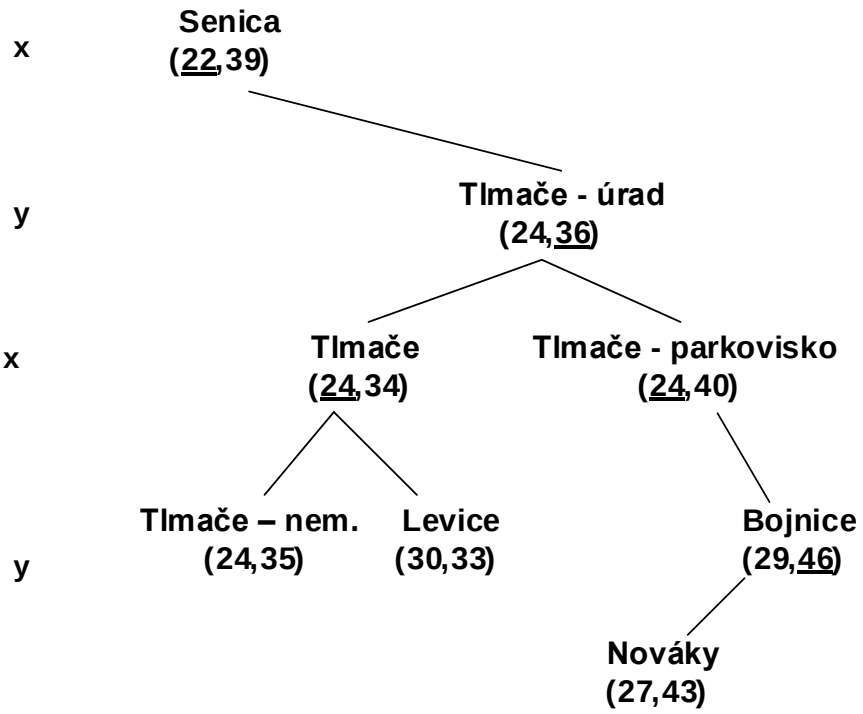
Ak sa v K-d strome vyskytujú neunikátne hodnoty kľúčov (čo je predpoklad toho, aby podporoval viacrozmerné vyhľadávanie podľa sekundárnych kľúčov), musíme pri implementácii mazania myslieť ešte na jeden prípad. Majme strom, kde rovnaké hodnoty kľúčov ukladáme vždy do ľavého podstromu. Postupným vkladaním kľúčov Nitra, Tlmače – úrad, Tlmače parkovisko, Senica, Senica – škola, Senica – úrad, Tlmače, Levice, Bojnice, Nováky, Senica – stanica, Hodonín, Tlmače - nem. do prázdneho stromu dostane strom uvedený na obrázku.



Ak chceme v tejto štruktúre vymazať Nitru, máme na výber dve možnosti. Buď z ľavého podstromu vyberieme najväčší prvok so súradnicou x (teda inorder predchodca podľa kľúča x) alebo z pravého podstromu vyberieme najmenší prvok so súradnicou X (teda inorder nasledovník podľa kľúča x). Ak sa rozhodneme použiť inorder predchodcu máme na výber 4 prvky (Senica, Senica – škola, Senica – úrad, Senica – stanica). Nech vyberiem ktorýkoľvek a nahradíme ním Nitru strom bude správne usporiadaný. Pri výbere jedného z prvkov Senica, Senica – škola a Senica – stanica musíme ešte tento prvok nahradiť ďalším. Ak by sme si napríklad zvolili, že Nitru nahradíme Senicou. Následne Senicu nahradíme prvkom Senica – úrad (inorder predchodca Senice podľa kľúča y) alebo prvkom Hodoním (inorder nasledovník Senice podľa kľúča y). Výsledný strom by mohol vyzerať nasledovne.



Ako môžeme vidieť, tu žiadny problém nevznikol, keďže sme vyberali z ľavého podstromu a v ľavom podstrome máme uložené menšie a rovnaké kľúče.
Nech postupne vymažeme celý ľavý podstrom koreňa, vznikne nám nasledujúca štruktúra.



Teraz ideme vymazať prvok Senica, ktorý má až štyroch inorder nasledovníkov podľa dimenzie x (Tlmače – úrad, Tlmače parkovisko, Tlmače, Tlmače - nem.). Ak by sme si napríklad zvolili ako náhradu Senice prvok Tlmače - nem., štruktúra nebude správna. V koreni vetvíme podľa kľúča x a teda pri vyhľadávaní prvkov so súradnicou x=24 by sme prvky Tlmače – úrad, Tlmače parkovisko a Tlmače už nikdy nenašli.

Na riešenie situácie použijeme nové pravidlo: Ak sa pri mazaní v strome presúva na nové miesto prvok, ktorého kľúč sa podľa dimenzie nového umiestnenia prvku v pravom podstrome nového umiestnenia prvku už nachádza, musíme následne všetky takéto prvky zo stromu tiež odobrať a opätovne vložiť do stromu. Toto pravidlo platí pre všetky presuny.

Ak sa teda rozhodneme Senicu nahradiť prvkom Tlmače - nem., skontroluje pravý podstrom koreňa (nové umiestnenie prvku Tlmače - nem.). V koreni vetvíme podľa kľuča x, to znamená, že v pravom podstrome nájdeme všetky prvky kde x=24 (Tlmače – úrad, Tlmače parkovisko, Tlmače). Tieto prvky postupne zo stromu tiež vymažeme a následne opätovne povkladáme.