Kopice in prioritetne vrste

Prioritetna vrsta:

- nova prazna vrsta
- vzemi prvega iz vrste
- ali je vrsta prazna?
- vstavi element x s prioriteto p v vrsto

Veljati mora: če ima x večjo prioriteto kot y, je prej na vrsti

Naivna implementacija:

Tabela $[(x_1, p_1), \ldots, (x_n, p_n)]$

a) urejena po prioritetah: $p_1 \geq p_2 \geq \cdots \geq p_n$

- vzemi prvega iz vrste $O(1)$
- vstavi element x s prioriteto p:

  $O(\log n)$ – poišči mesto v tabeli, kamor ga bomo vstavili (z bisekcijo)

  $O(n)$ – naredimo "luknjo" tako, da prestavimo elemente za 1 mesto

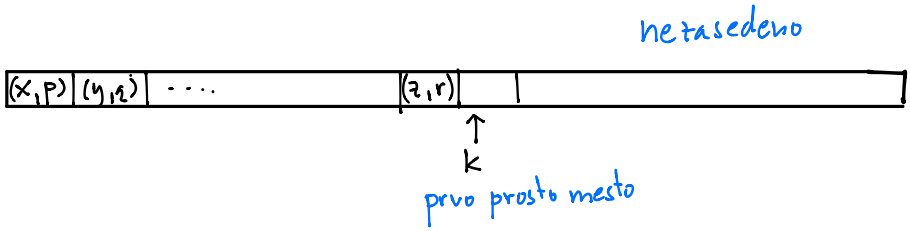
  $O(n)$ skupaj

b) ni urejena po prioritetah

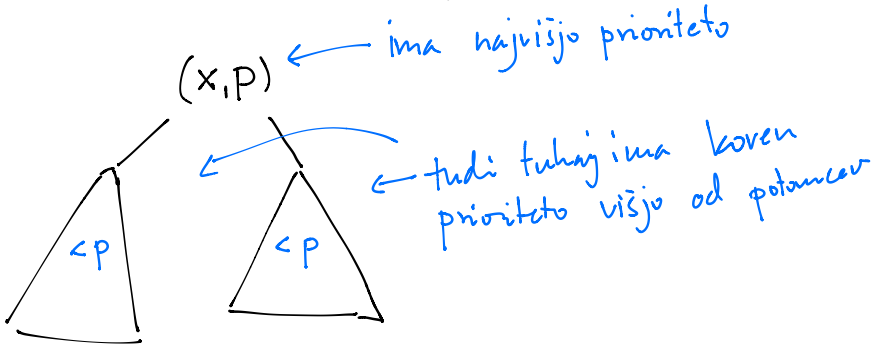
- vzemi prvega iz vrste $O(n)$
- vstavi (na konec) $O(1)$

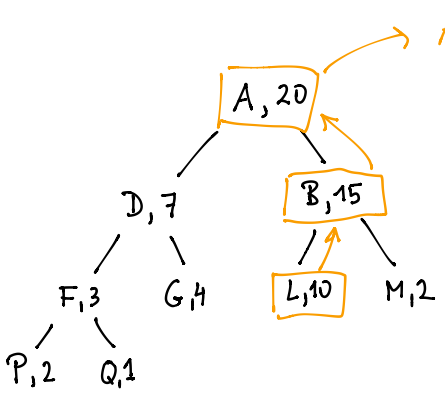
Boljša rešitev: uporabimo kopico

Imamo tabelo



Predstavljamo si, da so elementi razporejeni v dvojiško drevo:

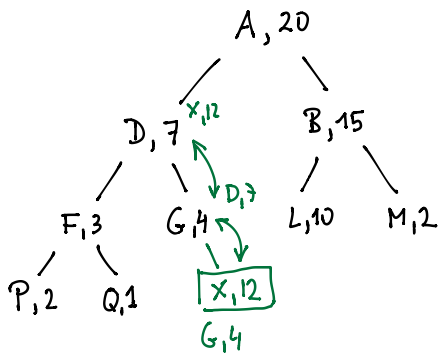




Vzemi prvega iz vrste
- v korenu je luknja
- izmed obeh sinov izberemo večjega in ga prestavimo v koren

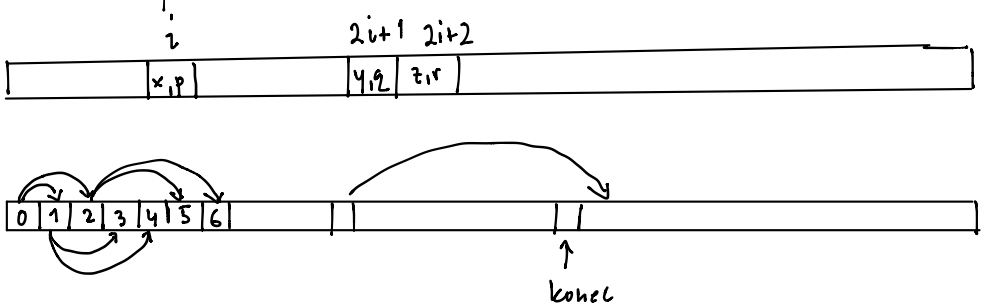
O(globina drevesa)

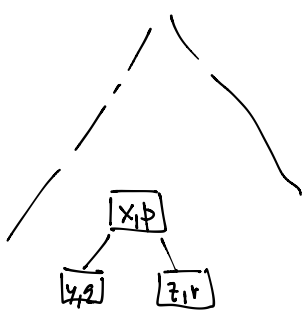
- Dodaj X,12 v list
- "Porivamo" ga navzgor po drevesu, dokler je prevelik.

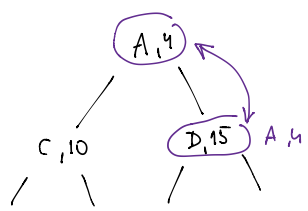


O(globina drevesa)

Kopico spravimo v tabelo:



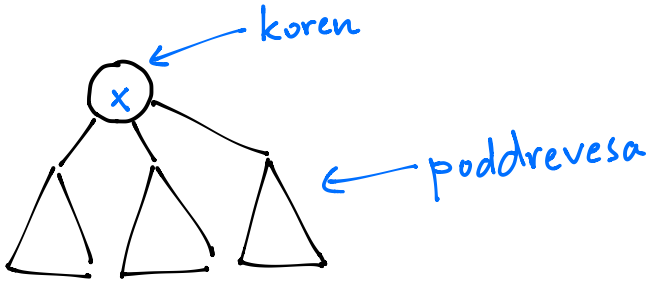




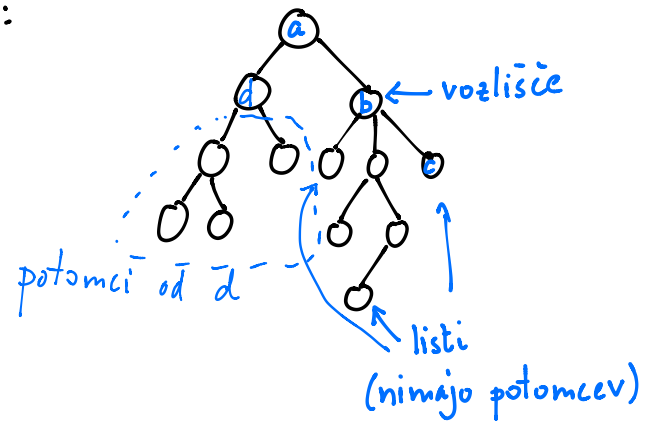
Drevesa

Podatkovna struktura, rekurzivna:

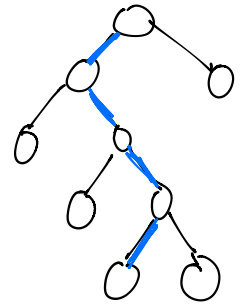
- prazno drevo
- sestavljeno drevo:



Primer:



a je oče od b
b je sin od a
d in b sta brata
a in b sta prednika c



globina 5

Globina:
število vozlišč v najdaljši poti od korena do lista

Dvojiško drevo:
vsako vozlišče ima največ dva sinova

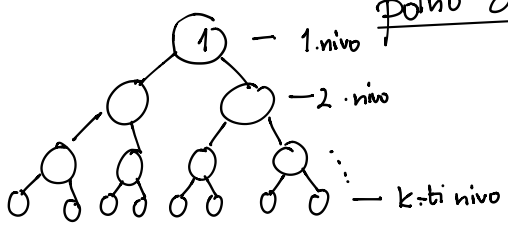
Denimo, da imamo dvojiško drevo z n vozlišči (velikost je n).
Kakšna je lahko globina?

Najglobje:

Drevo se ne veji, globina $= n$

Najmanjša globina:

Polno drevo globine k



$$n = 1 + 2 + 4 + \cdots + 2^k = 2^{k+1} - 1$$

$$n \approx 2^k$$

$$\text{globina} \approx \log_2 n$$

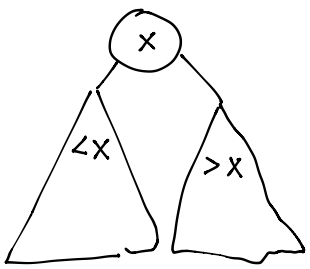
Sklep:

$$\log_2 n \leq \text{globina} \leq n$$

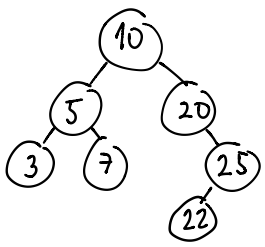
Iskalna drevesa

Dvojiško drevo je iskalno drevo, če:

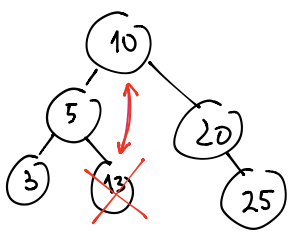
1) Vsi potomci na levi so manjši od korena
2) Vsi na desni so večji od korena
3) Enako velja za obe poddrevesi



Primer:



Protiprimer:



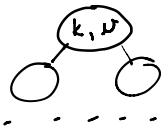
Osnovne operacije:

- prazno iskalno drevo
- ali je prazno?
- poišči element x v drevesu
- vstavi element x v drevo
- zbriši element x iz drevesa

Uporaba:

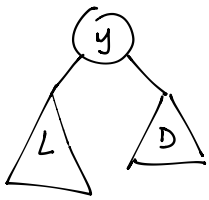
Z iskalnim drevesom predstavimo slovar:

{5:'a', 6:'b', 1:'z', ...}



V vozlišča damo pare (ključ, vrednost) in poskrbimo, da je drevo iskalno glede na ključe

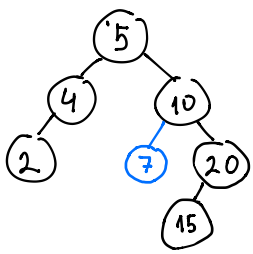
Poišči x:



if drevo je prazno:
    x ni v drevesu
elseif x = koren:
    smo našli
elseif x < koren:
    išči levo rekurzivno
else x > koren:
    išči desno

Vstavi x v drevo:

Vstavimo ga tja, kjer ga bi pričakovali, ko ga iščemo.

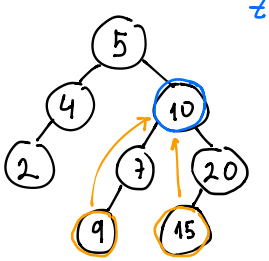


vstavimo 7

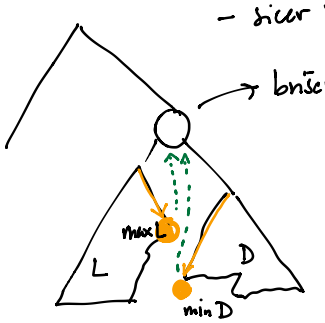
1) Z bisekcijo poiščemo mesto, kamor ga bomo vstavili
2) Ga vstavimo na najdeno mesto

Zbriši x iz drevesa:

zbriši 10



1) Z bisekcijo poišči x
2) Zbrišemo x in ustvarimo "luknjo"
   - če je x list, smo končali
   - sicer nadomestimo z največjim v levem ali z najmanjšim v desnem poddrevesu

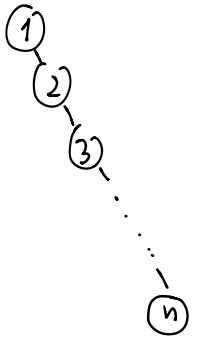


Časovna zahtevnost:

- iskanje : $O(\text{globina})$
- vstavljanje : $O(\text{globina})$
- brisanje : $O(\text{globina})$

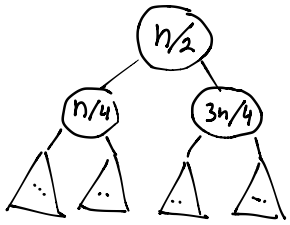
} dobro iskalno drevo mora imeti majhno globino.

Slabo iskalno drevo:



globina = n

Dobro:



globina = $\log_2 n$