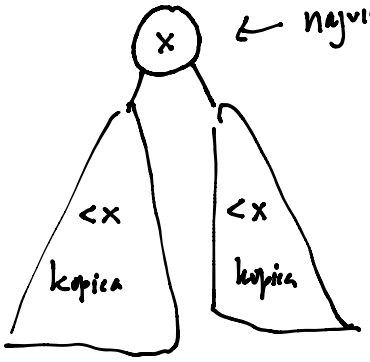
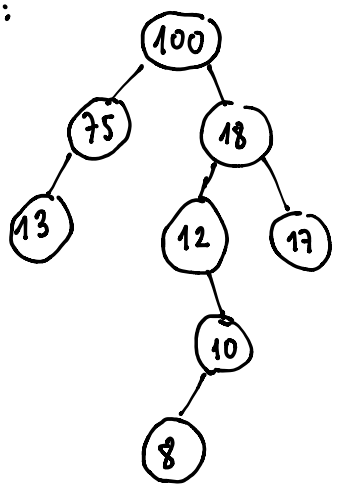
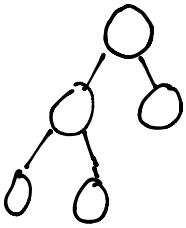
Kopice

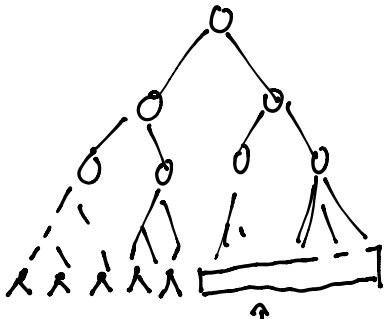


Primer:



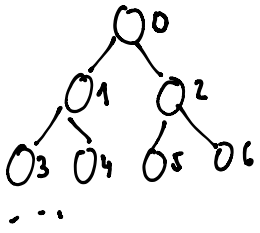


Poskrbimo, da je kopica "polna"

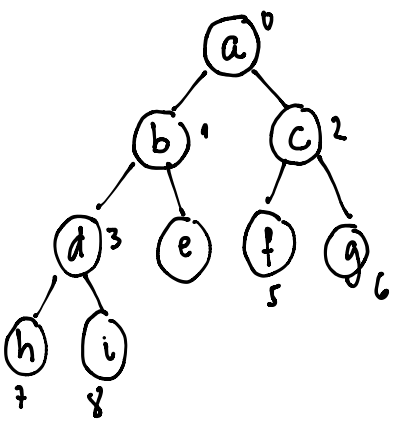


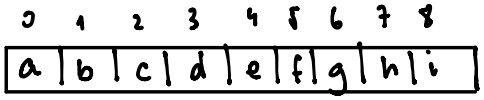
samo v spodnji vrsti manjkajo listi

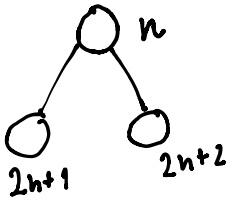
Kopico zložimo v tabelo

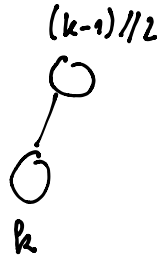


. . . .

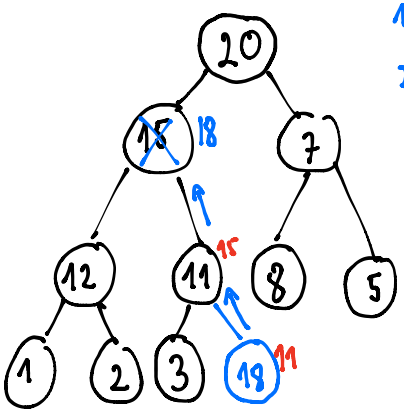






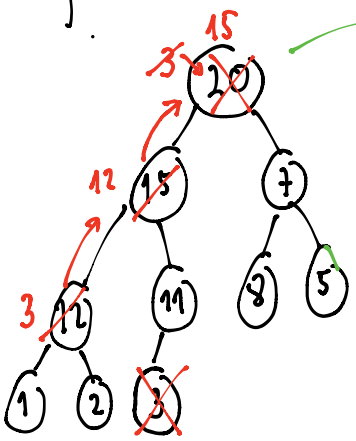


Dodajanje elementa:



1. dodamo 18 na konec
2. pomikamo 18 proti korenu, da spet vzpostavimo kopico

Odvzemanje elementa:



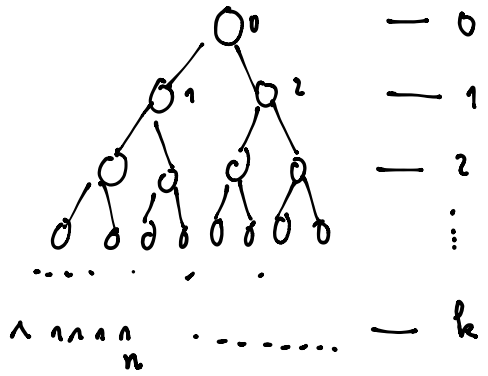
→ 20

1. Izpraznimo koren
2. Zadnji element damo v koren
3. Od korena ga potiskamo navzdol, da vzpostavimo kopico.

Časovna zahtevnost operacij: $O(\text{globina drevesa})$

Torej: če imamo kopico z n elementi, kolikšna je globina?

n vozlišč



globina k

vozlišča na nivoju i

Število vozlišč: $1 + 2 + 4 + \ldots + 2^i + \ldots + 2^k =$

$$2^{k+1} - 1 = n$$

$$k = \log_2(n+1) - 1$$

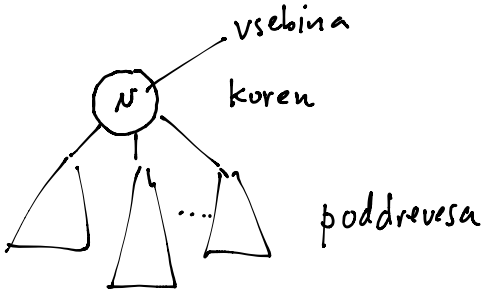
$$k \approx \log_2 n$$

Torej: Časovna zahtevnost operacij je $O(\log_2 n)$

Drevesa

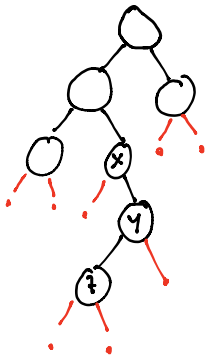
Podatkovna struktura:

- drevo je lahko prazno ali
- sestavljeno:



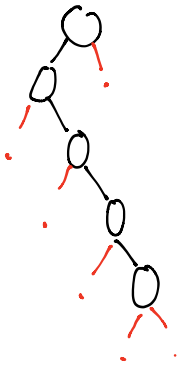
Rekurzivna struktura:
drevo je sestavljeno iz dreves

Dvojiško drevo: vozlišče ima dve poddrevesi, ali pa jih nima:



x ima dve poddrevesi:
- desno: 

- levo: prazno



Bolje: Dvojiško drevo, če ima vsako vozlišče 0, 1 ali 2 poddrevesi.

Drevo predstavimo z objekti:

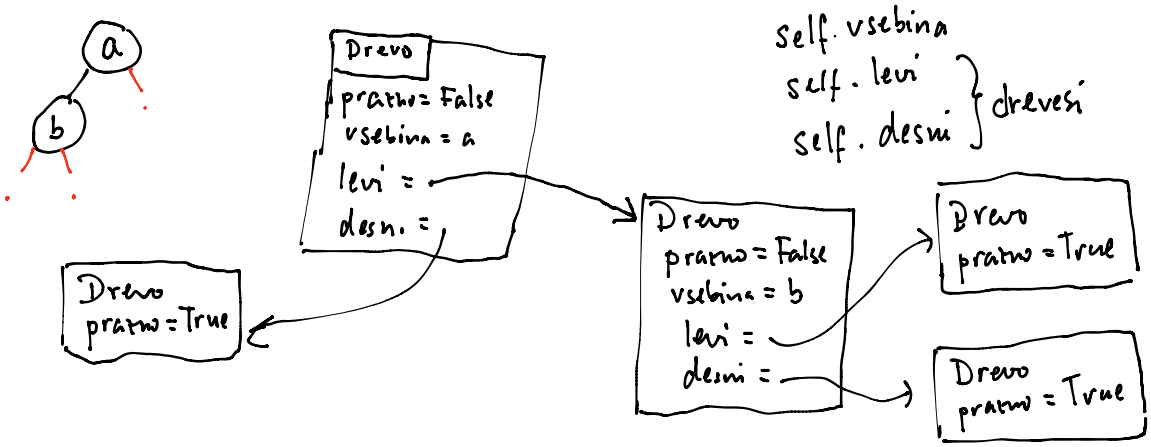
class Drevo():
⋮

atributi?

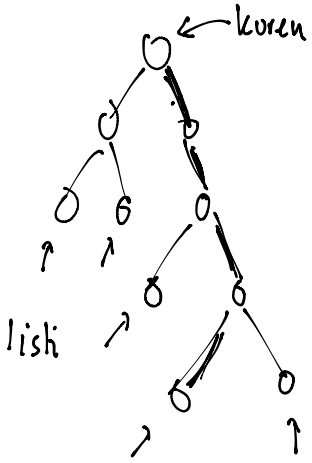
self.prazno = True / False

če ni prazno:

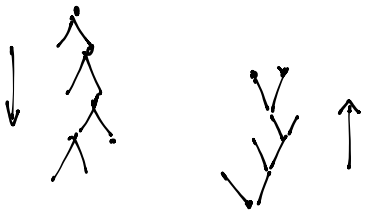
self.vsebina
self.levi
self.desni } drevesi



Globina

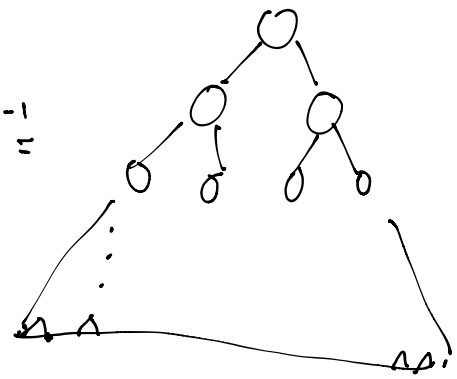


dolžina najdaljše poti od korena do lista



Polno drevo globine n:

Vozlišč je
$2^0 + 2^1 + 2^2 + \dots + 2^{n-1} =$
$2^n - 1$



Polno drevo:
→ globina 0 .... prazno
→ globina $n > 0$: