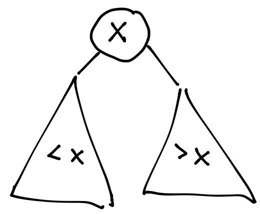
AVL drevesa

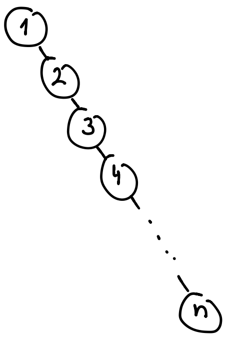
Imenujejo se po **A**delson-**V**elsky and **L**andis

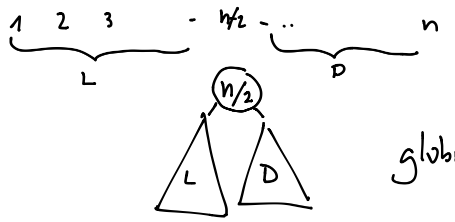
Iskalno drevo



rekurzivno

globina n

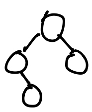




globina $\log_2 n$

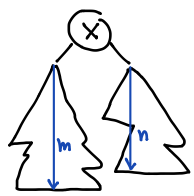
Operacije: iskanje, vstavljanje, brisanje } $O(\text{globina})$

Izboljšava: poskrbimo, da je globina vedno $O(\log n)$

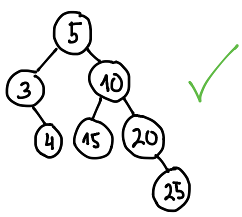


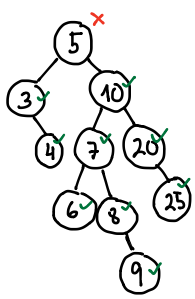
Def: Drevo je <u>uravnoteženo</u>, če za vsako vozlišče v drevesu velja: globini levega in desnega poddrevesa se razlikujeta za največ 1.

Dejstvo: globina uravnoteženega drevesa je $O(\log n)$.



$|m-n| \leq 1$





Kako izračunamo globino poddrevesa?

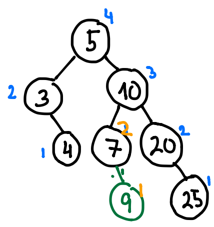
$$\text{globina}(\text{prazno}) = 0$$

$$\text{globina}(t) = 1 + \max(\text{globina}(l), \text{globina}(d))$$

Časovna zahtevnost računanja globine $O(n)$
↳ velikost drevesa

Ideja: globine ne računamo, globino drevesa hranimo kot podatek v korenu

globine

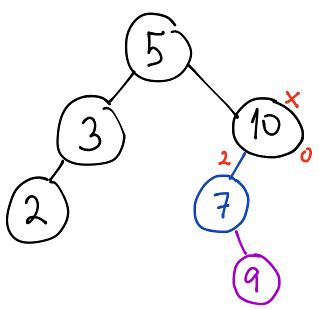


Pazimo, da se podatek o globini ustrezno osveži, ko drevo spreminjamo.

Operacije v AVL drevesu:

Iskanje: iščemo z bisekcijo, kot v iskalnem drevesu

Vstavljanje: vstavimo tja, kjer bi podatek iskali

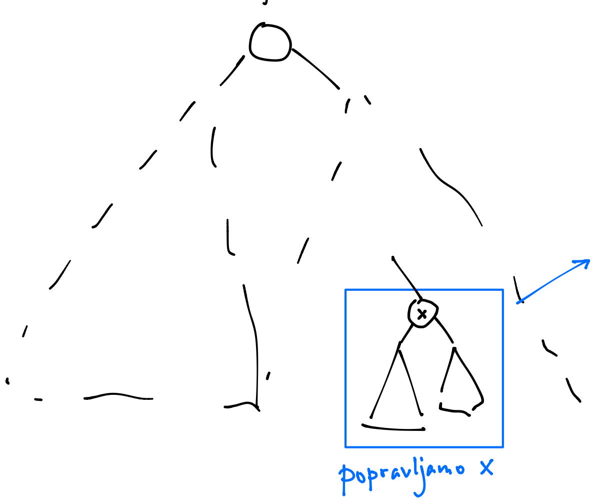


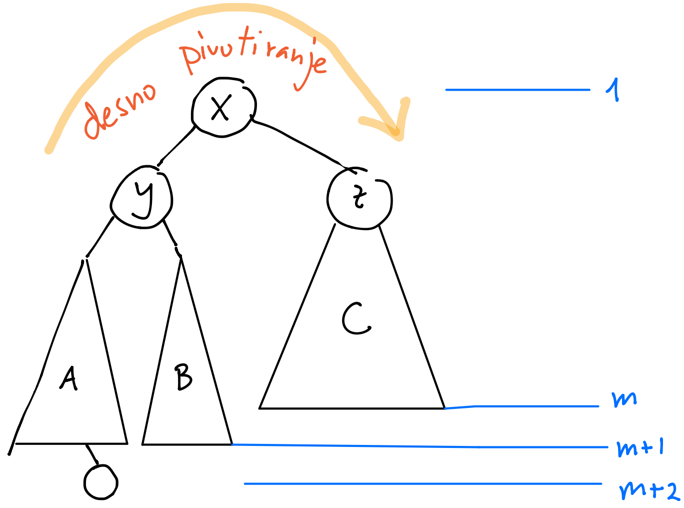
vstavimo 7
vstavimo 9

1) Vstavimo vozlišče v ustrezni list (tja, kjer bi ga iskali)

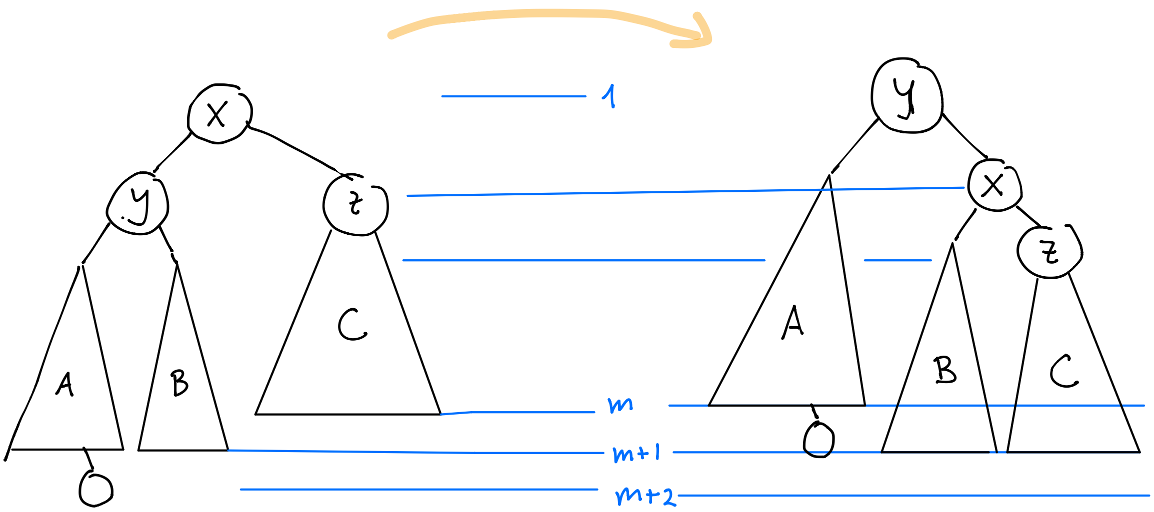
2) Pregledamo vozlišča od vstavljenega lista navzgor do korena:
   - popravimo podatek o globini
   - preverimo uravnoteženost
   - če najdemo neuravnoteženo vozlišče, ga popravimo s pivotiranjem (in ne potujemo več navzgor po drevesu)

Pivotiranje:

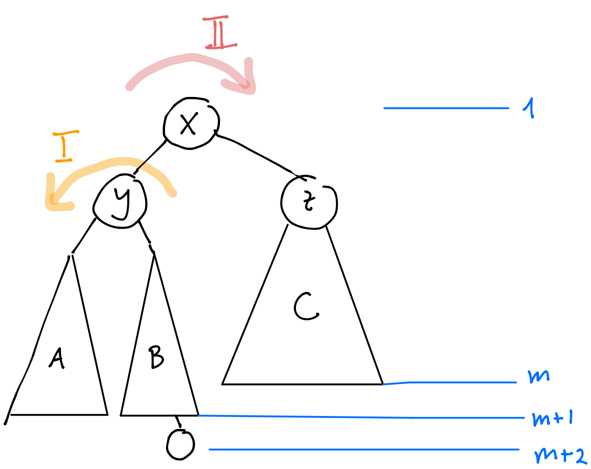




⇓ dobimo tole



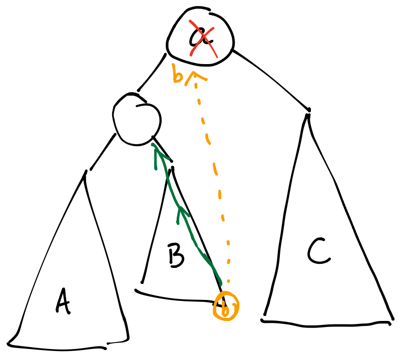
Še ena vrsta pivotiranja: najprej naredimo pomožno pivotiranje I, nato II



+ LEVO PIVOTIRANJE
(simetrično)

Brisanje:

1) Poiščemo vozlišče, ga zbrišemo, v luknjo vstavimo ustrezni list



2) Od izbrisanega lista potujemo proti korenu, popravljamo globine in preverjamo uravnoteženost; če je treba naredimo pivotiranje.