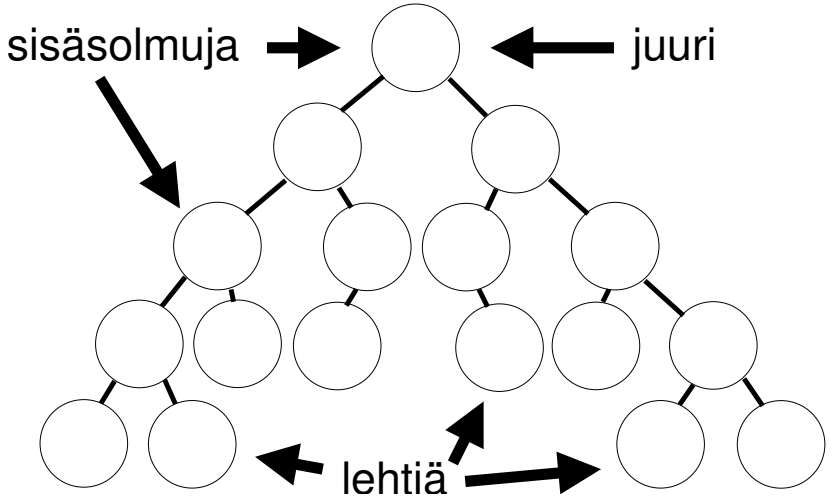
8.1 Puu

Ennen kuin käydään käsiksi kekoon, määritellään sen tueksi käsite *puu*.

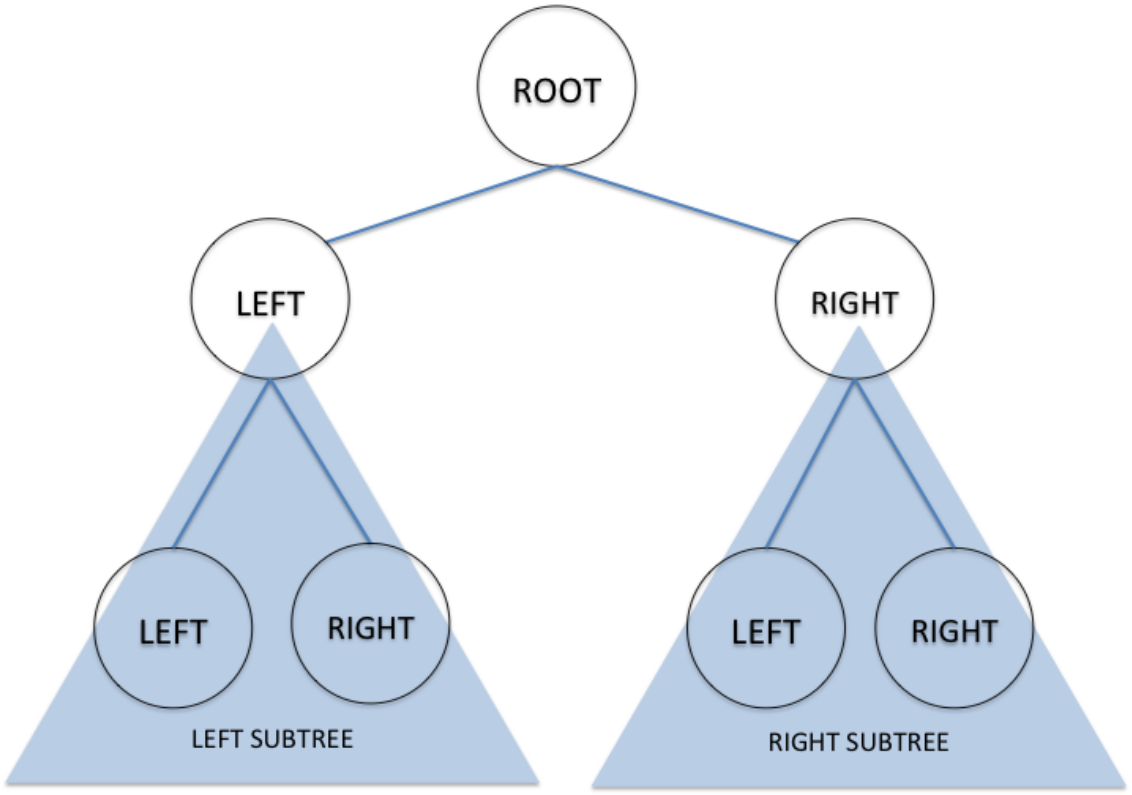
Puu on:



- rakenne, joka koostuu solmuista, joilla on mielivaltainen määrä lapsia.

- *Binääripuussa* lasten määrä on rajoitettu välille 0–2. Tällöin lapset nimetään *vasen (left)* ja *oikea (right)*
- solmu on lapsiensa *isä (parent)*
- lapseton solmu on *lehti (leaf)*, ja muut solmut ovat *sisäsolmuja (internal node)*
- puussa on korkeintaan yksi solmu, jolla ei ole isää. Isätön solmu on puun *juuri (root)*.
  - kaikki muut solmut ovat juuren lapsia, lastenlapsia jne.

- puun rakenne on rekursiivinen: kunkin solmun jälkeläiset muodostavat puun *alipuun*, jonka juuri kyseinen solmu on



Kuva 13: Binääripuun rekursiivisuus

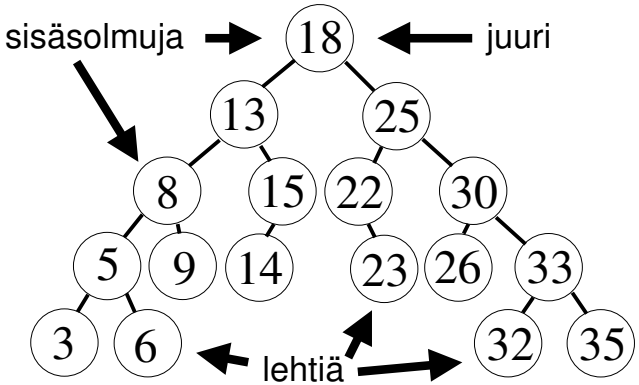
- puun solmun *korkeus (height)* on pisimmän solmusta

suoraan alas lehteen vievän polun pituus
  - pituus lasketaan kaarien mukaan, jolloin lehden korkeus on 0
- puun korkeus on sen juuren korkeus
- puu on *täydellisesti tasapainotettu (completely balanced)*, jos sen juuren lasten määräämien alipuiden korkeudet eroavat toisistaan enintään yhdellä, ja alipuut on täydellisesti tasapainotettu
- n-solmuisen puun korkeus on vähintään $\lfloor \lg n \rfloor$ ja korkeintaan n - 1 (logaritmin kantaluku riippuu lasten maksimimäärästä)
  $\Rightarrow O(n)$ ja $\Omega(\lg n)$

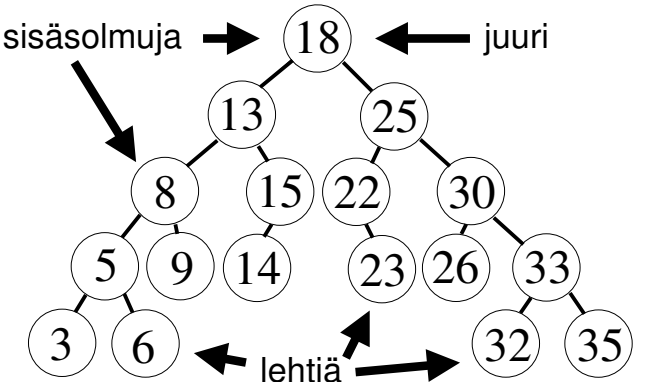
Puun solmut voidaan käsitellä monessa eri järjestyksessä.

- *esijärjestys* (*preorder*) eli ensin käsitellään juuri, sitten rekursiivisesti lapset.
  - kutsu: PREORDER-TREE-WALK$(T.root)$
  - esimerkin käsittelyjärjestys on 18, 13, 8, 5, 3, 6, 9, 15, 14, 25, 22, 23, 30, 26, 33, 32, 35

```
PREORDER-TREE-WALK(x)
1   if x ≠ NIL then
2       käsittele alkio x
3       for child in x→children do
4           PREORDER-TREE-WALK(child)
```



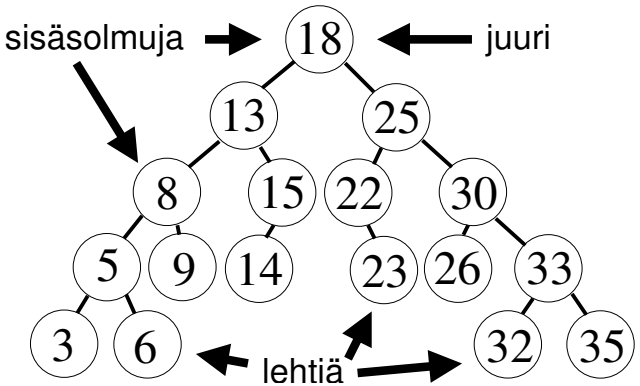
- *välijärjestys* (*inorder*)
  - välijärjestys koskee lähinnä *binääripuuta*, siinä käsitellään ensin rekursiivisesti vasen lapsi, sitten juuri ja lopuksi rekursiivisesti oikea lapsi
  - esimerkissä 3, 5, 6, 8, 9, 13, 14, 15, 18, 22, 23, 25, 26, 30, 32, 33, 35



INORDER-TREE-WALK(x)
1 **if** $x \neq$ NIL **then**
2 INORDER-TREE-WALK($x \rightarrow left$)
3 käsittele alkio x
4 INORDER-TREE-WALK($x \rightarrow right$)

- *jälkijärjestys* (*postorder*), eli ensin käsitellään rekursiivisesti lapset, lopuksi vasta juuri
  - esimerkissä 3, 6, 5, 9, 8, 14, 15, 13, 23, 22, 26, 32, 35, 33, 30, 25, 18

```
POSTORDER-TREE-WALK(x)
1  if x ≠ NIL then
2      for child in x→children do
3          POSTORDER-TREE-WALK(child)
4      käsittele alkio x
```



Puun läpikäynnin ajankäyttö:

- ajoaika $\Theta(n)$, algoritmit kutsuvat itseään kahdesti joka solmussa: kerran vasemmalle ja kerran oikealle lapselle
- lisämuistin tarve = Θ(rekursion maksimisyvyys) = $\Theta(h+1)$ = $\Theta(h)$