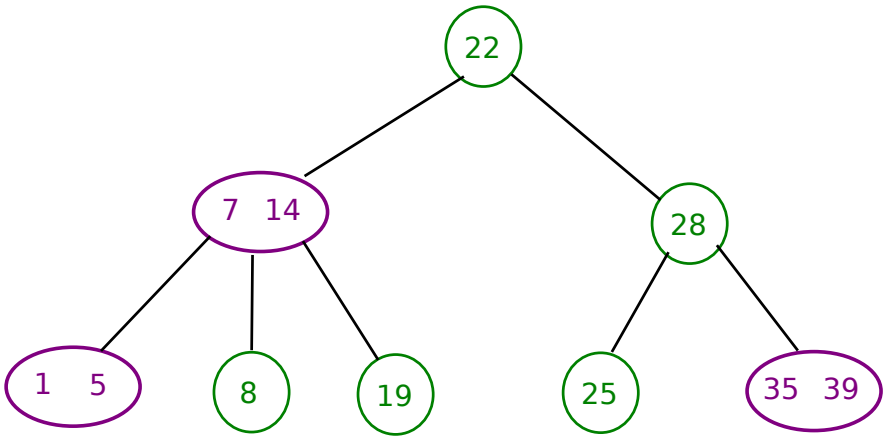
Balancerede søgetræer

- 2-3-træer er balancerede søgetræer
- Simulere 2-3-træer som binære søgetræer

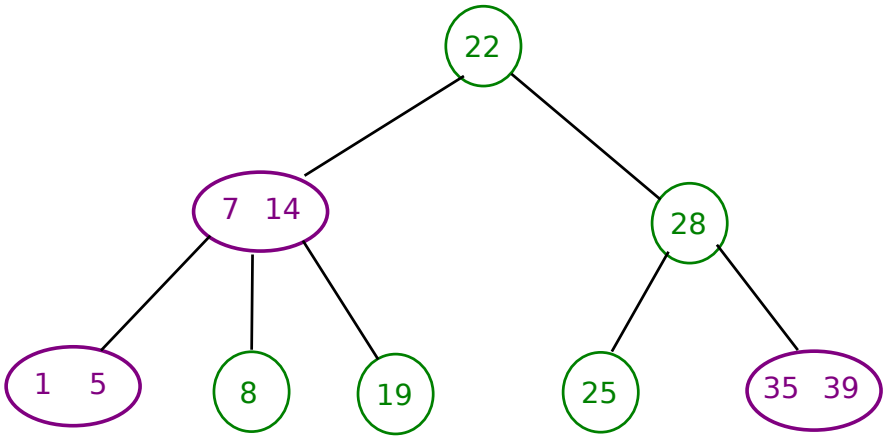
2-3-træer

- Rodfæstet træ.
- To slags knuder:
  - 2-knude: 1 nøgle og 2 børn
  - 3-knude: 2 nøgler og 3 børn
- Overholder søgetræorden:
  - 2-knude med nøgle x: alle i venstre barns deltræ $\leq x$, alle i højre barns deltræ $> x$.
  - 3-knude med nøgler y, z alle i venstre barns deltræ $\leq y$, alle i højre barns deltræ $> z$ og alle i midterste barns deltræ er mellem y og z.
- Perfekt balanceret:
  - Alle blade har samme afstand fra roden.
  - Højde? $O(\log n)$.



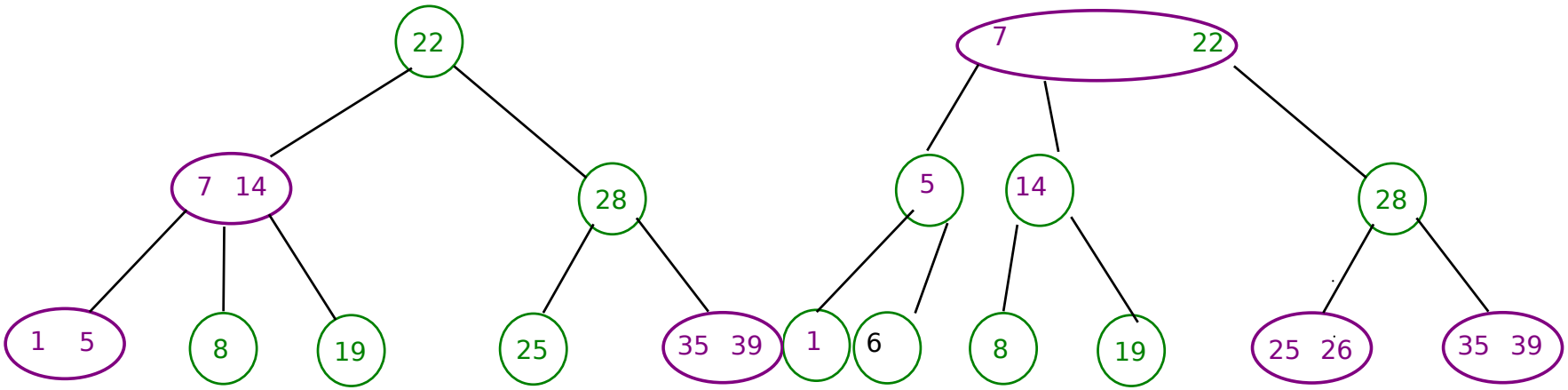
2-3-træer

- Søgning.
  - Findes 28?
  - Findes 4?
- Foregænger (predecessor)/efterfølger (successor).
  - Foregænger til 11?
  - Efterfølger til 26?
- Tid? $O(h) = O(\log n)$.

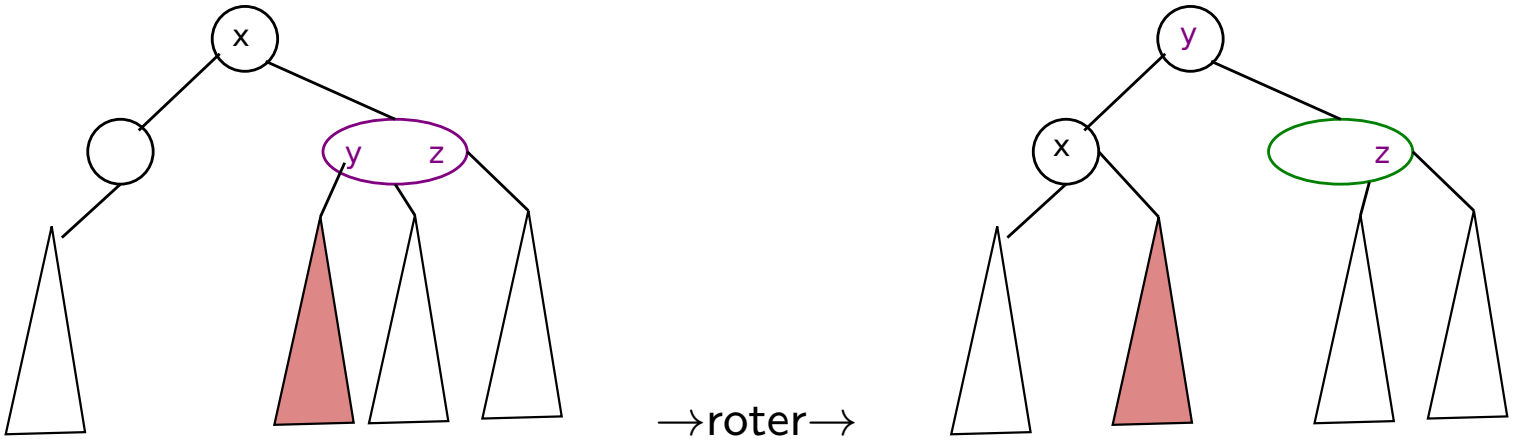


2-3-træer

- Indsætte element.
  - Foretag søgning. Find det passende blad.
  - Indsæt nøgle i blad.
  - Ved overflow, propager opad.
- Eksempel:
  - indsæt 26.
  - indsæt 6.
- Tid? $O(h) = O(\log n)$



- Slette element.
  - Hvis ikke blad, find efterfølger og "byt den ind" (som sidst).
  - Dvs. i alle tilfælde: kan antage, vi sletter et blad.
  - Hvis knuden bliver tom, to tilfælde:
    - "Stæle" fra sin søskend (rotation)
    - "Forene" med sin søskend.
  - Fortsæt evt. rekursivt hos forælder.

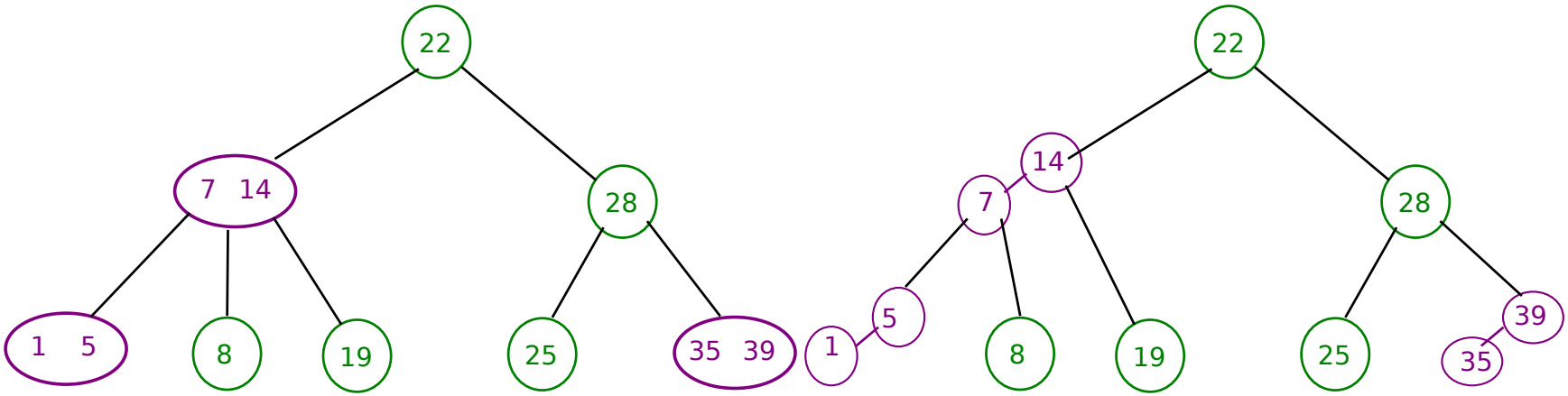


Balancerede søgetræer

- 2-3-træer er balancerede søgetræer
- Simulere 2-3-træer som binære søgetræer

Venstrelænende binære søgetræer

- Transformere/fortolke 2-3-træ som binært træ.
- To slags knuder:
  - 2-knude: 1 nøgle og 2 børn. Binær. :)
  - 3-knude: 2 nøgler og 3 børn. Ikke binær.



Konklusion

Der findes balancerede binære søgetræer:
Højde $O(\log n)$,
Slet/indsæt i $O(\log n)$ tid,
Forgænger/efterfølger i $O(\log n)$ tid.

Vi indså det ved at forstå 2-3-træer.

Kan man forestille sig knuder med k børn og $k-1$ nøgler?
Ja!
B-træer, B^ε-træer, ...
Praktiske (og teoretiske) fordele.