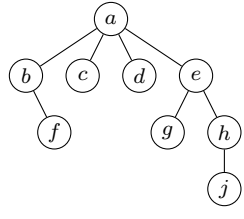
Laveste felles forfeder

Den *laveste felles forfederen* av to noder u og v i et tre T er den laveste (altså noden med høyest dybde) som har både u og v som en etterkommer.

Her er noen eksempler fra treet ved siden av:

- Laveste felles forfeder av a og a er a
- Laveste felles forfeder av b og e er a
- Laveste felles forfeder av g og j er e
- Laveste felles forfeder av e og j er e



(a) Forklar kort hvorfor det for hvert par av noder må eksistere en felles forfeder.

(b) Anta at hvis v er en node, har v kun følgende felter:

- v.`parent` – foreldernoden til v
- v.`children` – barnenodene til v
- v.`depth` – dybden til v

Gi en algoritme som, gitt to noder u og v, finner og returnerer deres laveste felles forfeder.

```
Input: To noder u of v
Output: Laveste felles forfeder av u og v
1 Procedure LowestCommonAncestor(u, v)
  |   // ···
```

(c) Anta nå at treet er et *binært søketre av unike heltall*. Hvis v er en node, så har v kun følgende felter:

- v.`element` – heltallet som er lagret i noden
- v.`left` – venstre barn av v
- v.`right` – høyre barn av v

Gi en algoritme som, gitt roten av treet og elementene x og y, returnerer den laveste felles forfeder av nodene som henholdsvis har elementene x og y der $x < y$. Du kan anta at begge elementene x og y finnes i treet.

```
Input: En node v, og to verdier x og y der x < y
Output: Laveste felles forfeder av nodene som har elementene x og y
1 Procedure LowestCommonAncestorBST(v, x, y)
  |   // ···
```

(d) Hva er kjøretidskompleksiteten til algoritmen din fra (c) for et tre med n noder i verste tilfelle? Hva er kjøretidskompleksiteten dersom du antar at treet er balansert?

Hvis du ikke har svart på (c) kan du oppgi svaret for din løsning på (b) i stedet.