Huffman-trær 10 poeng

Vi vil komprimere tekststrenger som består av symbolene `A`, `B`, `C` og `D` ved hjelp av Huffman-koding. De relative frekvensene er gitt av følgende frekvenstabell:

A	B	C	D
1	3	7	2

(a) Hva blir kodelengdene for de ulike symbolene i det tilhørende Huffman-treet?

2 poeng for riktig svar.

Kodelengdene for de ulike symbolene er:

A	B	C	D
3	2	1	3

(b) Hva inneholder løvnodene i det resulterende treet?

1 poeng for riktig svar.

De vil inneholde symbolene og de relative frekvensene.

Generelt i et Huffman-tre har hver node v:

- v.`symbol` – Symbolet til noden (eller **`null`**)
- v.`freq` – Frekvensen til noden
- v.`left` – Venstre barn av noden (eller **`null`**)
- v.`right` – Høyre barn av noden (eller **`null`**)

Et Huffman-tre bygges fra en mengde med par (s, f) der s er et symbol og f er en frekvens.

(c) Beskriv algoritmen for å bygge et Huffman-tre. Du kan skissere den med pseudokode, men en god beskrivelse med naturlig språk gir like god uttelling.

Maksimalt 7 poeng.

Algoritmen består av å opprette en prioritetskø som ordnes etter frekvensen til noden. Vi oppretter en node for hvert symbol- og frekvens par (uten barn) og plasserer disse på køen. Vi vil så ta ut to noder u og v fra prioritetskøen, og lage en ny node w (uten noe satt symbol) som har u og v som barn, og frekvens som svarer til summen av frekvensene til u og v. Den nye noden plasseres på prioritetskøen. Algoritmen terminerer når det kun er én node igjen på prioritetskøen, og den noden vil være rotnoden for Huffman-treet, som returneres som siste steg av algoritmen.

Input: En mengde C med par (s, f) der s er et symbol og f er en frekvens
Output: Et Huffman-tre

```
1  Procedure Huffman(C)
2      Q ← new PriorityQueue
3      for (s, f) ∈ C do
4          Insert(Q, new Node(s, f, null, null))
5      while Size(Q) > 1 do
6          v1 ← RemoveMin(Q)
7          v2 ← RemoveMin(Q)
8          f ← v1.freq + v2.freq
9          Insert(Q, new Node(null, f, v1, v2))
10     return RemoveMin(Q)
```

- 2.5 poeng for å få frem at det brukes en prioritetskø ordnet etter frekvens.
- 2.5 poeng for hvordan to noder slås sammen.
- 1 poeng for termineringskriterie.
- 1 poeng for returverdien.

En student som viser god forståelse for algoritmen kan få full uttelling selv hvis noen av momentene ovenfor ikke blir nevnt eksplisitt.