Løsningsforslag/sensorveiledning

Huffman-koding - Heaps og HeapSort - Rød-svarte trær

Innsetting i heap — 10 poeng

(a) Du er gitt et tomt array på størrelse 7.

0	1	2	3	4	5	6

Sett inn de følgende tallene med min-heap innsetting:

$$3, 5, 10, 8, 4, 2, 7$$

Fyll ut tabellen slik den ser ut etter alle tallene er satt inn. Skriv svaret som en kommaseparert liste.

Tallene settes inn slik:

3						
3	5					
3	5	10				
3	5	10	8			
3	5	10	8	4		
3	4	10	8	5		
3	4	10	8	5	2	
3	4	2	8	5	10	
2	4	3	8	5	10	7
0	1	2	3	4	5	6

Det er kun den siste linjen `2, 4, 3, 8, 5, 10, 7` som trenger inngå i svaret.

2 poeng for riktig svar. 1 poeng for *nesten* riktige svar, som vil si dersom

- svaret er en min-heap, men ikke den samme heapen som den gitte rekkefølgen vil gi opphav til, eller
- svaret er ett bytte unna løsningensforslaget.

(b) Forklar kort hvordan algoritmen for innsetting i en min-heap fungerer, og skisser algoritmen med pseudokode. Du kan anta at arrayet i input ikke er fullt.

```
Input: Et array A som representerer en heap med n elementer, og et element x
Output: Et array som representerer en heap, som inneholder x
1 Procedure HeapInsert(A, x)
  | // ...
```

Hovedidéen er å alltid legge til på neste ledige plass, det vil si indeks n. Det er der neste node *må* være for at treet fortsatt skal være komplett. Hvis noden på den nye plassen er mindre enn foreldrenoden så må de bytte plass. Dette gjøres iterativt (eller rekursivt).

I arrayrepresentasjonen av en heap ligger alltid roten på indeks 0. Foreldrenoden til en gitt indeks i ligger på `ParentOf(i)` $= \lfloor \frac{i-1}{2} \rfloor$ for en gitt indeks i.

Her er algoritmen presentert med pseudokode:

```
Input: Et array A som representerer en heap med n elementer, og et element x
Output: Et array som representerer en heap, som inneholder x
1 Procedure HeapInsert(A, x)
2     A[n] ← x
3     i ← n
4     while 0 < i and A[i] < A[ParentOf(i)] do
5         A[i], A[ParentOf(i)] ← A[ParentOf(i)], A[i]
6         i ← ParentOf(i)
7     return A
```

Det gis opptil 6 poeng for et godt svar på oppgaven.

- Opptil 1 poeng for å gi hvor elementet skal plasseres.
- Opptil 2 poeng for å få frem at noden skal bytte plass med foreldernoden dersom den er mindre.
- Opptil 3 poeng for en godt skissert algoritme.

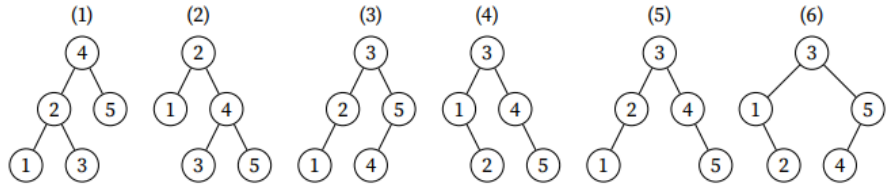
(c) Redgjør for kjøretidskompleksiteten til algoritmen.

Algoritmen for innsetting i heap er $\mathcal{O}(\log(n))$. Det er fordi n halveres i hver iterasjon.

Det gis 1 poeng for riktig kjøretidskompleksitet og 1 poeng for redegjørelsen.

Balanserte søketrær — 6 poeng

Følgende er alle AVL-trær som inneholder tallene 1 til 5.



(a) Hvilket av AVL-trærne ovenfor får man hvis man legger inn tallene 1 til 5 i rekkefølgen $2, 4, 3, 1, 5$? Oppgi svaret som et tall mellom 1 og 6.

5.

(b) Hvor mange av trærne ovenfor kan fargelegges som et rød-svart tre?

6. Alle AVL-trær kan fargelegges som et rød-svart tre.

(c) Hvis vi legger til 6 i alle AVL-trærne ovenfor, i hvilke trær vil det forekomme rotasjoner? Oppgi svaret som summen av alle trærne det gjelder. Altså, hvis det kun forekommer rotasjoner i det første treet, er svaret 1, og hvis det forekommer rotasjoner i alle trærne, er svaret 21 (fordi $1 + 2 + 3 + 4 + 5 + 6 = 21$).

$2 + 4 + 5 = 11$.

Huffmantrær 5 poeng

En tekststreng vi vil komprimere består av 5 ulike tegn a,b,c,d og e. De relative frekvensene er gitt av følgende frekvenstabell:

a	b	c	d	e
2	1	12	1	4

(a) Hva er den lengste kodelengden for et symbol i det tilhørende huffmantreet?

4.

(b) Hvor mange bits blir koden for tegnet e?

2.

(c) Hvor mange bits brukes for å kode strengen aaabbcdee?

26.

Kodelengdene for de ulike symbolene er:

a	b	c	d	e
3	4	1	4	2

Vi har 3 a-er, 2 b-er, 1 c, 1 d og 2 e-er, som gir $3 \cdot 3 + 2 \cdot 4 + 1 + 4 + 2 \cdot 2 = 9 + 8 + 1 + 4 + 4 = 26$.

Nå skal vi ikke jobbe med noe konkret huffmantre for et en gitt frekvenstabell, men heller tenke på generelle huffmantrær.

(d) Hvis en frekvenstabell består av 8 forskjellige tegn, hvor mange noder har det tilhørende huffmantreet?

15. Det vil ha 8 løvnoder og 7 interne noder.

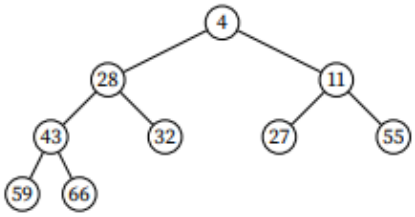
(e) I et huffmantre som har 8 løvnoder, hva er den lengste kodelengden et symbol kan ha?

7.

Det gis 1 poeng per deloppgave.

Binære heaps 6 poeng

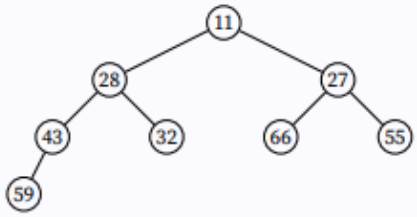
Du skal ta utgangspunkt i følgende binære heap H_1, hvor både tre-representasjonen og array-representasjonen er gitt:



4	28	11	43	32	27	55	59	66
0	1	2	3	4	5	6	7	8

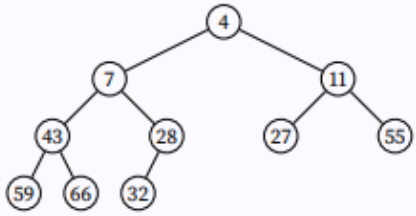
Du skal fylle inn tabellene som svarer til hver deloppgave i Inspera. Hver tabell svarer til array-representasjonen av H_1 etter de oppgitte operasjonene. Deloppgavene er uavhengig av hverandre, altså refererer H_1 til heapen som er gitt ovenfor, og endringer vi gjør i en deloppgave følger ikke med til neste deloppgave.

(a) Hvordan ser heapen ut etter ett kall på H_1.`RemoveMin`()?



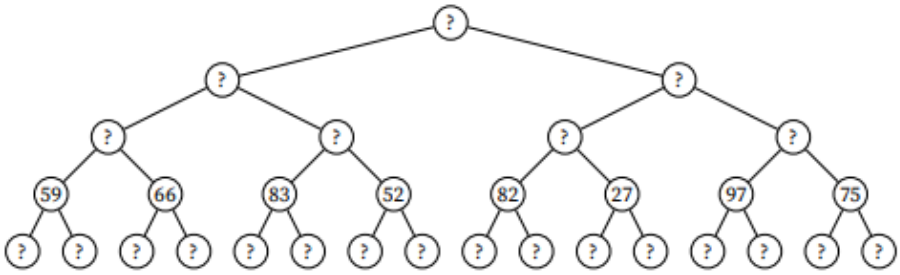
11	28	27	43	32	66	55	59
0	1	2	3	4	5	6	7

(b) Hvordan ser heapen ut etter ett kall H_1.`Insert`(7)?



4	7	11	43	28	27	55	59	66	32
0	1	2	3	4	5	6	7	8	9

I en annen binær min-heap H_2 får du kun gitt verdiene på dybde 3:

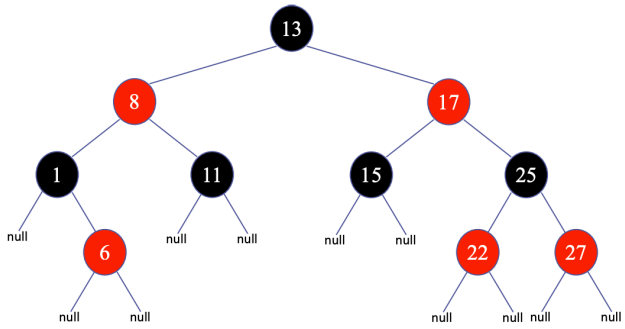


(c) Hva er den minste verdien som kan ligge på dybde 4 av H_2?

27. En heap tillater duplikater, og det ligger en node med verdi 27 på nivå 3 kan den noden ha et barn med samme verdi, altså 27.

Det gis 2 poeng per riktig svar (og 0 for feil svar). På oppgave (c) gis det 1 poeng for svaret 28.

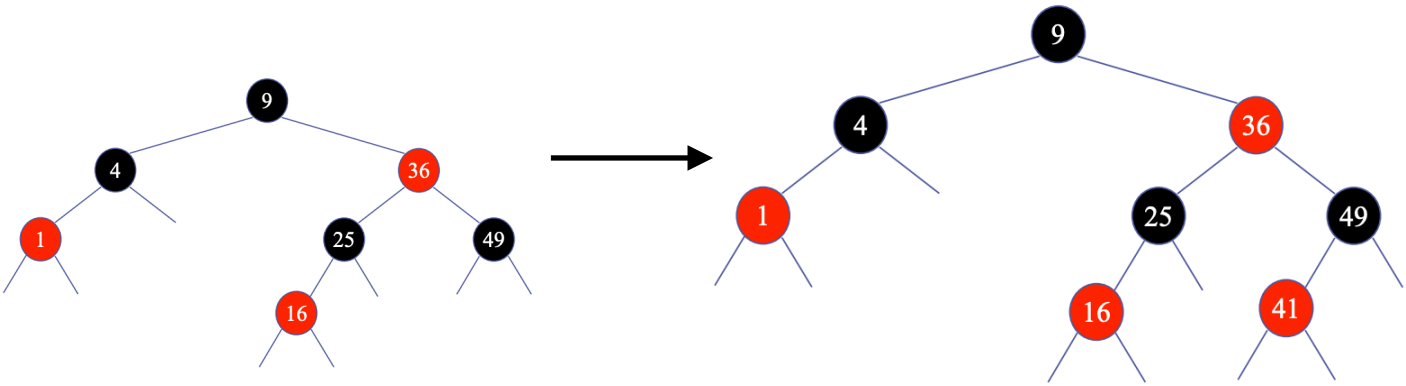
Er dette et rød-svart tre? Begrunn svaret.



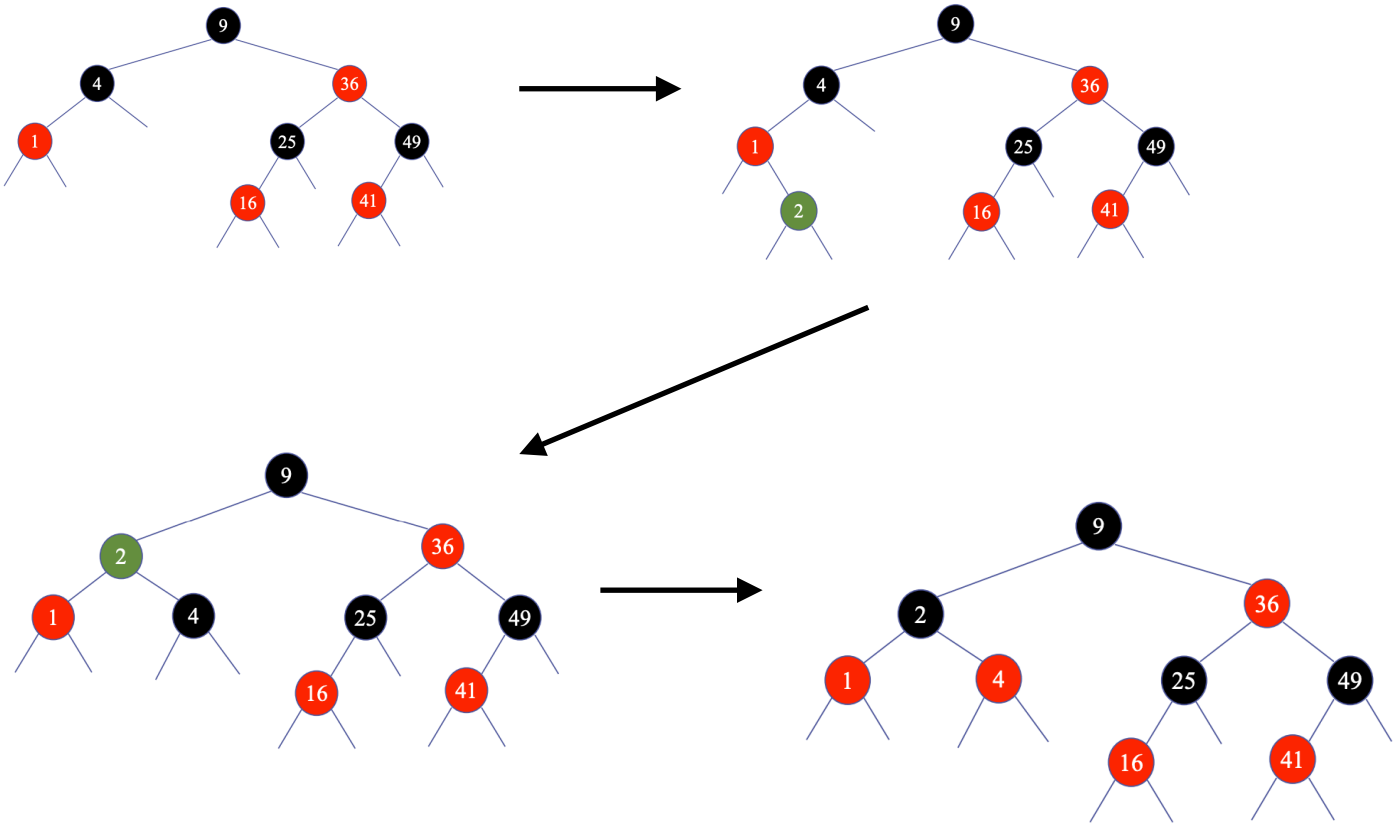
Svar: Ja. Det er et binært søketre der hver node er farget enten rød eller svart. Roten er sort. Hvis en node er rød, er barna svarte. Enhver sti fra en node til en nullpeker inneholder samme antall svarte noder.

Innsetting i rød-svarte trær

a) Sett inn tallet 41



b) Sett deretter inn tallet 2



1(i) Huffmankode

Løsningsforslag: GATTACA.

3(e) Krav til en heap

Løsningsforslag:

Strukturkrav: En (binær) heap er et komplett binærtre. På dypeste nivå er alle nodene til venstre. Ingen node har en nullpeker til venstre for seg. Når nodene ligger i en array (uten 'hull') er dette kravet tilfredsstilt.

Ordningskrav: En node har en verdi som er større enn eller lik verdien til nodens mor/far/forelder.

Eventuelt: En node er mindre enn eller lik verdien til barna og varianter av dette.