Øving 6
Oppgave 1

a) Vi observerer at det tar litt kortere tid å utføre sorteringen siden vi dropper å kjøre gjennom en løkke når vi ikke trenger det. (ved å bruke "break" i stedenfor å bruke boolean.

b) Ved å sortere to elementer samtidig vil tiden det tar å sortere kortere fordi når vi ser at det ene elementet er mindre enn det andre trenger vi først å sortere det største elementet så ser vi at det minste elementet må komme før dette. Derfor trenger vi ikke å sortere det minste elementet på delen bak det største.

Øving 6
Oppave 2
a)

Sortering ved innsetting

n	Antall målinger	Målt tid i ms. (gjennomsnitt)	Teoretisk tid c*f(n)
32000	6	11617	11617
64000	4	50082,2	46468
128000	3	415972,7	185872

Utvalgssortering

n	Antall målinger	Målt tid (gjennomsnitt)	Teoretisk tid c*f(n)
32000	6	8703,8	8703,8
64000	4	39450,8	34815,2
128000	3	178143,2	139260,8

Kvikksortering

n	Antall målinger	Målt tid (gjennomsnitt)	Teoretisk tid c*f(n)
32000	6	207,2	207,2
64000	4	277	443,16
128000	3	382	941,83

Flettesortering

n	Antall målinger	Målt tid (gjennomsnitt)	Teoretisk tid c*f(n)
32000	6	342	347
64000	4	483	740,37
128000	3	707,7	1573,49

For å regne ut teoretisk tid bruker vi følgende formel for å finne C

$$T(32000) = c \cdot f(n)$$

$$T(3200) = c \cdot O(f(n))$$

$$T(3209) = c \cdot n \log 2n$$

$$\frac{T(3200)}{32000 \log 2(32000)} = C$$

Bruker C fra første antall elementer (32000) for hver sorteringsmetode for å se hvordan målt tid stemmer med teoretisk tid for de forskjellige antallene elementer

b) ser at ved sortering av like elementer med Quicksort så tar det like lang tid som ellers før algoritmen går alikevell gjennom alle elementene.

Øving 7

Oppgave 1

a) Bi = to

Binær tre er et tre hvor hver node maksimalt kan ha to "barn" eller "løv"

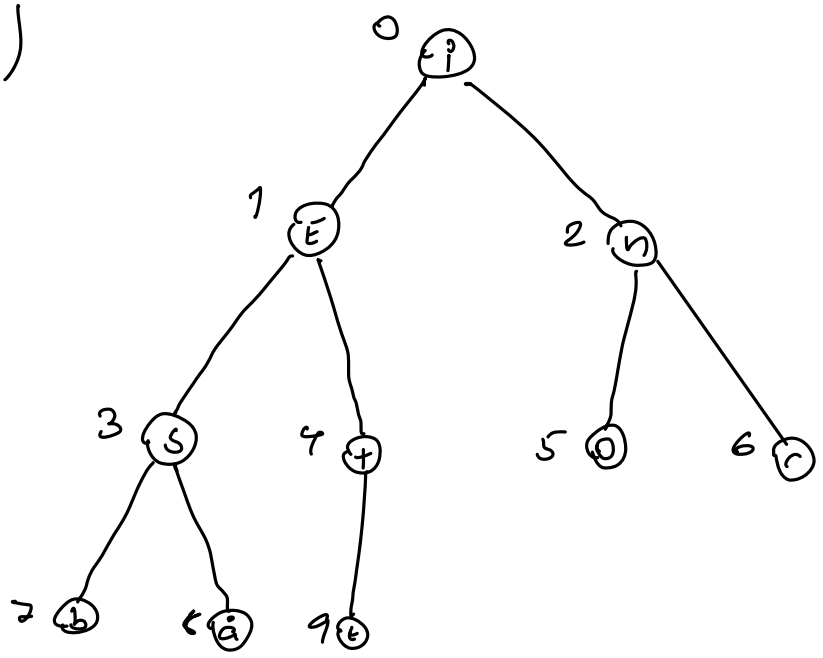
Høyden til et binært tre defineres som den lengste stien fra rot til et "løv"

Et fullt tre er et tre hvor alle "løvene" er på samme nivå. Alle noder er enten blad eller så har de nøyaktig k barn i et tre av orden k

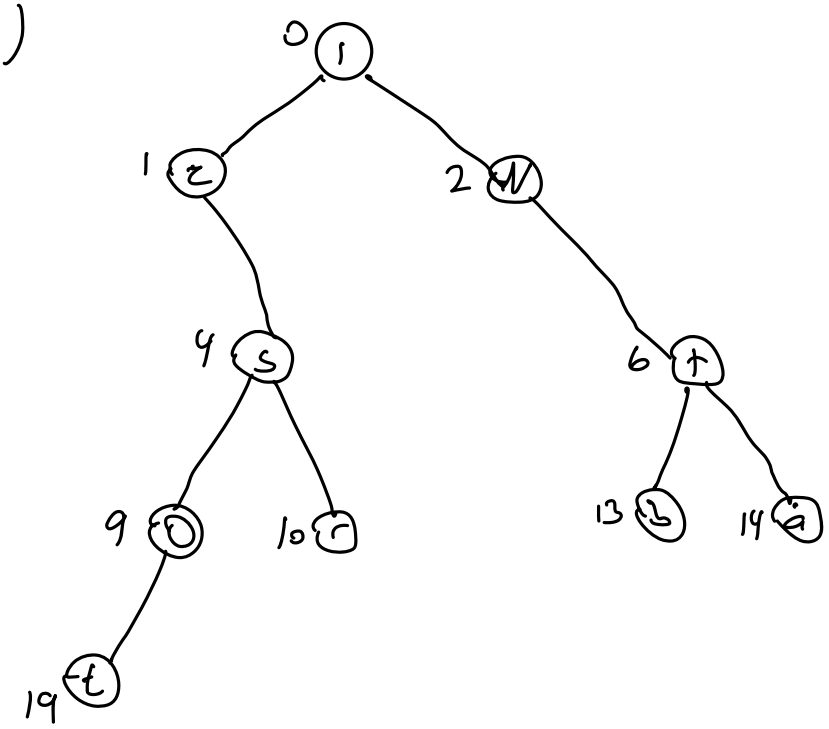
Et kampleft tre er et tre som er fullt eller fullt til vest sist nivå

Viktig at på siste nivå så er "løv"-noden plassert så langt til venstre som mulig

b/ i)



ii)



Postorden

c) i) i) i E s b å t t n O r

ii) i e s O t r N t b å

inorden

ii) i) b s å E t t i o n r

ii) e t o s r i N b t å

postorden

iii) i) b å s t t E o r n i

ii) t o r s e b å t N i

Nivåorden

i) i E n s t O r b å t

ii) i e N s t O r b å t

Øving 8

oppgave 4

c)

gj. høyde $= C \cdot \log_2 1023$

$$C = \frac{\text{gj. høyde}}{\log_2 1023}$$

$$C = \frac{21}{\log_2 1023}$$

$$C = 2,1003$$

gj. høyde for $n = 8191$

$$= C \cdot \log_2 8191$$

$$= 2,1003 \cdot \log_2 8191$$

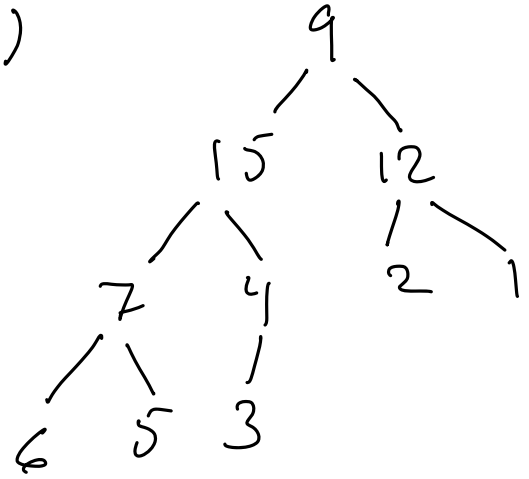
$$= \underline{\underline{27,3}}$$

Kode får 30 så stemmer regnet greit

Øving 8
oppgave 5

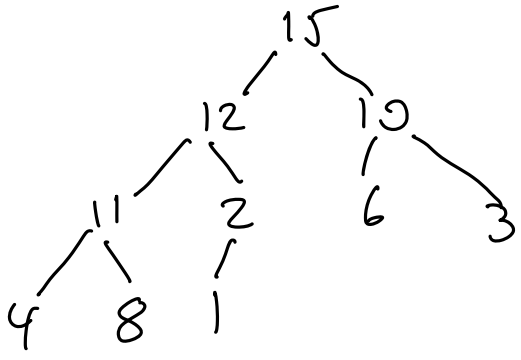
a) En haug (heap) er en datastruktur som brukes til å sortere data og lage prioritetskø.
Formen på en haug er et komplett tre

b) a)



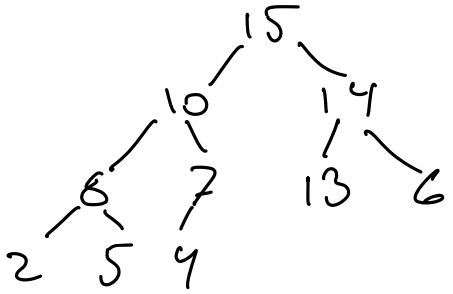
Ikke en maks haug

b)



Maks haug

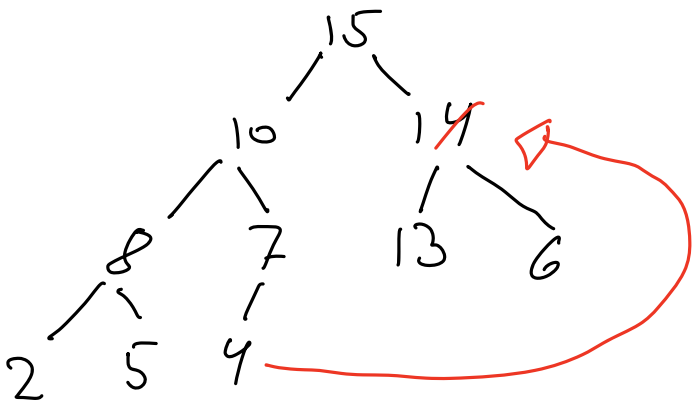
c)



Maks haug

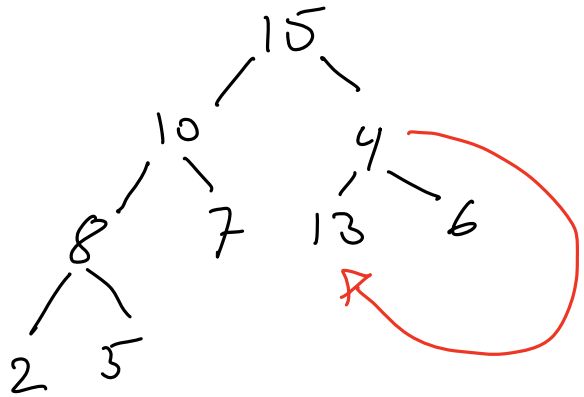
Tabell b og c danner maksheaper

c) i)



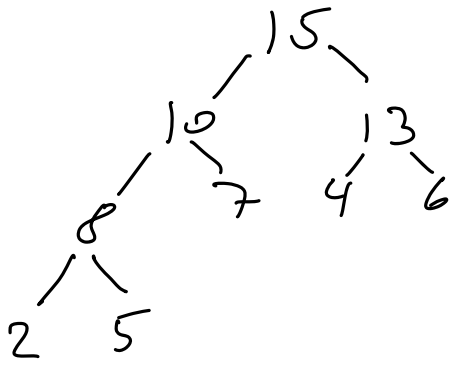
Flytter minste node til rot i undertre

→



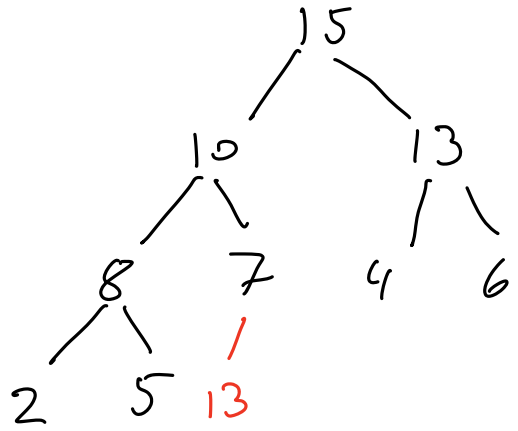
skjekker om 4 er større enn 13, nei så bytter plass

→

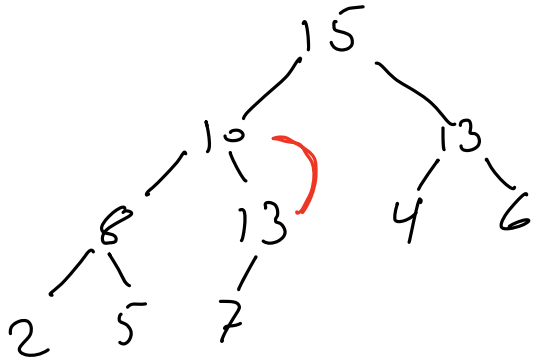


14 er fjernet og tredt danner en maks-heap.

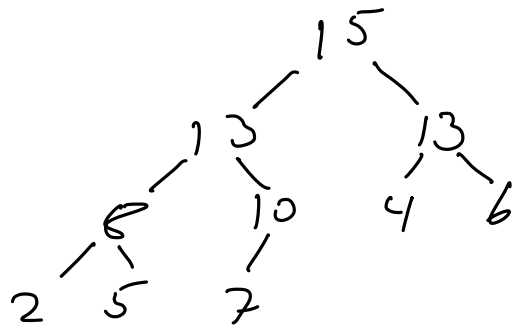
ii)



Setter inn 13 sist i heugen og sammenligner med noden.
13 er større en noden(7) sin så må bytte disse



sammenligner 13 og og noden sin (10)
13 er større så bytter plass



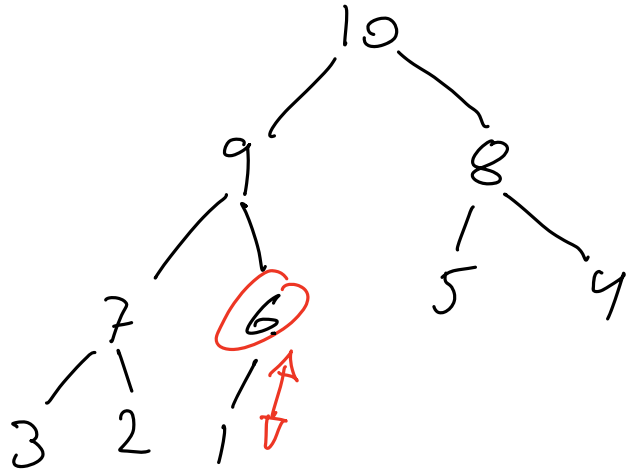
13 er satt inn og treet dannet en maksheug.

d) Et godt eksempel på hvordan prinsippet i C kan brukes til å sortere elementer er hvis elementene har ulik prioriteringsgrad. For eksempel på sykehus. Hvis det kommer en person inn som er akutthskadet

så kommer han før i køen pga at hans prioriteringsgrad er høyere enn de som sitter på venterommet

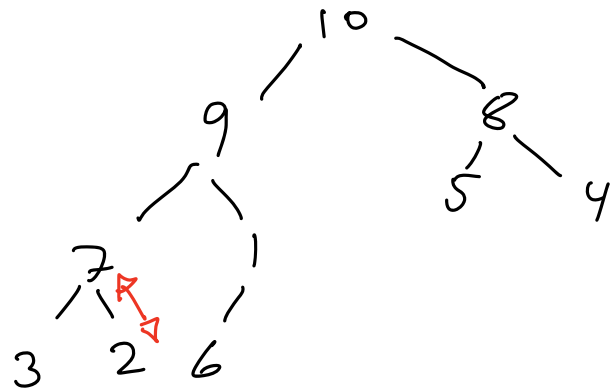
Du sammenligner prioriteringsgraden slik at de som trenger hjelp først får det.

e)



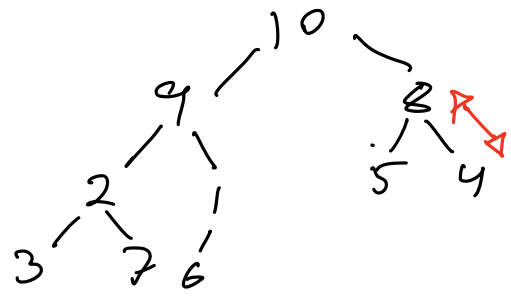
Starter på siste interne barn som ikke er et blad. Skjekker om 6 er større eller mindre en bladet. Det er det ikke så bytter disse

→

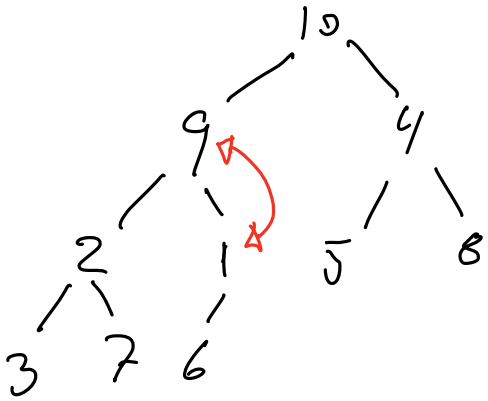


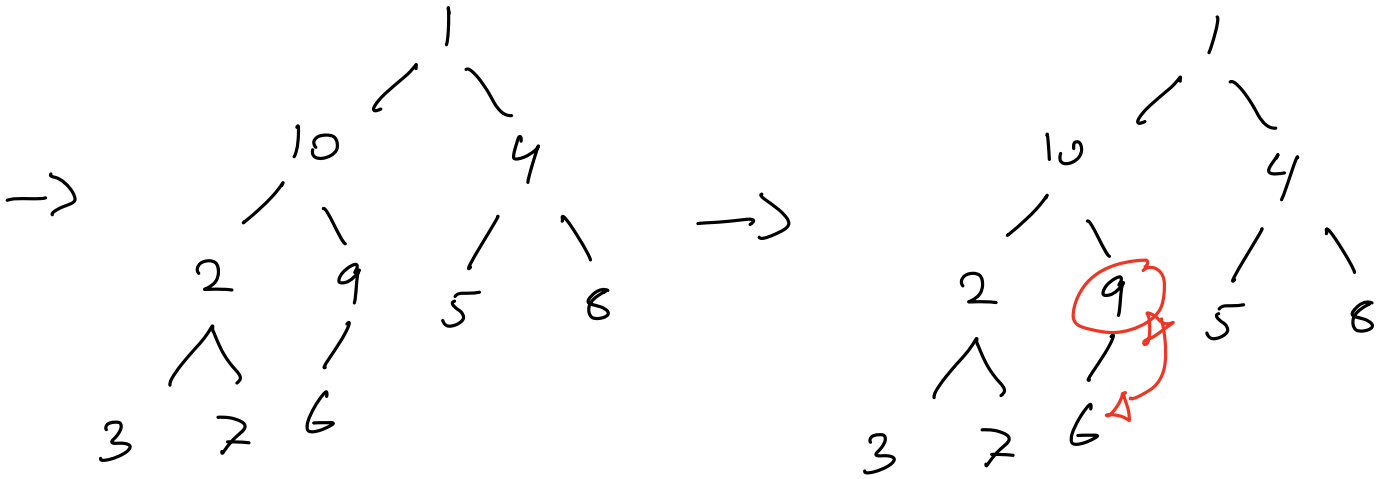
gjør samme operasjon som over med 7

→

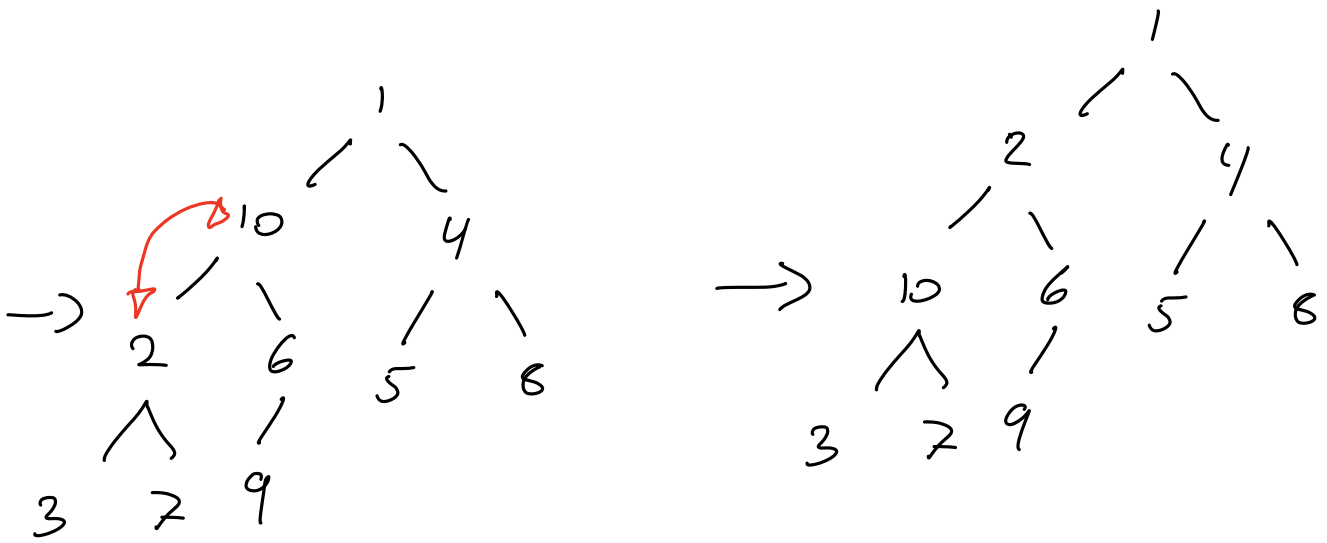


→

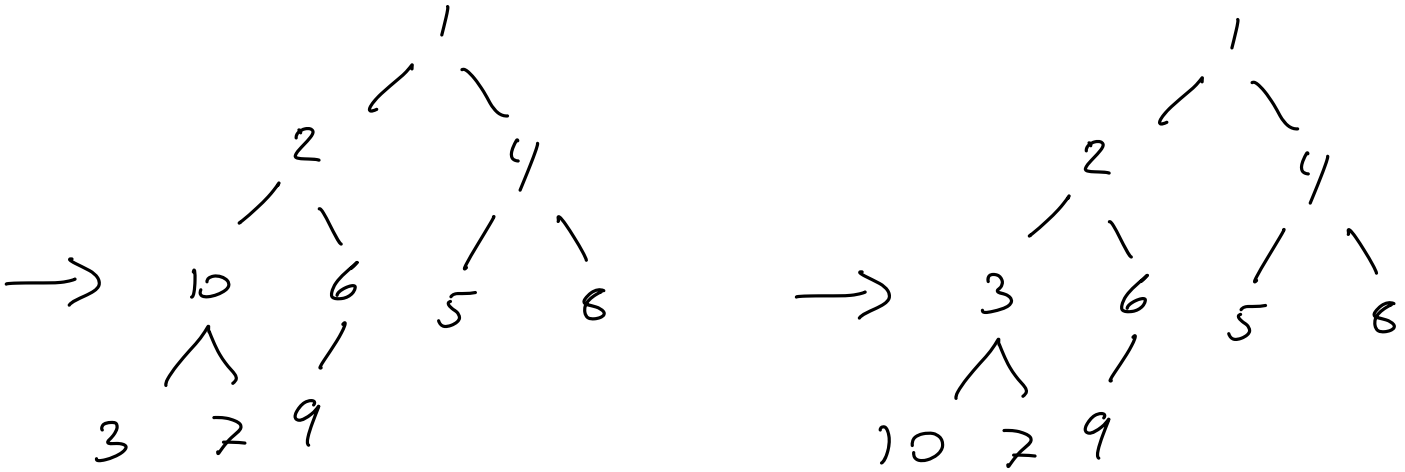




kjører metoden igjen rekursivt:
starter i siste node med minst ett barn



kjører igjen metoden rekursivt:



Har reparert Ned maksimumshaugen til en minimumshaug

7) Utskrift fra oppgave 5f

```
Verdiene i tabellen er n?:
1 2 18 10 33 100 30 200 300 54

Haugen i sortert rekkefoelge:
1  2  10  18  30  33  54  100  200  300
```