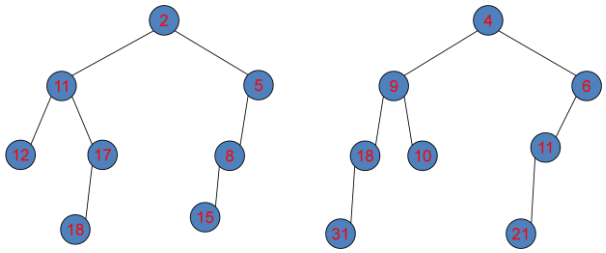
INF 4130 Oppgavesett 4, 27/09-2011 m/løsningsforslag

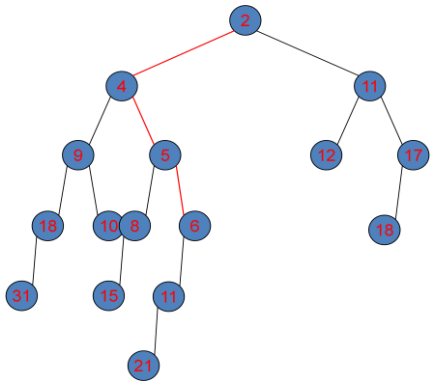
Oppgave 1

Løs oppgave 6.19 i Mark Allen Weiss *Algorithms and datastructures in Java* (INF 2220-boka).

Dette er trærne som skal merges:



Resultatet blir som følger, etter merge og snuing, med høyrestien uthevet i rødt, vi ser det er snudd i roten.



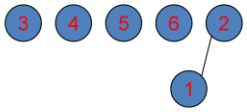
Oppgave 2

Løs oppgave 6.25 i MAW.

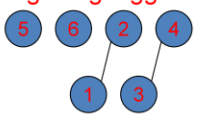
Følgende måte gir et tre som er mer venstrevridd enn bare å bygge en binærheap (som også tilfredsstiller venstrevridningskravet).

Nodene legges i kø. Numrene er ikke key, bare initielt nummer i køen.

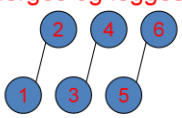
1 og 2 merges og legges bakerst i køen



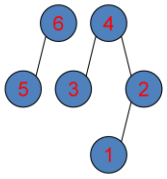
3 og 4 merges og legges bakerst i køen



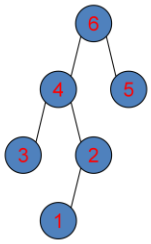
5 og 6 merges og legges bakerst i køen



(12) og (34) merges og legges bakerst i køen



(1234) og (56) merges. Ferdig.



Tidskompleksiteten blir:

$$\frac{n}{2} \cdot O(1) + \frac{n}{4} \cdot O(2) + \frac{n}{8} \cdot O(3) + \ldots = O(n)$$

$$\frac{\sum_{i=1}^{\text{Log}[2,n]} \left(\left(\frac{n}{2^i}\right) * (i-1)\right)}{\frac{-\text{Log}[2] + n\,\text{Log}[2] - \text{Log}[n]}{\text{Log}[2]}}$$

Oppgave 3

Løs oppgave 6.30 i MAW.

Det er jo i grunn opplagt, men det bør vises med et slags induksjonsbevis.

Basis: B1 har B0 som barn av roten.

Steg: Anta Bi har B0,...,B(i-1) som barn av roten.
Må vise at B(i+1) har B0,...,Bi som barn av roten.

B(i+1) lages ved å koble en Bi til roten i en annen Bi, altså vil B(i+1) bestå av den Bi som vi koblet til roten i den andre Bi'en, pluss de barn som allerede er koblet til roten i den andre Bi'en, disse er (ved antakelsen): B0,...,B(i-1). Altså har B(i+1) deltrærne B0,...,Bi.

Oppgave 4

Skriv en ikke-rekursiv implementasjon av `merge()` for leftist heaps.

Denne måten å merge på gjøres i to pass:

1) Nodene i heapenes høyre stier kan sees på som stier. Røttene er første element, nodenes .right-pekere tilsvarer next.

   Listene spleises (sortert rekkefølge). Man tar hele tiden det minste elementet, og kopierer dette over i et nytt re (en ny liste).

2) Gå igjennom den nye stien, fra slutten og mot roten (vi må ha en peker denne veien). Og sjekk om leftist-kravet er oppfylt (nullstilengdene til barna). Swap, om kravet er brutt.

Røff pseudokode kan se ca slik ut:

```
function merge(h1,h2)
   var list result
   while h1 <> nil and h2 <> nil
      if h1.key <= h2.key
         append h1.first to result        // antar .first funker.
         h1 = h1.right
      else
         append h2.first to result
         h2 = h1.right
      if h1 <> nil
         append h1 to result
      if h2 <> nil
         append h2 to result

   var elem node
   elem = result.last
   while elem <> result.first
      if elem.left.npl < elem.right.npl
         swapChildren(elem);
      elem = elem.parent                  // antar en parent-peker finnes.

   return result
end
```

Oppgave 5

Professor Pinoccio påstår at høyden i en N-noders Fibbonacci-heap er O(log N). Vis at professoren tar feil ved å vise at for ethvert positivt heltall N, så finnes det en sekvens av Fibonacci heap-operasjoner som konstruerer en heap som består av kun ett tre som er en lang kjede av N noder.

Forsøk gjerne å bruke appleten på

http://www.cs.yorku.ca/~aaw/Jason/FibonacciHeapAnimation.html

til å lage denne kjeden, og til å studere Fibonacci-heaper.

Her er det også en slags induksjon som ligger til grunn. Strengen bygges opp ved å bruke strukturen i binomialtrær som utgangspunkt:

Basis er et tre bestående av to noder: Dette kan lages ved å legge inn tre noder i en tom heap, og så kjøre en deleteMin.

Steget i induksjonen/konstruksjonen består i å legge inn tre noder med lavere key enn de som allerede ligger i heapen (kall disse a,b,c [sortert etter key]), så kjøre deleteMin, dette gir et tre med to grener, rot i dette treet er b, den ene grenen er strengen vi startet med, den andre er c, så slettes element c. Steget gjentas så mange ganger man vil.

[slutt]