## Algoritmer og datastrukturer - Avsnitt 5.2.15 - Algoritmeanalyse

### Punkt 5.2.15.3 - Gjennomsnittlig antall sammenligninger ved vellykket og mislykket søking i binære søketrær med *n* forskjellige verdier

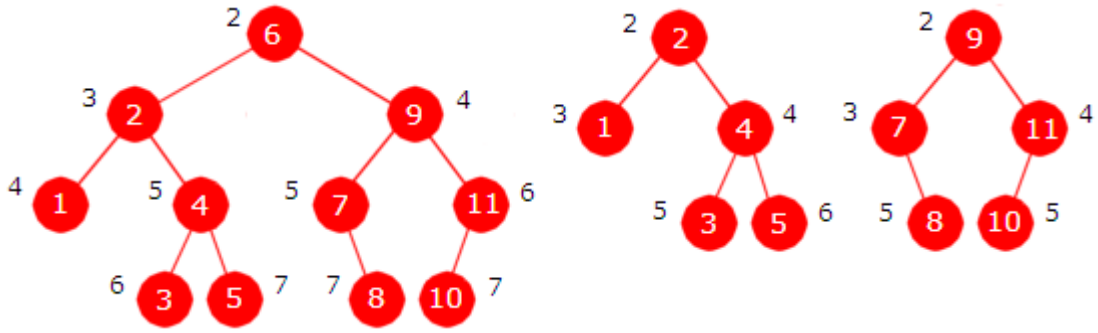
Den søkemetoden som vi skal analysere, ser slik ut:

```
public boolean inneholder(T verdi)            // søkemetoden
{
  Node<T> p = rot;                            // starter i roten

  while (p != null)                           // sjekker p
  {
    int cmp = comp.compare(verdi,p.verdi);    // sammenligner
    if (cmp < 0) p = p.venstre;               // går til venstre
    else if (cmp > 0) p = p.høyre;            // går til høyre
    else return true;                         // cmp == 0, funnet
  }

  return false;                               // ikke funnet
}
```

Figur 1 under viser til venstre et binært søketre med tallene fra 1 til 11 som verdier. Rotnodens to subtrær er satt opp som separate trær ved siden:




Figur 1

Ved siden av hver node står et tall som angir hvor mange sammenligninger som trengs for å finne verdien i noden. For eksempel blir det alltid to for å finne verdien i rotnoden – én gang for å avgjøre at verdien det søkes etter ikke er mindre og så en til for å avgjøre at den ikke er større. Figuren viser også at hvis en node er venstre barn, trengs det én sammenligning mer for å finne verdien i noden enn det trengs for å finne verdien i forelderen. Hvis derimot en node er høyre barn, trengs to sammenligninger mer for å finne nodeverdien.

Vi kaller summen av alle disse tallene for treets *sammenligningssum*. La $S_n$ være den gjennomsnittlige sammenligningssummen for binære søketrær med $n$ forskjellige verdier. Gjennomsnittet tas over de $n!$ binære søketrærne vi får ved å lage ett tre for hver permutasjon av tallene fra 1 til $n$. Det gir egentlig ikke mening å søke i et tomt tre, men vi definerer likevel at $S_0 = 0$.

La $S_n(k)$ være den gjennomsnittlige sammenligningssummen for de binære søketrærne som har $k$ , $1 \leq k \leq n$ som rotnodeverdi. For de trærne vil $S_{k-1}$ og $S_{n-k}$ være gjennomsnittlige sammenligningssummer for henholdsvis rotnodens venstre subtre og høyre subtre.

Figur 1 viser at tallet ved siden av hver node i det venstre subtreet er én mer når det er subtre enn når det er et separat tre. Tilsvarende er det to mer for høyre subtre. Sammenligningssummen er 56 for hele treet og henholdsvis 20 og 19 for de to subtrærne som separate trær. Her er $k = 6$ rotnodeverdi. Dermed er det $k - 1 = 5$ verdier i venstre subtre og $n - k = 11 - 6 = 5$ i høyre subtre. Dermed gjelder denne sammenhengen for treet i Figur 1:

(1) $56 = 2 + 20 + (6 - 1) + 19 + 2(11 - 6)$

Observasjonen i (1) gjelder treet i Figur 1, men kan generaliseres til dette:

(2) $S_n(k) = 2 + S_{k-1} + k - 1 + S_{n-k} + 2(n-k)$.

Da alle verdiene har samme sannsynlighet for å ligge i rotnoden, får vi også

(3) $S_n = \frac{1}{n}[S_n(1) + S_n(2) + \ldots . + S_n(n)]$ og ved hjelp av (1) at

(4) $S_n = \frac{2}{n}(S_0 + S_1 + \ldots . + S_{n-1}) + \frac{1}{2}(3n+1)$ .

Fremgangsmåten for å finne en formel for $S_n$ er nøyaktig den samme som for $V_n$. Derfor utelater vi de fleste detaljene her. Husk at $S_0 = 0$.

(5) $$\frac{S_n}{n+1} = \frac{3n-1}{n(n+1)} + \frac{S_{n-1}}{n} = \frac{4}{n+1} - \frac{1}{n} + \frac{S_{n-1}}{n}$$

$$= 4(\frac{1}{n+1} + \frac{1}{n} + \ldots + \frac{1}{2}) - (\frac{1}{n} + \frac{1}{n-1} + \ldots + 1)$$

$$= \frac{4}{n+1} + 4(H_n - 1) - H_n = \frac{4}{n+1} + 3H_n - 4$$

Vi ganger så med $n+1$ på begge sider av likhetstegnet og får

(6) $S_n = 3(n+1)H_n - 4n$

La $A_n$ være gjennomsnittlig antall sammenligninger, dvs. $A_n = \frac{S_n}{n}$. Ved hjelp av (7) får vi følgende formel for $A_n$:

(8) $A_n = 3(1+\frac{1}{n})H_n - 4$ , $n \geq 1$ .

Hvis verdien vi søker etter ikke ligger treet, må vi tenke litt annerledes når det gjelder treets verdier. La treet fortsatt inneholde tallene fra 1 til $n$, men nå som desimaltall, dvs. tallene 1.0 , 2.0 , 3.0 osv. Dermed kan vi søke etter verdier som ligger mellom to tall, f.eks. 1.5 som ikke ligger ikke i treet. Anta så at den verdien vi søker etter har samme sannsynlighet for å være mindre enn 1.0, være mellom 1.0 og 2.0, mellom 2.0 og 3.0, osv. oppover til å være større enn $n$. Det vil si $n + 1$ mulige tilfeller med samme sannsynlighet.

Hvis den søkte verdien ikke ligger i treet vil algoritmen gjøre at pekeren $p$ blir null. Binærtreet har $n+1$ nullpekere. Hvis vi tegner et tre og setter på ved hver nullpeker det antallet sammenligninger som algoritmen bruker for å komme dit, vil vi se et mønster av samme type som for den indre veilengden $V_n$ eller for sammenligningssummen $S_n$. La $F_n$ være summen av antallene ved hver nullpeker. Da må for eksempel $F_1 = 3$ siden det da er én node og to nullpekere med én sammenligning for den venstre og to for den høyre. Det gir egentlig ikke mening å søke i et tomt tre, men vi definerer likevel at $F_0 = 0$.

La videre $F_n(k)$ være gjennomsnittlig sum for de søketrærne som har $k$ , $1 \leq k \leq n$ som rotnodeverdi. Da gjelder

(9) $$F_n(k) = F_{k-1} + k + F_{n-k} + 2(n-k+1)$$

(10) $$F_n = \frac{1}{n}(F_n(1) + F_n(2) + \ldots . + F_n(n))$$

(11) $$F_n = \frac{2}{n}(F_0 + F_1 + \ldots . + F_{n-1}) + \frac{3}{2}(n+1)$$

(12) $$nF_n = 2(F_0 + F_1 + \ldots . + F_{n-1}) + \frac{3}{2}n(n+1)$$

Erstatt så $n$ med $n-1$ i (12)

(13) $$(n-1)F_{n-1} = 2(F_0 + F_1 + \ldots . + F_{n-2}) + \frac{3}{2}(n-1)n$$

Ved å ta differansen mellom (12) og (13) får vi

(14) $$nF_n - (n-1)F_{n-1} = 2\,F_{n-1} + 3n \ ,$$

og når ledd med $F_{n-1}$ samles blir det

(15) $$nF_n = 3n + (n+1)F_{n-1} \ .$$

Nå deler vi med $n(n+1)$ på begge sider av likhetstegnet i (15)

(16) $$\frac{F_n}{n+1} = \frac{3}{n+1} + \frac{F_{n-1}}{n}$$

Vi bruker (16) gjentatte ganger inntil $n = 2$ og bruker så at $F_1 = 3$

(17) $$\frac{F_n}{n+1} = \frac{3}{n+1} + \frac{3}{n} + \ldots + \frac{3}{2} = 3H_{n+1} - 3$$

Da alle de $n+1$ tilfellene er like sannsynlige, vil det gjennomsnittlige antallet sammenligninger $A_n$ algoritmen bruker for å avgjøre at en verdi ikke ligger i treet bli

(18) $$A_n = 3H_{n+1} - 3.$$