DMA

Ugeopgave 4

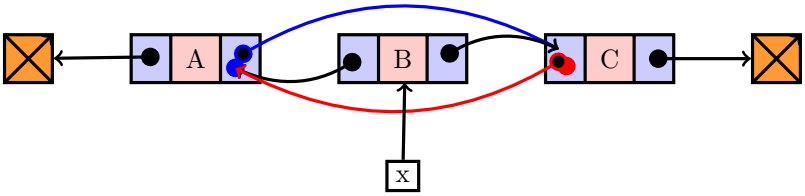
Beate Berendt Søegaard
Mathias Larsen
Simon Rotendahl

Datalogi

3. oktober 2016

Del 1

a



Da listen skal være dobbelthægtet skal en knude have tre felter. En previous, en key, (x'et) og en next felt.

b

```
(1) n = doubly-link list
(2) z = n.key
(3) x = S.head
(4)  fun F(S,z)
(5)  if z.key <= S.head then
(6)       z.next = S.head.next
(7)       z = S.head
(8)       z.prev = NULL
(9)  elif z.key => S.tail
(10)      z.next = NULL
(11)      S.tail.next = z
(12)      z.prev = S.tail
(13)      S.tail = z
(14) else
(15)      z.next = x.next
(16)      z.prev = x
(17)      x.next = z
(18) while z.key > x.key and x.next != NULL
(19)      x = x.next
```

c

I vores pseudokode har vi tre samlignings statements, hvilket køre i $\mathcal{O}(1)$. Tilsidst har vi et while-løkke som køre i $\mathcal{O}(n)$ tid. Dvs., at køretiden for pseudokoden er $T(n) = \mathcal{O}(1) + \mathcal{O}(1) + \mathcal{O}(1) + \mathcal{O}(n) = \mathcal{O}(n)$.
Udover det så da vi har flere nestede operatorer, f.eks linje 10 til linje 13 er nestede i linje 9. Det vil sige at køretiden for linje 10 - 13 skal summeres og derefter ganges på linje 9's kørertid. Det betyder dog ikke noget for vores køretid da alle nestede operationer er af konstanttid, og derved ikke øger den samlede størrelsesorden.

d

Lad listen S være tom til at starte med. Der indsættes herefter n heltal. Et af gangen. Når det sidste heltal er indsat vil S være en sorteret liste bestående af de n indsatte tal. Argumentér for at vi på denne måde har sorteret n tal i $\mathcal{O}(n^2)$ tid.

Køretiden for vores funktion kan deles op i to dele: Søge og indsættelse. Vi starter med at analysere indsættelse

Søge: Antallet af operationer for n, vil være summen af antallet af $n-1$ heltal (det skal være $n-1$ da det element der indsættes ikke skal sammenlignes med sig selv). F.eks hvis der er $n=4$ heltal der skal sættes ind i S, så vil summen være $0+1+2+3=6$ så 6 operationer er nødvendige. Antallet af operationer der er nødvendige for en given mængde heltal n, vil være lig med

$$\frac{n \times (n-1)}{2}$$

hvilket kan findes i CLRS side 1146 (dog går den sumfunktion til n og ikke n-1, derfor er vores ændret). Vi kan se fra regel S6 omkring størrelsesorden at $\frac{n\times(n+1)}{2}$ har samme størrelsesorden som $\frac{n\times(n+1)}{2} \times 2$ og derved kan vi fjerne konstanterne fra udtrykket, og derefter ganger vi parantesen ud, og får n^2+n^1. Med regel S3 kan vi vise at n^1 er af lavere størrelsesorden end n^2 og med regel S8 kan vi vise at en funktion af lavere størrelsesorden adderet med en funktion af højere størrelsesorden, vil have samme størrelsesorden som den høje funktion. Derved har vi vist at

$$\frac{n \times (n+1)}{2} \times 2 \in \Theta(n^2)$$

Indsættelse: Det tager dog også n tid at indsætte elementerne i S, og derved vil den samlede søge og indsættelses funktions størrelsesorden lyde

$$n^2 + n$$

Vi har dog lige vist at $+\ n$ ikke betyder noget for størrelsesordnen og derved ikke køretiden, så endeligt bliver køretiden $\mathcal{O}(n^2)$

Del 2

a

En liste S kan opdeles i k mindre sorterede lister, $l_1, l_2, \cdots l_k$, hvor hver af de k små lister, består af k elementer. Vi får endvidere oplyst at $k = \sqrt{n}$ er et heltal, dvs. at hver l_k skal være lige store. Knuderene i B bliver oprettet ved $k = \sqrt{n}$.

b

Da B består af k elementer tager det $\mathcal{O}(k)$ tid at løbe igennem det. k er defineret som $k = \sqrt{n}$, hvor n er antallet af elementer i S så derfor kan vi også skrive tiden det tager at løbe B igennem som $\mathcal{O}(\sqrt{n})$.

c

```
(1) G(S, B, x)
(2)  i = B.head
(3)  if i.pa > x then
(4)      x.prev = NIL
(5)      x.next = S.head
(6)      i.pa.prev = x
(7)      S.head = x
(8)   loop until i == NIL
(9)       if i.next.pa > x OR i.next == NIL then
(10)         I = i.pa
(11)         loop until I == i.next.pa OR I == NIL
(12)            if I >= x then
(13)                x.next = I
(14)                x.prev = I.prev
(15)                I.prev.next = x
(16)                I.prev = x
(17)                return S
(18)            I = I.next
```