Diskret Matematik og Algoritmer

Aflevering 4g

Aske Fjellerup
Adam Ingwersen
Peter Friborg
Hold 4

Datalogisk Institut
Københavns Universitet

October 4, 2016

Del 1

(a)

Hvilke attributer skal hver knude på listen indeholde?

Hvert objekt skal indeholde 3 attributer:

- prev (Previous objekt)
- key (Key value)
- next (Next objekt)

Algorithm 1 Pseudokode der indsætter heltallet z på listen L

function INSERTZ((L, z))
    z = Objekt
    L[head] = z.prev
    L[head+1] = z.key
    L[tail] = z.next
    z.next = NIL

(b)

I denne opgave skal der opskrives pseudokode til at indsætte et objekt på en sorteret liste.

Algorithm 2 Pseudokode, der indsætter objekt i rækkefølge på en sorteret liste

function INSERTSORT(L, z)
    if $F[i].key \leq z.key$
        $Z.next \leftarrow F[i].next$
        $Z.prev \leftarrow F[i].prev$
        $F[i].next.prev \leftarrow Z$
        $F[i].next \leftarrow Z$
        return F[i].next
    else $i \leftarrow i + 1$

(c)

Idet vi kører listen igennem et objekt ad gangen og der er n objekter, vil funktionen i værstetilfælde køre igennem n gange. Det er ikke muligt at bruge binærsøgning da det er en dobbelthægtet liste. Derfor kommer vi frem til at køretiden er $\theta(n)$

(d)

Hvis vi prøver at bruge vores funktion på den omvendt sorterede liste vil den køre igennem $n^2/2$ hvilket er netop $O(n^2)$.

Del 2

(a)

Opdeling af S

I denne opgave inddeler vi den hægtede liste, S af længden n, i k mindre hægtede lister, $l_1, l_2...l_k$ af længden k. k er defineret ved $k = \sqrt{n}$.
Listen S opdeles ved at dele listen i $\sqrt{n}$ dele elementer i hver inddeling.

Oprettelse af B

Listen B skal have k elementer med en peger. Pegerne peger mod det første element af henholdvist, $i_1, i_2...i_k$ for elementerne gående mod k.

Algorithm 3 Pseudokode, der opretter B via opdeling af S

---

function CREATEB(S)
  Lad B[0..$\sqrt{n}$] være en liste
  Lad S[0..n-1] være en sorteret liste
  $k = \sqrt{n}$
  $i \leftarrow 0$
  while $i \leq k$
    Lad L_i være en liste af længde k
    $L_i \leftarrow S[(i \cdot k)..(((i+1) \cdot k) - 1)]$
    $B[i] \leftarrow L_i$
      if $i = 0$
      $B.head \leftarrow NIL$
    $B[i-1].next \leftarrow B[i]$
    $B[i].prev \leftarrow B[i-1]$
    $i \leftarrow i + 1$

Oprettelse af B

Køretiden vil være $\theta(\sqrt{n})$, det tager $c_1\sqrt{n}$ tid at opdele S i k dele da S er en dobbelthægtet liste. B tager $c_2\sqrt{n}$ tid at oprette da der 3 pegere pr element der skal defineres og der er k elementer. tiden bliver derfor

$$\underline{\underline{c\sqrt{n} = \theta(\sqrt{n})}}$$

(b)

Der findes n elementer i S. Det betyder at der i den mindre liste B, vil være $\sqrt{n}$ elementer at løbe igennem. Vi kommer derfor frem til at køretiden vil være O($\sqrt{n}$).

(c)

Her opskrives en funktion, G, til at indsætte heltallet x i listen S, ved brug af den mindre liste B.

Algorithm 4 Pseudokode, der indsætter objekt i rækkefølge på listen S, ved brug af hjælpelisten B

```
     function G(S, B, x)
2:       i ← 0
         repeat
4:       if B[i].pa ≥ x
             repeat
6:           if S[i].key ≤ x
                 x.next ← S[i].next
8:               x.prev ← S[i]
                 S[i].next.prev ← x
10:              S[i].next ← x
                 return S[i].next
12:          else i ← i − 1
         else i ← i + 1
```

Det ses i linje 4 og 6 at vi først søger efter om pegeren på B er større end x. Herefter skal vi finde den helt rigtige placering på S og derfor spørger vi om S[i].key er mindre end, for ellers er vi nødt til at gå en lille smule tilbage på listen, for at finde det helt rigtige sted.