DMA Ugeopgave 4

Carl Dybdahl, Patrick Hartvigsen, Emil Søderblom

October 3, 2016

1 Del 1

1.1 a)

Et objekt i listen `S` skal indeholde 3 attributter, `pre`, `key`, og `next`. `pre` skal pege på den tidligere knude, hvor `next` skal pege på den næste knude, og `key` skal være elementet ved knudens position. Hvis det er det første objekt i listen skal `pre = NIL` og hvis det er det sidste skal `next = NIL`.

Pseudokode:

```
def singleton(z):
        return new [ prev = NIL; next = NIL; key = z ]
```

1.2 b)

```
def F(S, z):
        var prev = NIL
        var target = S
        while (target != NIL && target.key < z):
                prev = target
                target = target.next
        var node = singleton(z)
        if (target == NIL):
                prev.next = node
                node.prev = prev
        else:
                if (target.prev != NIL):
                        target.prev.next = node
                        node.prev = target.prev
                target.prev = node
                node.next = target
```

1.3 c)

Hvis `z` skal placeres sidst i listen er funktionen nødt til at iterere gennem hele listen. Dette tager $\Theta(n)$ tid.

1.4 d)

Worst-case tilfældet er, at vi ved hver indsættelse skal iterere gennem hele listen - hvis dette er nødvendigt kommer vi til at bruge $\Theta(n^2)$, da vi n gange skal iterere gennem en liste med gennemsnitlig størrelse $n/2$. Dette sker hvis vi hver gang skal indsætte et element der er større end alle de andre, altså hvis vi indsætte elementerne i sorteret rækkefølge.

I visse tilfælde, for eksempel hvis elementerne bliver indsat i omvendt sorteret rækkefølge, kan vi gøre det hurtigere end $\Theta(n^2)$, men det kan ikke blive langsommere. Dermed er kørselstiden $O(n^2)$ på sortering af lister med denne metode.

2 Del 2

2.1 a)

Vi laver en liste `B` og en løkke der kører gennem hver underliste l_i i `S`. Denne løkke itererer k gange, og hver gang indsætter vi i `B` en knude med en pejer på det tilsvarende element i `S`. Efter vi har indsat knuden gennemløber vi en anden løkke k gange, hvor vi bevæger os frem til næste underliste.

2.2 b)

Da `B` har en længde på $\sqrt{n}$ vil det tage $\Theta(\sqrt{n})$.

2.3 c)

```
1  def G (S,B,x):
2          var prev = NIL
3          var target = B
4          while (target != NIL && target.pa.key < z)):
5                  prev = target
6                  target = target.next
7          if (target == NIL):
8                  F(prev.pa, x)
9          elseif (prev == NIL):
10                 var node = new [
11                         prev = NIL,
12                         next = target.pa,
13                         key = x
14                 ]
15                 target.pa.prev = node
16         else
17                 F(prev.pa, x)
```

2.4 d)

Hvis en sorteret indsættelse kan gøres på $O(\sqrt{n})$ tid, kan man bare indsætte hvert element der skal sorteres, hvilket giver en samlet kørselstid på $O(n\sqrt{n})$. Dog skal man sørge for intelligent at vedligeholde `B`-listen, for ellers kan man ikke gentagende gange indsætte på $O(\sqrt{n})$ tid.