Diskret Matematik og Algoritmer 2016

Ugeopgave 4

Rasmus F, Aiyu L & Frederik KM

Indhold

1 Del 1

1.1 (a)

1.1.1

Den skal have:
container til dataen der skal oplærgers og 2 pointers som peger på det forrige og næste ellement eller til null

1.1.2

```
List::List (int z) : length (1) {
  head = new node(z)
  tail = head
}
```

1.2 (b)

Vi går ud fra at S er sorteret i ikke-faldene order

```
void node::insertBefore (value v) {
  if (prev == null) {
    prev = new node(v)
    prev.next = &this
  } else {
    temp = new node(v)
    temp.prev = prev
    temp.next = &this
    prev.next = temp
    prev = temp
  }

void node::insertAfter (value v) {
  if (next == null) {
    next = new node(v)
    next.prev = &this
  } else {
    temp = new node(v)
    temp.prev = &this
    temp.next = next
    next.prev = temp
    next = temp
  }
```

```
F(S, z) {
  i = S.head
  if (z < i.value) {i.insertBefore(z)}
  else
    while (i != null && z < (i.next).value())
      i = i.next
    i.prev.insertAfter( z)
}
```

1.3 (c)

Alle linjerne i F kører i konstant tid, med undtagelse af while-løkken. Den kører fra head til - i værst tænklige tilfælde - tails; altså over de n elementer, og den gør det højest 1 gang. Altså er kørertiden $O(n)$.

1.4 (d)

Hver gang et tal bliver indsat, skal den går over elementerne i listen en gang:

$$1 + 2 + ... + n - 1 + n = \frac{n^2 + n}{2}$$

Udtrykket oven over er $O(n^2)$

2 Del 2

2.1 (a)

En måde man kunne gører dette på er at gennemløbe S, og for hvert k'te element opretter man en knude som man hægter på B. Kørertiden ville således være $O(n)$.

Dette kræver dog at man kender længden på S. Enten ved at S er implementeret at længden er gemt i et medlem af S, ellers er man nødt til at gennemløbe S for at finde længden. Køretiden vil stadig være $O(n)$.

2.2 (b)

Det ville tage $O(\sqrt{n})$, da dette er antallet af elementer i B.

2.3 (c)