IDATT2101 – Øving 1

Algoritmen

Til denne øvingen kom jeg fram til denne algoritmen:

```
10
11  std::pair<int, int> optimum_stock_plan_nSq(int* stock, int count) {
12      if (count < 2)
13          return std::pair<int, int>(-1, -1);
14
15      int buy_idx  = -1;
16      int sell_idx = -1;
17      int current_profit = 0;
18
19      for (int buy = 0; buy < count; buy++) {
20          int temp_profit = 0;
21
22          for (int sell = buy; sell < count; sell++) {
23              temp_profit += stock[sell];
24
25              if (current_profit < temp_profit) {
26                  current_profit = temp_profit;
27                  buy_idx = buy;
28                  sell_idx = sell;
29              }
30          }
31      }
32
33      std::pair<int, int> result(buy_idx, sell_idx);
34
35      return result;
36  }
37
```

Kompleksitetsanalyse

Denne algoritmen benytter en såkalt «brute-force»-metode ved å gå gjennom alle mulige kombinasjoner av kjøps- og salgsdato. Den benytter en nestet for-løkke for dette. Den ytterste itererer over alle elementer én gang. Den innerste itererer ikke over alle elementer, men kun alle elementene fra elementet definert av den ytterste løkken.

Det samlede antallet iterasjoner disse løkkene gjennomfører, er i ordenen av et trekanttall, uttrykt på denne måten: $\frac{n(n-1)}{2}$. Dette uttrykket kan skrives om slik: $\frac{n(n-1)}{2} = \frac{1}{2}(n^2 - n)$. Leddet n^2 inni parentesen avslører at kompleksiteten til algoritmen avgrenses av $C * n^2$ der C er en vilkårlig konstant som gjør at dette uttrykket stemmer: $0 \leq \frac{1}{2}(n^2 - n) \leq C * n^2$.

Derfor er algoritmens kompleksitet avgrenset av $\Omega(n^2)$.

Tidsmålinger av algoritmen

Antall elementer	Måling 1	Måling 2	Måling 3
1000	0.000617s	0.000604s	0.000597s
10000	0.060497s	0.059902s	0.060694s
100000	5.946290s	5.996751s	6.019439s

Disse målingene er gjennomført i debug-modus, uten kompilator-optimaliseringer.

De forskjellige målingene benytter et antall elementer som stiger med en faktor på 10. Hvis kompleksitetsanalysen vår er riktig, vil vi se at tiden algoritmen bruker vil øke med en faktor på 100, siden $\Omega((10)^2) = \Omega(100)$. Denne antagelsen gjenspeiles i målingene, eksempelvis måling 1:

$$0.000617\text{s} * 100 \approx 0.060497\text{s}$$

$$0.060497\text{s} * 100 \approx 5.946290\text{s}$$

De tre målingene er nokså like, som viser at denne tidsberegningen er konstant.