IDATT2101 Øving 2

Rekursiv programmering

Trang Minh Duong

Fra oppgaven antar vi at n er alltid positiv.

Metode 1

$$n \cdot x = \begin{cases} x & \text{hvis } n = 1 \\ x + (n-1) & \text{hvis } n > 1 \cdot x \end{cases}$$

For metode 1, sjekker jeg betingelsen om n er lik eller ulik 1 (2 cases).

```
1 int recursion1 (int n, int x) {
2     if (n == 1) {
3         return n, x;
4     }
5     return x + recursion1(n-1,x);
6 }
```

Listing 1: Rekursjon metode 1

Metode 2

$$n \cdot x = \begin{cases} x & \text{hvis } n = 1 \\ \frac{n}{2} \cdot (x + x) & \text{hvis n er partall} \\ x + \frac{n-1}{2} \cdot (x + x) & \text{hvis n er oddetall} \end{cases}$$

For metode 2, sjekker jeg betingelsen om n er lik 1, om n er et partall eller om n er et oddetall (3 cases).

```
1 int recursion2(int n, int x) {
2     if (n == 1) {
3         return n, x;
4     }
5     if (n%2 == 0) {
6         return recursion2(n/2, x+x);
7     }
8     return x + recursion2((n-1)/2, x+x);
9 }
```

Listing 2: Rekursjon metode 2

Tidsmålinger

Under er tabellen for gjennomsnitsstidsmålinger i nanosekund (5 ganger) for x = 2 og ulike verdi av n (10 - 43 339). Hvorfor akkurat 43 339 er fordi det er det største tallet programmet kan kjøre uten å få minnesegmentsfeil:

n	**Metode 1**	**Metode 2**
10	580 ns	140 ns
100	1680 ns	240 ns
⋮	⋮	⋮
10 000	667 400 ns	240 ns
43 339	1 626 800 ns	240 ns

Tabell 1: Gjennomsnittstidsmålingene for metodene

Disse tidsmålingene stemmer med forventningen og formelene til metodene. I metode 1 kaller recursion1(n - 1, x) helt til n = 1. For n = 43 339 betyr det at ∼43 339 funksjonskall med like mange retur-hopp og addisjoner. For metode 2 konvergerer tidsmålingene mot 240 ns når $n \to \infty$. Dette er fordi for hver funksjonskall i metode 2 halveres n-verdien, og addisjon skjer bare når n er et oddetall. Dette er mye mindre kostbare på minnet sammenlignes med metode 1.

Kompleksitetsanalyse

La n = 4, x = 5 Hvis vi bruker metode 1, får vi følgende:

$$4 \cdot 5 \to 5 \cdot (3 + 5) \to 5 + (2 \cdot 5) \to 5 + (1 \cdot 5) \to 5 \tag{1}$$

Her ser vi at ved bruk av metode 1 benytter vi totalt 4 steg, som er det samme som n-verdien. Dette betyr at metode 1 er lineær og tidskompleksiteten for metode 1 er $\Theta(n)$

Vi prøver det samme med metode 2.

$$4 \cdot 5 \to 2 \cdot (5 + 5) \to 1 + (10 + 10) \to 20 \tag{2}$$

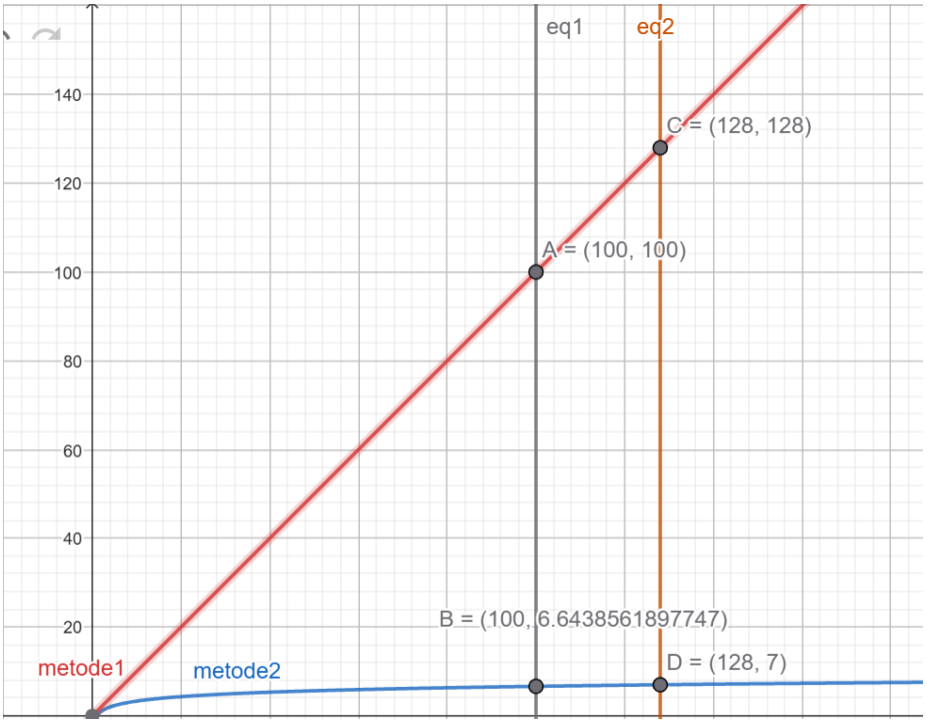
Metode 2 benytter bare 3 steg. Vi prøver igjen med en større n = 100 og x = 2:

$$\begin{aligned} 100 \cdot 2 \to 50 \cdot 4 \to 25 \cdot 8 \to 8 + 12 \cdot 16 = 8 + 192 \\ 192 \to 6 \cdot 32 \to 3 \cdot 64 \to 64 + 1 \cdot 128 \to 128 \end{aligned} \tag{3}$$

For n = 100, x = 2 med bruk av metode 2 benytter vi 7 steg. Hvis vi gjør det samme for n = 128, vil det være 8 steg (inkludert base casen hvor n = 1). Vi kan trekke $128 = 2^7$ og $100 \approx 2^{6.64}$. Dermed er metode 2 logaritmisk og tidskompleksiteten for metode 2 er

$\Theta(\log_2 n)$ eller $\Theta(\log n)$.

Vi sjekker antall stegene ved å plotte grafer for kompleksiteter.



Figur 1: Tidskompleksiteter til metodene

Vi kan se at grafene stemmer med funksjonene og analysen. Dermed er tidskompleksitetene $\Theta(n)$ for metode 1 of $\Theta(\log n)$ for metode 2.