Usman Ghafoorzai

Rekursiv programmering

Kildekode

```
public class Oving2 {

  // Tidskompelkitet θ(n) - lineær rekursjon
  private static double metode1(double x, int n) {
    if (n == 1) {
      return x;
    } else  {
      return x * metode1(x, n - 1);
    }
  }
  public  static double Metode1(double x, int n) {
    return metode1(x, n);
  }

  // Tidskompleksitet θ(log n) - eksponentiell rekursjon
  private  static double metode2(double x, int n) {
    if (n == 1) {
      return x;
    } else if (n % 2 == 0) {
      return metode2(x, n / 2) * metode2(x, n / 2);
    } else {
      return x * metode2(x, n - 1);
    }
  }
  public static double Metode2(double x, int n) {
    return metode2(x, n);
  }

  // Tidskompleksitet θ(1) - konstant tid
  private  static double metode3(double x, int n) {
    return Math.pow(x, n);
  }
  public static double Metode3(double x, int n) {
    return metode3(x, n);
  }

  public static void main(String[] args) {
    System.out.println("Metode 1:  " + Metode1(5, 11));
    System.out.println("Metode 2:  " + Metode2(5, 11));
    System.out.println("Metode 3:  " + Metode3(5, 11));
```

Usman Ghafoorzai

Beregning av 5^{11}

```
C:\Users\47968\.jdks\openjdk-21.0.2\bin\java.exe
Metode 1: 4.8828125E7
Metode 2: 4.8828125E7
Metode 3: 4.8828125E7
```

Tidtaking for $n = 1000$, $n = 3000$, $n = 5000$

```
C:\Users\47968\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent

For n = 1000:
Gjennomsnittlig kjøretid for Metode 1: 18377 nanosekunder
Gjennomsnittlig kjøretid for Metode 2: 12874 nanosekunder
Gjennomsnittlig kjøretid for Metode 3: 459 nanosekunder

Process finished with exit code 0
```

```
C:\Users\47968\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:

For n = 3000:
Gjennomsnittlig kjøretid for Metode 1: 44430 nanosekunder
Gjennomsnittlig kjøretid for Metode 2: 24686 nanosekunder
Gjennomsnittlig kjøretid for Metode 3: 403 nanosekunder

Process finished with exit code 0
```

```
C:\Users\47968\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:

For n = 5000:
Gjennomsnittlig kjøretid for Metode 1: 72918 nanosekunder
Gjennomsnittlig kjøretid for Metode 2: 27972 nanosekunder
Gjennomsnittlig kjøretid for Metode 3: 419 nanosekunder

Process finished with exit code 0
```

Analyse av tidsmålinger vha. asymptotisk analyse

De tre metodene har respektivt tidskompleksiteter $\theta(n)$, $\theta(\log n)$ og $\theta(1)$.

Metode 1 har $\theta(n)$ siden for hver økning i n må metoden utføre en ekstra multiplikasjon. Dette innebærer at kjøretiden øker lineært med størrelsen på input.

Metode 2 har $\theta(\log n)$ siden metoden deler problemet i to ved hver iterasjon (oddetall og partall). Dette betyr at kjøretiden øker logaritmisk med størrelsen på input, og derfor øke mye saktere enn for metode 1. Dette kan observeres i «Tidtaking for $n = 1000$, $n = 3000$, $n = 5000$».

For metode 3 er kjøretiden konstant, uavhengig av størrelsen på input – siden metoden bruker innebygde funksjoner for å beregne potensen. Derfor er tidskompleksiteten $\theta(1)$.

Disse tidskompleksitetene er i samsvar med tidsmålingene i koden. Metode 1 har den lengste kjøretiden, etterfulgt av metode 2. Metode 3 har den korteste kjøretiden og er så å si konstant. Dette er forventet, gitt tidskompleksitetene deres.