ANALISIS ALGORITMA

Disusun Oleh :

Aprischa Nauva Miliantari

140810180063

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJRAN

2020

1. Untuk $T(n) = 2 + 4 + 8 + 16 + \cdots + n^2$, tentukan nilai C, $f(n)$, n_0, dan notasi Big-O sedemikian sehingga $T(n) = O(f(n))$ jika $T(n) \leq C\ untuk\ semua\ n \geq n_0$

   Jawab :

   $T(n) = 2 + 4 + 8 + 16 + \ldots + 2^n$

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

   $T(n) = 2^{n+1} - 2 = O(2^n)$

   $T(n) \leq cf(n)$

   $2^{n+1} - 2 \leq c.2^n$

   $2.2^n - 2 \leq c.2^n$

   $2 - \frac{2}{2^n} \leq c$

   $n_o = 1$

   $2 - \frac{2}{2} \leq c$

   $c \geq 1$

2. Buktikan bahwa untuk konstanta-konstanta positif p, q, dan r: $T(n) = pn^2 + qn + r$ adalah $O(n^2)$, $\Omega(n^2)$, $\Theta(n^2)$

   Jawab :

   $T(n) = pn^2 + qn + r$

   - $O(n^2)$ -> Big O

     $T(n) \leq c.f(n)$

     $pn^2 + qn + r \leq c.n^2$

     $p + \frac{q}{n} + \frac{r}{n^2} \leq c$

     $n_o = 1$

     $p + q + r \leq c$

     $c \geq p + q + r$

   - $\Omega(n^2)$ -> Big Ω

     $T(n) \geq c.f(n)$

     $pn^2 + qn + r \geq c.n$

     $p + q + \frac{r}{n} \leq c$

     $n_o = 1$

     $p + q + r \geq c$

$c \leq p + q + r$

- Karena Big O = Big $\Omega = n^2$ maka Big $\Theta = n^2$

3. Tentukan waktu kompleksitas asimptotik (Big-O, Big-Ω, dan Big-Θ) dari kode program berikut:

```
for k ← 1 to n do
    for i ← 1 to n do
        for j ← to n do
            wij ← wij or wik and wkj
        endfor
    endfor
endfor
```

Jawab :

```
for k <- i to n do
   for i <- i to n do
      for j <- to n do
         wij <- wij or wik or wkj => n.n.n
      endfor                                T(n) = n^3
   endfor
endfor
```

- Big O
  $n^3 \leq c.\ n^3$
  $1 \leq c$
  $c \geq 1$

- Big Ω
  $n^3 \leq c.\ n^3$
  $c \geq 1$

- Big Θ
  Big O = Big Ω maka Big $\Theta = \Theta(n^3)$

4. Tulislah algoritma untuk menjumlahkan dua buah matriks yang masing-masing berukuran n x n. Berapa kompleksitas waktunya T(n)? dan berapa kompleksitas waktu asimptotiknya yang dinyatakan dalam Big-O, Big-Ω, dan Big-Θ?

Jawab :

Algoritma penjumlahan matiks n x m

```
for i <- i to n do
      for j <- i to n do
```

```
        mij <- aij + bij => n.n
      endfor                  T(n) = n^2
    endfor
```

- Big O
  $n^2 \leq c.\ n^2$
  $1 \leq c$
  $c \geq 1$

- Big Ω
  $n^2 \leq c.\ n^2$
  $1 \geq c$
  $c \leq 1$

- Big O = Big Ω maka Big Θ = $\Theta(n^2)$

5. Tulislah algoritma untuk menyalin (copy) isi sebuah larik ke larik lain. Ukuran elemen larik adalah n elemen. Berapa kompleksitas waktunya T(n)? dan berapa kompleksitas waktu asimptotiknya yang dinyatakan dalam Big-O, Big-Ω, dan Big-Θ?

   Jawab :

   Algoritma menyalin lirik

   ```
   for i <- i to n do
       ai <- bi          => n = T(n)
   endfor
   ```

   - Big O
     $n \leq cn$
     $1 \leq c$
     $c \geq 1$

   - Big Ω
     $n \leq cn$
     $1 \geq c$
     $c \leq 1$

   - Big O = Big Ω maka Big Θ = $\Theta(n)$

6. Diberikan algoritma Bubble Sort sebagai berikut:

```
procedure BubbleSort(input/output a1, a2, ..., an : integer)
{ Mengurut tabel integer TabInt[1..n] dengan metode pengurutan bubble-
sort
  Masukan: a1, a2, ..., an
  Keluaran: a1, a2, ..., an (terurut menaik)
}
Deklarasi
   k : integer      { indeks untuk traversal tabel }
   pass : integer { tahapan pengurutan }
   temp : integer { peubah bantu untuk pertukaran elemen tabel }
Algoritma
   for pass ← 1 to n - 1 do
     for k ← n downto pass + 1 do
       if ak < ak-1 then
          { pertukarkan ak dengan ak-1 }
          temp ← ak
          ak ← ak-1
          ak-1 ← temp
       endif
     endfor
   endfor
```

a. Hitung berapa jumlah operasi perbandingan elemen-elemen tabel!
b. Berapa kali maksimum pertukaran elemen-elemen tabel dilakukan?
c. Hitung kompleksitas waktu asimptotik (Big-O, Big-Ω, dan Big-Θ) dari algoritma Bubble Sort
tersebut!

Jawab :

a. Jumlah operasi perbandingan
   $1 + 2 + 3 + 4 + \ldots + (n-1)$
   $= \frac{n(n-1)}{2}$ kali
b. Berapa kali maksimum pertukaran elemen-elemen tabel dilakukan
   $= \frac{n(n-1)}{2}$ kali
c. Hitung kompleksitas
   - Bestcase (semua telah terurut)
   $\frac{n(n-1)}{2}$ kali, $T\ min(n) = \frac{n(n-1)}{2} = \frac{n^2-n}{2}$
   - Worstcase (semua data harus ditukar)
   Perbandingan -> $\frac{n(n-1)}{2}$
   Memasukkan nilai -> $\frac{3n(n-1)}{2}$
   $Tmax(n) = \frac{4n(n-1)}{2}$
   $= 2n^2-2n$
   ➔ Big O
   $2n^2-2n \leq cn^2$

$$2 - \frac{2}{n} \leq c$$

$n_0 = 1 \rightarrow 2-2 \leq c$

$$c \geq 0$$

➔ Big Ω

$$\frac{n^2 - n}{2} \geq cn^2$$

$$\frac{1}{2} - \frac{1}{2n} \geq c$$

$n_0 = 1 \rightarrow \frac{1}{2} - \frac{1}{2} \geq c$

$$c \leq 0$$

7. Untuk menyelesaikan problem X dengan ukuran N tersedia 3 macam algoritma:
    1. Algoritma A mempunyai kompleksitas waktu O(log N)
    2. Algoritma B mempunyai kompleksitas waktu O(N log N)
    3. Algoritma C mempunyai kompleksitas waktu O(N)

    Untuk problem X dengan ukuran N=8, algoritma manakah yang paling cepat? Secara asimptotik, algoritma manakah yang paling cepat?

    Jawab :

    a. Algoritma A -> O (log N)
    b. Algoritma B -> O (N log N)
    c. Algoritma C -> O (N^2)

        Jika N = 8 mana Algoritma yang paling efektif?

        d. O (log 8) = O (^{3}log 2)
        e. O (^{8}log 8) = O (^{24}Log 2)
        f. O (8^2) = O (64)

        Yang paling efektif adalah Algoritma A karena semakin kecil O() semakin efektif.

8. Algoritma mengevaluasi polinom yang lebih baik dapat dibuat dengan metode Horner berikut:

$$p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \ldots + x(a_{n-1} + a_n x)))\ldots))$$

```
function p2(input x : real) → real
{ Mengembalikan nilai p(x) dengan metode Horner}

Deklarasi
  k : integer
  b1, b2, ..., bn : real

Algoritma
  bn ← an
  for k ← n - 1 downto 0 do
     bk ← ak + bk+1 * x
  endfor
  return b0
```

Hitunglah berapa operasi perkalian dan penjumlahan yang dilakukan oleh algoritma diatas, Jumlahkan kedua hitungan tersebut, lalu tentukan kompleksitas waktu asimptotik (Big-O)nya. Manakah yang terbaik, algoritma p atau p2?

Jawab :

Operasi memasukan nilai

- bn <- an 1 kali
- bk <- ak + bk +1 + x n kali

T(n) = n + 1
O(n) = untuk p^2
Algoritma P
Penjumlahan n kali
Perkalian n kali
T(n) = 2n

P^2 lebih baik dari P