

ANALISIS ALGORITMA



Disusun oleh :

Meira Dwiana Anjani

140810180015

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

2020

Latihan Analisa

Minggu ini kegiatan praktikum difokuskan pada latihan menganalisa, sebagian besar tidak perlu menggunakan komputer dan mengkode program, gunakan pensil dan kertas untuk menjawab persoalan berikut!

1. Untuk $T(n) = 2 + 4 + 8 + 16 + \dots + n^2$, 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 $n \geq n_0$

$$\begin{aligned} 1) T(n) &= 2 + 4 + 8 + 16 + \dots + 2^n \\ &= \frac{2(2^n - 1)}{2 - 1} = 2(2^n - 1) = 2^{n+1} - 2 \\ T(n) &= 2^{n+1} - 2 = O(2^n) \\ T(n) &\leq C f(n) \\ 2^{n+1} - 2 &\leq C \cdot 2^n \\ 2 \cdot 2^n - 2 &\leq C \cdot 2^n \\ 2 - \frac{2}{2^n} &\leq C \end{aligned} \quad \left. \begin{array}{l} \rightarrow n_0 = 1 \\ 2 - \frac{2}{2} \leq C \\ C \geq 1 \end{array} \right\}$$

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

$$\begin{aligned} 2) T(n) &= pn^2 + qn + r \\ \rightarrow O(n^2) &\rightarrow \text{Big } O \\ T(n) &\leq C \cdot f(n) \\ pn^2 + qn + r &\leq C \cdot n^2 \\ p + \frac{q}{n} + \frac{r}{n^2} &\leq C \end{aligned} \quad \left. \begin{array}{l} \rightarrow n_0 = 1 \\ p + q + r \leq C \\ C \geq p + q + r \end{array} \right\}$$
$$\begin{aligned} \rightarrow \Omega(n^2) &\rightarrow \text{Big } \Omega \\ T(n) &\geq C \cdot f(n) \\ pn^2 + qn + r &\geq C \cdot n^2 \\ p + \frac{q}{n} + \frac{r}{n^2} &\geq C \end{aligned} \quad \left. \begin{array}{l} \rightarrow n_0 = 1 \\ p + q + r \geq C \\ C \leq p + q + r \end{array} \right\}$$

\rightarrow karena $\text{Big } O = \text{Big } \Omega = n^2$
maka $\text{Big } \Theta = n^2$

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 ← 1 to n do
       $w_{ij} \leftarrow w_{ij} \text{ or } w_{ik} \text{ and } w_{kj}$ 
    endfor
  endfor
endfor

```

3) for k ← 1 to n do
 for i ← 1 to n do
 for j ← 1 to n do
 $w_{ij} \leftarrow w_{ij} \text{ or } w_{ik} \text{ or } w_{kj} \Rightarrow n \cdot n \cdot n$
 endfor
 endfor
 endfor

* Big O
 $n^3 \leq c \cdot n^3$
 $1 \leq c$
 $c \geq 1$

* Big Ω
 $n^3 \geq c \cdot n^3$
 $c \leq 1$

* Big Θ
 Big O = Big Ω
 maka Big Θ pun
 sama $\Theta(n^3)$

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

4) Algoritma penjumlahan matriks $n \times n$

```

for i ← 1 to n do
  for j ← 1 to n do
     $m_{ij} \leftarrow a_{ij} + b_{ij} \Rightarrow n \cdot n$ 
  endfor
endfor

```

$T(n) = n^2$

* Big O
 $n^2 \leq c \cdot n^2$
 $1 \leq c$
 $c \geq 1$

* Big Ω
 $n^2 \geq c \cdot n^2$
 $1 \geq c$
 $c \leq 1$

* Big O = Big Ω
 maka Big Θ nilainya
 sama seperti Big O =
 Big Ω yaitu $\Theta(n^2)$

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 ?

5) Algoritma menyalin larik

```

for i ← 1 to n do
  a_i ← b_i    ⇒ n = T(n)
endfor

```

* Big O * Big-Ω

$n \leq cn$ $n \geq cn$

$1 \leq c$ $1 \geq c$

$c \geq 1$ $c \leq 1$

* Jika Big O = Big-Ω maka Big Θnya sama yaitu $\Theta(n)$

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

```

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

6) ① jumlah operasi perbandingan
 $1 + 2 + 3 + 4 + \dots + (n-1)$
 $= \frac{n(n-1)}{2}$ kali

② berapa kali maksimum pertukaran elemen-elemen tabel dilakukan
 $\frac{n(n-1)}{2}$ kali

③ hitung kompleksitas

- Best case (semua telah terurut)
 $\frac{(n-1)n}{2}$ kali, $T_{\min}(n) = \frac{n(n-1)}{2} = \frac{n^2 - n}{2}$
- Worst case (semua data harus ditukar)
 Perbandingan $\rightarrow \frac{n(n-1)}{2}$
 memasukan nilai $\rightarrow \frac{3n(n-1)}{2}$
 $T_{\max}(n) = \frac{4n(n-1)}{2}$
 $= 2n^2 - 2n$

Big O

$$2n^2 - 2n \leq Cn^2 \quad \left\{ \begin{array}{l} \text{Big-O} \\ \frac{n^2 - n}{2} \gg Cn^2 \\ 2 - \frac{2}{n} \leq C \\ n_0 = 1 \rightsquigarrow 2 - 2 \leq C \\ C \geq 0 \end{array} \right. \quad \left\{ \begin{array}{l} \text{Big-O} \\ \frac{n^2 - n}{2} \gg Cn^2 \\ \frac{1}{2} - \frac{1}{2n} \gg C \\ n_0 = 1 \rightsquigarrow \frac{1}{2} - \frac{1}{2} \gg C \\ C \leq 0 \end{array} \right.$$

7. Untuk menyelesaikan problem X dengan ukuran N tersedia 3 macam algoritma:

- Algoritma A mempunyai kompleksitas waktu $O(\log N)$
- Algoritma B mempunyai kompleksitas waktu $O(N \log N)$
- 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?

7) ① Algoritma A $\rightarrow O(\log N)$
 ② Algoritma B $\rightarrow O(N \log N)$
 ③ Algoritma C $\rightarrow O(N^2)$

Jika $N=8$ mana Algoritma yang paling efektif?

- $O(\log 8) = O(3 \log 2)$
- $O(8 \log 8) = O(24 \log 2)$
- $O(8^2) = O(64)$

yang paling efektif adalah Algoritma A
 Karena semakin kecil OC) semakin efektif

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

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

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

Deklarasi

k : integer
 b₁, b₂, ..., b_n : real

Algoritma

b_n ← a_n
 for k ← n - 1 downto 0 do
 b_k ← a_k + b_{k+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?

8) Operasi memasukkan nilai

- b_n ← a_n 1 kali
- b_k ← a_k + b_{k+1} * x n kali

$$T(n) = n + 1$$

$$O(n) = \text{untuk } p^2$$

Algoritma p

Penjumlahan n kali

Perkalian n kali

$$T(n) = 2n$$

p² lebih baik dari p