

# **LAPORAN PRAKTIKUM 3**

## **MATA KULIAH ANALISIS ALGORITMA**



**Disusun Oleh:**

**Falah Rizqi Abdullah Fairuz  
140810180069**

**PROGRAM STUDI S-1 TEKNIK INFORMATIKA  
DEPARTEMEN ILMU KOMPUTER  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS PADJADJARAN**

### Worksheet 3

1. Untuk  $T(n) = 2 + 4 + 6 + 8 + 16 + \dots + n^2$ , 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  $n \geq n_0$

1. Untuk  $T(n) = 2 + 4 + 6 + 8 + 16 + \dots + n^2$ , 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  $n \geq n_0$ .  
Jawab:

$$T(n) = 2 + 4 + 6 + 8 + 16 + \dots + 2^n \rightarrow \text{Deret geometri}$$

$$S = \frac{a(r^n - 1)}{r - 1} = \frac{2(2^n - 1)}{2 - 1}$$

$$= 2(2^n - 1)$$

$$= 2^{n+1} - 2$$

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

$$T(n) \leq C \cdot f(n)$$

$$\text{Misal } n_0 = 1$$

$$2^{n+1} - 2 \leq C \cdot 2^n$$

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

$$\frac{2^{n+1}}{2^n} - \frac{2}{2^n} \leq C$$

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

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

$$2 - 1 \leq C$$

$$C \geq 1$$

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

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

Big-O

$O(n^2)$

$$T(n) \leq C \cdot f(n)$$

$$pn^2 + qn + r \leq C \cdot n^2$$

$$\frac{pn^2}{n^2} + \frac{qn}{n^2} + \frac{r}{n^2} \leq C$$

$$C \geq p + \frac{q}{n} + \frac{r}{n^2}$$

$$\text{Misal } n_0 = 1$$

$$C \geq p + \frac{q}{1} + \frac{r}{1^2}$$

$$C \geq p + q + r$$

$$\text{Misal } p = q = r = 1$$

$$C \geq 1 + 1 + 1$$

$$C \geq 3$$

Terbukti benar

Big-Ω

$\Omega(n^2)$

$$T(n) \geq C \cdot f(n)$$

$$pn^2 + qn + r \geq C \cdot n^2$$

$$\frac{pn^2}{n^2} + \frac{qn}{n^2} + \frac{r}{n^2} \geq C$$

$$p + \frac{q}{n} + \frac{r}{n^2} \geq C$$

$$\text{Misal } n_0 = 1$$

$$C \leq p + \frac{q}{1} + \frac{r}{1^2}$$

$$C \leq p + q + r$$

$$\text{Misal } p = q = r$$

$$C \leq 1 + 1 + 1$$

$$C \leq 3$$

Terbukti benar

Big-Θ

$\Theta(n^2)$

karena  $O(n^2)$  dan  $\Omega(n^2)$  terbukti berderajat sama maka  $\Theta(n^2)$  otomatis terbukti benar

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
      wij ← wij or wik and wkj
    endfor
  endfor
endfor

```

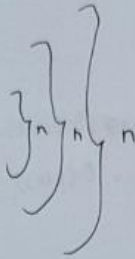
3. Kompleksitas waktu asimptotik

Tentukan Big O, Big Ω dan Big Θ

```

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

```



3 nested loop maka

$$T(n) = n \cdot n \cdot n \\ = n^3$$

Big O

$$O(n^3) \\ T(n) \leq c f(n) \\ n^3 \leq c n^3 \\ \frac{n^3}{n^3} \leq c \\ 1 \leq c \\ c \geq 1$$

Big Ω

$$\Omega(n^3) \\ T(n) \geq c f(n) \\ n^3 \geq c n^3 \\ \frac{n^3}{n^3} \geq c \\ 1 \geq c \\ c \leq 1$$

Big Θ  $O(n^3)$   $\Omega(n^3)$   
karena Big O dan Big Ω terbukti  
sederajat maka Big Θ =  $\Theta(n^3)$

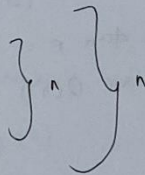
4. Tulislah algoritma untuk menjumlahkan dua buah matriks yang masing-masing berukuran  $n \times n$ . Berapa kompleksitas waktunya  $T(n)$ ? dan berapa kompleksitas waktu asimptotiknya yang dinyatakan dalam Big-O, Big-Ω, dan Big-Θ?

4. Algoritma menjumlahkan dua matriks  $n \times n$ , berapa kompleksitas asimptotik

```

for i ← 1 to n do
  for j ← 1 to n do
    mij ← aij + bij
  endfor
endfor

```



$$T(n) = n^2$$

$O(n^2)$

$$T(n) \leq c f(n) \\ n^2 \leq c n^2 \\ c \geq 1$$

$\Omega(n^2)$

$$T(n) \geq c f(n) \\ n^2 \geq c n^2 \\ c \leq 1$$

$\Theta(n^2)$

$O(n^2)$  dan  $\Omega(n^2)$   
berderajat  
sama maka  
 $\Theta(n^2)$

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- $\Omega$ , dan Big- $\Theta$ ?

Algoritma menyalin larik, Big O? Big  $\Omega$ ? Big  $\Theta$ ?

```
for i ← 1 to n do
  a_i ← b_i
endfor
```

}  $n$

$T(n) = n$

$O(n)$

$T(n) \leq C \cdot f(n)$

$n \leq C \cdot n$

$C \geq 1$

$\Omega(n)$

$T(n) \geq C \cdot f(n)$

$n \geq C \cdot n$

$C \leq 1$

$\Theta(n)$

$O(n)$  dan  $\Omega(n)$

sederajat, maka

$\Theta(n)$

6. Diberikan algoritma Bubble Sort sebagai berikut:

```
procedure BubbleSort(input/output a_1, a_2, ..., a_n: integer)
  { Mengurut tabel integer TabInt[1..n] dengan metode pengurutan bubble-
  sort
```

Masukan:  $a_1, a_2, \dots, a_n$

Keluaran:  $a_1, a_2, \dots, a_n$  (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  $a_k < a_{k-1}$  then
```

```
      { pertukarkan  $a_k$  dengan  $a_{k-1}$  }
```

```
      temp ←  $a_k$ 
```

```
       $a_k$  ←  $a_{k-1}$ 
```

```
       $a_{k-1}$  ← temp
```

```
    endif
```

```
  endfor
```

```
endfor
```

- Hitung berapa jumlah operasi perbandingan elemen-elemen tabel!
- Berapa kali maksimum pertukaran elemen-elemen tabel dilakukan?
- Hitung kompleksitas waktu asimptotik (Big-O, Big- $\Omega$ , dan Big- $\Theta$ ) dari algoritma Bubble Sort tersebut!

## 6. Bubble sort

a. Hitung berapa jumlah operasi perbandingan

$$T(n) = (n-1) + (n-2) + (n-3) + \dots + 1$$

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

b. Berapa kali maksimum pertukaran elemen \* tabel dilakukan?

$$\frac{n(n-1)}{2} \text{ kali atau } \frac{n^2 - n}{2} \text{ kali}$$

c. Kompleksitas waktu

• Best case

Perbandingan  $\rightarrow \frac{n(n-1)}{2}$  kali

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

• Worst case

Perbandingan  $\rightarrow \frac{n(n-1)}{2}$  kali

Assignment  $\rightarrow \frac{3n(n-1)}{2}$  kali

$$T_{\max}(n) = \frac{n(n-1)}{2} + \frac{3n(n-1)}{2} = \frac{4n(n-1)}{2} = 2n^2 - 2n$$

$$\begin{aligned} & \cdot O(n^2) \\ & 2n^2 - 2 \leq Cn^2 \\ & \frac{2n^2}{n^2} - \frac{2}{n^2} \leq C \\ & 2 - \frac{2}{n^2} \leq C \\ & \text{misal } n_0 = 1 \\ & C \geq 0 \end{aligned} \quad \left\{ \begin{aligned} & \Omega(n^2) \\ & 2n^2 - 2 \geq Cn^2 \\ & \frac{2n^2}{n^2} - \frac{2}{n^2} \geq C \\ & 2 - \frac{2}{n^2} \geq C \\ & \text{misal } n_0 = 1 \\ & C \leq 0 \end{aligned} \right.$$

$\Theta(n^2)$  karena  $O(n^2)$  dan  $\Omega(n^2)$  berderajat sama

a. Algoritma A  $\rightarrow n(1) \dots$

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

(a) Algoritma A mempunyai kompleksitas waktu  $O(\log N)$

(b) Algoritma B mempunyai kompleksitas waktu  $O(N \log N)$

(c) Algoritma C mempunyai kompleksitas waktu  $O(N^2)$

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

a. Algoritma A  $\rightarrow O(\log N)$

b. Algoritma B  $\rightarrow O(N \log N)$

c. Algoritma C  $\rightarrow O(N^2)$

untuk problem  $N=8$ , manakah yg paling cepat? Secara asimptotik?

$$A \rightarrow O(\log 8) = O(\log 2^3)$$

$$= O(3 \log 2)$$

$$B \rightarrow O(8 \log 8) = O(8 \cdot 3 \log 2)$$

$$= O(24 \log 2)$$

$$C \rightarrow O(8^2) = O(64)$$

Algoritma A memiliki hasil terkecil dibanding yg lain. Maka yang tercepat adalah algoritma A

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

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

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

**Deklarasi**

k : integer  
b<sub>1</sub>, b<sub>2</sub>, ..., b<sub>n</sub> : real

**Algoritma**

b<sub>n</sub> ← a<sub>n</sub>  
for k ← n - 1 downto 0 do  
    b<sub>k</sub> ← a<sub>k</sub> + b<sub>k+1</sub> \* x  
endfor  
return b<sub>0</sub>

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 Assignment  
Algoritma P

• b<sub>n</sub> ← a<sub>n</sub>

• b<sub>k</sub> ← a<sub>k</sub> + b<sub>k+1</sub> \* x      1 kali  
  n kali (loop)

T(n) = n + 1

O(n) untuk p2

Algoritma P

penjumlahan = n kali

perkalian = n kali

T(n) = n<sup>2</sup>

Algoritma p2 lebih baik daripada p