# LAPORAN PRAKTIKUM 3
## KOMPLEKSITAS WAKTU ASIMPTOTIK DARI ALGORITMA

**MATA KULIAH**
**ANALISIS ALGORITMA**
**D10G.4205 & D10K.0400601**


**PENGAJAR : (1) MIRA SURYANI, S.Pd., M.Kom**

**(2) INO SURYANA, Drs., M.Kom**
**(3) R. SUDRAJAT, Drs., M.Si**
**FAKULTAS : MIPA**
**SEMESTER : IV dan VI**

**DISUSUN OLEH : AHMAD FAAIZ A (140810180023)**

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


**MARET 2019**

## Latihan Analisa

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

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

**Penyelesaian:**

$T(n) = 2 + 4 + 8 + 16 + \cdots + 2^n = 2^{n+1} - 2 \quad (menggunakan \; rumus \; deret \; geometri)$

$T(n) = 2^{n+1} - 2 \; = 2 \, . \, 2^n - 2 = O(2^n)$

$T(n) \leq \; c \, . \, f(n), \qquad f(n) = 2^n$

$2 \, . \, 2^n - 2 \leq c \, . \, 2^n$
$2 \, . \, 2^n - 2 \leq 2 \, . \, 2^n + 2 \, . \, 2^n = 4 \, . \, 2^n, \qquad untuk \geq 1$
$2 \, . \, 2^n - 2 \leq 4 \, . \, 2^n$

Sehingga $C = 4, \;\; f(n) = 2^n, n_0 = 1$, didapat $T(n) = O(2^n)$

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

**Big-O Notation:**

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

Kita mengamati bahwa $n \; \geq \; 1$, maka $n \leq n^2$ dan $1 \; \leq \; n^2$ sehingga

$$pn^2 + qn + r = \boldsymbol{O(n^2)} \leq pn^2 + qn^2 + rn^2 = \boldsymbol{n^2(p + q + r)}, \qquad untuk \; n \geq 1$$

Maka kita bisa mengambil C= $p + q + r$ dan $n_0 = 1$ untuk memperlihatkan:

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

**Big-Ω Notation:**

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

Karena $pn^2 + qn + r \geq pn^2$ untuk $n \geq \; 1$, dengan mengambil C=p, kita memperoleh

$$pn^2 + qn + r = \Omega(n^2)$$

**Big-θ Notation:**

Karena $pn^2 + qn + r = O(n^2)$ dan $pn^2 + qn + r = \Omega(n^2)$, maka $pn^2 + qn + r = \theta(n^2)$

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
            W_ij ← W_ij or W_ik and W_kj
        endfor
    endfor
endfor
```

**Penyelesaian:**

```
for k ← 1 to n do                          O(n)
    for i ← 1 to n do                      O(n)
        for j ← to n do                    O(n)
            W_ij ← W_ij or W_ik and W_kj   O(1)
        endfor
    endfor
endfor
```

Penjelasan:

$O(n) \times O(n) \times O(n) \times O(1) = O(n \times n \times n \times 1) = O(n^3)$ Teorema 2(b)

Maka, didapat kompleksitas asimptotiknya:

Big-O = $O(n^3)$
Big-Ω = $\Omega(n^3)$
Big-Θ = $\theta(n^3)$

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?

**Penyelesaian:**

```
for k ← 1 to n do                          O(n)
    for j ← 1 to n do                      O(n)
        Sum[k][j] ← A[k][j] + B[k][j]      O(1)
    endfor
endfor
```

Kompleksitas Waktu:

$$T(n) = O(n) \times O(n) \times O(1)$$
$$= O(n \times n \times 1)$$
$$= O(n^2)$$

Maka, kompleksitas asimptotiknya:

Big-O = $O(n^2)$
Big-Ω = $\Omega(n^2)$
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?

**Penyelesaian:**

```
for i ← 1 to n do          O(n)
    Ai ← Bi                O(1)
endfor
```

Kompleksitas Waktu:

$$T(n) = O(n) \times O(1)$$
$$= O(n \times 1)$$
$$= O(n)$$

Maka, kompleksitas asimptotiknya:

Big-O = $O(n)$
Big-Ω = $\Omega(n)$
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!

**Jawab:**

$$1 + 2 + 3 + \cdots + (n-1) = \frac{n}{2}(n-1), \qquad (dengan\ rumus\ deret\ aritmatika)$$

(b) Berapa kali maksimum pertukaran elemen-elemen tabel dilakukan?

**Jawab:**

$$1 + 2 + 3 + \cdots + (n-1) = \frac{n}{2}(n-1), \qquad (dengan\ rumus\ deret\ aritmatika)$$

(c) Hitung kompleksitas waktu asimptotik (Big-O, Big-Ω, dan Big-Θ) dari algoritma Bubble Sort tersebut!

**Jawab:**

1. Best Case
   $T(n) = \frac{n}{2}(n-1)$ , maka Big-Ω = $\Omega(n^2)$

2. Worst Case
   $T(n) = \frac{n}{2}(n-1)$ , maka Big-O = $O(n^2)$

3. Average Case
   Karena Big-Ω = $\Omega(n^2)$ dan Big-O = $O(n^2)$, maka Big-Θ = $\theta(n^2)$

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?

**Penyelesaian:**

Jika N = 8, maka

(a) Algoritma A: $O(\log N) \Longrightarrow \log(8) = 0{,}9031$
(b) Algoritma B: $O(N \log N) \Longrightarrow 8\log(8) = 7{,}2247$
(c) Algoritma C: $O(N^2) \Longrightarrow 8^2 = 64$

Berdasarkan ketiga data di atas, dapat disimpulkan bahwa algoritma yang paling cepat adalah **Algoritma A** dengan waktu $\mathbf{0,9031}$

Namun, secara amsimptotik urutan Algoritma dari yang paling cepat ke yang lebih lambat adalah sebagai berikut:

$$O(\log N) < O(N \log N) < N^2 \quad Atau \quad A < B < C$$

Jadi**, Algoritma A** adalah algoritma yang **paling cepat** dibanding algoritma B dan C.

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?

**Penyelesaian:**

Algoritma P1:

Operasi perkalian: 1+2+3+ … + n = $\frac{n}{2}(n+1)$ kali
Operasi penjumlahan: $n$ kali
Kompleksitas waktu asimptotik: $T(n) = \frac{n}{2}(n+1) + n = O(n^2)$

Algoritma P2:

Operasi perkalian: $n$ kali
Operasi penjumlahan: $n$ kali
Kompleksitas waktu asimptotik: $T(n) = n + n = 2n = O(n)$

Dari kedua algoritma di atas, algoritma yang paling baik adalah **algoritma P2** karena memiliki kompleksitas waktu asimptotik yang lebih kecil dibandingkan dengan algoritma P.