# Analisis Algoritma

Laporan Praktikum 1

**Dibuat oleh:**
Muhammad Iqbal Alif Fadilla
140810180020

**Universitas Padjadjaran**
**Fakultas Matematika dan Ilmu Pengetahuan**
**2019**

**Worksheet 3**

1. Untuk $T(n) = 2 + 4 + 6 + 8 + 16 + \cdots + n^2$, tentukan nilai C, f(n), $n_o$, dan notasi Big-O sedemikian sehingga $T(n) = O(f(n))$ jika $T(n) \leq C$ *untuk semua* $n \geq n_0$

1. $T(n) = \frac{2(2^n - 1)}{2-1} = 2.2^n - 2$

Kompleksitas asimtotik = $O(f(n))$

$= O(2^n)$

$f(n) = 2^n$

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

Jika $n_0 = 1$

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

$2 \leq C.2$

$C \geq 1$

$C = 1$

Jadi

$C = 1$

$f(n) = 2^n$

$n_0 = 1$

$O(2^n)$

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

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

Maka T(n) berorde 2, sehingga kompleksitas asimtotiknya $O(n^2)$, $\Omega(n^2)$, $\theta(n^2)$

Berdasarkan Teorema 3

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
```

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

$= O(n^3) \rightarrow f(n)$

- Brg O: $O(f(n)) = O(n^3)$
- Brg Ω: $\Omega(f(n)) = \Omega(n^3)$
- Brg Θ = $\Theta(f(n)) = \Theta(n^3)$, Karena Brg O = Brg Ω

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?

```
for I ← 1 to n do
    for J ← 1 to n do
        hasil [i,j] ← a[ij] + b[ij]
    endfor
endfor
```

$T(n) = O(n) + O(n) = O(n^2) \rightarrow f(n)$

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

```
for i ← 1 to n do
    B(i) ← A(i)
end for
```

$T(n) = O(n) \rightarrow f(n)$

- Brg-O: $O(n)$
- Brg-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!
(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!

---

6.a.

| Pass | Jumlah Operasi |
|---|---|
| 1 | n - 1 |
| 2 | n - 2 |
| 3 | n - 3 |
| ⋮ | ⋮ |
| n | i |

$T(n) = (n-1) + (n-2) + \cdots + 1$

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

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

b. Maks pertukaran elemen : $\frac{n(n-1)}{2}$

c. Kompleksitas waktu Asimtotik

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

Big-Ω : $\Omega(n^2)$

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?

7. Algoritma A : O(log 8) — Yang Paling Cepat : Algoritma A
   Algoritma B : O(8 log 8) — karena semakin kecil angkanya
   Algoritma C : O(64) — semakin dikit operasinya

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?

8. Algoritma P
   • Jumlah : n kali
   • kali : n kali
   T(n) : n+n = 2n = n

   Algoritma P2
   $T_2(n)$ : 1+n = O(n)

   P dan $P_2$ sama-sama bagud . karena Big-O nya sama dengan O(n)