LAPORAN PRAKTIKUM
ANALISIS ALGORITMA

Disusun oleh:

Fadlan Mulya Priatna
140810180041
Kelas A

Program Studi S-1 Teknik Informatika
Departemen Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Padjadjaran

A. Materi

Definisi notasi BIG-O

$T(n) = O(f(n))$, artinya $T(n)$ berorde paling besar $f(n)$ bila terdapat konstanta C dan n_0 untuk $n \geq n_0$

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

Jika n dibuat semakin besar, waktu yg dibutuhkan tidak akan melebihi C dikalikan $f(n)$. Maka $f(n)$ adalah upper bound.

Dalam pembuktian Big-O, diperlukan nilai n_0 dan nilai C agar terpenuhi kondisi

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

Notasi Polinomial BIG-O Berderajat n

Digunakan untuk memperkirakan kompleksitas dengan mengabaikan suku berorde rendah. Contoh $T(n) = 3n^3 + 6n^2 + n + 8 = \mathrm{O}(n^3)$ dinyatakan pada:

Teorema 1 Bila $T(n) = a_m n^m + a_{m-1} n^{m-1} + a_1 n + a_0$ adalah polinom berderajat m maka $T(n) = O(n^m)$

Artinya:

Mengambil suku paling tinggi derajatnya yang diartikan laju pertumbuhan lebih cepat dibandingkan yang lainnya.

Teorema 2 Misalkan $T_1(n) = O(f(n))$ dan $T_2(n) = O(f(n))$, maka a) (i) $T_1(n) + T_2(n) = O(\max(f(n), g(n)))$ (ii) $T_1(n) + T_2(n) = O(f(n) + g(n))$ b) $T_1(n) \cdot T_2(n) = O(f(n))O(g(n)) = O(f(n) \cdot g(n))$ c) $O(cf(n)) = O(f(n))$, c adalah konstanta d) $f(n) = O(f(n))$

Contoh:

1. Misalkan $T_1(n) = O(n), T_2(n) = O(n^2)$, dan $T_3(n) = O(mn)$, dengan m sebagai peubah, maka:
   a. $T_1(n) + T_2(n) = O(\max(f(n, n^2))) = O(n^2)$ Teorema 2(a)(i)
   b. $T_2(n) + T_3(n) = O(n^2 + mn)$ Teorema 2(a)(ii)
   c. $T_1(n) \cdot T_2(n) = O(n \cdot n^2) = O(n^3)$ Teorema 2(b)
2. (a) $O(5n^2) = O(n^2)$ Teorema 2(c)

(b) $n^2 = O(n^2)$ Teorema 2(d)

Aturan Kompleksitas Waktu Asimptotik

Cara 1

Jika kompleksitas waktunya $T(n)$ sudah dihitung, maka kompleksitas waktu asimptotik dapat ditentukan dengan mengambil suku yang mendominasi fungsi T & menghilangkan koefisiennya (Teorema 1)

Contoh:

Pada algoritma cariMax, $T(n) = n - 1 = O(n)$

Cara 2

Big-O Notation:Pengisian nilai, perbandingan, operasi aritmatika, read, write, pengaksesan elemen larik, memilih field dari record, dan pemanggilan fungsi membutuhkan waktu $O(1)$. Cara penghitungan :

$T(n) \leq C \cdot g(n)$, dengan syarat nilai c dan n positif

Contoh

read (x)	O(1)
x <- x+1	O(1) + O(1) = O(1)
write (x)	O(1)

Kompleksitas waktu asimtotik algoritmanya O(1) + O(1) + O(1) = O(1)

Penjelasan :

O(1) + O(1) + O(1) = O(max(1,1)) + O(1)
= O(1) + O(1)
= O(max(1,1))
= O(1)

Notasi Big- Ω dan Big-Θ

Big-O hanya menyediakan upper bound.

Sedangkan untuk lower bound, dapat diperoleh dengan big-Ω dan big- Θ

Definisi

Notasi Big- Ω:

$T(n) = \Omega(f(n))$, artinya $T(n)$ berorde paling kecil $f(n)$ bila terdapat konstanta C dan n_0 sehingga :

$T(n) \geq C \cdot f(n)$, dengan syarat nilai c dan n positif

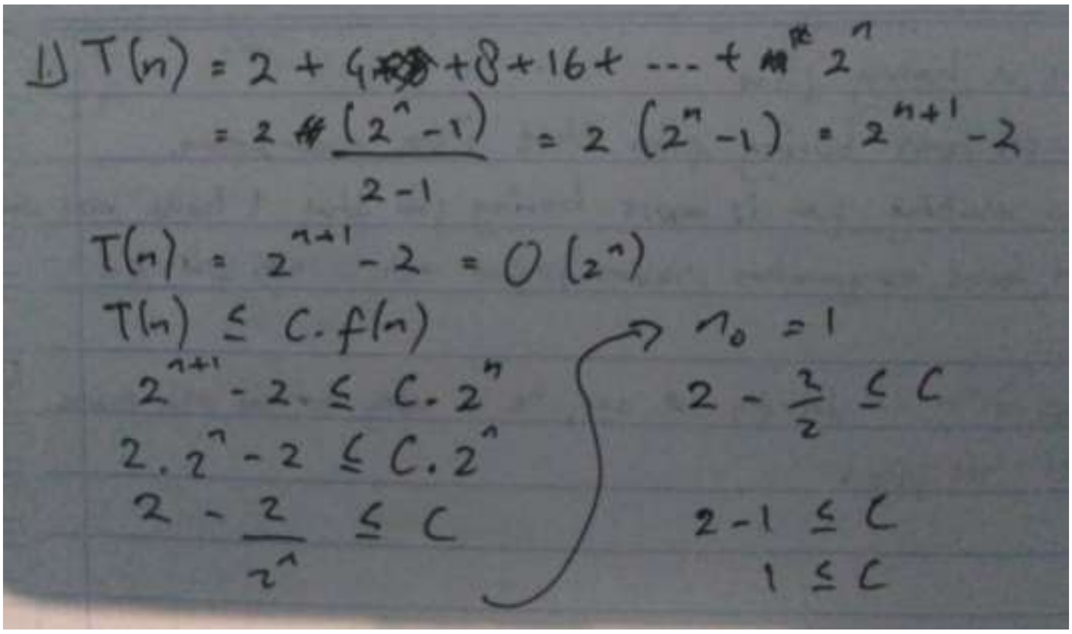
Notasi Big- Ω:

$T(n) = \Theta(h(n))$, artinya $T(n)$ berorde sama dengan $h(n)$ Jika $T(n) = O(h(n))$ dan $T(n) = \Omega(g(n))$.

$$C_1 f(n) \leq T(n) \leq C_2 f(n) \text{, dengan syarat nilai } c \text{ dan } n \text{ positif}$$

B. Bagian Analisis di Modul Praktikum

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



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

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

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

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

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

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

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

$n_0 = 1$

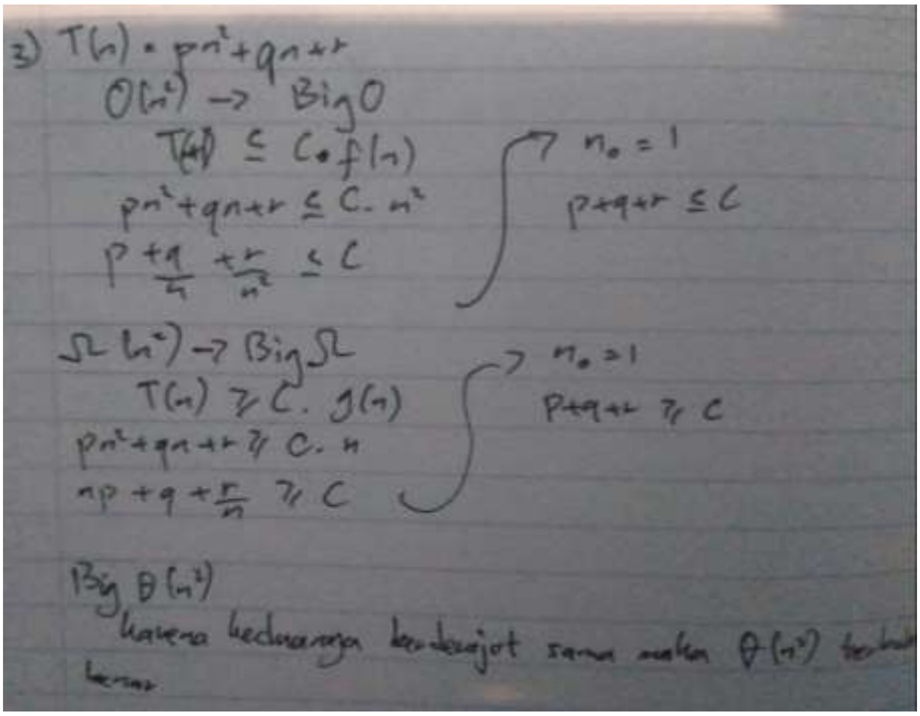
$2 - \dfrac{2}{2} \leq C$

$2 - 1 \leq C$

$1 \leq C$

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

$O(n^2) \rightarrow$ Big O

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

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

$p + \dfrac{q}{n} + \dfrac{r}{n^2} \leq C$

$n_0 = 1$

$p + q + r \leq C$

$\Omega(n^2) \rightarrow$ Big Ω

$T(n) \geq C.g(n)$

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

$np + q + \dfrac{r}{n} \geq C$

$n_0 = 1$

$p + q + r \geq C$

Big $\Theta(n^2)$

karena keduanya berderajat sama maka $\Theta(n^2)$ 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

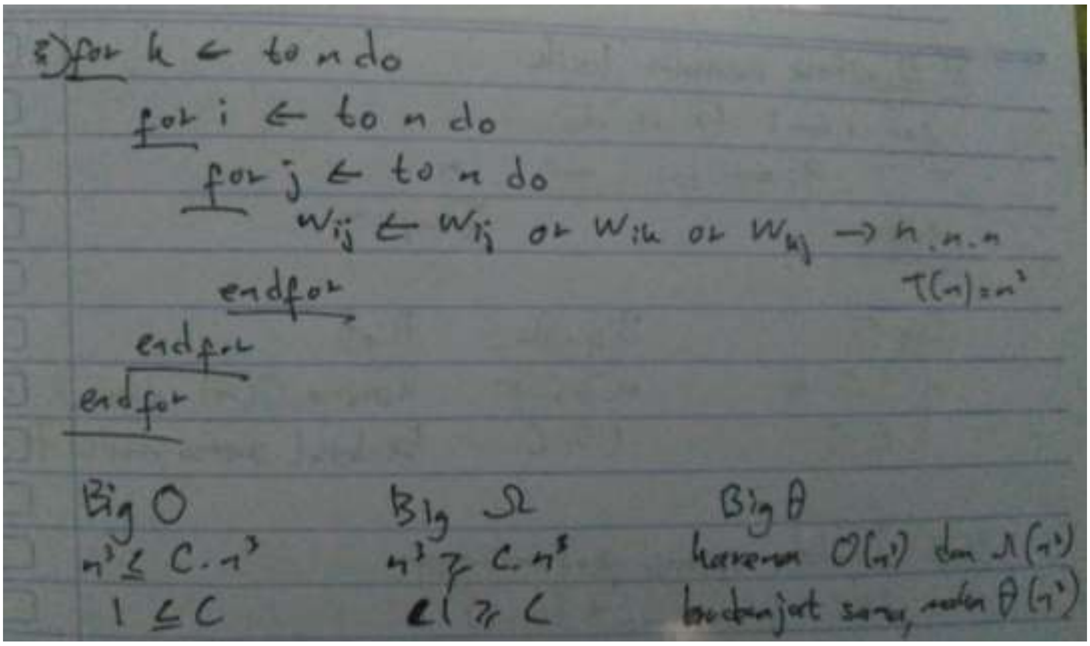
   for j ← to n do

   $w_{ij} \leftarrow w_{ij}$ or w_{ik} and w_{kj}

   endfor

   endfor

   endfor



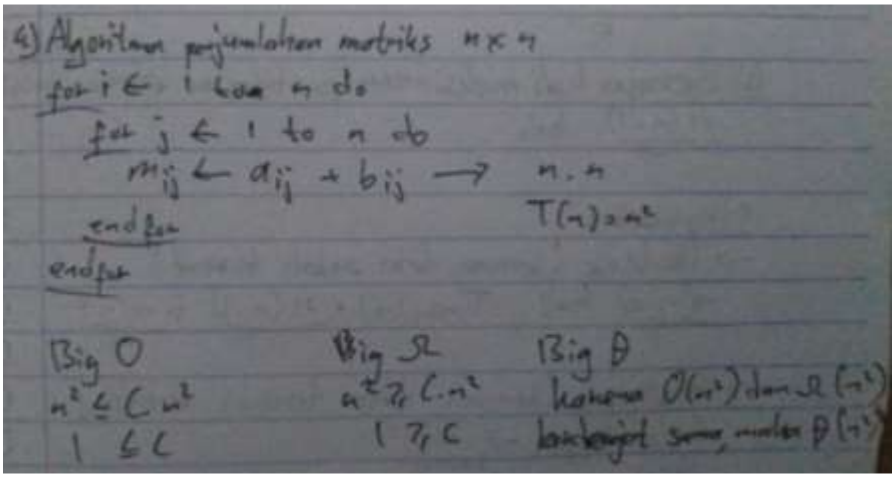
3) for k ← to n do
for i ← to n do
for j ← to n do
$w_{ij} \leftarrow w_{ij}$ or w_{ik} or w_{kj} → $n.n.n$
endfor
$T(n) = n^3$
endfor
endfor

Big O
$n^3 \le C \cdot n^3$
$1 \le C$

Big Ω
$n^3 \ge C \cdot n^3$
$1 \ge C$

Big θ
karena $O(n^3)$ dan $\Omega(n^3)$
berdanjat sama, maka $\theta(n^3)$

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
$m_{ij} \leftarrow a_{ij} + b_{ij}$ → $n.n$
endfor
$T(n) = n^2$
endfor

Big O
$n^2 \le C n^2$
$1 \le C$

Big Ω
$n^2 \ge C \cdot n^2$
$1 \ge C$

Big θ
karena $O(n^2)$ dan $\Omega(n^2)$
berdanjat 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-Ω, dan Big-Θ?

5.) Algoritma menyalin larik

for $i \leftarrow 1$ to n do
  $a_i \leftarrow b_i \longrightarrow n$
endfor
$T(n) = n$

Big O
$n \leq C.n$
$1 \leq C$

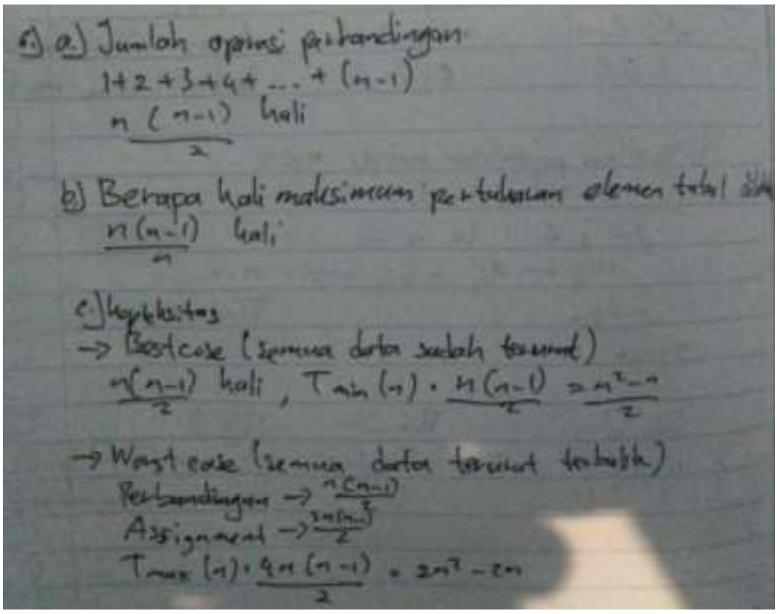
Big Ω
$n \geq C.n$
$1 \geq C$

Big θ
karena $O(n)$ dan $\Omega(n)$ berderet sama, maka $\theta(n)$

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.) Jumlah operasi perbandingan

$1+2+3+4+ \dots + (n-1)$

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

b.) Berapa kali maksimum pertukaran elemen tabel ...

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

c.) Kompleksitas

→ Bestcase (semua data sudah terurut)

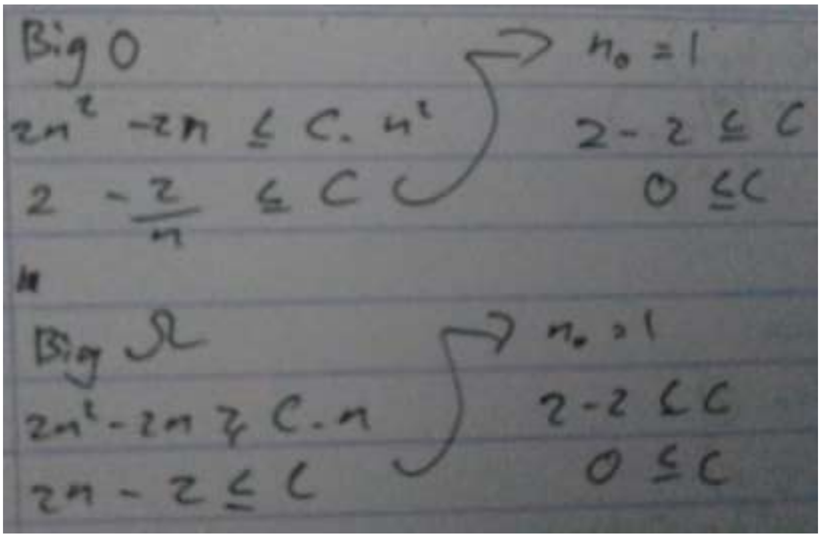
$\frac{n(n-1)}{2}$ kali, $T_{min}(n) = \frac{n(n-1)}{2} = \frac{n^2-n}{2}$

→ Worst case (semua data terurut terbalik)

Perbandingan → $\frac{n(n-1)}{2}$

Assignment → $\frac{3n(n-1)}{2}$

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



Big O

$2n^2 - 2n \le C \cdot n^2$

$2 - \frac{2}{n} \le C$

$n_0 = 1$

$2 - 2 \le C$

$0 \le C$

Big Ω

$2n^2 - 2n \ge C \cdot n$

$2n - 2 \le C$

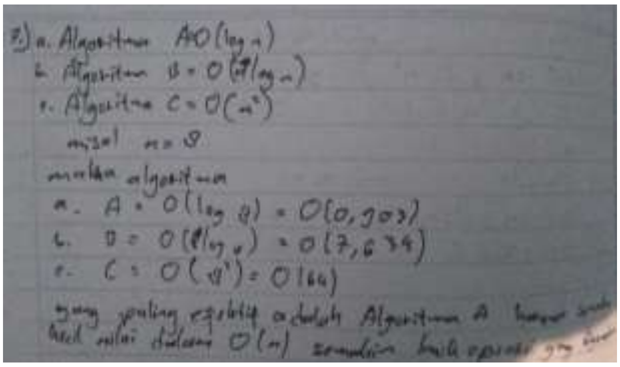
$n_0 = 1$

$2 - 2 \le C$

$0 \le C$

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(N2)

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



7.) a. Algoritma A O(log n)

b. Algoritma B = O(n log n)

c. Algoritma C = O(n^2)

misal n = 8

maka algoritma

a. A = O(log 8) = O(0,903)

b. B = O(8 log 8) = O(7,224)

c. C = O(8^2) = O(64)

yang paling efektif adalah Algoritma A karena ... hasil nilai dalam O(n) semakin kecil operasi yg ...

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?

d) Operasi Assignment

- $b_n \leftarrow a_n$ 1 kali
  $b_n \leftarrow a_n + b_{n+1} * x$ = n kali

$T(n) = n + 1$

$O(n)$ untuk p^2

Algoritma p

penjumlahan = n kali

perkalian = n kali

$T(n) = 2n$

Algoritma p^2 lebih baik dari pada p