Nama : Naufal Ariful Amri

NPM : 140810180009

Worksheet 3

1. Untuk T(n) = 2 + 4 + 8 + 16 + ⋯ + 2^n, tentukan nilai C, f(n), n_O, dan notasi Big-O sedemikian sehingga T(n) = 0(f(n)) jika T(n) ≤ C untuk semua n ≥ n_O

$$T(n) = 2 + 4 + 8 + \ldots + 2^n$$
$$T(n) = \frac{2(2^n - 1)}{2 - 1}$$
$$T(n) = 2^{n+1} + 2$$

Untuk big O. maka $f(n) = 2^n$

Bukti bahwa $T(n) = 2^{n+1} + 2 = O(2^n)$ adalah $T(n) \leq C.f(n)$

$= 2^{n+1} + 2 \leq C.2^n$, maka $2 < 2^n$

$= 2^{n+1} + 2 \leq 2^{n+1} + 2^n = 2^{2n+1}$ untuk $n \geq 1$

Maka bisa kita ambil C = 2 dan n_0 = 1 untuk memperlihatkan

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

2. Buktikan bahwa untuk konstanta-konstanta positif p, q, dan r: T(n) = pn^2 + qn + r adalah 0(n^2), Ω(n^2), dan Θ(n^2)

Bukti big O

Bukti bahwa $pn^2 + qn + 1 = O(n^2)$ adalah

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

$= pn^2 + qn + 1 \leq C.n^2.$

kita mengamati bahwa $n \geq 1$, $n^2 > n$, $dan\ n^2 > r$

Sehingga $pn^2 + qn + 1 \leq pn^2 + qn^2 + rn^2 = p + q + r$

Bukti big Ω

Bukti bahwa $pn^2 + qn + 1 = \Omega(n^2)$ adalah

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

$= pn^2 + qn + 1 \geq C.n^2$

Sehingga $pn^2 + qn + 1 \geq pn^2$

Dengan $C = p\ dan\ n \geq 1$

Bukti big θ

Karena $pn^2 + qn + 1 = O(n^2)$dan $pn^2 + qn + 1 = \Omega(n^2)$

Maka $pn^2 + qn + 1 = \theta(n^2)$

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
            wij ← wij or wik and wkj

        endfor
    endfor
endfor
```

Jawab :

- Untuk $k = 1$
  - $i = 1$, jumlah perhitungan sebanyak $n\ kali$
- Untuk $k = 2$
  - $i = 1$, jumlah perhitungan sebanyak $n\ kali$
  - $i = 2$, jumlah perhitungan sebanyak $n - 1\ kali$
- Untuk $k = n$
  - $i = 1$, jumlah perhitungan sebanyak $n\ kali$
  - $i = 2$, jumlah perhitungan sebanyak $n - 1\ kali$
  - $i = n$, jumlah perhitungan sebanyak $1\ kali$

Jumlah perhitungan = $n^2 + (n-1)^2 + (n-2)^2 + \ldots + 1^2$

$$T(n) = n^2 + (n-1)^2 + (n-2)^2 + \ldots + 1^2$$

$$T(n) = \frac{n\,(n+1)(2n+1)}{6}$$

$$T(n) = 2n^3 + 3n^2 + 1$$

$$T(n) = O(n^3) = \Omega(n^3) = \theta(n^3)$$

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 ?

Algoritma

```
for i <- 1 to baris do
    for j <- 1 to kolom do
    begin
        hasil [i , j] <- matrix1 [i , j] + matrix2 [ i , j ]
    endfor
endfor
```

- Untuk $i = 1$
  - $j = 1$, jumlah perhitungan sebanyak $1\ kali$
  - $j = 2$, jumlah perhitungan sebanyak $2\ kali$
  - $j = n$, jumlah perhitungan sebanyak $n\ kali$
- Untuk $i = 2$
  - $j = 1$, jumlah perhitungan sebanyak $1\ kali$
  - $j = 2$, jumlah perhitungan sebanyak $2\ kali$
  - $j = n$, jumlah perhitungan sebanyak $n\ kali$
- Untuk $i = n$
  - $j = 1$, jumlah perhitungan sebanyak $1\ kali$
  - $j = 2$, jumlah perhitungan sebanyak $2\ kali$
  - $j = n$, jumlah perhitungan sebanyak $n\ kali$

  Jumlah perhitungan = = $n\ .\ (n)$
  $T(n) = n\ (n)$
  $T(n) = n^2$
  $T(n) = O(n^2) = \Omega(n^2) = \theta(n^2)$

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?

   Algoritma

   ```
   Read(N)

   for I <- 1 to N

       B[ i ] <- A[ i ]

   endfor
   ```

   Kompleksitas waktu

   Waktu pemindahan larik -> $n\ kali\ , loop\ for\ sebanyak\ n\ kali$
   $T(n) = O(n) = \Omega(n) = \theta(n)$

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!

Jawab

a. Jumlah operasi perbandungan.

Perbandingan = (n – 1) + (n – 2) + (n – 3) + . . . + 1

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

b. maksimal pertukaran elemen

pertukaran = pertukaran terjadi ada

- line 5 -> $\frac{n}{2}(n-1)$
- line 6 -> $\frac{n}{2}(n-1)$
- line 7 -> $\frac{n}{2}(n-1)$

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

c. Kompleksitas waktu

big O

$$2n^2 - 2n \leq C.n^2$$
$$2 - \frac{2}{n} \leq C, n0\,! = 0\ dan\ n0\,! = \frac{1}{2}$$
$$c \geq 0$$

Big Ω

$$2n^2 - 2n \geq C.n^2$$
$$\frac{n^2}{2} - \frac{n}{2} \geq C.n^2$$
$$\frac{1}{2} - \frac{1}{2n} \geq C\ n0\,! = 0\ dan\ n0\,! = \frac{1}{2}$$
$$c \leq 0$$

Big θ

Karena $2n^2 - 2n = O(n^2)$dan $2n^2 - 2n = \Omega(n^2)$
Maka $2n^2 - 2n + 1 = \theta(n^2)$

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?

Jawab :

Secara asimtotik algoritma yang tercepat adalah O(log N) . pembuktiannya adalah inputkan N ke ketiga algoritma. Kompleksias O (log N) adalah log 8 = 0.903. Kompleksitas O (N log N) = 8 log 8 = 7.2222. Kompleksitas O (n^2) = 8 ^ 2 = 64. Jadi yang tercepat adalah O(log N).

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?

P1

Operasi penjumlahan = n kali -> loop for

Operasi perkalian = 1 + 2 + 3 + 4 + . . . + n = $\frac{n}{2}(n+1)$

Operasi penjumlahan + operasi perkalian = $n + \frac{n}{2}(n+1) = O(n^2)$

P2

Operasi bk = n kali - > loop for

Operasi total $O(n)$

Yang terbaik adalah algoritma adalah algoritma p2 karena kompleksitas waktunya lebih efektif yaitu $O(n)$