LAPORAN PRAKTIKUM

WORKSHEET 2

Oleh:

Muhamad Ilham Habib (140810180018)

Kelas B

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

JATINANGOR

2019

I. Tujuan

1. Mahasiswa paham mengenai kompleksitas algoritma secara umum
2. Mahasiswa paham cara menghitung kompleksitas waktu
3. Mahasiswa mengetahui operasi-operasi dalam menghitung kompleksitas waktu
4. Mahasiswa mengimplementasikan point di atas dengan masalah yang diberikan

II. Landasan Teori

Sebuah algoritma tidak saja harus benar, tetapi juga harus efisien. Algoritma yang bagus adalah algoritma yang efisien. Efisiensi suatu algoritma diukur dari berapa jumlah waktu dan ruang (space) memori yang dibutuhkan untuk menjalankannya. Algoritma yang efisien adalah algoritma yang meminimumkan kebutuhan waktu dan ruang.

Kompleksitas algoritma bergantung dengan jumlah waktu/kompleksitas waktu yang bisa dinotasikan dengan **T(n)** dan ruang memori yang bisa dinotasikan dengan **S(n)**

Kompleksitas waktu dibedakan menjadi 3 macam yaitu

Best Case = $T_{min}(n)$ yaitu kompleksitas waktu dengan jumlah terkecil

Average Case = $T_{avg}(n)$ yaitu rata-rata yang biasanya Penjumlahan dari Best Case dan Worst Case dibagi 2

Worst Case = $T_{max}(n)$ yaitu kompleksitas waktu dengan jumlah terbesar

III. Worksheet 2

1. **Studi Kasus 1: Pencarian Nilai Maksimal**

   Buatlah programnya dan hitunglah kompleksitas waktu dari algoritma berikut:

```
procedure CariMaks(input x1, x2, ... , xn: integer, output maks: integer)
{  Mencari elemen terbesar dari sekumpulan elemen larik integer x1, x2, ... , xn. Elemen terbesar akan
   disimpan di dalam maks
   Input: x1, x2, ... , xn
   Output: maks (nilai terbesar)
}

Deklarasi
        i : integer

Algoritma
        maks ← x1
        i ← 2
        while i ≤ n do
           if xi > maks then
                maks ← xi
           endif
           i ← i + 1
        endwhile
```

Jawaban :

a. Operasi Assigment (———) = 1 +1+ (n-1) + (n-1) = 2n

b. Operasi Perbandingan (———) = (n-1)

c. Operasi Penjumlahan (———) = (n-1)

   Maka $T_{\max}$ = 4n - 4

2. **Studi Kasus 2: Sequential Search**

   Diberikan larik bilangan bulan x_1, x_2, x_n... yang telah terurut menaik dan tidak ada elemen ganda. Buatlah programnya dengan C++ dan hitunglah kompleksitas waktu terbaik, terburuk, dan ratarata dari algoritma pencarian beruntun (sequential search). Algoritma sequential search berikut menghasilkan indeks elemen yang bernilai sama dengan y. Jika y tidak ditemukan, indeks 0 akan dihasilkan.

```
procedure SequentialSearch(input x1, x2, ... xn : integer,  y : integer, output idx : integer)
{  Mencari y di dalam elemen x1, x2, ... xn. Lokasi (indeks elemen) tempat y ditemukan diisi ke dalam idx.
   Jika y tidak ditemukan, makai idx diisi dengan 0.
   Input: x1, x2, ... xn
   Output: idx
}
```

```
Deklarasi
        i : integer
        found : boolean {bernilai true jika y ditemukan atau false jika y tidak ditemukan}
Algoritma
        i ← 1
        found ← false
        while (i ≤ n) and (not found) do
            if xi = y then
                found ← true
            else
                i ← i + 1
            endif
        endwhile
        {i < n or found}

        If found then {y ditemukan}
                idx ← i
        else
                idx ← 0 {y tidak ditemukan}
        endif
```

Jawaban :

T_{min}(n) :

a. Operasi Assigment (——) = 4
b. Operasi Perbandingan (——) = 2

T_{min}(n) : 4 + 2 = 6

T_{max}(n) :

a. Operasi Assigment (——) = 1 + 1 + n + 1 = 3 + n
b. Operasi Perbandingan (——) = n + 1
c. Operasi Penjumlahan (——) = n

T_{max}(n) : 3 + n + n + 1 + n = 3n + 4

T_{avg}(n) :

(T_{max}(n) + T_{min}(n)) / 2 =(6 + 3n + 4)/2 = 3n + 10 / 2

3. Studi Kasus 3: Binary Search

Diberikan larik bilangan bulan x_1, x_2, x_n... yang telah terurut menaik dan tidak ada elemen ganda. Buatlah programnya dengan C++ dan hitunglah kompleksitas waktu terbaik, terburuk, dan ratarata dari algoritma pencarian bagi dua (binary search). Algoritma binary search berikut menghasilkan indeks elemen yang bernilai sama dengan y. Jika y tidak ditemukan, indeks 0 akan dihasilkan.

```
procedure BinarySearch(input x1, x2, ... xn : integer, x : integer, output : idx : integer)
{   Mencari y di dalam elemen x1, x2, ... xn. Lokasi (indeks elemen) tempat y ditemukan diisi ke dalam idx.
    Jika y tidak ditemukan makai dx diisi dengan 0.
    Input: x1, x2, ... xn
    Output: idx
}
Deklarasi
        i, j, mid : integer
        found : Boolean
Algoritma
        i ← 1
        j ← n
        found ← false
        while (not found) and ( i ≤ j) do
                mid ← (i + j) div 2
                if xmid = y then
                    found ← true
                else
                    if xmid < y then      {mencari di bagian kanan}
                        i ← mid + 1
                    else                  {mencari di bagian kiri}
                        j ← mid – 1
                    endif
                endif
        endwhile
        {found or i > j }

        If found then
                Idx ← mid
        else
                Idx ← 0
        endif
```

Jawaban :

T$_{min}$(n) :

a. Operasi Assigment (——) = 4
b. Operasi Perbandingan (——) = 2

T$_{min}$(n) : 6 + 2 = 8

T$_{max}$(n) :

Panjang array akan berubah pada setiap iterasi:

- Iterasi 1 = n
- Iterasi 2 = n/2
- Iterasi 3 = $n/2^2$
- Iterasi x = $n/2^k$-1 ~ $n/2^k$ *(-1 diabaikan karena kecil dibanding $n/2^k$)*

Panjang array menjadi 1.

Maka,

$n/2^k$ = 1

n = 2^k

log 2(n)= log 2(2^k) = k log 2(2)

k = log 2(n)

Sehingga

T_{max}(n) : log 2n

T_{avg}(n) :

(T_{max}(n) + T_{min}(n)) / 2 = (8 + log 2n)/2

4. Studi Kasus 4: Insertion Sort

a. Buatlah program insertion sort dengan menggunakan bahasa C++

b. Hitunglah operasi perbandingan elemen larik dan operasi pertukaran pada algoritma insertion sort.

c. Tentukan kompleksitas waktu terbaik, terburuk, dan rata-rata untuk algoritma insertion sort.

```
procedure InsertionSort(input/output x1, x2, ... xn : integer)
{   Mengurutkan elemen-elemen x1, x2, ... xn dengan metode insertion sort.
    Input: x1, x2, ... xn
    OutputL x1, x2, ... xn (sudah terurut menaik)
}
Deklarasi
        i, j, insert : integer
Algoritma
        for i ← 2 to n do
            insert ← xi
            j ← i
            while (j < i) and (x[j-i] > insert) do
                x[j]← x[j-1]
                j←j-1
            endwhile
            x[j] = insert
        endfor
```

Jawaban :

T_{min}(n) :

a. Operasi Assigment (——) = 2 (n-1) + (n-1) = 3n-3

b. Operasi Perbandingan = 2*((n-1) + (n-1)) = 2 * (2n-2) = 4n – 4

c. Operasi Pertukaran = (n – 1) * n = n^2- n

T_{min}(n) : 3n – 3 + 4n – 4 + 1 = 7n – 6

T_{max}(n) :

T_{max}(n) : 3n – 3 + 4n – 4 + n^2- n = n^2+ 6n – 6

T_{avg}(n) :

(T_{max}(n) + T_{min}(n)) / 2 = (7n – 6 + n^2+6n – 6) / 2 =(n^2+ 13n -12)/2

5. Studi Kasus 5: Selection Sort

1. Buatlah program selection sort dengan menggunakan bahasa C++

2. Hitunglah operasi perbandingan elemen larik dan operasi pertukaran pada algoritma selection sort.

3. Tentukan kompleksitas waktu terbaik, terburuk, dan rata-rata untuk algoritma insertion sort.

```
procedure SelectionSort(input/output x1, x2, ... xn : integer)
{   Mengurutkan elemen-elemen x1, x2, ... xn dengan metode selection sort.
    Input: x1, x2, ... xn
    OutputL x1, x2, ... xn (sudah terurut menaik)
}
Deklarasi
          i, j, imaks, temp : integer
Algoritma
          for i ← n downto 2 do {pass sebanyak n-1 kali}
               imaks ← 1
               for j ← 2 to i do
                 if xj > ximaks then
                    imaks ← j
                 endif
               endfor
               {pertukarkan ximaks dengan xi}
               temp ← xi
               xi ← ximaks
               ximaks ← temp
          endfor
```

1. Operasi Perbandingan =

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

2. Operasi Pertukaran = n – 1

   T_{min}(n) :

   T_{min}(n) : (4n – 4) + $\frac{1}{2}$ (n^2- n) + 1 ~ n^2

   T_{max}(n) :

   T_{min}(n) : $\frac{1}{2}$ (n^2- n) + (n – 1) ~ n^2

   T_{avg}(n) :

   (T_{max}(n) + T_{min}(n)) / 2 = (n^2 + n^2) / 2