

# **LAPORAN PRAKTIKUM 2 DESAIN DAN ANALISIS ALGORITMA**



**DISUSUN OLEH:**  
**SURIADI VAJRAKARUNA** **140810180038**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN  
ALAM  
UNIVERSITAS PADJADJARAN  
SUMEDANG  
2020**

## **I. Tujuan**

1. Mahasiswa mengerti kompleksitas algoritma secara umum.
2. Mahasiswa mengerti cara menghitung kompleksitas waktu dari suatu algoritma.
3. Mahasiswa mengetahui faktor-faktor yang mempengaruhi kompleksitas waktu dari suatu algoritma.
4. Mahasiswa mengetahui operasi-operasi yang mempengaruhi atau ikut disertakan dalam menghitung kompleksitas waktu dari suatu algoritma.
5. Mahasiswa dapat mengimplementasikan perhitungan algoritma saat membuat sebuah program.

## II. Landasan Teori

1. Sebuah algoritma tidak hanya harus benar tetapi harus efisien dan optimal.
2. Kompleksitas algoritma bergantung pada jumlah waktu/kompleksitas waktu yang dinotasikan dengan  **$T(n)$**  dan ruang memori yang dinotasikan dengan  **$S(n)$**  yang dibutuhkan untuk mengeksekusi algoritma tersebut.
3. Sebuah algoritma dikatakan efisien apabila waktu dan ruang memori yang dibutuhkan untuk mengeksekusinya kecil.
4. Kompleksitas waktu dapat dihitung dengan
  - a. Menetapkan ukuran input ( $n$ ).
  - b. Menghitung banyak operasi seperti penjumlahan (+), pengurangan (-), perbandingan (<, >, =) kecuali pada *loop*, Pembagian (/), pembacaan atau *assignment* (<-), pemanggilan prosedur atau *function*, dan lain-lain.
  - c. Jika operasi berada didalam *loop*, maka operasi itu dihitung sesuai berapa kali *looping* itu terjadi.
5. Jika kita mengetahui besaran waktu (detik) untuk melaksanakan operasi-operasi yang terdapat di sebuah algoritma, maka kita dapat menghitung waktu sesungguhnya yang diperlukan sebuah mesin untuk mengeksekusi algoritma tersebut.
6. Ada tiga macam kompleksitas waktu yang mungkin terjadi pada sebuah algoritma yaitu
  - a. *Best Case* =  $T_{\min}(n)$  adalah kompleksitas waktu dengan jumlah paling kecil.
  - b. *Average Case* =  $T_{\text{avg}}(n)$  adalah kompleksitas dengan jumlah rata-rata keseluruhan kemungkinan. Jadi , hasil perhitungan dari *worst case* dan *best case* dijumlah lalu dibagi dengan  $n$  (biasanya  $n=2$ ).
  - c. *Worst Case* =  $T_{\max}(n)$  adalah kompleksitas waktu dengan jumlah paling besar.

### III. Worksheet 2

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

Buatlah programnya dan hitunglah kompleksitas waktu dari algoritma berikut:

```
procedure CariMaks(input  $x_1, x_2, \dots, x_n$ : integer, output maks: integer)
{ Mencari elemen terbesar dari sekumpulan elemen larik integer  $x_1, x_2, \dots, x_n$ . Elemen terbesar akan
  disimpan di dalam maks
  Input:  $x_1, x_2, \dots, x_n$ 
  Output: maks (nilai terbesar)
}

Deklarasi
  i : integer

Algoritma
  maks  $\leftarrow x_1$ 
  i  $\leftarrow 2$ 
  while i  $\leq$  n do
    if  $x_i >$  maks then
      maks  $\leftarrow x_i$ 
    endif
    i  $\leftarrow i + 1$ 
  endwhile
```

Jawab:

1. Operasi Assignment =  $1 + 1 + (n-1) + (n-1) = 2n$

2. Operasi Perbandingan =  $n-1$

3. Operasi Penjumlahan =  $n-1$

Maka  $T_{\max}(n) = 4n-2$

## 2. Studi Kasus 2: Sequential Search

Diberikan larik bilangan bulat  $x_1, x_2, \dots, x_n$  yang telah terurut menaik dan tidak ada elemen ganda. Buatlah programnya dengan C++ dan hitunglah kompleksitas waktu terbaik, terburuk, dan rata-rata 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  $x_1, x_2, \dots, x_n$  : integer,  $y$  : integer, output  $idx$  : integer)
{ Mencari  $y$  di dalam elemen  $x_1, x_2, \dots, x_n$ . Lokasi (indeks elemen) tempat  $y$  ditemukan diisi ke dalam  $idx$ .
  Jika  $y$  tidak ditemukan, maka  $idx$  diisi dengan 0.
  Input:  $x_1, x_2, \dots, x_n$ 
  Output:  $idx$ 
}
```

```
Deklarasi
   $i$  : integer
  found : boolean { bernilai true jika  $y$  ditemukan atau false jika  $y$  tidak ditemukan}

Algoritma
   $i \leftarrow 1$ 
  found  $\leftarrow$  false
  while ( $i \leq n$ ) and (not found) do
    if  $x_i = y$  then
      found  $\leftarrow$  true
    else
       $i \leftarrow i + 1$ 
    endif
  endwhile
  {  $i < n$  or found }

  If found then {  $y$  ditemukan }
     $idx \leftarrow i$ 
  else
     $idx \leftarrow 0$  {  $y$  tidak ditemukan }
  endif
```

Jawab:

$T_{\min}(n)$ :

1. Operasi Assignment = 4

2. Operasi Perbandingan = 2

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

$T_{\max}(n)$ :

1. Operasi Assignment =  $1 + 1 + n + 1 = 3 + n$

2. Operasi Perbandingan =  $n + 1$

3. Operasi Penjumlahan =  $n$

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

$T_{\text{avg}}(n)$ :

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

$$T_{\text{avg}}(n) = (10 + 3n) / 2$$

### 3. Studi Kasus 3: Binary Search

Diberikan larik bilangan bulat  $x_1, x_2, \dots, x_n$  yang telah terurut menaik dan tidak ada elemen ganda. Buatlah programnya dengan C++ dan hitunglah kompleksitas waktu terbaik, terburuk, dan rata - rata 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  $x_1, x_2, \dots, x_n$  : integer,  $x$  : integer, output :  $idx$  : integer)
{ Mencari  $y$  di dalam elemen  $x_1, x_2, \dots, x_n$ . Lokasi (indeks elemen) tempat  $y$  ditemukan diisi ke dalam  $idx$ .
  Jika  $y$  tidak ditemukan maka  $idx$  diisi dengan 0.
  Input:  $x_1, x_2, \dots, x_n$ 
  Output:  $idx$ 
}
Deklarasi
   $i, j, mid$  : integer
  found : Boolean
Algoritma
   $i \leftarrow 1$ 
   $j \leftarrow n$ 
  found  $\leftarrow$  false
  while (not found) and ( $i \leq j$ ) do
     $mid \leftarrow (i + j) \div 2$ 
    if  $x_{mid} = y$  then
      found  $\leftarrow$  true
    else
```

```

        if  $x_{mid} < y$  then {mencari di bagian kanan}
             $i \leftarrow mid + 1$ 
        else {mencari di bagian kiri}
             $j \leftarrow mid - 1$ 
        endif
    endwhile
    {found or  $i > j$ }

    If found then
         $idx \leftarrow mid$ 
    else
         $idx \leftarrow 0$ 
    endif

```

Jawab:

$T_{min}(n)$ :

1. Operasi Assignment = 6
2. 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  $x = n/2^{k-1} \sim n/2^k$  (-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

$$T_{max}(n) = \lceil \log_2(n) \rceil$$

**$T_{avg}(n)$ :**

$$(T_{min}(n) + T_{max}(n)) / 2 = (1 + \log 2(n)) / 2$$

$$T_{avg}(n) = (1 + \log 2(n)) / 2$$

#### 4. Studi Kasus 4: Insertion Sort

1. Buatlah program insertion sort dengan menggunakan bahasa C++.
2. Hitunglah operasi perbandingan elemen larik dan operasi pertukaran pada algoritma insertion sort.
3. Tentukan kompleksitas waktu terbaik, terburuk, dan rata-rata untuk algoritma insertion sort.

```
procedure InsertionSort(input/output  $x_1, x_2, \dots, x_n$  : integer)
{  Mengurutkan elemen-elemen  $x_1, x_2, \dots, x_n$  dengan metode insertion sort.
  Input:  $x_1, x_2, \dots, x_n$ 
  Output:  $x_1, x_2, \dots, x_n$  (sudah terurut menaik)
}
Deklarasi
  i, j, insert : integer
Algoritma
  for i  $\leftarrow$  2 to n do
    insert  $\leftarrow$   $x_i$ 
    j  $\leftarrow$  i
    while (j < i) and ( $x[j-i] >$  insert) do
       $x[j] \leftarrow x[j-1]$ 
      j  $\leftarrow$  j-1
    endwhile
     $x[j] =$  insert
  endfor
```

Jawab:

1. Operasi Assignment:  $2(n-1) + (n-1) = 3n-3$

2. Operasi Perbandingan:  $2*((n-1) + (n-1)) = 2*(2n-2) = 4n-4$

3. Operasi Pertukaran:  $(n-1) * n = n^2-n$

**$T_{min}(n)$ :**

$$T_{min}(n) = 3n-3 + 4n-4 + 1 = 7n - 6$$

**$T_{max}(n)$ :**



$$T_{\max}(n) = 3n-3 + 4n-4 + n^2-n = n^2+6n-6$$

$T_{\text{avg}}(n)$ :

$$(T_{\min}(n) + T_{\max}(n)) / 2 = (7n-6 + n^2+6n-6) / 2$$

$$T_{\text{avg}}(n) = (n^2 + 13n - 12) / 2$$

## 5. Studi Kasus 5: Selection Sort

1. Buatlah program insertion 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 selection sort.

```

procedure SelectionSort(input/output  $x_1, x_2, \dots, x_n$  : integer)
{ Mengurutkan elemen-elemen  $x_1, x_2, \dots, x_n$  dengan metode selection sort.
  Input:  $x_1, x_2, \dots, x_n$ 
  Output:  $x_1, x_2, \dots, x_n$  (sudah terurut menaik)
}
Deklarasi
  i, j, imaks, temp : integer
Algoritma
  for i  $\leftarrow$  n downto 2 do {pass sebanyak n-1 kali}
    imaks  $\leftarrow$  1
    for j  $\leftarrow$  2 to i do
      if  $x_j$   $\geq$   $x_{\text{imaks}}$  then
        imaks  $\leftarrow$  j
      endif
    endfor
    {pertukarkan  $x_{\text{imaks}}$  dengan  $x_i$ }
    temp  $\leftarrow$   $x_i$ 
     $x_i$   $\leftarrow$   $x_{\text{imaks}}$ 
     $x_{\text{imaks}}$   $\leftarrow$  temp
  endfor

```

Jawab:

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 =  $n-1$

**$T_{\min}(n)$ :**

$$T_{\min}(n) = (4n-4) + \frac{1}{2}(n^2-n) + 1 \sim n^2$$

**$T_{\max}(n)$ :**

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

**$T_{\text{avg}}(n)$ :**

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

$$T_{\text{avg}}(n) = n^2$$

## IV. Program Worksheet 2

### 1. Studi Kasus 1

Sourcecode (.cpp)

```
/*  
NAMA      : SURIADI VAJRAKARNA  
NPM       : 140810180038  
KELAS    : B  
TANGGAL   : 8 MARET 2020  
TUGAS 2 - STUDI KASUS 1 - PRAKTIKUM ANALISIS ALGORITMA  
*/  
  
#include <iostream>  
using namespace std;  
  
int main(int argc, char const *argv[])  
{  
    int bil[5] = {1632, 3, 230, 23, 28};  
    int n = sizeof(bil)/sizeof(bil[0]);  
    int max = bil[0];  
    int i = 2;  
  
    while (i <= n)  
    {  
        if (bil[i] > max)  
            max = bil[i];  
        i = i + 1;  
    }  
    cout << "Max = " << max;  
  
    return 0;  
}
```

Screenshot

```
PS D:\MEGASync\SEN\SEMESTER 4\ANALGO\Praktikum\LaporanProgramPraktikum\AnalgoKu2> g++ -o sk1 studikases1.cpp  
PS D:\MEGASync\SEN\SEMESTER 4\ANALGO\Praktikum\LaporanProgramPraktikum\AnalgoKu2> ./sk1  
Max = 230
```

## 2. Studi Kasus 2

### Sourcecode (.cpp)

```
/*
NAMA      : SURIADI VAJRAKARNA
NPM       : 140810180038
KELAS    : B
TANGGAL   : 8 MARET 2020
TUGAS 2 - STUDI KASUS 2 - PRAKTIKUM ANALISIS ALGORITMA
*/

#include <iostream>
using namespace std;

int main(int argc, char const *argv[])
{
    int bil[5] = {15, 387, 22, 63, 74};
    int cari = 63;
    int n = sizeof(bil) / sizeof(bil[0]);

    int idx;
    int i = 1;
    bool found = false;

    while (i <= n && !found)
    {
        if (bil[i] == cari)
            found = true;
        else
            i = i + 1;
    }

    if (found == true)
    {
        idx = i;
        cout << "Found at index " << idx;
    }

    else
    {
        idx = 0;
        cout << "Not Found";
    }
}
```

```
    return 0;
}
```

### Screenshot

```
PS D:\MEGASync\SEN\SEMESTER 4\ANALGO\Praktikum\LaporanProgramPraktikum\AnalgoKu2> g++ -o sk2 studikases2.cpp
PS D:\MEGASync\SEN\SEMESTER 4\ANALGO\Praktikum\LaporanProgramPraktikum\AnalgoKu2> ./sk2
Found at index 3
```

## 3. Studi Kasus 3

### Sourcecode (.cpp)

```
/*
NAMA      : SURIADI VAJRAKARNA
NPM       : 140810180038
KELAS     : B
TANGGAL   : 8 MARET 2020
TUGAS 2 - STUDI KASUS 3 - PRAKTIKUM ANALISIS ALGORITMA
*/

#include <iostream>
using namespace std;

main()
{
    int bil[5] = {14, 31, 59, 72, 98};
    int cari = 72;
    int n = sizeof(bil) / sizeof(bil[0]);
    int idx;

    int i = 1;
    int j = n;
    int mid;
    bool found = false;
    while (!found && i <= j)
    {
        mid = (i + j) / 2;
        if (bil[mid] == cari)
            found = true;
        else if (bil[mid] < cari)
            i = mid + 1;
        else

```

```

        j = mid - 1;
    }

    if (found == true)
    {
        idx = mid;
        cout << "Found at index " << idx;
    }
    else
    {
        idx = 0;
        cout << "Not Found";
    }

    return 0;
}

```

#### Screenshot

```

PS D:\MEGASync\SEN\SEMESTER 4\ANALGO\Praktikum\LaporanProgramPraktikum\AnalgoKu2> g++ -o sk3 studikusus3.cpp
PS D:\MEGASync\SEN\SEMESTER 4\ANALGO\Praktikum\LaporanProgramPraktikum\AnalgoKu2> ./sk3
Found at index 3

```

#### 4. Studi Kasus 4

##### Sourcecode (.cpp)

```

/*
NAMA      : SURIADI VAJRAKARNA
NPM       : 140810180038
KELAS    : B
TANGGAL   : 8 MARET 2020
TUGAS 2 - STUDI KASUS 4 - PRAKTIKUM ANALISIS ALGORITMA
*/

#include <iostream>
using namespace std;

int main(int argc, char const *argv[])
{
    int bil[5] = {3, 45, 72, 31, 96};
    int n = sizeof(bil) / sizeof(bil[0]);

```

```

int i, j, insert;

for (i = 1; i < n; i++)
{
    insert = bil[i];
    j = i - 1;
    while (j >= 0 && bil[j] > insert)
    {
        bil[j + 1] = bil[j];
        j = j - 1;
    }
    bil[j + 1] = insert;
}

cout << "Insertion Sort: ";
for (j = 0; j < n; j++)
{
    cout << bil[j] << " ";
}
return 0;
}

```

### Screenshot

```

PS D:\MEGASync\SEN\SEMESTER 4\ANALGO\Praktikum\LaporanProgramPraktikum\AnalgoKu2> g++ -o sk4 studikases4.cpp
PS D:\MEGASync\SEN\SEMESTER 4\ANALGO\Praktikum\LaporanProgramPraktikum\AnalgoKu2> ./sk4
Insertion Sort: 3 31 45 72 96

```

## 5. Studi Kasus 5

### Sourcecode (.cpp)

```

/*
NAMA      : SURIADI VAJRAKARNA
NPM       : 140810180038
KELAS     : B
TANGGAL   : 8 MARET 2020
TUGAS 2 - STUDI KASUS 5 - PRAKTIKUM ANALISIS ALGORITMA
*/

#include <iostream>
using namespace std;

```

```

int main(int argc, char const *argv[])
{
    int bil[5] = {42, 12, 57, 289, 48};
    int n = sizeof(bil) / sizeof(bil[0]);

    int i, j, imaks, temp;

    for (i = 2; i < n; i++)
    {
        imaks = 1;
        for (j = 2; j < i; j++)
        {
            if (bil[j] > bil[imaks])
                imaks = j;
        }
        temp = bil[i];
        bil[i] = bil[imaks];
        bil[imaks] = temp;
    }

    cout << "Selection Sort: ";
    for (int i = 0; i < n; i++)
    {
        cout << bil[i] << " ";
    }
    return 0;
}

```

### Screenshot

```

PS D:\MEGASync\SEN\SEMESTER 4\ANALGO\Praktikum\LaporanProgramPraktikum\AnalgoKu2> g++ -o sk5 studikusus5.cpp
PS D:\MEGASync\SEN\SEMESTER 4\ANALGO\Praktikum\LaporanProgramPraktikum\AnalgoKu2> ./sk5
Selection Sort: 42 48 12 57 289

```



## Daftar Pustaka

Suryani, Mira, Ino Suryana, R. Sudrajat. 2019. *ANALISIS ALGORITMA: MODUL PRAKTIKUM 2*. Jatinangor: Universitas Padjadjaran.

Hatta, Mouhamad, Rifaldi, Agung, dan Siddiq. *Analisis Algoritma: Best, Worst, dan Average*. Satelit di (diakses 6 Maret).

Gautama, Elliana. *Kompleksitas Algoritma*. Satelit di (diakses 6 Maret).

Mukharl, Adam. *Analisis Algoritma – Pengantar Kompleksitas Algoritma*. Satelit di (diakses 6 Maret).