LAPORAN PRAKTIKUM 2
Analisis Algoritma

Disusun oleh :

Asep Budiyana Muharam
140810180029

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

Studi Kasus 1 - Pencarian Nilai Maksimal :

Program:

```
#include <iostream>
using namespace std;

int main()
{
    int max, i, arr[] = { 1, 45, 54, 71, 66, 12 };
    int n = sizeof(arr)/sizeof(arr[0]);

    //Start Algorithm
    max = arr[0];
    i = 2;
    while (i<=n) {
        if (arr[i]>max) {
            max = arr[i];
        }
        i++;
    }
    // End Algorithm

    cout<<"Array: ";
    for (int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }

    cout<<"\nMax = "<<max;
    return 0;
}
```

Analisis:

Algoritma

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

(i) Operasi Assignment

maks <- x1	1 kali
i <- 2	1 kali
maks <- xi	n kali
i <- i + 1	n kali

t1 = 1 + 1 + n + n = 2+2n

(ii) Operasi Perbandingan

if xi > maks then	n kali

t2 = n

(iii) Operasi Penjumlahan

i + 1	n kali

t3 = n

Jadi, T(n) = t1 + t2 + t3 = 2 + 2n + n + n = 2 + 4n

Studi Kasus 2 - Sequential Search :

Program:

```
#include <iostream>
using namespace std;

int main()
{
    int i, idx, arr[] = { 10, 20, 80, 30, 60, 50, 110, 100, 130, 170 };
    bool found;
    int n = sizeof(arr)/sizeof(arr[0]);
    int y = 110;

    // Start Algorithm
    i = 1;
    found = false;
    while (i<=n && !found) {
        if (arr[i] == y) found = true;
        else i++;
    }
    if (found) idx = i;
    else idx = 0;
    // End Algorithm
```

```
    cout<<"Array: ";
    for (int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }
    cout<<"\nElemen "<<y<<" Berada pada index ke-"<<idx;
    return 0;
}
```

Analisis:

```
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
```

1. Tmin(n)

Terjadi ketika n (besar inputan) = 1 dan nilai yang dicari berada di index ke-0;
t(assignment) = 4
t(perbandingan) = 2

sehingga:
Tmin(n) = 4 + 2 = 6 = Ω(1)

2. Tmax(n)

Terjadi ketika nilai yang dicari berada di index terakhir atau tidak ditemukan.
t(assignment) = 3 + n
t(perbandingan) = 1 + n
t(penjumlahan) = n

sehingga:
Tmax(n) = 3 + n + 1 + n + n = 4 + 3n = O(n)

3. Tavg(n)

Tavg(n) = (Tmin(n) + Tmax(n)) / 2 = (6 + 4 + 3n) / 2 = (10 + 3n) / 2 = Θ(n)

Studi Kasus 3 - Binary Search :

Program:

```
#include <iostream>
using namespace std;

int main()
{
    int i, j, mid, idx, arr[] = { 10, 20, 80, 30, 60, 50, 110, 100,
130, 170 };
    bool found;
    int n = sizeof(arr)/sizeof(arr[0]);
    int y = 130;

    // Start Algorithm
    i = 1;
    j = n;
    found = false;
    while (!found && i<=j) {
        mid = (i + j) / 2;
        if (arr[mid] == y) found = true;
        else {
            if (arr[mid] < y) i = mid + 1;
            else j = mid - 1;
        }
    }
    if (found) idx = mid;
    else idx = 0;
    // End Algorithm

    cout<<"Array: ";
    for (int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }
    cout<<"\nElemen "<<y<<" Berada pada index ke-"<<idx;
    return 0;
}
```

[redacted]

Analisis:

```
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
```

1. Tmin(n)

Terjadi ketika nilai yang dicari berada di index ke-0;
t(assignment) = 6
t(perbandingan) = 2

sehingga:
Tmin(n) = 6 + 2 = 8 = Ω(1)

2. Tmax(n)

Terjadi ketika nilai yang dicari berada di index terakhir atau tidak ditemukan.

perubahan panjang array pada tiap iterasi:
iterasi 1 = n kali
iterasi 2 = n/2 kali
iterasi 3 = n/2^2
iterasi k = n/2^k-1 ~ n/2^k

setelah pembagian ke k , panjang array menjadi 1
maka:

n/2^k   = 1
n   = 2^k
log_2(n) = log_2(2^k)
log_2(n) = k log_2(2)
k   = log_2(n)

sehingga:
Tmax(n) = O(log_2(n))

3. Tavg(n)

Tavg(n) = (Tmin(n) + Tmax(n)) / 2 = (1 + log_2(n)) / 2 = **Θ**(log_2(n))

Studi Kasus 4 - Insertion Sort :

Program:

```
#include <iostream>
using namespace std;

int main()
{
    int i, j, insert, arr[] = { 1, 45, 54, 71, 66, 12 };
    int n = sizeof(arr)/sizeof(arr[0]);

    cout<<"Unsorted Array: ";
    for (int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }

    // Start Algorithm
    for (i=1;i<n;i++) {
        insert = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > insert) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = insert;
    }
    // End Algorithm

    cout<<"\nSorted Array: ";
```

```
    for (int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }
    return 0;
}
```

Analisis:

```
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
```

Jumlah Operasi:

1. Operasi Perbandingan = 2*((n-1) + (n-1)) = 2*(2n-2) = 4n - 4
2. Operasi Pertukaran = (n-1) * n = (n^2)-n

Kompleksitas Waktu:

1. Tmin(n)

Terjadi ketika hanya terjadi satu kali pertukaran

sehingga:
Tmin(n) = 4n - 4 + 1 = 4n - 3 = Ω(n)

2. Tmax(n)

Terjadi ketika terjadi kurang lebih n kali pertukaran.

sehingga:
Tmax(n) = 4n - 4 + (n^2) - n = (n^2) + 3n - 4 = O(n^2)

3. Tavg(n)

Tavg(n) = (Tmin(n) + Tmax(n)) / 2 = (n + n^2) / 2 = **Θ**(n^2)

Studi Kasus 5 - Selection Sort :

Program:

```
#include <iostream>
```

```
using namespace std;

int main()
{
    int i, j, imin, temp, arr[] = { 1, 45, 54, 71, 66, 12 };
    int n = sizeof(arr)/sizeof(arr[0]);

    cout<<"Unsorted Array: ";
    for (int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }

    // Start Algorithm
    for (i=0;i<n-1;i++) {
        imin = i;
        for (j=i+1;j<n;j++) {
            if (arr[j] < arr[imin]) imin = j;
        }
        temp = arr[i];
        arr[i] = arr[imin];
        arr[imin] = temp;
    }
    // End Algorithm

    cout<<"\nSorted Array: ";
    for (int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }
    return 0;
}
```

***Analisis*:**

Algoritma

```
for i ← n downto 2 do {pass sebanyak n-1 kali}
    imaks ← 1
    for j ← 2 to i do
      if x_j > x_imaks then
        imaks ← j
      endif
    endfor
    {pertukarkan x_imaks dengan x_i}
    temp ← x_i
    x_i ← x_imaks
    x_imaks ← temp
endfor
```

Jumlah Operasi:

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)

Kompleksitas Waktu:

1. Tmin(n)

Terjadi ketika hanya terjadi satu kali pertukaran

sehingga:

Tmin(n) = ½ (n^2 - n) + 1 = Ω(n^2)

2. Tmax(n)

Terjadi ketika terjadi kurang lebih n kali pertukaran.

sehingga:

Tmax(n) = ½ (n^2 - n) + (n-1) = O(n^2)

3. Tavg(n)

Tavg(n) = (Tmin(n) + Tmax(n)) / 2 = (n^2 + n^2) / 2 = Θ(n^2)