LAPORAN PRAKTIKUM 2
Analysis Algorithm

Kompleksitas Waktu dari Algoritma

Disusun oleh :

Mohammad Dhikri
140810180075

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

PENDAHULUAN

Dalam memecahkan suatu masalah dengan komputer seringkali kita dihadapkan pada pilihan berikut:

1. Menggunakan algoritma yang waktu eksekusinya cepat dengan komputer standar.
2. Menggunakan algoritma yang waktu eksekusinya tidak terlalu cepat dengan komputer yang cepat.

Dikarenakan keterbatasan sumber daya, pola pemecahan masalah beralih ke pertimbangan menggunakan algoritma. Oleh karena itu diperlukan algoritma yang efektif dan efisien atau lebih tepatnya Algoritma yang mangkus.

Algoritma yang mangkus diukur dari berapa **jumlah waktu dan ruang (space) memori** yang dibutuhkan untuk menjalankannya. Algoritma yang mangkus ialah algoritma yang meminimumkan kebutuhan waktu dan ruang. Penentuan kemangkusan algoritma adakah dengan melakukan pengukuran kompleksitas algoritma.

Kompleksitas algoritma terdiri dari kompleksitas waktu dan ruang. Terminologi yang diperlukan dalam membahas kompleksitas waktu dan ruang adalah:

1. Ukuran input data untuk suatu algoritma, . Contoh algoritma pengurutan elemen-elemen larik, adalah jumlah elemen larik. Sedangkan dalam algoritma perkalian matriks n adalah ukuran matriks .
2. Kompleksitas waktu, (), adalah jumlah operasi yang dilakukan untuk melaksanakan algoritma sebagai fungsi dari input .
3. Kompleksitas ruang, (), adalah ruang memori yang dibutuhkan algoritma sebagai fungsi dari input .

KOMPLEKSITAS WAKTU

Kompleksitas waktu sebuah algoritma dapat dihitung dengan langkah-langkah sebagai berikut:

1. Menetapkan ukuran input.
2. Menghitung banyaknya operasi yang dilakukan oleh algoritma. Dalam sebuah algoritma terdapat banyak jenis operasi seperti operasi penjumlahan, pengurangan, perbandingan, pembagian, pembacaan, pemanggilan prosedur, dsb.

STUDI KASUS

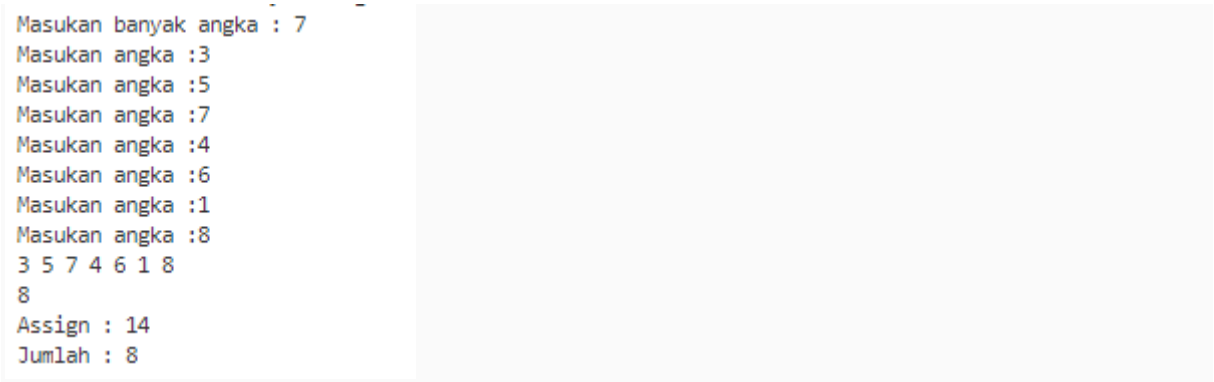
Studi Kasus 1 : Pencarian Nilai Maksimum

Program :

```
#include <iostream>
using namespace std;
main(){
    int arr[100];
    int n;
    cout<<"Masukan banyak angka : "; cin>>n;
    for(int i=0; i<n; i++){
        cout<<"Masukan angka :"; cin>>arr[i];
    }
    for(int i=0; i<n; i++){
        cout<<arr[i]<<" ";
    }
```

```
    cout<<endl;
    int maks;
    int jumlah = 0;
    int assign = 3;
    maks=arr[0];
    int i=0;
    while(i<=n){
        if(arr[i] > maks){
            maks=arr[i];
            assign+=1;
        }
        i=i+1;
        assign+=1;
        jumlah+=1;
    }
    cout<<maks<<endl;
    cout<<"Assign : "<<assign<<endl;
    cout<<"Jumlah : "<<jumlah<<endl;
}
```

Screenshot :



Kompleksitas waktu : $T(n) = n - 1$

Studi Kasus 2 : Sequential Search

Program :

```
#include <iostream>
using namespace std;
main(){
    int arr[100];
    int n;
    cout<<"Masukan banyak angka : "; cin>>n;
    for(int i=0; i<n; i++){
        arr[i]=i+1;
    }
    for(int i=0; i<n; i++){
        cout<<arr[i]<<" ";
    }
```

```
    int i;
    bool found;
    int y;
    //Algoritma
    i=0;
    cout<<"Masukan angka yang ingin dicari : "; cin>>y;
    found = false;
    while(i<=n && not found){
        if(arr[i] == y){
            found=true;
        }
        else{
            i=i+1;
        }
    }
    if(found){
        y=i+1;
    }
    else{
        y=0;
    }
    cout<<y<<endl;
}
```

Screenshot :

```
E:\Kuliah\Semester 4\Analysis Algorithm\Praktikum\Weel
uliah\Semester 4\Analysis Algorithm\Praktikum\Week 2\
Masukan banyak angka : 6
1 2 3 4 5 6 Masukan angka yang ingin dicari : 5
5

E:\Kuliah\Semester 4\Analysis Algorithm\Praktikum\Weel
uliah\Semester 4\Analysis Algorithm\Praktikum\Week 2\
Masukan banyak angka : 6
1 2 3 4 5 6 Masukan angka yang ingin dicari : 7
0
```

Kompleksitas waktu terbaik, terburuk dan rata-rata :

1. Kasus terbaik : ini terjadi bila a_1 = x
   $T_{min}(n) = 1$

2. Kasus terburuk : bila $a_n = x$ atau x tidak ditemukan.

   $T_{max}(n) = 4+2n$

3. Kasus rata-rata : Jika x ditemukan pada posisi ke-j, maka semua operasi (a_k = x) akan dieksekusi sebanyak j kali.

   $T_{avg}(n) = (1+2+3+..+n)/n = (1/2n(1+n))/n = (n+1)/2 \sim n$

Studi Kasus 3 : Binary Search

Program :

```
#include <iostream>
using namespace std;
main(){
    int arr[100];
    int n;
    cout<<"Masukan banyak angka : "; cin>>n;
    for(int i=0; i<n; i++){
        arr[i]=i+1;
    }
    for(int i=0; i<n; i++){
        cout<<arr[i]<<" ";
    }
    //Deklarasi
    int i, j, mid, y;
    bool found;
    //Algoritma
    i=1;
    j=n;
    cout<<"Masukan angka yang ingin dicari : "; cin>>y;
    found = false;
    while(not 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){
        y=mid+1;
    }
    else{
        y=0;
    }
    cout<<y<<endl;
}
```

Screenshot :

```
E:\Kuliah\Semester 4\Analysis Algorithm\Praktikum\Week 2\C
uliah\Semester 4\Analysis Algorithm\Praktikum\Week 2\Code
Masukan banyak angka : 7
1 2 3 4 5 6 7 Masukan angka yang ingin dicari : 6
6

E:\Kuliah\Semester 4\Analysis Algorithm\Praktikum\Week 2\C
uliah\Semester 4\Analysis Algorithm\Praktikum\Week 2\Code
Masukan banyak angka : 7
1 2 3 4 5 6 7 Masukan angka yang ingin dicari : 4
4
```

Kompleksitas waktu terbaik, terburuk dan rata-rata :

- Kasus terbaik : $T_{min}(n) = 1$
- Kasus average: $T_{avg}(n/2) : {}^{2}\log(n/2)$
- Kasus terburuk : $T_{max}(n) = {}^{2}\log n$

Studi Kasus 4 : Insertion Sort

Program :

```
#include <iostream>
using namespace std;
main(){
    int n;
    int arr[100];
    cout<<"Masukan banyak angka : "; cin>>n;
    for(int i=0; i<n; i++){
        cout<<"Masukan angka : "; cin>>arr[i];
    }
    //Deklarasi
    int i, j, insert;
    //Algoritma
    for(i=0; i<n; i++){
        insert = arr[i];
        j=i-1;
        while((arr[j] > insert) && (j>=0)){
            arr[j+1]=arr[j];
            j=j-1;
        }
        arr[j+1]=insert;
    }
    for(int i=0; i<n; i++){
        cout<<arr[i]<<" ";
    }
}
```

Screenshot :

```
E:\Kuliah\Semester 4\Analysis Algo
uliah\Semester 4\Analysis Algorith
Masukan banyak angka : 5
Masukan angka : 2
Masukan angka : 4
Masukan angka : 3
Masukan angka : 7
Masukan angka : 5
2 3 4 5 7
```

Jumlah operasi perbandingan dan operasi pertukaran :

J	Perbandingan	Perpindahan	Total Operasi
2	1	1	2
3	2	2	4
4	3	3	6
5	4	4	8
N	$(n-1)$	$(n-1)$	$2(n-1)$

Jumlah operasi perbandingan : (n - 1)
Jumlah operasi pertukaran : (n - 1)

Kompleksitas waktu terbaik, terburuk dan rata-rata :
Kompleksitas waktu T(n) untuk worst case yang dibutuhkan adalah :

$$T(n) = 2(1) + 2(2) + 2(3) + 2(4) + 2(5) + \cdots + 2(n-1))$$
$$= 2\big(1 + 2 + 3 + 4 + 5 + \cdots + (n-1)\big)$$
$$= 2\frac{(n-1)(n)}{2}$$
$$= (n-1)(n)$$

$$T(n) = n^2 - n \text{ dengan } n > 1\ ;\ n \epsilon \mathbb{Z}$$

Ditentukan worst case-nya ternyata :

$$T(n)_{Insertion\ Sort} = O(n^2)$$

- Kasus terbaik : $T_{min}(n) = 1$
- Kasus average: $T_{avg}\ (n/2) : n^2/4$
- Kasus terburuk : $T_{max}\ (n) = n^2$

Jadi worst case dari Algoritma Insertion Sort adalah $O(n^2)$

Studi Kasus 5 : Selection Sort

Program :

```
#include <iostream>
using namespace std;
main(){
    int n, arr[100], temp;
    cout<<"Masukan banyak angka : "; cin>>n;
    for(int i=0; i<n; i++){
```

```
        cout<<"Masukan angka : "; cin>>arr[i];
    }
    for(int i=0; i<n; i++){
        for(int j=i+1; j<n; j++){
            if(arr[i]>arr[j]){
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
    cout<<"Hasil algoritma sendiri : ";
    for(int i=0; i<n; i++){
        cout<<arr[i]<<" ";
    }
    //Deklarasi
    int i, j, imaks;
    //Algoritma
    for(i=n; i>=1; i--){
        imaks=1;
        for(j=1; j<i; j++){
            if(arr[j] > arr[imaks]){
                imaks=j;
            }
        }
        temp = arr[i];
        arr[i] = arr[imaks];
        arr[imaks] = temp;
    }
    cout<<endl<<"Hasil algoritma Modul : ";
    for(int i=0; i<n; i++){
        cout<<arr[i]<<" ";
    }
}
```

Screenshot :

```
E:\Kuliah\Semester 4\Analysis Algorithm\P
uliah\Semester 4\Analysis Algorithm\Prakt
Masukan banyak angka : 3
Masukan angka : 4
Masukan angka : 2
Masukan angka : 1
Hasil algoritma sendiri : 1 2 4
Hasil algoritma Modul : 1 1877811344 2
```

Jumlah operasi perbandingan dan operasi pertukaran :

➔ Jumlah Operasi Perbandingan

Jumlah operasi perbandingan element. Untuk setiap *pass* ke-*i*,

i = 1 –> jumlah perbandingan = *n* – 1

$i = 2$ –> jumlah perbandingan = $n - 2$

$i = 3$ –> jumlah perbandingan = $n - 3$

:

$i = k$ –> jumlah perbandingan = $n - k$

:

$i = n - 1$ –> jumlah perbandingan = 1

Jumlah seluruh operasi perbandingan elemen-elemen larik adalah :

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

➔ Jumlah Operasi Pertukaran

Untuk setiap i dari 1 sampai $n - 1$, terjadi satu kali pertukaran elemen, sehingga jumlah operasi **pertukaran seluruhnya adalah $T(n) = n - 1$.**

Jadi, algoritma pengurutan maksimum membutuhkan $n(n-1)/2$ buah operasi perbandingan elemen dan $n - 1$ buah operasi pertukaran.

Kompleksitas waktu terbaik, terburuk dan rata-rata :

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

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

Berarti kompleksitasnya secara simtotik adalah $O(n^2)$

- Kasus terbaik : $T_{min}(n) = 1$
- Kasus average: $T_{avg}(n/2) : n^2/4$
- Kasus terburuk : $T_{max}(n) = n^2$

DAFTAR PUSTAKA

[1] Prosiding Annual Research Seminar 2017 Computer Science and ICT ISBN : 979-587-705-4

[2] Makalah Kompleksitas Algoritma Pengurutan Selection Sort dan Insertion Sort (Setia Negara B. Tjaru (13508054))

[3] https://dosen.perbanas.id/kompleksitas-algoritma/?print=print

[4] https://bertzzie.com/knowledge/analisis-algoritma/KompleksitasAlgoritma.html

[5] http://dhiekalantana.blog.unas.ac.id/files/2012/09/binary-search-analysis.pdf

[6] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2009-2010/Makalah0910/MakalahStrukdis0910-074.pdf