LAPORAN PRAKTIKUM 2

ANALISIS ALGORITMA

Disusun oleh :

Hafidh Akhdan Najib

140810180061

PROGRAM STUDI S-1 INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

2020

Studi Kasus 1: Pencarian Nilai Maksimal

Buatlah programnya dan hitunglah kompleksitas waktu dari algoritma berikut:

Algoritma Pencarian Nilai Maksimal

```
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
```

Jawab :

```
/*
Nama        : Hafidh Akhdan N
NPM         : 140810180061
Kelas       : A
Program     : Studi Kasus 1 "Pencarian nilai maksimal"
*/

#include<iostream>
using namespace std;

int main(){
        int x[5]={31,45,100,4,35};
        int n= sizeof(x)/sizeof(x[0]);

        int maks = x[0];
        int i= 2;
```

```
        while (i<= n){
                if(x[i] > maks){
                        maks = x[i];
                }
                        i=i+1;
        }

        cout<<"Nilai maksimum adalah : "<<maks;

}
```

Kompleksitas waktu

T(n) = 2+2+(n-1)+(n-2)+2(n-2)+2n
= 6n-3

Studi Kasus 2: *Sequential Search*

Diberikan larik bilangan bulan $x_1, x_2, \ldots\ x_n$ 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 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 ← o  {y tidak ditemukan}
        endif
```

Jawab:

```
/*
Nama        : Hafidh Akhdan N
NPM         : 140810180061
Kelas       : A
Program     : Studi Kasus 2 "Sequental Search"
*/


#include<iostream>
using namespace std;

int main(){
        int x[5] = {1,2,3,4,5};
        int y = 4;
        int n = sizeof(x)/sizeof(x[0]);



        int i = 1;
```

```
    int idx;
    bool found = false;

    while(i<=n && !found){
        if(x[i] == y){
            found = true;
        }else
        i = i+1;
    }
    if(found == true){
        idx = i;
    }else
    idx = 0;
    cout<<"Hasil yang dicari index elemen : "<<idx;
}
```

Kompleksitas Waktu

Kasus terbaik (Best case) : ini terjadi bila $x_1 = y$

$T_{min}(n) = 1$

Kasus terburuk (Worst case): bila $x_n = y$ atau y tidak ditemukan.

$T_{max}(n) = n$

Kasus rata-rata (Average case): Jika *y* ditemukan pada posisi ke-*j*, maka operasi perbandingan ($a_k = y$) akan dieksekusi sebanyak *j*kali.

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

Studi Kasus 3: *Binary Search*

Diberikan larik bilangan bulan $x_1, x_2, \ldots\ x_n$ 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 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
```

Jawab :

```
/*
Nama        : Hafidh Akhdan N
NPM         : 140810180061
Kelas       : A
Program     : Studi Kasus 3 "Binary Search"
*/
```

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

int main(){
	int x[5]={1,2,3,4,5};
	int idx;
	int y = 5;
	int n = sizeof(x)/sizeof(x[0]);

	int i, j, mid;
	bool found;

	i = 1;
	j = n;
	found = false;
	while(!found && i<= j){
		mid = (i + j)/2;
		if (x[mid] == y){
			found = true;
		}
		else if(x[mid] < y){
			i = mid+1;
		}
		else{
			j = mid - 1;
		}


	}
	if(found == true){
```

```
            idx=mid;
        }else
        idx= 0;

        cout<<"Hasil yang dicari indeks elemen : "<<idx;

}
```

Kompleksitas Waktu

Kasus terbaik (best case) : Jika ditemukan pada arr[mid] atau indeks di tengah

$T_{min}(n) = 1$

Kasus terburuk (worst case) : Jika tidak ditemukan sama sekali

$T_{max}(n) = {}^{2}\log n$

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 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
```

Jawab :

```
/*
Nama        : Hafidh Akhdan N
NPM         : 140810180061
Kelas       : A
Program     : Studi Kasus 4 "Insertion Sort"
*/
```

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

int main(){
	int x[5]={2,3,1,5,4};
	int n = sizeof(x)/sizeof(x[0]);

	int i , j, insert;

	for(i=1; i<n; i++){
		insert= x[i];
		j = i - 1;

		while(j >= 0 && x[j] > insert){
			x[j+1] = x[j];
			j = j - 1;
		}
		x[j+1] = insert;
	}
	for(j = 0; j < n ; j++ ){
			cout<<x[j]<<" ";
	}
}
```

Kompleksitas waktu

Kasus terbaik (best case) : Jika array sudah terurut sehingga loop while tidak dijalankan
Kasus rata-rata (average case) : Jika sebagian elemen array sudah terurut
Kasus terburuk (worst case) : Jika array harus diurutkan sebanyak n kali = O (n2).

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
```

Jawab :

```
/*
Nama        : Hafidh Akhdan N
NPM         : 140810180061
Kelas       : A
Program     : Studi Kasus 5 "Selection Sort"
*/

#include<iostream>
using namespace std;

int main(){
    int x[5] = {1,7,3,9,5};
    int n = sizeof(x)/sizeof(x[0]);

    int i, j, imaks, temp;
```

```
    for ( i=2 ; i<n; i++){
        imaks = 1;
        for( j=2; j<i; j++){
            if ( x[j] > x[imaks]){
                imaks = j;
            }
        }
        temp = x[i];
        x[i] = x[imaks];
        x[imaks] = temp;
    }
    for (int i=0; i<n; i++){
        cout<<x[i]<<" ";
    }
}
```

Kompleksitas waktu

1. Jumlah operasi perbandingan elemen

   Untuk setiap loop ke-i,

   i = 1 → jumlah perbandingan = n-1
   i = 2 → jumlah perbandingan = n-2
   i = k → jumlah perbandingan = n-k
   i = n-1 → jumlah perbandingan = 1

Sehingga T(n) = (n-1) + (n-2) + … + 1 = n(n-1)/2

2. Jumlah operasi pertukaran

Untuk setiap loop ke-1 sampai n-1 terjadi satu kali pertukaran elemen sehingga T(n) = n-1.