LAPORAN PRAKTIKUM 2
KOMPLEKSITAS WAKTU DARI ALGORITMA

MATA KULIAH
ANALISIS ALGORITMA
D10G.4205 & D10K.0400601

SHEILA AZHAR ALMUFARIDA

140810180001

PROGRAM STUDI S-1 TEKNIK INFORMATIKA
DEPARTEMEN ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PADJADJARAN
MARET 2019

WORKSHEET 2

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
```

Jawaban Studi Kasus 1

(i) Operasi Assignment

Maks <- x1	1 kali
l <- 2	1 kali
Maks <- xi	n-1 kali
l <- l + 1	n-1 kali

Jumlah seluruh operasi assignment = 1 + 1 + n-1 + n-1 = 2 + 2(n-1) = 2n

(ii) Operasi Perbandingan

If xi maks then	n-1 kali

Jumlah seluruh operasi perbandingan = n-1 kali

(iii) Operasi Penjumlahan

i <- i + 1	n-1 kali

Jumlah seluruh operasi penjumlahan = n-1 kali

Kompleksitas waktu algoritma =2n + n-1 + n-1 = 2n + 2(n-1) = 4n - 2

Studi Kasus 2: *Sequential Search*

Diberikan larik bilangan bulat yang telah terurut menaik dan tidak ada elemen ganda. Buatlah program 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 : integer, y : integer, output idx : integer)
{ Mencari di dalam elemen elemen) tempat ditemukan diisi ke dalam idx.
  diisi dengan 0.
  Jika tidak ditemukan, maka idx'
  Input:

  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 Studi Kasus 2

(i) Operasi Assignment

i <- 1	1 kali
Found <- false	1 kali
Found <- true	n kali
i <- i + 1	n kali
Idx <- i	1 kali
Idx <- 0	1 kali

Jumlah Operasi Assignment = 1 + 1 + n + n + 1 + 1 = 4 + 2n

(ii) Operasi Perbandingan

If xi = y then	n kali
If found then	1 kali

Jumlah Operasi Perbandingan = n + 1 = 1 + n

(iii) Operasi Pertambahan

i <- i + 1	n kali

Jumlah Operasi Pertambahan = n

Kompleksitas Waktu Algoritma = 4 + 2n + 1 + n + n = 5 + 4n

1. Best Case, apabila ai=x
   Operasi perbandungan elemen (ai=x) hanya dilakukan satu kali maka
   Tmin(n)=1
2. Worst Case, apabila an=x atau x tidak ditemukan.
   Seluruh elemen larik dibandingkan, maka jumlah perbandingan elemen larik (ai=x) adalah
   Tmax(n)=n
3. Average Case, jika x ditemuka pada posisi ke-j, maka operasi perbandungan (ai=x) dilakukan sebanyak j kali. Jadi, kebutuhan waktu rata-rata algoritma ini adalah:
   Tavg(n)= (1+2+3+… +n)/n = ½ n(1+n) / n = (n+1)/2

Studi Kasus 3: *Binary Search*

Diberikan larik bilangan bulan yang telah terurut menaik dan tidak ada elemen ganda.

dan hitunglah kompleksitas waktu terbaik, terburuk, dan rata-

Buatlah programnya dengan C++

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                : integer, x : integer, output : idx : integer)
{ Mencari y di dalam elemen                 . Lokasi (indeks elemen) tempat y ditemukan diisi ke dalam idx.
  Jika y tidak ditemukan makai dx diisi dengan 0.
  Input:
  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 Studi Kasus 3

(i) Operasi Assignment

i <- 1	1 kali
j <- n	1 kali
found <- false	1 kali
mid ← (i + j) div 2	n kali
found ← true	1 kali
i <- mid + 1	n kali
j <- mid – 1	n kali
idx <- mid	1 kali
idx <- 0	1 kali

Jumlah operasi assignment = 6 + 3n

(ii) Operasi Pertambahan

mid ← (i + j) div 2	n kali
i <- mid + 1	n kali

Jumlah operasi pertambahan = 2n

(iii) Operasi Pengurangan

j <- mid – 1	n kali

Jumlah Operasi pengurangan = n

(iv) Operasi Perbandingan

If xmid = y then	n kali
If xmid < y then	n kali
If found then	1 kali

Jumlah operasi perbandingan = 1 + 2n

(v) Operasi Pembagian

mid ← (i + j) div 2	n kali

Jumlah operasi pembagian = n

Waktu kompleksitasnya = 6 + 3n + 2n + n + 1 + 2n + n = 9n + 7

1. Best Case, apabila x ditemukan pada elemen pertengahan amid, dan operasi perbandingan elemen (amid=x) yang dilakukan hanya satu kali. Pada kasus ini
   Tmid(n)=1

2. Worst Case, apabila elemen x ditemukan ketika ukuran larik =1. Pada kasusu ini, ukuran larik setiap kali memasuki looping while-do adalah n, n/2, n/4, n/8, … , 1 (sebanyak 2log n kali)
   Jumlah operasi perbandungan elemen (amid = x) adalah:
   Tmax(n)=2log n

3. Average Case, sulit ditentukan

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                    : integer)
Input:                                                  dengan metode insertion sort.
{ Mengurutkan elemen-elemen

                    (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 Studi Kasus 4

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 = n2-n

Tmin(n):
Tmin(n) = 3n-3 + 4n-4 + 1 = 7n - 6

Tmax(n):
Tmax(n) = 3n-3 + 4n-4 + n2-n = n2+6n-6

Tavg(n):
(Tmin(n) + Tmax(n)) / 2 = (7n–6 + n2+6n-6) / 2
Tavg(n) = (n2 + 13n - 12) / 2

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                    : integer)
{  Mengurutkan elemen-elemen

                                (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
```

Jawaban Studi Kasus 5

Operasi Perbandingan =

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

Operasi Pertukaran = n-1

Tmin(n):
Tmin(n) = (4n-4) + ½(n2-n) +1 ~ n2

Tmax(n):
Tmax(n) = ½(n2-n) + (n-1) ~ n2

Tavg(n):
(Tmin(n) + Tmax(n)) / 2 = (n2+ n2) / 2
Tavg(n) = n2

```
/*
Nama            : Sheila Azhar Almufarida
NPM             : 140810180001
Nama Program: Mencari nilai maksimum
*/

#include <iostream>

using namespace std;

main(){
	int n, maks, i, x[99];

	cout<<"-----------------------------------------"<<endl;
	cout<<"- - - Menghitung Nilai Maksimum - - -"<<endl;
	cout<<"-----------------------------------------"<<endl;

	for(;;){
		cout<<"Masukkan jumlah data: ";
		cin>>n;
		if(n<2){
			cout<<"Masukan minimal 2 data!"<<endl;
			continue;
		}
		break;
	}

	cout<<"Masukkan data:"<<endl;
	for (i=0; i<n; i++){
		cout<<"ke-"<<i+1<<": ";
		cin>>x[i];
	}

	i=1;
	maks=x[0];
	do {
		if(x[i]>maks){
			maks=x[i];
		}
		i=i+1;
	}
	while(i<n);
	cout<<" "<<endl;
	cout<<"Nilai terbesar dari data input adalah: "<<maks<<endl;
}
```

```
/*
Nama            : Sheila Azhar Almufarida
NPM             : 140810180001
Nama Program: Sequential Search
*/

#include<iostream>
using namespace std;

void PencarianBeruntun(int x[], int tanya, int n){
	bool ketemu=false;
	int i=0;
	while(i<n && !ketemu){
		if(x[i] == tanya){
			ketemu=true;
		}else{
			i++;
		}
	}

	int lokasi;
	if(ketemu){
		lokasi = i+1;
	} else{
		lokasi = 0;
	}
	cout<<"Elemen yang dicari berada di indeks: "<< lokasi;
}

main(){
	int n, tanya;
	cout<<"Masukkan banyak elemen array: ";
	cin>>n;

	int x[n];
	cout<<"Masukan elemen array berupa angka:"<<endl;
	for(int i=0; i<n; i++){
		cout<<"elemen["<<i+1<<"] = ";
		cin>>x[i];
	}

	cout<<endl<<"Masukan elemen array yang ingin dicari: ";
	cin>>tanya;

	PencarianBeruntun(x, tanya, n);
}
```

```
/*
Nama                : Sheila Azhar Almufarida
NPM                 : 140810180001
Nama Program: Binary Search
*/

#include <iostream>
#include <conio.h>
#include <iomanip>

using namespace std;


int data[7] = {1, 8, 2, 5, 4, 9, 7};
int cari;

void selection_sort()
{
    int temp, min, i, j;

    for(i=0; i<7;i++)
    {
        min = i;
        for(j = i+1; j<7; j++)
        {
            if(data[j]<data[min])
            {
                min=j;
            }
        }
        temp = data[i];
        data[i]  = data[min];
        data[min] = temp;
    }
}

void binarysearch()
{
    //searching
    int awal, akhir, tengah, b_flag = 0;
    awal = 0;
    akhir = 7;
    while (b_flag == 0 && awal<=akhir)
    {
        tengah = (awal + akhir)/2;
        if(data[tengah] == cari)
        {
            b_flag = 1;
            break;
        }
        else if(data[tengah]<cari)
            awal = tengah + 1;
        else
            akhir = tengah -1;
    }

     if(b_flag == 1)
        cout<<"\nData ditemukan pada urutan ke-"<<tengah+1<<endl;
    else
        cout<<"\nData tidak ditemukan\n";
}

int main()
```

```
{
    cout<<"\t   'BINARY SEARCH'"<<endl;
    cout<<"\t====================="<<endl;
    cout<<"\nData           : ";
    //tampilkan data awal
    for(int x = 0; x<7; x++)
        cout<<setw(3)<<data[x];
    cout<<endl;

    cout<<"\nMasukkan data yang ingin Anda cari : ";
    cin>>cari;
    cout<<"\nData diurutkan : ";

    //urutkan data dengan selection sort
    selection_sort();
    //tampilkan data setelah diurutkan
    for(int x = 0; x<7;x++)
        cout<<setw(3)<<data[x];

    cout<<endl;
    binarysearch();

 }
```

```
/*
Nama                : Sheila Azhar Almufarida
NPM                 : 140810180001
Nama Program: Insertion Search
*/

#include <iostream>
#include <conio.h>

using namespace std;

int data[10],data2[10];
int n;

void tukar(int a, int b)
{
 int t;
 t = data[b];
 data[b] = data[a];
 data[a] = t;
}

void insertion_sort()
{
 int temp,i,j;
 for(i=1;i<=n;i++){
  temp = data[i];
  j = i -1;
  while(data[j]>temp && j>=0)
  {
   data[j+1] = data[j];
   j--;
  }
 data[j+1] = temp;
 }
}

main()
{
 cout<<"===PROGRAM INSERTION SORT==="<<endl;
 cout<<"Masukkan Jumlah Data : ";
 cin>>n;
 for(int i=1;i<=n;i++)
 {
  cout<<"Masukkan data ke "<<i<<" : ";
  cin>>data[i];
  data2[i]=data[i];
 }

 insertion_sort();
 cout<<"Data Setelah di Sort : ";
 for(int i=1; i<=n; i++)
 {
  cout<<" "<<data[i];
 }
 cout<<"\n\nSorting dengan insertion sort selesai";
 getch();
}
```

```
/*
Nama                : Sheila Azhar Almufarida
NPM                 : 140810180001
Nama Program: Selection Search
*/

#include <iostream>
#include <conio.h>

using namespace std;

int data[10],data2[10];
int n;

void tukar(int a, int b)
{
 int t;
 t = data[b];
 data[b] = data[a];
 data[a] = t;
}
void selection_sort()
{
 int pos,i,j;
 for(i=1;i<=n-1;i++)
 {
  pos = i;
  for(j = i+1;j<=n;j++)
  {
   if(data[j] < data[pos]) pos = j;
  }
  if(pos != i) tukar(pos,i);
 }
}

main()
{
 cout<<"===PROGRAM SELECTION SORT==="<<endl;

 cout<<"Masukkan Jumlah Data : ";
 cin>>n;
 for(int i=1;i<=n;i++)
 {
  cout<<"Masukkan data ke "<<i<<" : ";
  cin>>data[i];
  data2[i]=data[i];
 }

 selection_sort();
 cout<<"Data Setelah di Sort : ";
 for(int i=1; i<=n; i++)
 {
  cout<<" "<<data[i];
 }
 cout<<"\n\nSorting dengan selection sort selesai";
 getch();
}
```