TUGAS 2 PRAKTIKUM
ANALISIS ALGORITMA

Asisten Praktikum:

Faradilla Azranur, Felia Sri Indriyani, Agnes Hata

Sitti Ufairoh Azzahra

140810180002

Dikumpulkan tanggal

4 Maret 2020

PROGRAM STUDI S-1 TEKNIK 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
        {i > n}
```

Jawaban Studi Kasus 1

Operasi pengisian nilai (*assignment*)

maks ← x_1	1 kali
i ← 2,	1 kali
maks ← x_1	n-1 kali (worst case)
	0 kali (best case)
i ← i + 1,	n kali

Jumlah seluruh operasi pengisian nilai (*assignment*) adalah

$t_1 = 1 + 1 + n - 1 + n = 2n + 1 \ (worst\ case)$

$t_1 = 1 + 1 + 0 + n = n + 2 \ (best\ case)$

Operasi penjumlahan

$i + 1_{k,}$	n kali

Jumlah seluruh operasi penjumlahan adalah

$t_2 = n$

$$T_{min}(n) = t_1 + t_2 = n + 2 + n = 2n + 2$$

$$T_{max}(n) = t_1 + t_2 = 2n + 1 + n = 3n + 1$$

Source Code Studi Kasus 1

```
/*  Nama        : Sitti Ufairoh Azzahra
    NPM         : 140810180002
    Kelas       : B
    Deskripsi   : Program Pencarian Nilai Maksimal
*/

#include <iostream>
using namespace std;

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

    cout<<"Masukkan jumlah elemen : ";cin>>n;
    for(int i=0;i<n;){
        cout<<"Bilangan ke - "<<++i<<" : ";cin>>x[i];
    }

    maks = x[1];

    i = 2;

    do{
        if(x[i]>maks){
            maks = x[i];
        }
        i = i+1;
    }while(i<=n);
    cout<<"output = " <<maks<<endl;
}
```

Hasil Program

```
Masukkan jumlah elemen : 5
Bilangan ke - 1 : 23
Bilangan ke - 2 : 7
Bilangan ke - 3 : 19
Bilangan ke - 4 : 16
Bilangan ke - 5 : 70
output = 70
```

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 x_1, x_2, ... x_n : integer,  y : integer, output idx : integer)
{  Mencari y di dalam elemen x_1, x_2, ... x_n.
   Lokasi (indeks elemen) tempat y ditemukan diisi ke dalam idx.
   Jika y tidak ditemukan, makai idx diisi dengan 0.
   Input: x_1, x_2, ... x_n
   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 x_i = 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

Kompleksitas waktu

Best Case :

i ←1	1 kali	
found ←false	1 kali	
found ←true	1 kali	
idx ←I	1 kali	

$$T_{min}(n) = 1 + 1 + 1 + 1 = 4$$

Average Case :

i ←1	1 kali	
found ←false	1 kali	
i ←i + 1		½ n kali
found ←true	1 kali	
idx ←I	1 kali	

$$T_{avg}(n) = 1 + 1 + \frac{1}{2}n + 1 + 1 = \frac{1}{2}n + 4$$

Worst Case :

i ←1	1 kali	
found ←false	1 kali	
i ←i + 1		n kali
found ←true	1 kali	
idx ←I	1 kali	

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

Source Code Studi Kasus 2

```
/*  Nama        : Sitti Ufairoh Azzahra
    NPM         : 140810180002
    Kelas       : B
    Deskripsi   : Program Sequential Search
*/

#include <iostream>
using namespace std;

int main() {
  int n,y,x[10],idx;
  int i = 0;
  bool found = false;

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

    cout << "\nMasukkan yang dicari : ";cin >> y;

    while ((i < n) && (!found)){
        if (x[i] == y)
            found = true;
        else
            i++;
    }

    if (found)
        idx = i+1;
    else
        idx = 0;

    cout << "\nYang dicari berada di urutan : " << idx << endl;

    return 0;
}
```

Hasil Program

```
Masukkan Jumlah Elemen : 3
Bilangan ke - 1 : 12
Bilangan ke - 2 : 34
Bilangan ke - 3 : 52

Masukkan yang dicari : 34

Yang dicari berada di urutan : 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
          {i < n or found}
          If found then {y ditemukan}
                    idx ← i
          else
                    idx ← 0          {y tidak ditemukan}
          endif
```

Jawaban Studi Kasus 3

Best Case :

i ←1	1 kali
j ←n	1 kali
found ←false	1 kali
mid ←(i + j) div2	1 kali
found ←true	1 kali
Idx ←mid	1 kali

$T_{min}(n) = 1 + 1 + 1 + 1 + 1 + 1 = 6$

Average Case :

i ←1	1 kali
j ←n	1 kali
found ←false	1 kali
mid ←(i + j) div2	½ n + 1 kali
i ←mid + 1 or j ←mid –1	½ n kali
found ←true	1 kali
Idx ←mid	1 kali

$T_{avg}(n) = 1 + 1 + 1 + \frac{1}{2}n + 1 + \frac{1}{2}n + 1 + 1 = n + 6$ `

Worst Case :

i ←1	1 kali
j ←n	1 kali
found ←false	1 kali
mid ←(i + j) div2	n + 1 kali
i ←mid + 1 or j ←mid –1	n kali
found ←true	1 kali
Idx ←mid	1 kali

$T_{max}(n) = 1 + 1 + 1 + n + 1 + n + 1 + 1 = 2n + 6$

Source Code Studi Kasus 3

```
/*  Nama        : Sitti Ufairoh Azzahra
    NPM         : 140810180002
    Kelas       : B
    Deskripsi   : Program Binary Search
*/
#include <iostream>
using namespace std;

int main(){
    int i,j,mid,n,x[100],y,idx;
    bool found=false;

    cout<<"Masukkan Jumlah Elemen  : ";cin>>n;
    for (int i=0; i<n;){
        cout<<"Bilangan ke - "<<++i<<" : ";cin>>x[i];
    }
    i=1;j=n;
    cout<<"\nMasukkan yang dicari : ";cin>>y;
    do{
        mid=(i+j)/2;
        if (x[mid]==y)
            found=true;
        else if (x[mid]<y)
            i=mid+1;
        else
            j=mid-1;
    }while(found==false&&i<=j);

    if (found==true)
        idx=mid;
    else
        idx=0;

    cout<<"\nBilangan berada di index ke - "<<idx;
}
```

Hasil Program

```
Masukkan Jumlah Elemen  : 5
Bilangan ke - 1 : 43
Bilangan ke - 2 : 11
Bilangan ke - 3 : 99
Bilangan ke - 4 : 78
Bilangan ke - 5 : 10

Masukkan yang dicari : 99

Bilangan berada di index ke - 3
```

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, ... x_n : integer)
{       Mengurutkan elemen-elemen x_1, x_2, ... x_n dengan metode insertion sort.
        Input: x_1, x_2, ... x_n
        OutputL x_1, x_2, ... x_n (sudah terurut menaik)
}
Deklarasi
        i, j, insert : integer
Algoritma
        for i ← 2 to n do
            insert ← x_i
            j ← i
            while (j < i) and (x[j-i] > insert) do
                x[j]← x[j-1]
                j←j-1
            endwhile
            x[j] = insert
        endfor
        {i < n or found}

        If found then {y ditemukan}
                idx ← i
        else
                idx ← o{y tidak ditemukan}
        endif
```

Jawaban Studi Kasus 4

Best Case :

For i ←2 to n do	1 kali
insert ←xi	n kali
j ←i	n kali
x[j] = insert	n kali

$$T_{min}(n) = 1 + n + n + n = 3n + 1$$

Average Case :

For i ←2 to n do	1 kali
insert ←xi	n kali
j ←I	n kali
x[j]←x[j-1]	n * ½ n kali
j←j-1	n * ½ n kali
x[j] = insert	n kali

$$T_{avg}(n) = 1 + n + n + \frac{1}{2}\,n^2 + \frac{1}{2}\,n^2 + n = n^2 + 3n + 1$$

Worst Case :

For i ←2 to n do	1 kali
insert ←xi	n kali
j ←i	n kali
x[j]←x[j-1]	n * n kali
j←j-1	n * n kali
x[j] = insert	n kali

$$T_{max}(n) = 1 + n + n + \ n^2 + \ n^2 + n = 2n^2 + 3n + 1$$

Source Code Studi Kasus 4

```
/*  Nama        : Sitti Ufairoh Azzahra
    NPM         : 140810180002
    Kelas       : B
    Deskripsi   : Program Insertion Sort
*/

#include <iostream>
using namespace std;

int main(){
    int n,x[10],j;
    int insert;

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

    cout << "\nSebelum di Sorting : ";
    for (int i = 0; i < n; i++)
        cout << x[i] << " ";
        cout << endl;

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

    cout << "Setelah di Sorting : ";
    for (int i = 0; i < n; i++)
        cout << x[i] << " ";

    return 0;
}
```

Hasil Program

```
Masukkan Jumlah Elemen : 3
Bilangan ke - 1 : 11
Bilangan ke - 2 : 2
Bilangan ke - 3 : 99

Sebelum di Sorting : 11 2 99
Setelah di Sorting : 2 11 99
```

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 x_1, x_2, ... x_n : integer)
{  Mengurutkan elemen-elemen x_1, x_2, ... x_n dengan metode selection sort.
   Input: x_1, x_2, ... x_n
   OutputL x_1, x_2, ... x_n (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 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
```

Jawaban Studi Kasus 5

Best Case :

for i ←n downto 2 do	1 kali
imaks ←1	n kali
for j ←2 to i do	n kali
imaks ←j	n*1 kali
temp ←xi	n kali
xi←ximaks	n kali
ximaks←temp	n kali

$$T_{min}(n) = 1 + n + n + n * 1 + n + n + n = 6n + 1$$

Average Case :

for i ←n downto 2 do	1 kali
imaks ←1	nkali
for j ←2 to i do	n kali
imaks ←j	n * ½ n kali
temp ←xi	n kali
xi←ximaks	n kali
ximaks←temp	n kali

$$T_{avg}(n) = 1 + n + n + \frac{1}{2}n^2 + n + n + n = \frac{1}{2}n^2 + 5n + 1$$

Worst Case :

for i ←n downto 2 do	1 kali
imaks ←1	n kali
for j ←2 to i do	n kali
imaks ←j	n * n kali
temp ←xi	n kali
xi←ximaks	n kali
ximaks←temp	n kali

$$T_{max}(n) = 1 + n + n + n^2 + n + n + n = n^2 + 5n + 1$$

Source Code Studi Kasus 5

```
/*  Nama        : Sitti Ufairoh Azzahra
    NPM         : 140810180002
    Kelas       : B
    Deskripsi   : Program Selection Sort
*/

#include <iostream>
using namespace std;

int main(){
    int n,x[10],imaks,temp;

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

    cout << "\nSebelum di Sorting : ";
    for (int i = 0; i < n; i++)
        cout << x[i] << " ";
        cout << endl;

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

    cout << "Setelah di Sorting : ";
    for (int i = 0; i < n; i++)
        cout << x[i] << " ";

    return 0;
}
```

Hasil Program

```
Masukkan Jumlah Elemen : 4
Bilangan ke - 1 : 17
Bilangan ke - 2 : 4
Bilangan ke - 3 : 2
Bilangan ke - 4 : 98

Sebelum di Sorting : 17 4 2 98
Setelah di Sorting : 2 4 17 98
```