Nama : Hana Meilina Fauziyyah
NPM : 140810180012
Kelas : B

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 :

- Program

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

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

        cout<<"================================="<<endl;
        cout<<"======Mencari Nilai Maksimal======"<<endl;
        cout<<"================================="<<endl;

        for(;;){
         cout<<"Masukkan jumlah data : ";
                cin>>n;
                if(n<2){
                        cout<<"Minimal berisi 2 data"<<endl;
                        continue;
                }
                break;
        }
        cout<<"Masukkan data : "<<endl;
        for (i=0; i<n; i++){
                cout<<"Data 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<<"Nilai maksimal adalah : "<<maks<<endl;
}
```

- Kompleksitas

  Operator Assignment

  Baris 1 1 kali

  Baris 2 1 kali

  Baris 5 n-1 kali

  Baris 7 n-1 kali

  T1 = 1 + 1 + (n-1) + (n-1) = 2n

  Operator Penjumlahan

  Baris 7 n-1 kali

  T2 = n-1

  Operator Perbandingan

  Baris 4 n-1 kali

  T3 = n-1

  Jadi T(n) = t1 + t2 + t3 = 2n + (n-1) + (n-1) = 4n-2

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 ← 0 {y tidak ditemukan}
        endif
```

Jawab :

- Program

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

main(){
    int n, cari, A[100], index, jwb;
       bool found = false;
       cout<<"==========SEQUENTIAL SEARCH=========="<<endl;
       cout<<"Masukan banyak data =  ";
    cin>>n;
       cout<<"====================================="<<endl;

       for(int i=0; i<n; i++){
```

```
            cout<<"Data ke-" << i+1 << " : ";
            cin>>A[i];
        }

        cout<<"Masukan data yang dicari : ";
        cin>>cari;
        cout<<"====================================="<<endl;


        for(int i=0; i<n; i++){
            if(A[i] == cari){
                    found = true;
                index = i;
                i = n;
            }
        }
        if(found == true){
            cout<<"Data ditemukan pada data ke-"<<index+1;
            cout<<endl;
        }
        else{
            cout<<"Data tidak Ada!"<<endl;
        }
        cout<<"====================================="<<endl;
        cout<<"\n";
}
```

- Kompleksitas

Operator Assignment

Baris 1	1 kali
Baris 2	1 kali
Baris 5	n kali
Baris 7	n kali
Baris 12	1 kali
Baris 14	1 kali

T1 = 1 + 1 + n + n + 1 + 1 = 4 + 2n

Operator Perbandingan

Baris 4	n kali
Baris 11	1 kali

T2 = n + 1 = n + 1

Operator Penjumlahan

Baris 7	n kali

T3 = n

T(n) = t1 + t2 + t3 = 4 + 2n + n + 1 + n = 5 + 4n

- Case
    1. Best Case
       Apabila ai=x. Operasi perbandingan elemen ai=x hanya dilakukan satu kali maka Tmin(n) = 1
    2. Average Case
       Jika x ditemukan pada posisi ke-j, maka operasi perbandingan (ai=x) dilakukan sebanyak j kali, jadi Tavg(n) = (1+2+3+…+n)/n =1/2 n(1+n)/n = (n+1)/2
    3. Worst Case
       Apabila an=x atau x tidak ditemukan. Seluruh elemen larik dibandingkan, maka jumlah perbandingan elemen larik (ai=x) adalah Tmax(n) = n

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 :

- Program

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

main(){
        int n, i, array[100], cari, awal, akhir, tengah;
        cout<<"==========Binary Search============="<<endl;
```

```
        cout<<"Masukkan banyak data : ";
        cin>>n;
        cout<<"==================================="<<endl;

        for (i=0; i<n; i++){
                cout<<"Data ke-"<<i+1<<" :";
                cin>>array[i];
        }

        cout<<"Masukkan data yang di cari : ";
        cin>>cari;
        awal=0;
        akhir=n-1;
        cout<<"==================================="<<endl;

        while(awal <= akhir){
         tengah = (awal+akhir)/2;
                if(array[tengah]<cari){
                        awal = tengah + 1;
         }
                else if(array[tengah]==cari){
                        cout<<"Ditemukan pada data ke-"<<tengah+1<<"\n";
                        break;
                }
                else{
                         akhir = tengah - 1;
                }
                tengah = (awal + akhir)/2;
        }

        if(awal>akhir){
                cout<<"Data tidak ditemukan"<<endl;
        }
        cout<<"==================================="<<endl;
        //getch();
}
```

- Kompleksitas

Operator Assignment

Baris 1	1 kali
Baris 2	1 kali
Baris 3	1 kali
Baris 5	n kali
Baris 7	1 kali
Baris 10	n kali
Baris 12	n kali
Baris 18	1 kali
Baris 20	1 kali

T1 =1 + 1 + 1 + n + 1 + n + n + 1 + 1 = 6 + 3n

Operator Perbandingan

Baris 6	n kali
Baris 9	n kali

Baris 17 1 kali
T2 = n + n + 1 = 2n + 1

Operator Penjumlahan
Baris 5 n kali
Baris 10 n kali
T3 = 2n

Operator Pengurangan
Baris 12 n kali
T4 = n

Operator Pembagian
Baris 5 n kali
T5 = n

T(n) = t1 + t2 + t3 t4 = 6 + 3n + 2n + 1 + 2n + n + n= 7 + 9n

- Case
  1. Best Case
     Apabila x ditemukan pada elemen pertengahan amid dan operasi perbandingan elemen (amid=x) yang hanya dilakukan satu kali. Pada kasus ini Tmid(n)=1
  2. Average Case
     Sulit ditentukan
  3. Worst Case
     Apabila elemen x ditemukan ketika ukuran larik = 1. Pada kasus ini, ukuran larik setiap kali memasuki looping while-do adalah n, n/2, n/4, n/8, …, 1 (sebanyak 2lon n kali), jumlah operasi perbandingan elemen (amid = x) adalah Tmax(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 :

- Program

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

int data[100],data2[100],n;
void insertSort();

main(){
        cout<<"=========Insertion Short==========="<<endl;
        cout<<"Masukkan Jumlah Data : ";
        cin>>n;
        cout<<"================================="<<endl;

        for(int i=1;i<=n;i++){
          cout<<"Masukkan data ke-"<<i<<" : ";
          cin>>data[i];
          data2[i]=data[i];
        }
        cout << "=================================" << endl;
        insertSort();
        cout<<"Data Setelah di Urutkan : "<<endl;

        for(int i=1; i<=n; i++){
          cout<<data[i]<<" ";
        }
        cout<<endl;
        cout<<"================================="<<endl;
}

void insertSort(){
        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;
        }
}
```

- Kompleksitas

Operator Assignment
T1 = 2(n-1) + n-1 = 3n-3

Operator Perbandingan
T2 = 2*((n-1)+(n-1)) = 2*(2n-2) = 4n-4

Operator Pertukaran
T3 = (n-1)*n = n^2- n

Tmin(n) = 3n-3 + 4n-4 + 1 = 7n-6
Tmax(n) = 3n-3 + 4n-4 + n^2-n = n^2+ 6n – 6
Tavg(n) = (Tmin(n) + Tmax(n)) / 2 = (7n-6 + n^2+ 6n – 6) / 2 = n^2+ 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 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 :

- Program

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

int data[100],datdata[100], n;
void change(int s, int r);
void selectionSort();

main(){
        cout<<"==========Selection Short==============";
        cout<<endl;
        cout<<"Masukkan jumlah data : ";
        cin>>n;
        cout<<"====================================="<<endl;

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

        selectionSort();
        cout<<"======================================="<<endl;
        cout<<"Data Setelah di Urutkan : "<<endl;
        for(int i=1; i<=n; i++){
                cout<<" "<<data[i];
        }
        cout<<endl;
```

```
        cout<<"====================================";
}

void change(int s, int r){
        int t;
        t = data[r];
        data[r] = data[s];
        data[s] = t;
}

void selectionSort(){
        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) change(pos,i);
    }
}
```

- Kompleksitas

  Operator Perbandingan

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

  Operator Pertukaran = n-1

  Tmin(n) = (4n-4) + ½ (n^2 - n) + 1 ~ n^2

  Tmax(n) = ½ (n^2 - n) + (n-1) ~ n^2

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