Nama : Sharashena Chairani
NPM : 140810180022
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
```

Jawaban Studi Kasus 1

Kompleksitas :

Operator Assignment : t1

Baris 1 : 1 Kali

Baris 2 : 1 Kali

Baris 5 : n-1 Kali

Baris 7 : n-1 Kali

T1 = 2 + n-1 + n-1 = 2n

T1 =2n

Operator Perbandingan : t2

Baris 4 : n-1 Kali

T2 = n-1

Operator Penjumlahan : t3

Baris 7 : n-1 Kali

T3 : n-1

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

Kompleksitas waktu Algoritma T(n) = 4n -2

Studi Kasus 2: *Sequential Search*

Diberikan larik bilangan bulan , , ... yang telah terurut menaik dan tidak ada elemen ganda. Buatlah programnya dengan C++ dan hitunglah kompleksitas waktu terbaik, terburuk, dan ratarata 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 , , ... . Lokasi (indeks elemen) tempat  ditemukan diisi ke dalam idx. Jika  tidak
    ditemukan, makai idx diisi dengan 0.
    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

Kompleksitas

Operator Assignment : t1

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

T1 = 4 + 2n

Operator Perbandingan : t2

Baris 4 : n Kali

Baris 11 : 1 Kali

T2 = n + 1

Operator Penjumlahan : t3

Baris 7 : n Kali

T3 : n

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

Kompleksitas waktu algoritma T(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 ditemukan 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. Buatlah programnya dengan C++ dan hitunglah kompleksitas waktu terbaik, terburuk, dan ratarata 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 x_mid = y then
                                found ←
true                 else
                    if x_mid < y then  {mencari di bagian kanan}
      i ← mid + 1                      {mencari di bagian kiri}
                 else
      j ← mid – 1
                endif
             endif
      endwhile
      {found or i > j }

      If found then
             Idx ← mid
      else
             Idx ← 0
      endif
```

Jawaban Studi Kasus 3

Operator Assignment : t1

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 + 1 + 1 + 1 + n + n + n

T1 = 6 + 3n

Operator Perbandingan : t2

Baris 6 : n Kali

Baris 9 : n Kali

Baris 17 : 1 Kali

T2 = n + n + 1

T2 = 2n + 1

Operator Penjumlahan : t3

Baris 5 : n Kali

Baris 10 : n Kali

T3 : n + n

T3 : 2n

Operator Pengurangan : t4

Baris 12 : n Kali

T4 : n

Operator Pembagian : t5

Baris 5 : n Kali

T5 = n

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

Waktu Kompleksitas T (n) = 7 + 9n

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)
{   Mengurutkan elemen-elemen    ,    , ...    dengan metode insertion sort.
    Input:    ,    , ...
    OutputL    ,    , ...    (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

Operator Assignment : t1

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

Operasi Perbandingan : t2

T2 = 2*((n-1) + (n-1))

T2 = 2*(2n-2)

T2 = 4n-4

Operasi Pertukaran : t3

T3 = (n-1) * n

T3 = n^2 – n

Tmin(n) = 3n-3 + 4n-4 + 1

Tmin(n) = 7n-6

Tmax(n) = 3n-3 + 4n-4 + n^2 – n

Tmax(n)= n^2 + 6n – 6

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

Tavg(n) = (7n-6 + n^2 + 6n – 6) / 2

Tavg(n) = 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    ,   , ...     : integer)
{ Mengurutkan elemen-elemen      ,   , ...     dengan metode selection sort.
   Input:    ,   , ...
   OutputL    ,   , ...     (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 =

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

Operasi 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

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

Tavg(n) = n^2

PROGRAM STUDI KASUS 1-5

```
/*
Nama          : Sharashena Chairani
NPM           : 140810180022
Kelas                 : B
Nama Program    : Studi Kasus 1. Nilai Maksimal
*/

#include <iostream>
using namespace std;

main(){
	//deklarasi
	int jumlah, maksimum, i, x[50];

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

	for(;;){
		cout<<endl;
		cout<<"Masukkan jumlah elemen: ";
		cin>>jumlah;

		if(jumlah<2){
			cout<<"Maaf, data minimal 2"<<endl;
			continue;
		}
		break;
	}

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

	//Algoritma
	i=1;
	maksimum=x[0];
	do {
```

```
                if(x[i]>maksimum){
                        maksimum=x[i];
                }
                i=i+1;
        }
        while(i<jumlah);
        cout<<endl;
        cout<<"Nilai terbesar adalah: "<<maksimum<<endl;
        cout<<"--------------------------------"<<endl;
}


/*
Nama          : Sharashena Chairani
NPM           : 140810180022
Kelas                 : B
Nama Program  : Studi Kasus 2. Sequential Search
*/

#include <iostream>
using namespace std;

main()
{
   int jumlah, carielemen, x[100], index, jwb;
        bool found = false;
        cout << "~~~~~~~~~~Sequential Search~~~~~~~~~~";
        cout << "\nMasukan banyak elemen =  ";
   cin  >> jumlah;
        cout << "\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

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

        cout << "\nMasukan elemen yang dicari : ";
        cin  >> carielemen;
        cout << "\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;


        for(int i=0; i<jumlah; i++){
           if(x[i] == carielemen){
                   found = true;
               index = i;
                  i = jumlah;
             }
```

```
        }
        if(found == true){
          cout << "Elemen ditemukan pada data ke-" << index+1;
        }
        else{
          cout << "Elemen tidak Ada!";
        }
        cout << "\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;
        cout << "\n";
        system("pause");
}


/*
Nama          : Sharashena Chairani
NPM           : 140810180022
Kelas                 : B
Nama Program  : Studi Kasus 3. Binary Search
*/

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

using namespace std;
int main()
{
        int jumlah, i, x[100], cariElemen, begin, end, middle;
        cout << "~~~~~~~~~~Binary Search~~~~~~~~~~~~~~"<<endl;
        cout <<"\nMasukkan banyak elemen : ";
        cin  >>jumlah;
        cout << "\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

        for (i=0; i<jumlah; i++)
        {
                cout<<"Elemen ke-"<<i+1<<" :";
                cin>>x[i];
        }
        cout <<"\nMasukkan elemen yang di cari :";
        cin  >>cariElemen;
        begin  = 0;
        end = jumlah-1;
        cout << "\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

        while (begin <= end)
        {
                middle = (begin+end)/2;
                if(x[middle] < cariElemen)
                {
```

```
                begin = middle + 1;

            }
            else if(x[middle] == cariElemen)
            {
                cout<<"Data "<<cariElemen<<" ditemukan pada elemen ke-
"<<middle+1<<"\n";
                break;
            }
            else
            {
                 end = middle - 1;
            }
            middle = (begin + end)/2;
        }

        if(begin > end)
        {
            cout<<"Data "<<cariElemen<<" Tidak Ditemukan!";
        }
        cout << "\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"<<endl;
        getch();
}



/*
Nama          : Sharashena Chairani
NPM           : 140810180022
Kelas                 : B
Nama Program    : Studi Kasus 4. Insertion Sort
*/

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

using namespace std;

int x[100],y[100],jumlah;

//Fungsi Insertion Discending
void Insert_sort()
{
        int temp,i,j;
        for(i=1;i<=jumlah;i++){
           temp = x[i];
                j = i -1;
           while(x[j]>temp && j>=0){
```

```
                x[j+1] = x[j];
            j--;
        }
        x[j+1] = temp;
    }
}
int main()
{
    cout <<"\~~~~~~~~~~Insertion Short~~~~~~~~~~"<<endl;
    cout <<"\nMasukkan Jumlah x : ";
    cin  >>jumlah;
    cout <<"\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

    for(int i=1;i<=jumlah;i++){
      cout<<"Masukkan x ke-"<<i<<" : ";
      cin>>x[i];
      y[i]=x[i];
    }
    cout << "\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;
    Insert_sort();
    cout<<"\nx Diurutkan dengan Insertion Sort : "<<endl;

    for(int i=1; i<=jumlah; i++)
    {
      cout<<x[i]<<" ";
    }
    cout << "\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"<<endl;
    getch();
}


/*
Nama        : Sharashena Chairani
NPM         : 140810180022
Kelas               : B
Nama Program    : Studi Kasus 5. Selection Sort
*/

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

using namespace std;

int x[10],y[10];
int n;

void tukar(int a, int b)
{
```

```
 int t;
 t = x[b];
 x[b] = x[a];
 x[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(x[j] < x[pos]) pos = j;
  }
  if(pos != i) tukar(pos,i);
 }
}

main()
{
 cout <<"\~~~~~~~~~~Selection sort~~~~~~~~~~"<<endl;

 cout<<"\nMasukkan Jumlah x : ";
 cin>>n;
 for(int i=1;i<=n;i++)
 {
  cout<<"\nMasukkan x ke "<<i<<" : ";
  cin>>x[i];
  y[i]=x[i];
 }

 selection_sort();
 cout<<"\nx Setelah Selection Sort : ";
 for(int i=1; i<=n; i++)
 {
  cout<<" "<<x[i];
 }
 cout<<"\n\nSorting dengan selection sort selesai";
 cout << "\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;
 getch();
}
```