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 🡨 x1 1 kali

i 🡨 2, 1 kali

maks 🡨 x1 n-1 kali (worst case)

0 kali (best case)

i 🡨 i + 1, n kali

Jumlah seluruh operasi pengisian nilai (*assignment*) adalah

Operasi penjumlahan

i + 1k, n kali

Jumlah seluruh operasi penjumlahan adalah

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];

        }

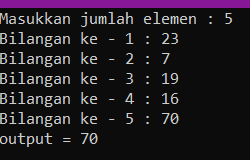
        i = i+1;

    }while(i<=n);

    cout<<"output = " <<maks<<endl;

}

Hasil Program



# 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 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 .

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 waktu

### Best Case :

i 1 1 kali

found false 1 kali

found true 1 kali

idx I 1 kali

### Average Case :

i 1 1 kali

found false 1 kali

i i + 1 ½ n kali

found true 1 kali

idx I 1 kali

### Worst Case :

i 1 1 kali

found false 1 kali

i i + 1 n kali

found true 1 kali

idx I 1 kali

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

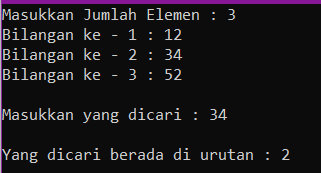
        idx = 0;

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

    return 0;

}

Hasil Program



# 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 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 : 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

{*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

### 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

`

### 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

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;

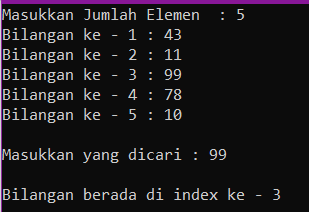
    else

        idx=0;

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

}

Hasil Program



# 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

{*i < n or found*}

If found then {*y ditemukan*}

idx 🡨 i

else

idx 🡨 0 {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

### 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

### 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

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 : ";

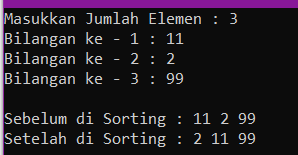
    for (int i = 0; i < n; i++)

        cout << x[i] << " ";

    return 0;

}

Hasil Program



# 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

### 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

### 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

### 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

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

