

**LAPORAN PRAKTIKUM ANALISIS ALGORITMA  
KOMPLEKSITAS WAKTU DARI ALGORITMA**



Oleh

Shalvina Zahwa Aulia (140810180052)

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

## Studi Kasus 1: Pencarian Nilai Maksimal

Buatlah programnya dan hitunglah kompleksitas waktu dari algoritma berikut:

### Algoritma Pencarian Nilai Maksimal

```
procedure CariMaks(input  $x_1, x_2, \dots, x_n$ : integer, output maks: integer)
{   Mencari elemen terbesar dari sekumpulan elemen larik integer  $x_1, x_2, \dots, x_n$ . Elemen
    terbesar akan disimpan di dalam maks
    Input:  $x_1, x_2, \dots, x_n$ 
    Output: maks (nilai terbesar)
}
```

#### Deklarasi

$i$  : integer

#### Algoritma

```
    maks  $\leftarrow x_1$ 
 $x_1$      $i \leftarrow 2$ 
    while  $i \leq n$  do
        if  $x_i > \text{maks}$  then
            maks  $\leftarrow x_i$ 
        endif
         $i \leftarrow i + 1$ 
    endwhile
```

### Jawaban Studi Kasus 1

#### 1. Operasi assignment :

$\text{maks} \leftarrow x_1$	1 kali
$i \leftarrow 2$	1 kali
$\text{maks} \leftarrow x_i$	$n-1$ kali
	0 kali
$i \leftarrow i+1$	$n-1$ kali

jadi  $t_1 = 1+1+(n-1)+(n-1) = 2n$

#### 2. Operasi penjumlahan

$i + 1$	$n-1$ kali
---------	------------

jadi  $t_2 = n-1$

### 3. Operasi perbandingan

If  $x_i > \text{maks}$        $n-1$  kali

Jadi  $t_3 = n-1$

$$T_{\max}(n) = t_1 + t_2 + t_3 = 2n + n - 1 + n - 1 = 4n - 2$$

### Program

```
/*  
Nama      : Shalvina Zahwa Aulia  
Kelas    : B  
Deskripsi : Program mencari nilai maksimum  
*/  
  
#include <iostream>  
using namespace std;  
  
int main(){  
    int i, n, maks;  
  
    cout << "Jumlah data : "; cin >> n;  
    int x[n];  
    for(i=1; i<=n; i++){  
        cout << "x[" << i << "] : "; cin >> x[i];  
    }  
    maks = x[1];  
    i = 2;  
    while(i<n){  
        if(x[i] > maks){  
            maks = x[i];  
        }  
        i = i+1;  
    }  
    cout << "Maks = " << maks;  
}
```

```
"C:\Users\HP\Documents\UNPAD\Himatif\K U L I A H\Semester4\analisisAlg  
Jumlah data : 7  
x[1] : 5  
x[2] : 3  
x[3] : 8  
x[4] : 3  
x[5] : 9  
x[6] : 1  
x[7] : 2  
Maks = 9  
Process returned 0 (0x0)   execution time : 13.370 s  
Press any key to continue.
```

### 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, maka 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  $\square$  1

found  $\square$  false

while (i  $\leq$  n) and (not found) do

if  $x_i = y$  then

        found  $\square$  true

else

        i  $\square$  i + 1

endif

endwhile

{ i < n or found }

If found then { y ditemukan }

    idx  $\square$  i

else

        idx  $\square$  0 { y tidak ditemukan }

endif

## Jawaban Studi Kasus 2 :

- Best case jika nilai y ada pada  $x_1$   
Jadi  $T_{\min}(n) = 1$
- Average case jika y ada pada posisi ke i dan operasi perbandingan dilakukan sebanyak i kali  
 $T_{\text{avg}}(n) = (n+1)/2$
- Worst case jika y sulit atau tidak ditemukan  
 $T_{\max}(n) = n$

## Program :

/\*

Nama : Shalvina Zahwa Aulia

NPM : 140810180052

Deskripsi : Program mencari index key nilai y

\*/

```

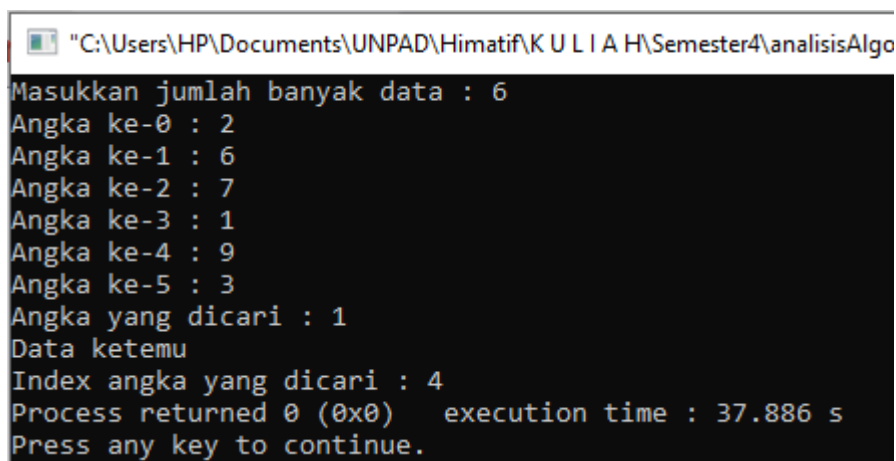
#include <iostream>
using namespace std;

int main(){
    int n, index,y;
    bool found = false;

    cout << "Masukkan jumlah banyak data : "; cin >> n;
    int x[n];

    for(int i=0; i<n; i++){
        cout << "Angka ke-"<<i<< " : "; cin >> x[i];
    }
    cout << "Angka yang dicari : "; cin >> y;
    for(int i=0; i<n; i++){
        if(x[i] == y){
            found = true;
            index = i;
            i = n;
        }
    }
    if (found == true){
        cout << "Data ketemu\n";
        cout << "Index angka yang dicari : " << index + 1;
    }
    else{
        cout << "Data tidak ditemukan";
        cout << "Index = 0";
    }
    return 0;
}

```



```

"C:\Users\HP\Documents\UNPAD\Himatif\K U L I A H\Semester4\analisisAlgo
Masukkan jumlah banyak data : 6
Angka ke-0 : 2
Angka ke-1 : 6
Angka ke-2 : 7
Angka ke-3 : 1
Angka ke-4 : 9
Angka ke-5 : 3
Angka yang dicari : 1
Data ketemu
Index angka yang dicari : 4
Process returned 0 (0x0)   execution time : 37.886 s
Press any key to continue.

```

### Studi Kasus 3: *Binary Search*

Diberikan larik bilangan bulat  $a_1, a_2, \dots, a_n$  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  $a_1, a_2, \dots, a_n$  : integer,  $x$  : integer, output : idx : integer)
{ Mencari  $y$  di dalam elemen  $a_1, a_2, \dots, a_n$ . Lokasi (indeks elemen) tempat  $y$  ditemukan diisi ke dalam idx. Jika  $y$  tidak ditemukan maka idx diisi dengan 0.
  Input:  $a_1, a_2, \dots, a_n$ 
  Output: idx
}
Deklarasi  $i, j$ ,
mid : integer
found :
Boolean
Algoritma
 $i \leftarrow 1$ 
 $j \leftarrow n$ 
found  $\leftarrow$  false
while (not found) and ( $i \leq j$ )
do
  mid  $\leftarrow (i + j) \text{ div } 2$ 
  if  $x_{\text{mid}} = y$  then
    found
  else
    if  $x_{\text{mid}} < y$  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 :

- Best Case :  
 $T_{\min}(n) = 1$
- Worst Case :  
 $T_{\max}(n) = {}^2\log n$

#### Program :

```

/*
Nama      : Shalvina Zahwa Aulia
NPM       : 140810180052
Deskripsi : Program mencari index key nilai y
*/

```

```

#include <iostream>
using namespace std;

int main(){

    int n, y, first, mid, last;

    cout << "Masukkan banyak angka : "; cin >> n;
    int x[n];
    for(int i=0; i<n; i++){
        cout << "Angka ke-"<<i<< " : "; cin >> x[i];
    }
    cout << "Angka yang dicari : "; cin >> y;
    first = 0;
    last = n-1;

```



```

while(first <= last){
    mid = (first+last)/2;
    if(x[mid] < y){
        first = mid +1;
    }
    else if(x[mid] == y){
        cout << "Angka " << y << " ditemukan di array ke-" << mid+1 << endl;
        break;
    }
    else{
        last = mid-1;
    }
    mid = (first+last)/2;
}
if(first>last){
    cout << "angka tidak ditemukan\n";
}
return 0;
}

```

"C:\Users\HP\Documents\UNPAD\Himatif\K U L I A H\Semester4\analisisAlgo

```

Masukkan banyak angka : 6
Angka ke-0 : 1
Angka ke-1 : 2
Angka ke-2 : 3
Angka ke-3 : 4
Angka ke-4 : 5
Angka ke-5 : 6
Angka yang dicari : 4
Angka 4 ditemukan di array ke-4

Process returned 0 (0x0)   execution time : 5.912 s
Press any key to continue.

```

#### 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  $\square$  2 to n do
  insert  $\square$  xi
  j  $\square$  i
  while (j < i) and (x[j-i] > insert) do
    x[j]  $\square$  x[j-1]
    j  $\square$  j-1
  endwhile
  x[j] = insert
endfor

```

### Jawaban Studi Kasus 4 :

Kompleksitas waktu keseluruhan =  $O(n+f(n))$

$f(n)$  : jumlah inversi

Jika jumlah inversi  $O(n)$  maka kompleksitas waktu jenis penyisipan adalah  $O(n)$

Worst case jika array urutannya terbalik, jadi muncul kemungkinan inversi  $n*(n-1)/2$  dan kompleksitas waktu penyisipan untuk worst case adalah  $O(n^2)$ ;

### Program :

```

/*
Nama : Shalvina Zahwa Aulia
NPM : 140810180052
Deskripsi : Program mencari index key nilai y
*/

```

```

#include <iostream>
using namespace std;

```

```

int x[100], x1[100], n;

```

```

void insertionSort(){
  int temp,j;
  for(int i=1; i<=n; i++){

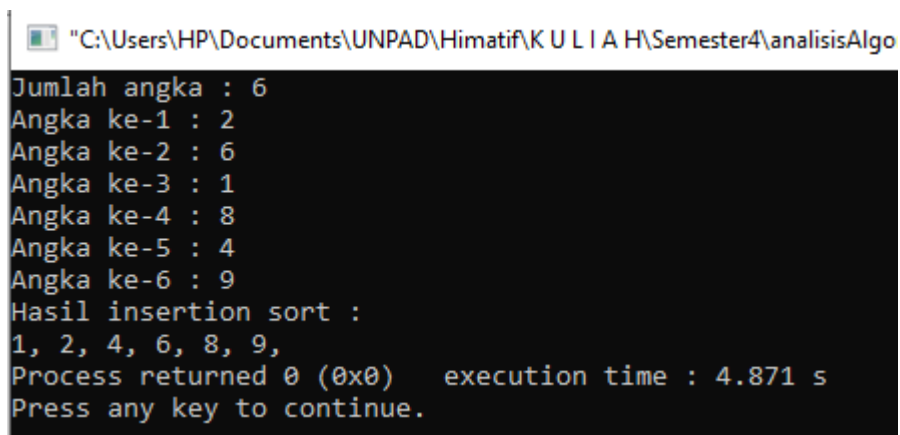
```

```

        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 << "Jumlah angka : "; cin >> n;
    for(int i=1; i<=n; i++){
        cout << "Angka ke-"<<i<<" : "; cin >> x[i];
        x1[i]=x[i];
    }
    insertionSort();
    cout << "Hasil insertion sort : \n";
    for(int i=1; i<=n; i++){
        cout << x[i] <<" , ";
    }
    return 0;
}

```



The screenshot shows a Windows command prompt window with the title bar "C:\Users\HP\Documents\UNPAD\Himatif\K U L I A H\Semester4\analisisAlgo". The program's output is as follows:

```

Jumlah angka : 6
Angka ke-1 : 2
Angka ke-2 : 6
Angka ke-3 : 1
Angka ke-4 : 8
Angka ke-5 : 4
Angka ke-6 : 9
Hasil insertion sort :
1, 2, 4, 6, 8, 9,
Process returned 0 (0x0)   execution time : 4.871 s
Press any key to continue.

```

### 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  $\square$  n downto 2 do {pass sebanyak n-1 kali}
    imaks  $\square$  1
    for j  $\square$  2 to i do
      if  $x_j > x_{imaks}$  then
        imaks  $\square$ 
j      endif
    endfor
    {pertukarkan  $x_{imaks}$  dengan  $x_i$ }
    temp  $\square$   $x_i$ 
     $x_i$   $\square$   $x_{imaks}$ 
     $x_{imaks}$   $\square$  temp
  endfor

```

### Jawaban Studi Kasus 5 :

- Jumlah operasi perbandingan. Untuk setiap pass ke i :  
 $i=1$ , jumlah perbandingan =  $n-1$   
 $i=2$ , jumlah perbandingan =  $n-2$   
 $i=3$ , jumlah perbandingan =  $n-3$   
 $i=k$ , jumlah perbandingan =  $n-k$   
 $i=n-1$ , jumlah perbandingan = 1  
 Jadi  $T(n) = (n-1) + (n-2) + \dots + 1$  (kompleksitas waktu untuk worst case dan best case karena algoritma urut tidak bergantung pada datanya sudah terurut atau belum.
- Jumlah operasi pertukaran  
 Untuk setiap i sampai  $n-1$  pertukaran elemen terjadi 1 kali sehingga  $T(n) = n-1$   
 Anggota pengurutan maksimum memerlukan  $n(n-1)/2$  jumlah operasi perbandingan elemen dan  $n-1$  jumlah operasi pertukaran.

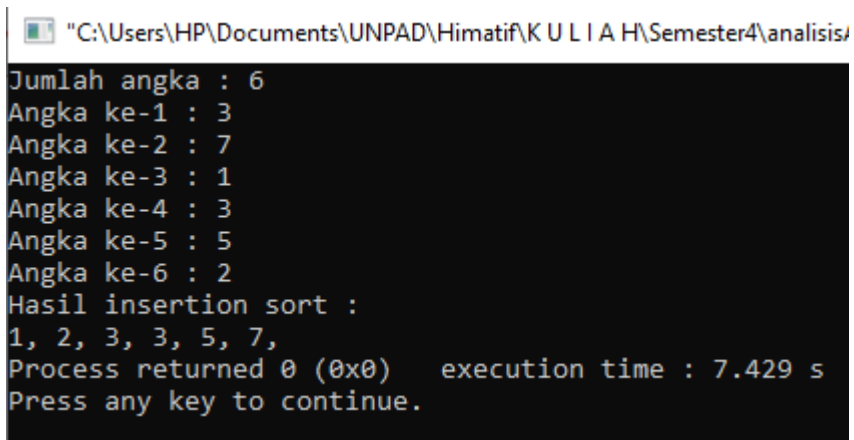
**Program :**

```
/*  
Nama      : Shalvina Zahwa Aulia  
NPM       : 140810180052  
Deskripsi : Program selection sort  
*/  
  
#include <iostream>  
using namespace std;  
  
int x[100], x1[100], n;  
  
void swipe(int a, int b){  
    int t;  
    t = x[b];  
    x[b]=x[a];  
    x[a]=t;  
}  
  
void selectionSort(){  
    int temp,i,j;  
    for(int i=1; i<=n-1; i++){  
        temp = i;  
        for(j=i+1; j<=n; j++){  
            if(x[j]<x[temp])  
                temp =j;  
        }  
        if(temp!=i)  
            swipe(temp,i);  
    }  
}  
  
int main(){  
    cout << "Jumlah angka : "; cin >> n;  
    for(int i=1; i<=n; i++){
```

```

        cout << "Angka ke-"<<i<<" : "; cin >> x[i];
        x1[i]=x[i];
    }
    selectionSort();
    cout << "Hasil insertion sort : \n";
    for(int i=1; i<=n; i++){
        cout << x[i] <<" , ";
    }
    return 0;
}

```



```

"C:\Users\HP\Documents\UNPAD\Himatif\K U L I A H\Semester4\analisis/
Jumlah angka : 6
Angka ke-1 : 3
Angka ke-2 : 7
Angka ke-3 : 1
Angka ke-4 : 3
Angka ke-5 : 5
Angka ke-6 : 2
Hasil insertion sort :
1, 2, 3, 3, 5, 7,
Process returned 0 (0x0)   execution time : 7.429 s
Press any key to continue.

```