

LAPORAN PRAKTIKUM 2

ANALISIS ALGORITMA



Bunga Azizha N
140810180016

Asisten Praktikum:
Faradilla Azranur, Felia Sri Indriyani, Agnes Hata

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  $x_1, x_2, \dots, x_n$ : 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  $\leftarrow x_1$ 
 $i \leftarrow 2$ 
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

Kompleksitas Waktu

- Operator Assignment
Baris 1) 1 kali
Baris 2) 1 kali
Baris 5) n kali
Baris 7) n kali
 $t_1 = 1 + 1 + n + n = 2 + 2n$
- Operator Perbandingan
Baris 3) $n-1$ kali
Baris 4) n kali
 $t_2 = n-1 + n = 2n - 1$
- Operator Penjumlahan
Baris 7) n kali
 $t_3 = n$

Kompleksitas

$$T(n) = t_1 + t_2 + t_3 = 2 + 2n + 2n - 1 + n = 5n + 1$$

Source Code :

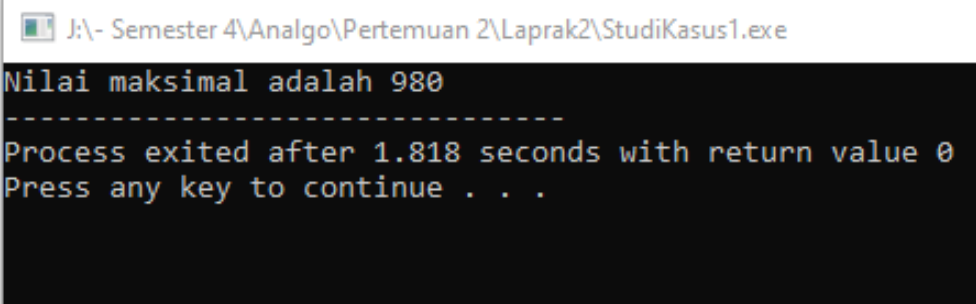
```
/* Bunga Azizha N
140810180016 - Kelas B
Studi Kasus 1 "Algoritma Pencarian Nilai Maksimal" */

#include <iostream>
using namespace std;

#define N 10
int FindMaks(int x[]){
    int maks = x[0];
    for(int i = 1; i < N; i++){
        if(x[i] > maks)
            maks = x[i];
    }
    return maks;
}

int main(){
    int x[N] = {4,8,980,7,56,15,20,1,80,571};
    cout << "Nilai maksimal adalah "<<FindMaks(x);
}
```

Screenshoot Hasil Program



```
J:\- Semester 4\Analgo\Pertemuan 2\Labrak2\StudiKasus1.exe
Nilai maksimal adalah 980
-----
Process exited after 1.818 seconds with return value 0
Press any key to continue . . .
```

Studi Kasus 2: Sequential Search

Diberikan larik bilangan bulat x_1, x_2, \dots, 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  $x_1, x_2, \dots, x_n$  : integer,  $y$  : integer, output  $idx$  : integer)
{ Mencari  $y$  di dalam elemen  $x_1, x_2, \dots, x_n$ . Lokasi (indeks elemen) tempat  $y$  ditemukan diisi ke dalam  $idx$ .
  Jika  $y$  tidak ditemukan, maka  $idx$  diisi dengan 0.
  Input:  $x_1, x_2, \dots, x_n$ 
  Output:  $idx$ 
}
```

Deklarasi

i : integer

found : boolean { bernilai true jika y ditemukan atau false jika y tidak ditemukan }

Algoritma

$i \leftarrow 1$

found \leftarrow false

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

if $x_i = y$ then

 found \leftarrow true

else

$i \leftarrow i + 1$

endif

endwhile

{ $i < n$ or found }

If found then { y ditemukan }

$idx \leftarrow i$

else

$idx \leftarrow 0$ { y tidak ditemukan }

endif

Jawaban Studi Kasus 2

Kompleksitas Waktu

- Operator Assignment
 - Baris 1) 1 kali
 - Baris 2) 1 kali
 - Baris 5) n kali
 - Baris 7) n kali
 - Baris 12) 1 kali
 - Baris 14) n kali
 - $t1 = 1 + 1 + n + n + 1 + n = 3 + 2n$
- Operator Perbandingan
 - Baris 3) $n - 1$ kali
 - Baris 4) $n - 1$ kali
 - Baris 10) $n - 1$ kali
 - $t2 = n - 1 + n - 1 + n - 1 = 3n - 3$
- Operator Penjumlahan
 - Baris 7) n kali
 - $t3 = n$

Kompleksitas

$$T(n) = t_1 + t_2 + t_3 = 3 + 2n + 3n - 3 + n = 6n$$

Source Code

```
/* Bunga Azizha N
140810180016 - Kelas B
Studi Kasus 2 "Sequential Search" */


#include <iostream>
using namespace std;

#define N 4
int SequentialSearch(int *x, int y){
    int idx;
    int i = 0;
    bool found = false;
    while( i < sizeof(x) && !found){
        if(x[i] == y)
            found = true;
        else
            i++;
    }

    if(found)
        idx = i;
    else
        idx = 0;
    return idx;
}

int main(){
    int x[N] = {1,3,99,2};
    cout << "Index key : " << SequentialSearch(x,2);
}
```

Screenshoot Hasil Program

 J:\- Semester 4\Analgo\Pertemuan 2\Labrak2\StudiKasus2.exe

```
Index key : 3
-----
Process exited after 0.09542 seconds with return value 0
Press any key to continue . . .
```

Studi Kasus 3: *Binary Search*

Diberikan larik bilangan x_1, x_2, \dots, 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  $x_1, x_2, \dots, x_n$  : integer,  $x$  : integer, output :  $idx$  : integer)
{ Mencari  $y$  di dalam elemen  $x_1, x_2, \dots, x_n$ . Lokasi (indeks elemen) tempat  $y$  ditemukan diisi ke dalam  $idx$ .
  Jika  $y$  tidak ditemukan maka  $idx$  diisi dengan 0.
  Input:  $x_1, x_2, \dots, x_n$ 
  Output:  $idx$ 
}
```

Deklarasi

i, j, mid : integer
 $found$: Boolean

Algoritma

```
 $i \leftarrow 1$ 
 $j \leftarrow n$ 
 $found \leftarrow \text{false}$ 
while (not  $found$ ) and ( $i \leq j$ ) do
   $mid \leftarrow (i + j) \text{ div } 2$ 
  if  $x_{mid} = y$  then
     $found \leftarrow \text{true}$ 
  else
```

```
    if  $x_{mid} < y$  then {mencari di bagian kanan}
       $i \leftarrow mid + 1$ 
    else {mencari di bagian kiri}
       $j \leftarrow mid - 1$ 
    endif
  endif
```

```
endif
endwhile
{ $found$  or  $i > j$ }
```

```
If  $found$  then
   $idx \leftarrow mid$ 
else
   $idx \leftarrow 0$ 
endif
```

Jawaban Studi Kasus 3

1. Kasus terbaik $T_{min}(n) = 1$
2. Kasus terburuk $T_{max}(n) = 2 \log n$

Source Code

```
/* Bunga Azizha N
140810180016 - Kelas B
Studi Kasus 3 "Binary Search"
*/


#include <iostream>
using namespace std;

#define N 5

int BinarySearch(int *x, int y){
    int i = 0, j = N, mid;
    bool found = false;
    while (!found && i <= j){
        mid = (i+j)/2;
        if( x[mid] == y )
            found = true;
        else if( x[mid] < y)
            i = mid + 1;
        else
            j = mid - 1;
    }
}

int main(){
    int x[N] = {1,3,99,2,4};
    cout << "Index key : " << BinarySearch(x,2);
}
```

Screenshoot Hasil Program

 J:\- Semester 4\Analgo\Pertemuan 2\Labrak2\StudiKasus3.exe

```
Index key : 1
-----
Process exited after 0.07551 seconds with return value 0
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  $x_1, x_2, \dots, x_n$  : integer)
{  Mengurutkan elemen-elemen  $x_1, x_2, \dots, x_n$  dengan metode insertion sort.
  Input:  $x_1, x_2, \dots, x_n$ 
  Output:  $x_1, x_2, \dots, x_n$  (sudah terurut menaik)
}
Deklarasi
  i, j, insert : integer
Algoritma
  for i  $\leftarrow$  2 to n do
    insert  $\leftarrow$   $x_i$ 
    j  $\leftarrow$  i
    while (j < i) and ( $x[j-i]$  > insert) do
       $x[j]$   $\leftarrow$   $x[j-1]$ 
      j  $\leftarrow$  j-1
    endwhile
     $x[j]$  = insert
  endfor
```

Jawaban Studi Kasus 4

Loop sementara dijalankan hanya jika $i > j$ dan $arr[i] < arr[j]$. Jumlah total iterasi loop sementara (Untuk semua nilai i) sama dengan jumlah inversi. Kompleksitas waktu keseluruhan dari jenis penyisipan adalah $O(n + f(n))$ di mana $f(n)$ adalah jumlah inversi. Jika jumlah inversi adalah $O(n)$, maka kompleksitas waktu dari jenis penyisipan adalah $O(n)$.

Dalam kasus terburuk, bisa ada inversi $n * (n-1) / 2$. Kasus terburuk terjadi ketika array diurutkan dalam urutan terbalik. Jadi kompleksitas waktu kasus terburuk dari jenis penyisipan adalah $O(n^2)$.

Source Code

```
/* Bunga Azizha N
140810180016 - Kelas B
Studi Kasus 4 "Insertion Sort" */

#include <iostream>
using namespace std;

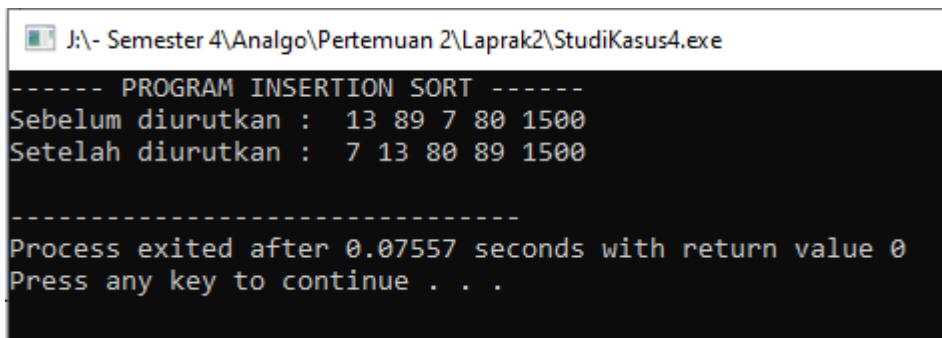
#define N 5

void InsertionSort(int *x){
    int insert,j;
    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;
    }
}

void print(int *x){
    for(int i = 0; i < N; i++)
    {
        cout << " " << x[i];
    }
    cout << endl;
}

int main(){
    int x[N] = {13,89,7,80,1500};
    cout<<"----- PROGRAM INSERTION SORT -----"<<endl;
    cout << "Sebelum diurutkan : "; print(x);
    InsertionSort(x);
    cout << "Setelah diurutkan : "; print(x);
}
```

Screenshoot Hasil Program



```
J:\- Semester 4\Analgo\Pertemuan 2\Labrak2\StudiKasus4.exe
----- PROGRAM INSERTION SORT -----
Sebelum diurutkan : 13 89 7 80 1500
Setelah diurutkan : 7 13 80 89 1500

-----
Process exited after 0.07557 seconds with return value 0
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  $x_1, x_2, \dots, x_n$  : integer)
{ Mengurutkan elemen-elemen  $x_1, x_2, \dots, x_n$  dengan metode selection sort.
  Input:  $x_1, x_2, \dots, x_n$ 
  Output:  $x_1, x_2, \dots, x_n$  (sudah terurut menaik)
}
Deklarasi
  i, j, imaks, temp : integer
Algoritma
  for i  $\leftarrow$  n downto 2 do {pass sebanyak n-1 kali}
    imaks  $\leftarrow$  1
    for j  $\leftarrow$  2 to i do
      if  $x_j > x_{\text{imaks}}$  then
        imaks  $\leftarrow$  j
      endif
    endfor
    {pertukarkan  $x_{\text{imaks}}$  dengan  $x_i$ }
    temp  $\leftarrow$   $x_i$ 
     $x_i \leftarrow x_{\text{imaks}}$ 
     $x_{\text{imaks}} \leftarrow$  temp
  endfor
```

Jawaban Studi Kasus 5

a. Jumlah operasi perbandingan element.

Untuk setiap pass ke-i,

$i = 1 \rightarrow$ jumlah perbandingan = $n - 1$

$i = 2 \rightarrow$ jumlah perbandingan = $n - 2$

$i = 3 \rightarrow$ jumlah perbandingan = $n - 3$:

$i = k \rightarrow$ jumlah perbandingan = $n - k$:

$i = n - 1 \rightarrow$ jumlah perbandingan = 1

Jumlah seluruh operasi perbandingan elemen-elemen larik adalah $T(n) = (n - 1) + (n - 2) + \dots + 1$

Ini adalah kompleksitas waktu untuk kasus terbaik dan terburuk, karena algoritma Urut tidak bergantung pada batasan apakah data masukannya sudah terurut atau acak.

b. Jumlah operasi pertukaran

Untuk setiap i dari 1 sampai $n - 1$,

terjadi satu kali pertukaran elemen, sehingga jumlah operasi pertukaran seluruhnya adalah $T(n) = n - 1$.

Jadi, algoritma pengurutan maksimum membutuhkan $n(n - 1)/2$ buah operasi perbandingan elemen dan $n - 1$ buah operasi pertukaran.

Source Code

```
/* Bunga Azizha N
140810180016 - Kelas B
Studi Kasus 5 "Selection Sort" */

#include <iostream>
using namespace std;

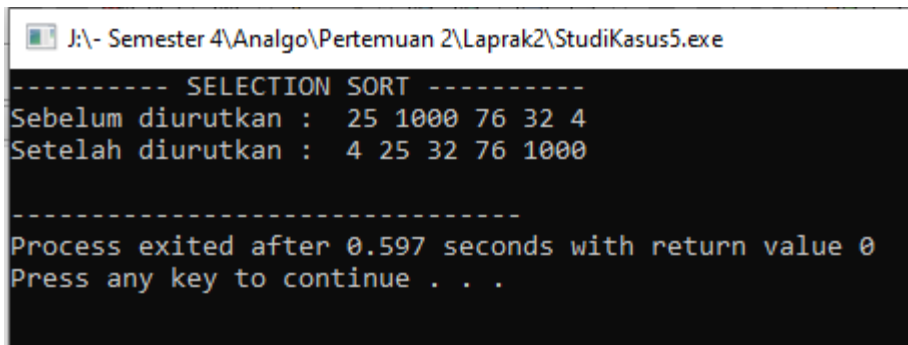
#define N 5

void SelectionSort(int *x){
    int imaks,temp;
    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;
    }
}

void print(int *x){
    for(int i = 0; i < N; i++)
    {
        cout << " " << x[i];
    }
    cout << endl;
}

int main(){
    int x[N] = {25,1000,76,32,4};
    cout<<"----- SELECTION SORT -----"<<endl;
    cout << "Sebelum diurutkan : "; print(x);
    SelectionSort(x);
    cout << "Setelah diurutkan : "; print(x);
}
```

Screenshoot Hasil Program



```
J:\- Semester 4\Analgo\Pertemuan 2\Labrak2\StudiKasus5.exe
----- SELECTION SORT -----
Sebelum diurutkan : 25 1000 76 32 4
Setelah diurutkan : 4 25 32 76 1000

-----
Process exited after 0.597 seconds with return value 0
Press any key to continue . . .
```