Tugas 2 : Kopleksitas Waktu

Praktikum Analisis Algoritma

Disusun oleh :

Syifa Fauziyah N. I. (140810160026)

S-1 Teknik Informatika

Fakultas Matematika & Ilmu Pengetahuan Alam

Universitas Padjadjaran

Jalan Raya Bandung - Sumedang Km. 21 Jatinangor 45363

1. Mencari nilai max

- **Algoritma**

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

Deklarasi
i : integer

Algoritma
maks ← x_1
i ← 2
while i ≤ n do
if x_i > maks then
maks ← x_i
endif
i ← i + 1
endwhile
{i > n}

- **Code**

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

int main()
{
   int n;
   int x[10];
   cout << "Masukkan Jumlah Data : ";
   cin >> n;
   for (int i = 0; i < n; i++)
   {
         cout << "Masukkan Data ke - " << i+1 << " : ";
         cin >> x[i];
   }

   int maks = x[0];
   int i = 1;
   while (i <= n)
   {
         if (x[i] > maks)
               maks = x[i];
```

```
            i++;
        }
        cout << "Maksimum Number : " << maks << endl;

        return 0;
    }
```

- **Kompleksitas waktu**

maks ← x_1	1 kali
i ← 2	1 kali
maks ← x_i	n kali
i ← i + 1	n kali

$$T(n) = 1 + 1 + n + n = 2n + 2$$

2. Sequential Search

- **Algoritma**

procedure SequentialSearch(input $x_1, x_2, \ldots\ x_n$: integer, y : integer, output idx : integer)
{ Mencari y di dalam elemen $x_1, x_2, \ldots\ x_n$. Lokasi (indeks elemen) tempat y ditemukan diisi ke dalam idx. Jika y tidak ditemukan, makai idx diisi dengan 0.
Input: $x_1, x_2, \ldots\ x_n$
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 x_i = 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

- **Code**

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

int main()
{
   int n;
   int x[10];
   cout << "Masukkan Jumlah Data : ";
   cin >> n;
   for (int i = 0; i < n; i++)
   {
         cout << "Masukkan Data ke - " << i+1 << " : ";
```

```
        cin >> x[i];
    }

    int y;
    cout << "Masukkan yang dicari : ";
    cin >> y;

    int i = 0;
    bool found = false;
    int idx;
    while ((i < n) && (!found))
    {
        if (x[i] == y)
            found = true;
        else
            i++;
    }
    if (found)
        idx = i+1;
    else
        idx = 0;

    cout << "Yang dicari berada di urutan : " << idx << endl;

    return 0;
}
```

- **Kompleksitas waktu**

- Best Case :

i ←1	1 kali
found ←false	1 kali
found ←true	1 kali
idx ←I	1 kali

$T_{min}(n) = 1 + 1 + 1 + 1 = 4$

- Average Case :

i ←1	1 kali
found ←false	1 kali
i ←i + 1	½ n kali
found ←true	1 kali
idx ←I	1 kali

$T_{avg}(n) = 1 + 1 + \frac{1}{2}\,n + 1 + 1 = \frac{1}{2}\,n + 4$

- Worst Case :

i ←1	1 kali
found ←false	1 kali
i ←i + 1	n kali
found ←true	1 kali
idx ←I	1 kali

$T_{max}(n) = 1 + 1 + n + 1 + 1 = n + 4$

3. Binary Search

- **Algoritma**

procedure BinarySearch(input $x_1, x_2, \ldots\ x_n$: integer, x : integer, output : idx : integer)
{ Mencari y di dalam elemen $x_1, x_2, \ldots\ x_n$. Lokasi (indeks elemen) tempat y ditemukan diisi ke dalam idx. Jika y tidak ditemukan makai dx diisi dengan 0.
Input: $x_1, x_2, \ldots\ x_n$
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
else {*mencari di bagian kiri*}
j ← mid – 1
endif
endif
endwhile
{*found or i > j* }

If found then
Idx ← mid
else
Idx ← 0
endif

- **Code**

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

int main()
{
   int n;
   int x[10];
```

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

    int y;
    cout << "Masukkan yang dicari : ";
    cin >> y;

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

    }

    if (found)
          idx = mid+1;
    else
          idx = 0;

    cout << "Yang dicari berada di urutan : " << idx << endl;

    return 0;
}
```

- **Kompleksitas waktu**

- 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

$T_{min}(n) = 1 + 1 + 1 + 1 + 1 + 1 = 6$

- Average Case :

i ←1	1 kali
j ←n	1 kali
found ←false	1 kali
mid ←(i + j) div2	½ n + 1 kali
i ←mid + 1or j ←mid –1	½ n kali
found ←true	1 kali
Idx ←mid	1 kali

$T_{avg}(n) = 1 + 1 + 1 + \frac{1}{2}n + 1 + \frac{1}{2}n + 1 + 1 = n + 6$ `

- Worst Case :

i ←1	1 kali
j ←n	1 kali
found ←false	1 kali
mid ←(i + j) div2	n + 1 kali
i ←mid + 1or j ←mid –1	n kali
found ←true	1 kali
Idx ←mid	1 kali

$T_{max}(n) = 1 + 1 + 1 + n + 1 + n + 1 + 1 = 2n + 6$

4. Insertion Sort

- **Algoritma**

procedure InsertionSort(input/output $x_1, x_2, \ldots\ x_n$: integer)
{ Mengurutkan elemen-elemen $x_1, x_2, \ldots\ x_n$ dengan metode insertion sort.
Input: $x_1, x_2, \ldots\ x_n$
OutputL $x_1, x_2, \ldots\ x_n$ (sudah terurut menaik)
}
Deklarasi
i, j, insert : integer
Algoritma
for i ← 2 to n do
insert ← x_i
j ← i
while (j < i) and (x[j-i] > insert) do
x[j]← x[j-1]
j←j-1
endwhile
x[j] = insert
endfor

- **Code**

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

int main()
{
	int n;
	int x[10];
	cout << "Masukkan Jumlah Data : ";
	cin >> n;
	for (int i = 0; i < n; i++)
	{
		cout << "Masukkan Data ke - " << i+1 << " : ";
		cin >> x[i];
	}
	cout << "Data Sebelum di Sorting : ";
	for (int i = 0; i < n; i++)
		cout << x[i] << " ";
	cout << endl;

	int insert;
```

```
        int 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;
        }

        cout << "Data setelah di Sorting : ";
        for (int i = 0; i < n; i++)
              cout << x[i] << " ";

        return 0;
  }
```

- **Kompleksitas waktu**

  - Best Case :

fori ←2 to n do	1 kali
insert ←xi	n kali
j ←i	n kali
x[j] = insert	n kali

    $T_{min}(n) = 1 + n + n + n = 3n + 1$

  - Average Case :

fori ←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

    $T_{avg}(n) = 1 + n + n + \frac{1}{2}n^2 + \frac{1}{2}n^2 + n = n^2 + 3n + 1$

  - Worst Case :

fori ←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

$T_{max}(n) = 1 + n + n + \ n^2 + \ n^2 + n = 2n^2 + 3n + 1$

5. Selection Sort

- **Algoritma**

procedure SelectionSort(input/output $x_1, x_2, \ldots\ x_n$: integer)
{ Mengurutkan elemen-elemen $x_1, x_2, \ldots\ x_n$ dengan metode selection sort.
Input: $x_1, x_2, \ldots\ x_n$
OutputL $x_1, x_2, \ldots\ x_n$ (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 $x_j > x_{imaks}$ then
imaks ← j
endif
endfor
{pertukarkan x_{imaks} dengan x_i}
temp ← x_i
x_i ← x_{imaks}
x_{imaks} ← temp
endfor

- **Code**

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

int main()
{
   int n;
   int x[10];
   cout << "Masukkan Jumlah Data : ";
   cin >> n;
   for (int i = 0; i < n; i++)
   {
        cout << "Masukkan Data ke - " << i+1 << " : ";
        cin >> x[i];
   }
   cout << "Data Sebelum di Sorting : ";
   for (int i = 0; i < n; i++)
        cout << x[i] << " ";
   cout << endl;

   int imaks;
```

```
    int 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;
    }

    cout << "Data setelah di Sorting : ";
    for (int i = 0; i < n; i++)
          cout << x[i] << " ";

    return 0;
}
```

- **Kompleksitas waktu**

- Best Case :

fori ←n downto2 do	1 kali
imaks ←1	n kali
forj ←2 toi do	n kali
imaks ←j	n*1 kali
temp ←xi	n kali
xi←ximaks	n kali
ximaks←temp	n kali

$T_{min}(n) = 1 + n + n + n * 1 + n + n + n = 6n + 1$

- Average Case :

fori ←n downto2 do	1 kali
imaks ←1	nkali
forj ←2 toi do	n kali
imaks ←j	n * ½ n kali
temp ←xi	n kali
xi←ximaks	n kali
ximaks←temp	n kali

$T_{avg}(n) = 1 + n + n + \frac{1}{2}n^2 + n + n + n = \frac{1}{2}n^2 + 5n + 1$

- Worst Case :

fori ←n downto2 do	1 kali
imaks ←1	n kali
forj ←2 toi do	n kali
imaks ←j	n * n kali
temp ←xi	n kali
xi←ximaks	n kali
ximaks←temp	n kali

$T_{max}(n) = 1 + n + n + n^2 + n + n + n = n^2 + 5n + 1$