Tugas 2

Analisis Algoritma

Disusun oleh :

Afifah Kho'eriah (140810160008)

S-1 Teknik Informatika

Fakultas Matematika & Ilmu Pengetahuan Alam

Universitas Padjadjaran

Jalan Raya Bandung - Sumedang Km. 21 Jatinangor 45363

1. Pencarian nilai maksimal

```
#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

```
#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

```
#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

```
#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 :

for i ← 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 :

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

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

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

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

5. **Selection Sort**

```
#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 :

for i ←n down to2 do	1 kali
i maks ←1	n kali
for j ←2 to i do	n kali
i maks ←j	n*1 kali
temp ←xi	n kali
xi←xi maks	n kali
xi maks←temp	n kali

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

Average Case :

for i ←n down to 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←xi maks	n kali
xi maks←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 :

for i ←n down to 2 do	1 kali
maks ←1	n kali
for j ←2 to i do	n kali
imaks ← j	n * n kali
temp ←xi	n kali
xi←xi maks	n kali
ximaks←temp	n kali

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