Analisis Algoritma

Laporan Praktikum 1

Dibuat oleh:
Muhammad Iqbal Alif Fadilla
140810180020

Universitas Padjadjaran
Fakultas Matematika dan Ilmu Pengetahuan
2019

Worksheet 2

1. 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
```

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.      int arr[100];
6.      int n, maks, i;
7.
8.      cout << "Masukkan Banyak Angka : ";
9.      cin >> n;
10.     for (int i = 0; i < n; i++)
11.     {
12.         cout << "Bilangan ke - " << i + 1 << " : ";
13.         cin >> arr[i];
14.     }
15.     maks = arr[0];
16.     for (int i = 0; i < n; i++)
17.     {
18.         if (arr[i] > maks)
19.             maks = arr[i];
20.     }
21.     cout << "Output = " << maks << endl;
22. }
```

```
d:\Kuliah\Semester 4\Analisis Algoritma\AnalgoKu2>01
Masukkan Banyak Angka : 5
Bilangan ke - 1 : 1
Bilangan ke - 2 : 2
Bilangan ke - 3 : 3
Bilangan ke - 4 : 4
Bilangan ke - 5 : 5
Nilai Maksimal = 5
```

- Operasi Assignment = 1 + 1 + (n-1) + (n-1) = 2n
- Operasi Perbandingan = n-1
- Operasi Penjumlahan = n-1

Maka Tmax(n) = 4n-2

2. Studi Kasus 2 : Sequential Search

   Diberikan larik bilangan bulan x_1, x_2, …, 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 x1, x2, ... xn : integer, y : integer, output idx : integer)
{   Mencari y di dalam elemen x1, x2, ... xn. Lokasi (indeks elemen) tempat y ditemukan diisi ke dalam idx.
    Jika y tidak ditemukan, makai idx diisi dengan 0.
    Input: x1, x2, ... xn
    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
```

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int main()
5.  {
6.      int n, target, arr[100], index;
7.      bool found = false;
8.      cout << "\nInput the amount of data =  ";
9.      cin >> n;
10.
11.     for (int i = 0; i < n; i++)
12.     {
```

```
13.         cout << "Data ke-" << i + 1 << " : ";
14.         cin >> arr[i];
15.     }
16.
17.     cout << "\nInput target : ";
18.     cin >> target;
19.
20.     for (int i = 0; i < n; i++)
21.     {
22.         if (arr[i] == target)
23.         {
24.             found = true;
25.             index = i;
26.             i = n;
27.         }
28.     }
29.     if (found == true)
30.     {
31.         cout << "Data found at index " << index + 1 << endl;
32.     }
33.     else
34.     {
35.         cout << "Data not found" << endl;
36.     }
37.     return 0;
38. }
```

```
Input the amount of data =  5
Data ke-1 : 1
Data ke-2 : 2
Data ke-3 : 3
Data ke-4 : 4
Data ke-5 : 5

Input target : 4
Data found at index 4
```

```
Input the amount of data =  5
Data ke-1 : 1
Data ke-2 : 2
Data ke-3 : 3
Data ke-4 : 4
Data ke-5 : 5

Input target : 7
Data not found
```

$T_{min}(n) = 4 + 2 = 6$

- Operasi Assignment = 4
- Operasi Perbandingan = 2

$T_{max}(n) = 3+n+n+1+n = 3n+4$

- Operasi Assignment = 1 + 1 + n + 1 = 3 + n
- Operasi Perbandingan = n + 1
- Operasi Penjumlahan = n

$T_{avg}(n) = (10+3n) / 2$

- $(T_{min}(n) + T_{max}(n)) / 2 = (6+4+3n) / 2 = (10+3n) / 2$

3. Studi Kasus 3 : Binary Search

Diberikan larik bilangan bulan x_1, x_2, ... 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 x1, x2, ... xn : integer, x : integer, output : idx : integer)
{   Mencari y di dalam elemen x1, x2, ... xn. Lokasi (indeks elemen) tempat y ditemukan diisi ke dalam idx.
    Jika y tidak ditemukan makai dx diisi dengan 0.
    Input: x1, x2, ... xn
    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
```

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int main()
5.  {
6.      int n, i, arr[100], target, low, high, mid;
7.      cout << "\nInput the amount of data : ";
8.      cin >> n;
9.      for (i = 0; i < n; i++)
10.     {
11.         cout << "Data ke-" << i + 1 << " : ";
12.         cin >> arr[i];
13.     }
14.     cout << "\nInput target : ";
15.     cin >> target;
16.     low = 0;
17.     high = n - 1;
```

```
18.     while (low <= high)
19.     {
20.         mid = (low + high) / 2;
21.         if (arr[mid] < target)
22.         {
23.             low = mid + 1;
24.         }
25.         else if (arr[mid] == target)
26.         {
27.             cout << target << " found at index-" << mid + 1 << endl;
28.             break;
29.         }
30.         else
31.         {
32.             high = mid - 1;
33.         }
34.         mid = (low + high) / 2;
35.     }
36.     if (low > high)
37.     {
38.         cout << target << " not found" << endl;
39.     }
40.     return 0;
41. }
```

```
Input the amount of data : 5
Data ke-1 : 1
Data ke-2 : 2
Data ke-3 : 3
Data ke-4 : 4
Data ke-5 : 5

Input target : 5
5 found at index-5
```

```
Input the amount of data : 5
Data ke-1 : 1
Data ke-2 : 2
Data ke-3 : 3
Data ke-4 : 4
Data ke-5 : 5

Input target : 7
7 not found
```

$T_{min}(n) = 6 + 2 = 8$

- Operasi Assignment = 6
- Operasi Perbandingan = 2

$T_{max}(n) = (\log 2(n))$

Panjang array akan berubah pada setiap iterasi:

- Iterasi 1 = n
- Iterasi 2 = n/2
- Iterasi 3 = $n/2^2$
- Iterasi x = $n/2^k$-1 ~ $n/2^k$ *(-1 diabaikan karena kecil dibanding $n/2^k$)*

Panjang array menjadi 1.

Maka,

$n/2^k = 1$

$n = 2^k$

$\log 2(n) = \log 2(2^k) = k \log 2(2)$

$k = \log 2(n)$

Sehingga

$T_{avg}(n) = (1 + \log 2(n)) / 2$

$(T_{min}(n) + T_{max}(n)) / 2 = (1 + \log 2(n)) / 2$

4. Studi Kasus 4 : Insertion Sort

- Buatlah program insertion sort dengan menggunakan bahasa C++
- Hitunglah operasi perbandingan elemen larik dan operasi pertukaran pada algoritma insertion sort.
- Tentukan kompleksitas waktu terbaik, terburuk, dan rata-rata untuk algoritma insertion sort.

```
procedure InsertionSort(input/output x1, x2, ... xn : integer)
{   Mengurutkan elemen-elemen x1, x2, ... xn dengan metode insertion sort.
    Input: x1, x2, ... xn
    OutputL x1, x2, ... xn (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
```

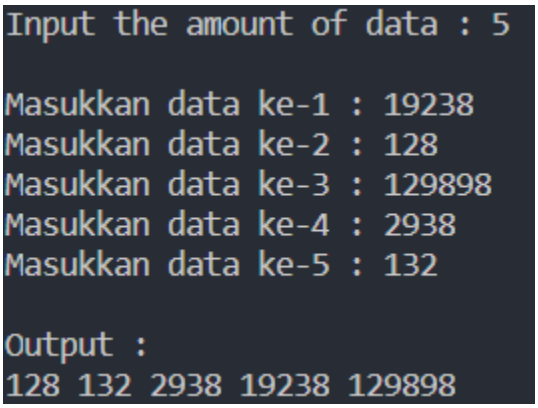
```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int arr[100], arr2[100], n;
5.
6.  void InsertionSort()
7.  {
8.      int temp, i, j;
9.      for (i = 1; i <= n; i++)
10.     {
11.         temp = arr[i];
12.         j = i - 1;
13.         while (arr[j] > temp && j >= 0)
14.         {
15.             arr[j + 1] = arr[j];
16.             j--;
17.         }
18.         arr[j + 1] = temp;
19.     }
20. }
21.
```

```cpp
int main()
{
    cout << "Input the amount of data : ";
    cin >> n;
    cout << endl;

    for (int i = 1; i <= n; i++)
    {
        cout << "Masukkan data ke-" << i << " : ";
        cin >> arr[i];
        arr2[i] = arr[i];
    }

    InsertionSort();
    cout << "\nOutput : " << endl;

    for (int i = 1; i <= n; i++)
    {
        cout << arr[i] << " ";
    }
    return 0;
}
```



$T_{min}(n) = 3n-3 + 4n-4 + 1 = 7n - 6$

1. Operasi Assignment: 2(n-1) + (n-1) = 3n-3
2. Operasi Perbandingan: 2*((n-1) + (n-1)) = 2*(2n-2) = 4n-4
3. Operasi Pertukaran: (n-1) * n = n^2-n

$T_{max}(n) = 3n-3 + 4n-4 + n^2-n = n^2+6n-6$

$T_{avg}(n) = (n^2 + 13n - 12) / 2$

($T_{min}(n) + T_{max}(n)) / 2 = (7n–6 + n^2+6n-6) / 2$

5. Studi Kasus 5 : Selection Sort

- Buatlah program selection sort dengan menggunakan bahasa C++
- Hitunglah operasi perbandingan elemen larik dan operasi pertukaran pada algoritma selection sort.
- Tentukan kompleksitas waktu terbaik, terburuk, dan rata-rata untuk algoritma insertion sort.

```
procedure SelectionSort(input/output x1, x2, ... xn : integer)
{   Mengurutkan elemen-elemen x1, x2, ... xn dengan metode selection sort.
    Input: x1, x2, ... xn
    OutputL x1, x2, ... xn (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
```

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

int arr[100], arr2[100];
int n;

void swap(int a, int b)
{
    int t;
    t = arr[b];
    arr[b] = arr[a];
    arr[a] = t;
}

void SelectionSort()
{
    int pos, i, j;
    for (i = 1; i <= n - 1; i++)
    {
        pos = i;
        for (j = i + 1; j <= n; j++)
        {
            if (arr[j] < arr[pos])
                pos = j;
        }
        if (pos != i)
            swap(pos, i);
    }
}

int main()
{
    cout << "\nInput the amount of data : ";
    cin >> n;

    for (int i = 1; i <= n; i++)
    {
        cout << "Masukkan data ke-" << i << " : ";
        cin >> arr[i];
```

```
        arr2[i] = arr[i];
    }

    SelectionSort();
    cout << "Output : " << endl;
    for (int i = 1; i <= n; i++)
    {
        cout << " " << arr[i];
    }
    return 0;
}
```

```
Input the amount of data : 5
Masukkan data ke-1 : 1238
Masukkan data ke-2 : 1923819
Masukkan data ke-3 : 92389
Masukkan data ke-4 : 12322
Masukkan data ke-5 : 9238
Output :
 1238 9238 12322 92389 1923819
```

1. Operasi Perbandingan =

$$\sum_{i=1}^{n-1} i = \frac{(n-1)+1}{2}(n-1) = \frac{1}{2}n(n-1) = \frac{1}{2}(n^2-n)$$

2. Operasi Pertukaran = n-1

$T_{min}(n) = (4n-4) + ½(n^2-n) +1 \sim n^2$
$T_{max}(n) = ½(n^2-n) + (n-1) \sim n^2$
$T_{avg}(n) = n^2$
$(T_{min}(n) + T_{max}(n)) / 2 = (n^2 + n^2) / 2$