**LAPORAN PRAKTIKUM 2**

**DESAIN DAN ANALISIS ALGORITMA**



**DISUSUN OLEH:**

**SURIADI VAJRAKARUNA 140810180038**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

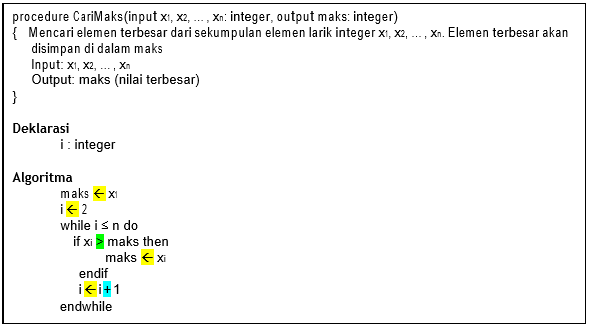
**UNIVERSITAS PADJADJARAN**

**SUMEDANG**

**2020**

1. **Tujuan**
2. Mahasiswa mengerti kompleksitas algoritma secara umum.
3. Mahasiswa mengerti cara menghitung kompleksitas waktu dari suatu algoritma.
4. Mahasiswa mengetahui faktor-faktor yang mempengaruhi kompleksitas waktu dari suatu algoritma.
5. Mahasiswa mengetahui operasi-operasi yang mempengeruhi atau ikut disertakan dalam menghitung kompleksitas waktu dari suatu algoritma.
6. Mahasiswa dapat mengimplementasikan perhitungan algoritma saat membuat sebuah program.
7. **Landasan Teori**
8. Sebuah algoritma tidak hanya harus benar tetapi harus efisien dan optimal.
9. Kompleksitas algoritma bergantung pada jumlah waktu/kompleksitas waktu yang dinotasikan dengan **T(n)** dan ruang memori yang dinotasikan dengan **S(n)** yang dibutuhkan untuk mengeksekusi algoritma tersebut.
10. Sebuah algoritma dikatakan efisien apabila waktu dan ruang memori yang dibutuhkan untuk mengeksekusinya kecil.
11. Kompleksitas waktu dapat dihitungan dengan
    1. Menetapkan ukuran input (n).
    2. Menghitung banyak operasi seperti penjumlahan (+), pengurangan (-), perbandingan (<, >, =) kecuali pada *loop*, Pembagian (/), pembacaan atau *assignment* (<-), pemanggilan prosedur atau *function*, dan lain-lain.
    3. Jika operasi berada didalam *loop*, maka operasi itu dihitung sesuai berapa kali *looping* itu terjadi.
12. Jika kita mengetahui besaran waktu (detik) untuk melaksanakan operasi-operasi yang terdapat di sebuah algoritma, maka kita dapat menghitung waktu sesungguhnya yang diperlukan sebuah mesin untuk mengeksekusi algoritma tersebut.
13. Ada tiga macam kompleksitas waktu yang mungkin terjadi pada sebuah algoritma yaitu
14. *Best Case* = T­min(n) adalah kompleksitas waktu dengan jumlah paling kecil.
15. *Average Case* = T­avg(n) adalah kompleksitas dengan jumlah rata-rata keseluruhan kemungkinan. Jadi , hasil perhitungan dari *worst case* dan *best case* dijumlah lalu dibagi dengan n (biasanya n=2).
16. *Worst Case* = T­max(n) adalah kompleksitas waktu dengan jumlah paling besar.
17. **Worksheet 2**
18. **Studi Kasus 1: Pencarian Nilai Maksimal**

Buatlah programnya dan hitunglah kompleksitas waktu dari algoritma berikut:



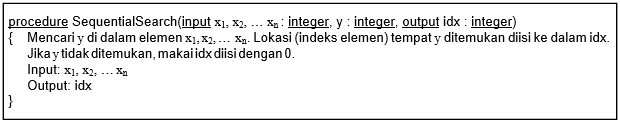
Jawab:

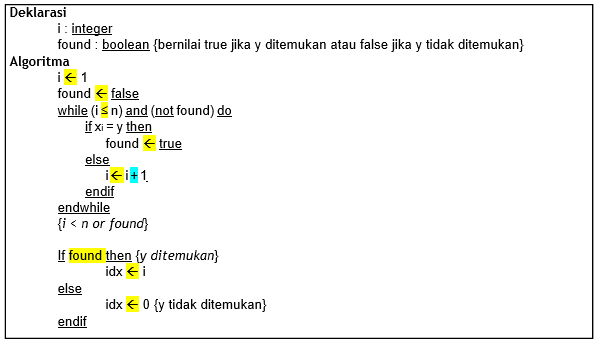
1. Operasi Assignment = 1 + 1 + (n-1) + (n-1) = 2n
2. Operasi Perbandingan = n-1
3. Operasi Penjumlahan = n-1

**Maka Tmax(n) = 4n-2**

1. **Studi Kasus 2: Sequential Search**

Diberikan larik bilangan bulan x1, x2, … xn 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.





Jawab:

**Tmin(n):**

1. Operasi Assignment = 4
2. Operasi Perbandingan = 2

**Tmin(n) = 4 + 2 = 6**

**Tmax(n):**

1. Operasi Assignment = 1 + 1 + n + 1 = 3 + n
2. Operasi Perbandingan = n + 1
3. Operasi Penjumlahan = n

**Tmax(n) = 3+n+n+1+n=3n+4**

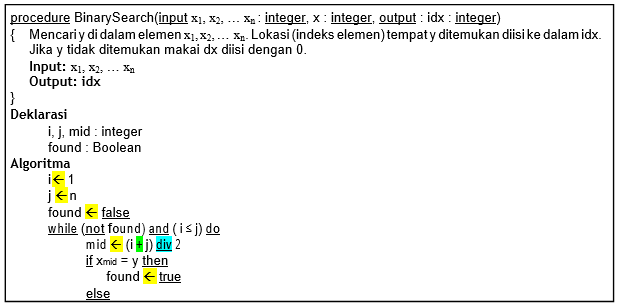
**Tavg(n):**

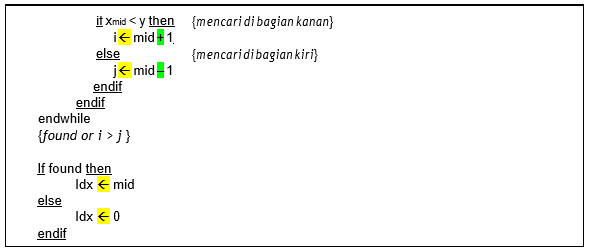
(Tmin(n) + Tmax(n)) / 2 = (6+4+3n) / 2 = (10+3n) / 2

**Tavg(n) = (10+3n) / 2**

1. **Studi Kasus 3: Binary Search**

Diberikan larik bilangan bulan x1, x2, … xn 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.





Jawab:

**Tmin(n):**

1. Operasi Assignment = 6
2. Operasi Perbandingan = 2

**Tmin(n) = 6 + 2 = 8**

**Tmax(n):**

Panjang array akan berubah pada setiap iterasi:

* Iterasi 1 = n
* Iterasi 2 = n/2
* Iterasi 3 = n/22
* Iterasi x = n/2k-1 ~ n/2k *(-1 diabaikan karena kecil dibanding n/2k)*

Panjang array menjadi 1.

Maka,

n/2k = 1

n = 2k

log 2(n) = log 2(2k) = k log 2(2)

k = log 2(n)

Sehingga

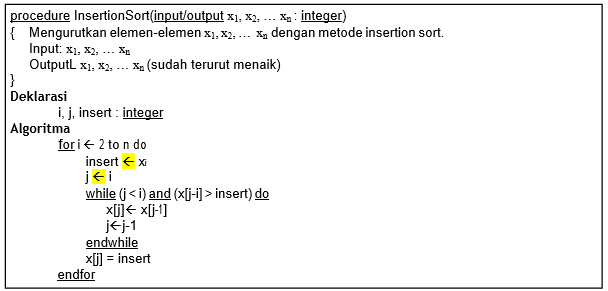
**Tmax(n) = )(log 2(n))**

**Tavg(n):**

(Tmin(n) + Tmax(n)) / 2 = (1 + log 2(n)) / 2

**Tavg(n) = (1 + log 2(n)) / 2**

1. **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.



Jawab:

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 = n2-n

**Tmin(n):**

**Tmin(n) = 3n-3 + 4n-4 + 1 = 7n - 6**

**Tmax(n):**

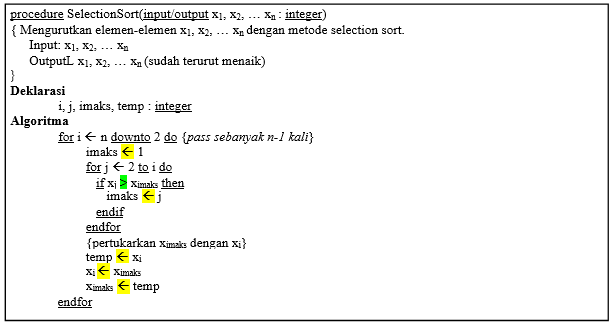
**Tmax(n) = 3n-3 + 4n-4 + n2-n = n2+6n-6**

**Tavg(n):**

(Tmin(n) + Tmax(n)) / 2 = (7n–6 + n2+6n-6) / 2

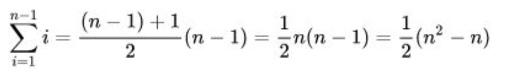
**Tavg(n) = (n2 + 13n - 12) / 2**

1. **Studi Kasus 5: Selection Sort**
   * + 1. Buatlah program insertion 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 selection sort.



Jawab:

1. Operasi Perbandingan =



1. Operasi Pertukaran = n-1

**Tmin(n):**

**Tmin(n) = (4n-4) + ½(n2-n) +1 ~ n2**

**Tmax(n):**

**Tmax(n) = ½(n2-n) + (n-1) ~ n2**

**Tavg(n):**

(Tmin(n) + Tmax(n)) / 2 = (n2+ n2) / 2

**Tavg(n) = n2**

1. **Program Worksheet 2**
2. **Studi Kasus 1**

**Sourcecode (.cpp)**

/\*

NAMA    : SURIADI VAJRAKARNA

NPM     : 140810180038

KELAS   : B

TANGGAL : 8 MARET 2020

TUGAS 2 - STUDI KASUS 1 - PRAKTIKUM ANALISIS ALGORITMA

\*/

#include <iostream>

using namespace std;

int main(int argc, char const \*argv[])

{

    int bil[5] = {1632, 3, 230, 23, 28};

    int n = sizeof(bil)/sizeof(bil[0]);

    int max = bil[0];

    int i = 2;

    while (i <= n)

    {

        if (bil[i] > max)

            max = bil[i];

        i = i + 1;

    }

    cout << "Max = " << max;

    return 0;

}

**Screenshot**



1. **Studi Kasus 2**

**Sourcecode (.cpp)**

/\*

NAMA    : SURIADI VAJRAKARNA

NPM     : 140810180038

KELAS   : B

TANGGAL : 8 MARET 2020

TUGAS 2 - STUDI KASUS 2 - PRAKTIKUM ANALISIS ALGORITMA

\*/

#include <iostream>

using namespace std;

int main(int argc, char const \*argv[])

{

    int bil[5] = {15, 387, 22, 63, 74};

    int cari = 63;

    int n = sizeof(bil) / sizeof(bil[0]);

    int idx;

    int i = 1;

    bool found = false;

    while (i <= n && !found)

    {

        if (bil[i] == cari)

            found = true;

        else

            i = i + 1;

    }

    if (found == true)

    {

        idx = i;

        cout << "Found at index " << idx;

    }

    else

    {

        idx = 0;

        cout << "Not Found";

    }

    return 0;

}

**Screenshot**



1. **Studi Kasus 3**

**Sourcecode (.cpp)**

/\*

NAMA    : SURIADI VAJRAKARNA

NPM     : 140810180038

KELAS   : B

TANGGAL : 8 MARET 2020

TUGAS 2 - STUDI KASUS 3 - PRAKTIKUM ANALISIS ALGORITMA

\*/

#include <iostream>

using namespace std;

main()

{

    int bil[5] = {14, 31, 59, 72, 98};

    int cari = 72;

    int n = sizeof(bil) / sizeof(bil[0]);

    int idx;

    int i = 1;

    int j = n;

    int mid;

    bool found = false;

    while (!found && i <= j)

    {

        mid = (i + j) / 2;

        if (bil[mid] == cari)

            found = true;

        else if (bil[mid] < cari)

            i = mid + 1;

        else

            j = mid - 1;

    }

    if (found == true)

    {

        idx = mid;

        cout << "Found at index " << idx;

    }

    else

    {

        idx = 0;

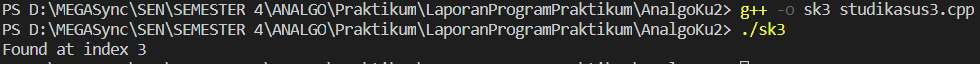
        cout << "Not Found";

    }

    return 0;

}

**Screenshot**



1. **Studi Kasus 4**

**Sourcecode (.cpp)**

/\*

NAMA    : SURIADI VAJRAKARNA

NPM     : 140810180038

KELAS   : B

TANGGAL : 8 MARET 2020

TUGAS 2 - STUDI KASUS 4 - PRAKTIKUM ANALISIS ALGORITMA

\*/

#include <iostream>

using namespace std;

int main(int argc, char const \*argv[])

{

    int bil[5] = {3, 45, 72, 31, 96};

    int n = sizeof(bil) / sizeof(bil[0]);

    int i, j, insert;

    for (i = 1; i < n; i++)

    {

        insert = bil[i];

        j = i - 1;

        while (j >= 0 && bil[j] > insert)

        {

            bil[j + 1] = bil[j];

            j = j - 1;

        }

        bil[j + 1] = insert;

    }

    cout << "Insertion Sort: ";

    for (j = 0; j < n; j++)

    {

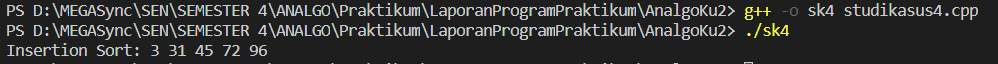
        cout << bil[j] << " ";

    }

    return 0;

}

**Screenshot**



1. **Studi Kasus 5**

**Sourcecode (.cpp)**

/\*

NAMA    : SURIADI VAJRAKARNA

NPM     : 140810180038

KELAS   : B

TANGGAL : 8 MARET 2020

TUGAS 2 - STUDI KASUS 5 - PRAKTIKUM ANALISIS ALGORITMA

\*/

#include <iostream>

using namespace std;

int main(int argc, char const \*argv[])

{

    int bil[5] = {42, 12, 57, 289, 48};

    int n = sizeof(bil) / sizeof(bil[0]);

    int i, j, imaks, temp;

    for (i = 2; i < n; i++)

    {

        imaks = 1;

        for (j = 2; j < i; j++)

        {

            if (bil[j] > bil[imaks])

                imaks = j;

        }

        temp = bil[i];

        bil[i] = bil[imaks];

        bil[imaks] = temp;

    }

    cout << "Selection Sort: ";

    for (int i = 0; i < n; i++)

    {

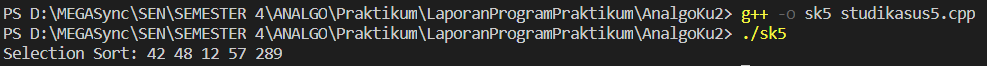
        cout << bil[i] << " ";

    }

    return 0;

}

**Screenshot**



**Daftar Pustaka**

Suryani, Mira, Ino Suryana, R. Sudrajat. 2019. *ANALISIS ALGORITMA: MODUL PRAKTIKUM 2*. Jatinangor: Universitas Padjadjaran.

Hatta, Mouhamad, Rifaldi, Agung, dan Siddiq. *Analisis Algoritma: Best, Worst, dan Average*. Satelit di (diakses 6 Maret).

Gautama, Elliana. *Kompleksitas Algoritma*. Satelit di (diakses 6 Maret).

Mukharl, Adam. *Analisis Algoritma – Pengantar Kompleksitas Algoritma*. Satelit di (diakses 6 Maret).