TUGAS PRAKTIKUM 4
DESAIN DAN ANALISIS ALGORITMA

DISUSUN OLEH:
SURIADI VAJRAKARUNA 140810180038

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PADJADJARAN
SUMEDANG
2020

Studi Kasus 1: MERGE SORT

Setelah Anda mengetahui Algoritma Merge-Sort mengadopsi paradigma divide & conquer, lakukan Hal berikut:

1. Buat program Merge-Sort dengan bahasa C++
2. Kompleksitas waktu algoritma merge sort adalah O(n lg n). Cari tahu kecepatan komputer Anda dalam memproses program. Hitung berapa running time yang dibutuhkan apabila input untuk merge sort-nya adalah 20?

```
/*
NAMA    : SURIADI VAJRAKARNA
NPM     : 140810180038
KELAS   : B
TANGGAL : 8 MARET 2020
TUGAS 4 - STUDI KASUS 1 - PRAKTIKUM DESAIN DAN ANALISIS ALGORITMA
*/

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

void merge(int *elmn, int low, int high, int mid)
{
    int i, j, k, c[100];
    i = low;
    k = low;
    j = mid + 1;
    while (i <= mid && j <= high)
    {
        if (elmn[i] < elmn[j])
        {
            c[k] = elmn[i];
            k++;
            i++;
        }
        else
        {
            c[k] = elmn[j];
            k++;
            j++;
        }
    }
    while (i <= mid)
    {
        c[k] = elmn[i];
        k++;
```

```
        i++;
    }
    while (j <= high)
    {
        c[k] = elmn[j];
        k++;
        j++;
    }
    for (i = low; i < k; i++)
    {
        elmn[i] = c[i];
    }
}

void merge_sort(int *elmn, int low, int high)
{
    int mid;
    if (low < high)
    {
        mid = (low + high) / 2;
        merge_sort(elmn, low, mid);
        merge_sort(elmn, mid + 1, high);
        merge(elmn, low, high, mid);
    }
}

int main()
{
    int jmlhElmn;
    cout << "Banyak elemen yang akan diurutkan: ";
    cin >> jmlhElmn;

    int elmn[jmlhElmn];
    cout << "Masukkan " << jmlhElmn << " elemen yang akan diurutkan:\n";
    for (int i = 0; i < jmlhElmn; i++)
    {
        cout << "Elemen ke-" << i + 1 << ": ";
        cin >> elmn[i];
    }

    auto start = chrono::steady_clock::now();
    merge_sort(elmn, 0, jmlhElmn - 1);
    auto end = chrono::steady_clock::now();

    cout << "Hasil Pengurutan\n";
```

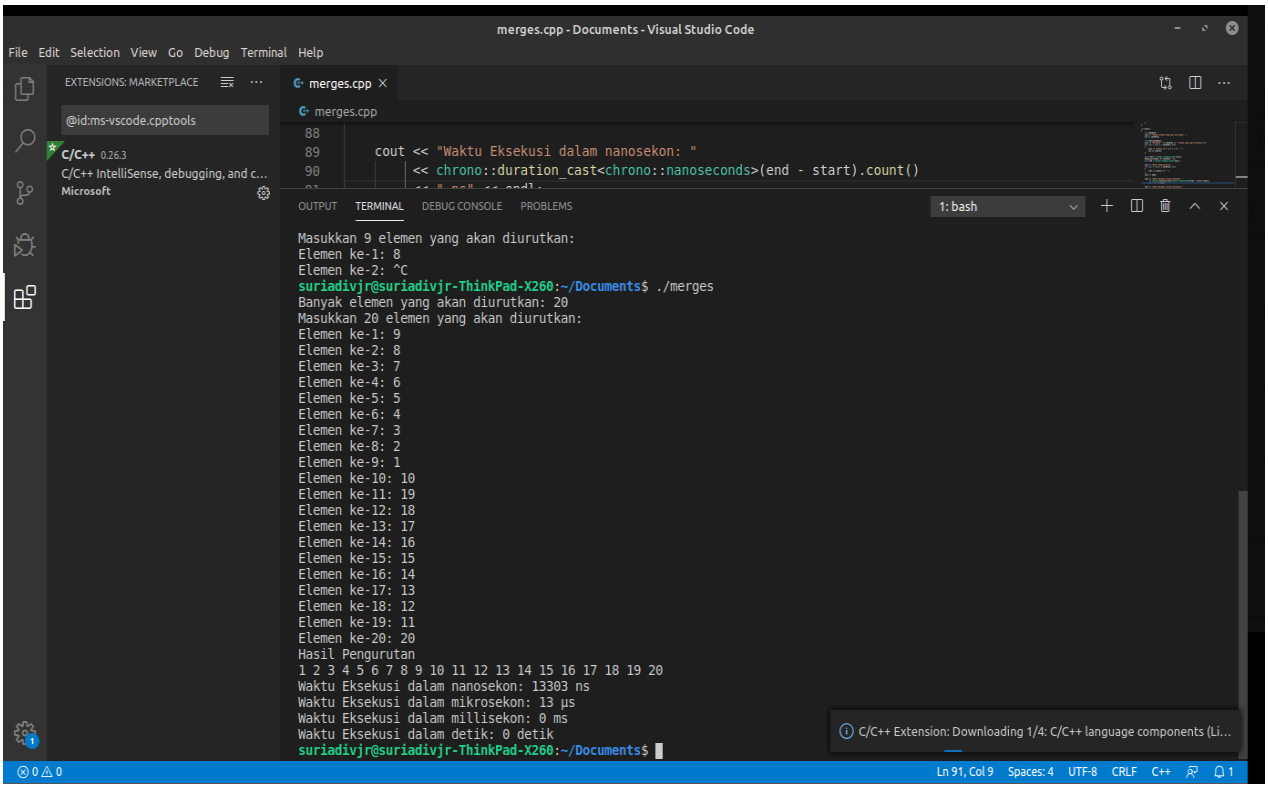
```cpp
    for (int i = 0; i < jmlhElmn; i++)
    {
        cout << elmn[i] << " ";
    }
    cout << endl;

    cout << "Waktu Eksekusi dalam nanosekon: "
         << chrono::duration_cast<chrono::nanoseconds>(end - start).count()
         << " ns" << endl;

    cout << "Waktu Eksekusi dalam mikrosekon: "
         << chrono::duration_cast<chrono::microseconds>(end - start).count()
         << " µs" << endl;

    cout << "Waktu Eksekusi dalam millisekon: "
         << chrono::duration_cast<chrono::milliseconds>(end - start).count()
         << " ms" << endl;

    cout << "Waktu Eksekusi dalam detik: "
         << chrono::duration_cast<chrono::seconds>(end - start).count()
         << " detik" << endl;
    return 0;
}
```



Studi Kasus 2: SELECTION SORT

Selection sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma selection sort, lakukan langkah-langkah berikut:

- Pelajari cara kerja algoritma selection sort
- Tentukan T(n) dari rekurensi (pengulangan) selection sort berdasarkan penentuan rekurensi divide & conquer:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

- Selesaikan persamaan rekurensi T(n) dengan **metode recursion-tree** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ
- Lakukan implementasi koding program untuk algoritma selection sort dengan menggunakan bahasa C++

Studi Kasus 2

- Algoritma Selection Sort

```
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
    temp ← xi
    xi ← x
    ximaks ← temp
endfor
```

Subproblem = 1

Masalah pada subproblem = n-1

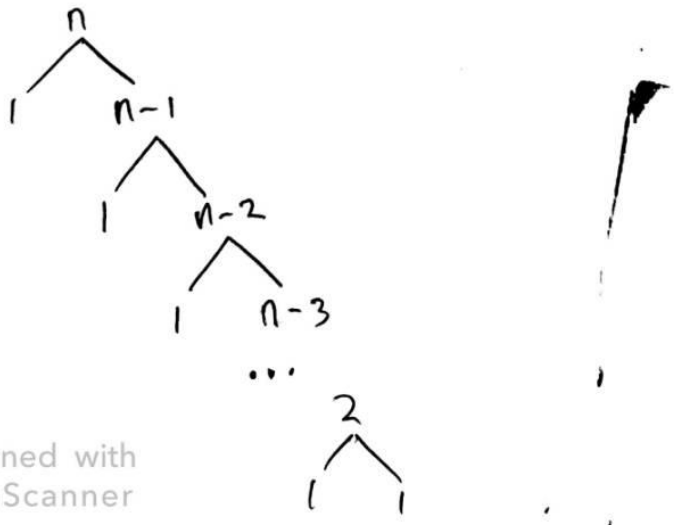
Pembagian = n

Penggabungan = n

Sehingga

$T(n) = \Theta(1)T(n-1) + \Theta(n)$

Recursion Tree



Worst Case

$T(n) = cn + cn - c + cn - 2c + \dots + 2c + cn$

$= c\left(\frac{(n-1)(n-2)}{2}\right) + cn$

$= c\left(\frac{(n^2-3n+2)}{2}\right) + cn$

$= c\left(\frac{n^2}{2}\right) - \frac{3n}{2} + 1 + cn$ → Suku belakang diabaikan, ambil n^2 saja.

$= O(n^2)$

Best Case

$T(n) = cn + cn - c + cn - 2c + \dots + 2c + cn$

$= c\left(\frac{(n-1)(n-2)}{2}\right) + cn$

$= c\left(\frac{(n^2-3n+2)}{2}\right) + cn$

$= c\left(\frac{(n^2-3n+2)}{2}\right) + cn$

$= c\left(\frac{n^2}{2}\right) - \frac{3n}{2} + 1 + cn$ → Suku belakang diabaikan, ambil n^2 saja.

$= \Omega(n^2)$

Average Case

$T(n) = \frac{n^2+n^2}{2} = n^2$

$= \Theta(n^2)$

```
/*
NAMA    : SURIADI VAJRAKARNA
NPM     : 140810180038
KELAS   : B
TANGGAL : 8 MARET 2020
TUGAS 4 - STUDI KASUS 2 - PRAKTIKUM DESAIN DAN ANALISIS ALGORITMA
*/

#include <iostream>
using namespace std;

void swap(int &a, int &b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}

void hasil(int *elmn, int jmlhElmn)
{
    for (int i = 0; i < jmlhElmn; i++)
        cout << elmn[i] << " ";
    cout << endl;
}

void selection_sort(int *elmn, int jmlhElmn)
{
    int i, j, imin;
    for (i = 0; i < jmlhElmn - 1; i++)
    {
        imin = i;
        for (j = i + 1; j < jmlhElmn; j++)
            if (elmn[j] < elmn[imin])
                imin = j;
        swap(elmn[i], elmn[imin]);
    }
}

int main()
{
    int jmlhElmn;
    cout << "Banyak elemen yang akan diurutkan: ";
    cin >> jmlhElmn;
```

```
    int elmn[jmlhElmn];
    cout << "Masukkan " << jmlhElmn << " elemen yang akan diurutkan:\n";
    for (int i = 0; i < jmlhElmn; i++)
    {
        cout << "Elemen ke-" << i+1 << ": ";
        cin >> elmn[i];
    }

    selection_sort(elmn, jmlhElmn);

    cout << "Hasil Pengurutan\n";
    hasil(elmn, jmlhElmn);
}
```

```
PS D:\MEGA\SEMESTER 4\ANALGO\Praktikum\AnalgoKu\AnalgoKu4> g++ -o selections selections.cpp
PS D:\MEGA\SEMESTER 4\ANALGO\Praktikum\AnalgoKu\AnalgoKu4> ./selections
Banyak elemen yang akan diurutkan: 5
Masukkan 5 elemen yang akan diurutkan:
Elemen ke-1: 3
Elemen ke-2: 4
Elemen ke-3: 2
Elemen ke-4: 1
Elemen ke-5: 5
Hasil Pengurutan
1 2 3 4 5
```

Studi Kasus 3: INSERTION SORT

Insertion sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma insertion sort, lakukan langkah-langkah berikut:

- Pelajari cara kerja algoritma insertion sort
- Tentukan T(n) dari rekurensi (pengulangan) insertion sort berdasarkan penentuan rekurensi divide & conquer:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

- Selesaikan persamaan rekurensi T(n) dengan **metode subtitusi** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ
- Lakukan implementasi koding program untuk algoritma insertion sort dengan menggunakan bahasa C++

Studi kasus 3.

- Algoritma Insertion Sort

```
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
end for
```

Subproblem = 1

Masalah pada subproblem = n-1

Pembagian = n

Penggabungan = n

Sehingga

$T(n) = (1)\,T(n-1) + (n)$

Worst Case

$T(n) = cn + cn-c + cn-2c + \ldots + 2c + cn \leq 2cn^2 + cn^2$

$= c\left(\frac{(n-1)(n-2)}{2}\right) + cn \leq 2cn^2 + cn^2$

$= c\left(\frac{cn^2-3n+2}{2}\right) + cn \leq 2cn^2 + cn^2$

$= c\left(\frac{n^2}{2}\right) - c\left(\frac{3n}{2}\right) + c + cn \leq 2cn^2 + cn^2$

$= O(n^2)$

Best Case

$T(n) = cn \leq cn$

$= \Omega(n)$

Average Case

$T(n) = \frac{cn + cn^2}{n}$

$= \ominus(n)$


```
/*
NAMA    : SURIADI VAJRAKARNA
NPM     : 140810180038
KELAS   : B
TANGGAL : 8 MARET 2020
TUGAS 4 - STUDI KASUS 3 - PRAKTIKUM DESAIN DAN ANALISIS ALGORITMA
*/

#include <iostream>
```

```cpp
using namespace std;

void hasil(int *elmn, int jmlhElmn)
{
    for (int i = 0; i < jmlhElmn; i++)
        cout << elmn[i] << " ";
    cout << endl;
}

void insertionSort(int *elmn, int jmlhElmn)
{
    int key, j;
    for (int i = 1; i < jmlhElmn; i++)
    {
        key = elmn[i];
        j = i;
        while (j > 0 && elmn[j - 1] > key)
        {
            elmn[j] = elmn[j - 1];
            j--;
        }
        elmn[j] = key;
    }
}

int main()
{
    int jmlhElmn;
    cout << "Banyak elemen yang akan diurutkan: ";
    cin >> jmlhElmn;

    int elmn[jmlhElmn];
    cout << "Masukkan " << jmlhElmn << " elemen yang akan diurutkan:\n";
    for (int i = 0; i < jmlhElmn; i++)
    {
        cout << "Elemen ke-" << i+1 << ": ";
        cin >> elmn[i];
    }

    insertionSort(elmn, jmlhElmn);

    cout << "Hasil: ";
    hasil(elmn, jmlhElmn);
}
```

```
PS D:\MEGA\SEMESTER 4\ANALGO\Praktikum\AnalgoKu\AnalgoKu4> g++ -o insertions insertions.cpp
PS D:\MEGA\SEMESTER 4\ANALGO\Praktikum\AnalgoKu\AnalgoKu4> ./insertions
Banyak elemen yang akan diurutkan: 5
Masukkan 5 elemen yang akan diurutkan:
Elemen ke-1: 3
Elemen ke-2: 2
Elemen ke-3: 4
Elemen ke-4: 1
Elemen ke-5: 5
Hasil: 1 2 3 4 5
```

Studi Kasus 4: BUBBLE SORT

Bubble sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma bubble sort, lakukan langkah-langkah berikut:

- Pelajari cara kerja algoritma bubble sort
- Tentukan T(n) dari rekurensi (pengulangan) insertion sort berdasarkan penentuan rekurensi divide & conquer:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

- Selesaikan persamaan rekurensi T(n) dengan **metode master** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ
- Lakukan implementasi koding program untuk algoritma bubble sort dengan menggunakan bahasa C++

Studi Kasus 4

Algoritma Bubble Sort

```
Swapped ← false
for i ← 1 to n-1 do
    if array[i-1] > array[i] then
        swap array[i-1] and array[i]
        swapped ← true
    endif
endfor
```

Subproblem = 1

Masalah pada subproblem = n-1

Pembagian = n

Penggabungan = n

Sehingga

$$T(n) = \text{①}(1)\,T(n-1) + \text{②}(n)$$

Worst Case

$$T(n) = cn + cn - c + cn - 2c + \cdots + 2c + c \leq 2cn^2 + cn^2$$

$$= c\left(\frac{(n-1)(n-2)}{2}\right) + c \leq 2cn^2 + cn^2$$

$$= c\left(\frac{(n^2-3n+2)}{2}\right) + c \leq 2cn^2 + cn^2$$

$$= c\left(\frac{n^2}{2}\right) - c\left(\frac{3n}{2}\right) + c \leq 2cn^2 + cn^2$$

$$= O(n^2)$$

Best Case

$$T(n) = cn + cn - c + cn - 2c + \dots + 2c + c \leq 2cn^2 + cn^2$$

$$= c\left(\frac{(n-1)(n-2)}{2}\right) + c \leq 2cn^2 + cn^2$$

$$= c\left(\frac{(n^2+3n+2)}{2}\right) + c \leq 2cn^2 + cn^2$$

$$= c\left(\frac{n^2}{2}\right) + c\left(\frac{3n}{2}\right) + c \leq 2cn^2 + cn^2$$

$$= \Omega(n^2)$$

Average Case

$$T(n) = cn^2 + cn^2$$

$$= \Theta(n^2)$$


```
/*
NAMA    : SURIADI VAJRAKARNA
NPM     : 140810180038
KELAS   : B
TANGGAL : 8 MARET 2020
TUGAS 4 - STUDI KASUS 4 - PRAKTIKUM DESAIN DAN ANALISIS ALGORITMA
*/

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

```
void swap(int &a, int &b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}
void hasil(int *elmn, int jmlhElmn)
{
    for (int i = 0; i < jmlhElmn; i++)
        cout << elmn[i] << " ";
    cout << endl;
}

void bubble_sort(int *elmn, int jmlhElmn)
{
    for (int i = 0; i < jmlhElmn; i++)
    {
        int swaps = 0;
        for (int j = 0; j < jmlhElmn - i - 1; j++)
        {
            if (elmn[j] > elmn[j + 1])
            {
                swap(elmn[j], elmn[j + 1]);
                swaps = 1;
            }
        }
        if (!swaps)
            break;
    }
}

int main()
{
    int jmlhElmn;
    cout << "Banyak elemen yang akan diurutkan: ";
    cin >> jmlhElmn;

    int elmn[jmlhElmn];
    cout << "Masukkan " << jmlhElmn << " elemen yang akan diurutkan:\n";
    for (int i = 0; i < jmlhElmn; i++)
    {
        cout << "Elemen ke-" << i + 1 << ": ";
        cin >> elmn[i];
    }
```

```
    bubble_sort(elmn, jmlhElmn);

    cout << "Hasil: ";
    hasil(elmn, jmlhElmn);
}
```

```
PS D:\MEGA\SEMESTER 4\ANALGO\Praktikum\AnalgoKu\AnalgoKu4> g++ -o bubbles bubbles.cpp
PS D:\MEGA\SEMESTER 4\ANALGO\Praktikum\AnalgoKu\AnalgoKu4> ./bubbles
Banyak elemen yang akan diurutkan: 5
Masukkan 5 elemen yang akan diurutkan:
Elemen ke-1: 3
Elemen ke-2: 4
Elemen ke-3: 1
Elemen ke-4: 2
Elemen ke-5: 5
Hasil: 1 2 3 4 5
```