

LAPORAN PRAKTIKUM 4
Analisis Algoritma



Disusun oleh :

Asep Budi yana Muharam
140810180029

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

program merge-sort.cpp

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

void merge(int arr[], int p, int q, int r)
{
    int i, j, k;
    int n1 = q - p + 1;
    int n2 = r - q;
    int L[n1], R[n2];

    for (i = 0; i < n1; i++)
        L[i] = arr[p + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[q + 1 + j];

    i = 0;
    j = 0;
    k = p;
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1)
    {
        arr[k] = L[i];
        i++;
        k++;
    }
}
```

Asep Budiyana M

140810180029

Tugas 4

```
        while (j < n2)
        {
            arr[k] = R[j];
            j++;
            k++;
        }
    }

void mergeSort(int arr[], int p, int r)
{
    if (p < r)
    {
        int q = p+(r-p)/2;
        mergeSort(arr, p, q);
        mergeSort(arr, q+1, r);
        merge(arr, p, q, r);
    }
}

int main()
{
    int arr[] = {20, 63, 3, 36, 61, 52, 42, 69, 18, 6, 82, 34, 62, 27, 67,
65, 33, 79, 88, 23};
    int p = 0, r = sizeof(arr)/sizeof(arr[0]);

    cout<<"\nBefore: \n";
    for (int i = 0; i<r; i++){
        cout<<arr[i]<<" ";
    }

    mergeSort(arr,p,r-1);

    cout<<"\nAfter: \n";
    for (int i = 0; i<r; i++){
        cout<<arr[i]<<" ";
    }

    return 0;
}

#include <iostream>
using namespace std;

void merge(int arr[], int p, int q, int r)
{

```

Asep BudiYana M

140810180029

Tugas 4

```
int i, j, k;
int n1 = q - p + 1;
int n2 = r - q;
int L[n1], R[n2];

for (i = 0; i < n1; i++)
    L[i] = arr[p + i];
for (j = 0; j < n2; j++)
    R[j] = arr[q + 1 + j];

i = 0;
j = 0;
k = p;
while (i < n1 && j < n2)
{
    if (L[i] <= R[j])
    {
        arr[k] = L[i];
        i++;
    }
    else
    {
        arr[k] = R[j];
        j++;
    }
    k++;
}

while (i < n1)
{
    arr[k] = L[i];
    i++;
    k++;
}

while (j < n2)
{
    arr[k] = R[j];
    j++;
    k++;
}
}

void mergeSort(int arr[], int p, int r)
{

```

Asep Budiyan M

140810180029

Tugas 4

```
if (p < r)
{
    int q = p+(r-p)/2;
    mergeSort(arr, p, q);
    mergeSort(arr, q+1, r);
    merge(arr, p, q, r);
}

int main()
{
    int arr[] = {20, 63, 3, 36, 61, 52, 42, 69, 18, 6, 82, 34, 62, 27, 67,
65, 33, 79, 88, 23};
    int p = 0, r = sizeof(arr)/sizeof(arr[0]);

    cout<<"\nBefore: \n";
    for (int i = 0; i<r; i++){
        cout<<arr[i]<<" ";
    }

    mergeSort(arr,p,r-1);

    cout<<"\nAfter: \n";
    for (int i = 0; i<r; i++){
        cout<<arr[i]<<" ";
    }

    return 0;
}
```

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?

running time = 1.541 sec(s)

compiled and executed in 1.541 sec(s)

```
Before:
20 63 3 36 61 52 42 69 18 6 82 34 62 27 67 65 33 79 88 23
After:
3 6 18 20 23 27 33 34 36 42 52 61 62 63 65 67 69 79 82 88 |
```

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:

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

$T(n)$ dari rekurensi selection sort:

$$T(n) = \begin{cases} a & , n = 1 \\ T(n-1) + cn & , n > 1 \end{cases}$$

- Selesaikan persamaan rekurensi $T(n)$ dengan **metode recursion-tree** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ **metode recursion-tree**

② Selection Sort - recursive type

$$T(n) = \begin{cases} n & , n = 1 \\ T(n-1) + cn & , n > 1 \end{cases}$$

{

$$\begin{array}{ccc}
 cn & \longrightarrow & cn \\
 \downarrow & & \\
 c(n-1) & \longrightarrow & cn - c \\
 \downarrow & & \\
 c((n-1)-1) & \longrightarrow & cn - 2c \\
 \vdots & & \\
 T(1) & \longrightarrow & \theta(n)
 \end{array}$$

Cost the recursion tree, yaitu :

$$\begin{aligned}
 T(n) &= cn + cn - c + cn - 2c + \theta(n) \\
 &= \sum_{i=0}^{n-1} cn - ic + \theta(n) \\
 &= cn(n-1) + \theta(n) \\
 &= cn^2 - c + \theta(n) \\
 &= O(n^2)
 \end{aligned}$$

kompleksitas waktu asimptotiknya adalah $\Omega(n^2)$, $\Theta(n^2)$, dan $O(n^2)$

Asep BudiYana M

140810180029

Tugas 4

- Lakukan implementasi koding program untuk algoritma selection sort dengan menggunakan bahasa C++
program selection-sort.cpp

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

void selectionSort(int arr[], int n)
{
    int i, j, imin, temp;

    for (i=0;i<n-1;i++) {
        imin = i;
        for (j=i+1;j<n;j++) {
            if (arr[j] < arr[imin]) imin = j;
        }
        temp = arr[i];
        arr[i] = arr[imin];
        arr[imin] = temp;
    }
}

int main()
{
    int arr[] = { 1, 45, 54, 71, 66, 12 };
    int n = sizeof(arr)/sizeof(arr[0]);

    cout<<"Unsorted Array: ";
    for (int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }

    selectionSort(arr,n);

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

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:

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

$T(n)$ dari rekurensi insertion sort:

$$T(n) = \begin{cases} a & , n = 1 \\ T(n-1) + cn & , n > 1 \end{cases}$$

- Selesaikan persamaan rekurensi $T(n)$ dengan **metode substitusi** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ *metode substitusi*

③ insertion sort - metode substitusi

$$T(n) = \begin{cases} a & , n = 1 \\ T(n-1) + cn & , n > 1 \end{cases}$$

$$\begin{aligned} T(n) &= cn + T(n-1) \\ &= cn + \{cn-1 + T(n-2)\} \\ &= cn + cn-1 + \{cn-2 + T(n-3)\} \\ &= \dots \\ &= cn + cn-1 + cn-2 + \dots + c2 + T(1) \\ &= c\{n + (n-1) + (n-2) + \dots + 2\} + a \\ &= c\{(n-1)(n+2)/2\} + a \\ &= cn^2/2 + cn/2 + (a-c) \\ &= O(n^2) \end{aligned}$$

kompleksitas waktu asimptotiknya adalah $\Omega(n^2)$, $\Theta(n^2)$, dan $O(n^2)$

- Lakukan implementasi koding program untuk algoritma insertion sort dengan menggunakan bahasa C++
program insertion-sort.cpp

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

void insertionSort(int arr[], int n)
{
    int i, j, insert;
    for (i=1; i<n; i++) {
```



```

        insert = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > insert) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = insert;
    }
}

int main()
{
    int arr[] = { 1, 45, 54, 71, 66, 12 };
    int n = sizeof(arr)/sizeof(arr[0]);

    cout<<"Unsorted Array: ";
    for (int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }

    insertionSort(arr,n);

    cout<<"\nSorted Array: ";
    for (int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }
    return 0;
}

```

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:

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

$T(n)$ dari rekurensi bubble sort:

$$T(n) = \begin{cases} a, & n = 1 \\ T(n-1) + cn, & n > 1 \end{cases}$$

Tugas 4

- Selesaikan persamaan rekurensi $T(n)$ dengan **metode master** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ

metode master

Karena $T(n) = \begin{cases} a & , n=1 \\ T(n-1) + cn & , n>1 \end{cases}$, sehingga $a=1$, dan $b=1$. yang mana tidak memenuhi syarat utk metode master (yaitu $a \geq 1$ dan $b > 1$), maka rekurensi algoritma bubble sort tidak bisa diselesaikan dengan metode master.

- Lakukan implementasi koding program untuk algoritma bubble sort dengan menggunakan bahasa C++

program bubble-sort.cpp

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

void bubbleSort(int arr[], int n)
{
    int temp;
    if (n == 1)
        return;

    for (int i=0; i<n-1; i++) {
        if (arr[i] > arr[i+1]) {
            temp = arr[i];
            arr[i] = arr[i+1];
            arr[i+1] = temp;
        }
    }
    bubbleSort(arr, n-1);
}

int main()
{
    int arr[] = { 64, 34, 25, 12, 22, 11, 90 };
    int n = sizeof(arr)/sizeof(arr[0]);

    cout<<"Unsorted Array: ";
    for (int i=0; i<n; i++){
        cout<<arr[i]<<" ";
    }

    bubbleSort(arr, n);

    cout<<"\nSorted Array: ";
    for (int i=0; i<n; i++){
```

Asep Budiyan M

140810180029

Tugas 4

```
        cout<<arr[i]<<" ";  
    }  
    return 0;  
}
```