LAPORAN PRAKTIKUM 4

MATA KULIAH
ANALISIS ALGORITMA

Disusun Oleh:

Falah Rizqi Abdullah Fairuz
140810180069

PROGRAM STUDI S-1 TEKNIK INFORMATIKA
DEPARTEMEN ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PADJADJARAN

Worksheet 4

Studi Kasus

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?

Jawab:

a.

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


void Merge(int *a, int low, int high, int mid)
{

    int i, j, k, temp[high-low+1];
    i = low;
    k = 0;
    j = mid + 1;


    while (i <= mid && j <= high)
    {
        if (a[i] < a[j])
        {
            temp[k] = a[i];
            k++;
            i++;
        }
        else
        {
            temp[k] = a[j];
            k++;
            j++;
        }
    }


    while (i <= mid)
    {
        temp[k] = a[i];
        k++;
```

```
        i++;
    }


    while (j <= high)
    {
        temp[k] = a[j];
        k++;
        j++;
    }



    for (i = low; i <= high; i++)
    {
        a[i] = temp[i-low];
    }
}


void MergeSort(int *a, int low, int high)
{
    int mid;
    if (low < high)
    {
        mid=(low+high)/2;

        MergeSort(a, low, mid);
        MergeSort(a, mid+1, high);


        Merge(a, low, high, mid);
    }
}

int main()
{

    int n, i;
    cout<<"\nEnter the number of data element to be sorted: ";
    cin>>n;

    int arr[n];
    for(i = 0; i < n; i++)
    {
        cout<<"Enter element "<<i+1<<": ";
        cin>>arr[i];
    }
```

```
    auto start = chrono::steady_clock::now();
    MergeSort(arr, 0, n-1);
    auto end = chrono::steady_clock::now();

    // Printing the sorted data.
    cout<<"\nSorted Data ";
    for (i = 0; i < n; i++)
        cout<<" - "<<arr[i];
    cout << endl;
    cout << "Elapsed time in nanoseconds : "
        << chrono::duration_cast<chrono::nanoseconds>(end - start).count()
        << " ns" << endl;

    cout << "Elapsed time in microseconds : "
        << chrono::duration_cast<chrono::microseconds>(end - start).count()
        << " µs" << endl;

    cout << "Elapsed time in milliseconds : "
        << chrono::duration_cast<chrono::milliseconds>(end - start).count()
        << " ms" << endl;

    cout << "Elapsed time in seconds : "
        << chrono::duration_cast<chrono::seconds>(end - start).count()
        << " sec";
    return 0;
}
```

```
Enter the number of data element to be sorted: 20
Enter element 1: 45
Enter element 2: 34
Enter element 3: 87
Enter element 4: 65
Enter element 5: 43
Enter element 6: 12
Enter element 7: 50
Enter element 8: 98
Enter element 9: 76
Enter element 10: 34
Enter element 11: 58
Enter element 12: 13
Enter element 13: 32
Enter element 14: 41
Enter element 15: 91
Enter element 16: 29
Enter element 17: 30
Enter element 18: 49
Enter element 19: 87
Enter element 20: 70

Sorted Data  - 12 - 13 - 29 - 30 - 32 - 34 - 34 - 41 - 43 - 45 - 49 - 50 - 58 - 65 - 70 - 76 - 87 - 87 - 91 - 98
Elapsed time in nanoseconds : 2900 ns
Elapsed time in microseconds : 2 µs
Elapsed time in milliseconds : 0 ms
Elapsed time in seconds : 0 sec
```

b.

Untuk di program hasilnya : 2900 ns

Tapi jika sesuai dengan O -> $T(20\ \log_{10} 20) = 26$

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++

Jawab:

```
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
```

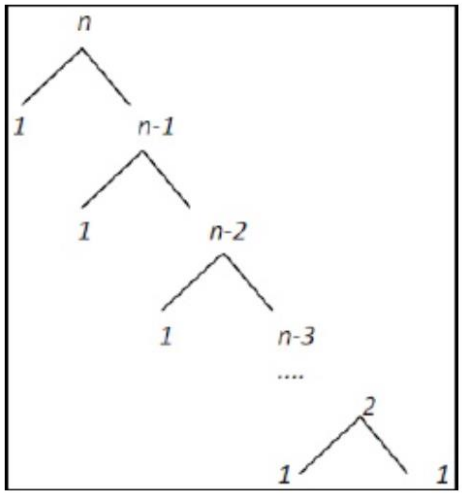
Subproblem = 1

Masalah setiap subproblem = n-1

Waktu proses pembagian = n

Waktu proses penggabungan = n

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



T(n) = cn + cn-c +cn-2c + ... + 2c +cn

= c((n-1)(n-2)/2) + cn

= c((n^2-3n+2)/2) + cn

= c(n^2/2)-(3n/2)+1 + cn

=O(n^2)

T(n) = cn + cn-c +cn-2c + ... + 2c +cn

= c((n-1)(n-2)/2) + cn

= c((n^2-3n+2)/2) + cn

= c(n^2/2)-(3n/2)+1 + cn

= Ω (n^2)

T(n) = cn^2

= Θ(n^2)

Source code:

```
#include <iostream>
#include<conio.h>

using namespace std;

int data[100],data2[100];
int n;
```

```
void tukar(int a, int b)
{
    int t;
    t = data[b];
    data[b] = data[a];
    data[a] = t;
}
void selection_sort()
{
    int pos,i,j;
    for(i=1;i<=n-1;i++)
    {
        pos = i;
        for(j = i+1;j<=n;j++)
        {
           if(data[j] < data[pos]) pos = j;
        }
        if(pos != i) tukar(pos,i);
    }
}

int main()
{
    cout<<"\nEnter the number of data element to be sorted: ";cin>>n;
    for(int i=1;i<=n;i++)
    {
        cout<<"Enter element "<<i<<": ";
        cin>>data[i];
        data2[i]=data[i];
    }

    selection_sort();
    cout<<"Sorted Data: "<<endl;
    for(int i=1; i<=n; i++)
    {
        cout<<" "<<data[i];
    }

}
```

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++

Jawab:

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
```

Subproblem = 1

Masalah setiap subproblem = n-1

Waktu proses penggabungan = n

Waktu proses pembagian = n

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

$T(n) = cn + cn\text{-}c + cn\text{-}2c + \ldots + 2c + cn <= 2cn^2 + cn^2$

$= c((n-1)(n-2)/2) + cn <= 2cn^2 + cn^2$

$= c((n^2\text{-}3n+2)/2) + cn <= 2cn^2 + cn^2$

$= c(n^2/2)-c(3n/2)+c+cn <= 2cn^2 + cn^2$

$=O(n^2)$

$T(n) = cn <= cn$

$= \Omega(n)$

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

$= \Theta(n)$

Source code:

```
#include <iostream>
#include <conio.h>

using namespace std;

int data[100],data2[100],n;

void insertion_sort()
{
    int temp,i,j;
    for(i=1;i<=n;i++){
        temp = data[i];
        j = i -1;
        while(data[j]>temp && j>=0){
            data[j+1] = data[j];
            j--;
        }
        data[j+1] = temp;
    }
}
int main()
{
    cout<<"Enter the number of data element to be sorted: "; cin>>n;
    cout<<endl;

    for(int i=1;i<=n;i++)
    {
      cout<<"Enter element "<<i<<": ";
      cin>>data[i];
      data2[i]=data[i];
    }
```

```
    insertion_sort();
    cout<<"\nSorted Data: "<<endl;
    for(int i=1; i<=n; i++)
    {
      cout<<data[i]<<" ";
    }
}
```

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++

Jawab:

Algoritma

```
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 setiap subproblem = n-1

Waktu proses pembagian = n

Waktu proses penggabungan = n

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

T(n) = cn + cn-c +cn-2c + ... + 2c +c <= $2cn^2 + cn^2$

= c((n-1)(n-2)/2) + c<= $2cn^2 + cn^2$

= c((n^2-3n+2)/2) + c<= $2cn^2 + cn^2$

= c(n^2/2)-c(3n/2)+2c <= $2cn^2 + cn^2$

=O(n^2)

T(n) = cn + cn-c +cn-2c + ... + 2c +c <= $2cn^2 + cn^2$

= c((n-1)(n-2)/2) + c<= $2cn^2 + cn^2$

= c((n^2-3n+2)/2) + c<= $2cn^2 + cn^2$

= c(n^2/2)-c(3n/2)+2c <= $2cn^2 + cn^2$

= Ω (n^2)

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

= Θ(n^2)

Source code:

```
#include <iostream>
#include <conio.h>

using namespace std;


int main(){
    int arr[100],n,temp;

    cout<<"Enter the number of data element to be sorted: ";cin>>n;


    for(int i=0;i<n;++i){
        cout<<"Enter element "<<i+1<<" : ";cin>>arr[i];
    }

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

```
                arr[j+1]=temp;
            }
        }
    }

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

}
```