Nama : Shania Salsabila
NPM : 140810180014
Tugas 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?

SOURCE CODE

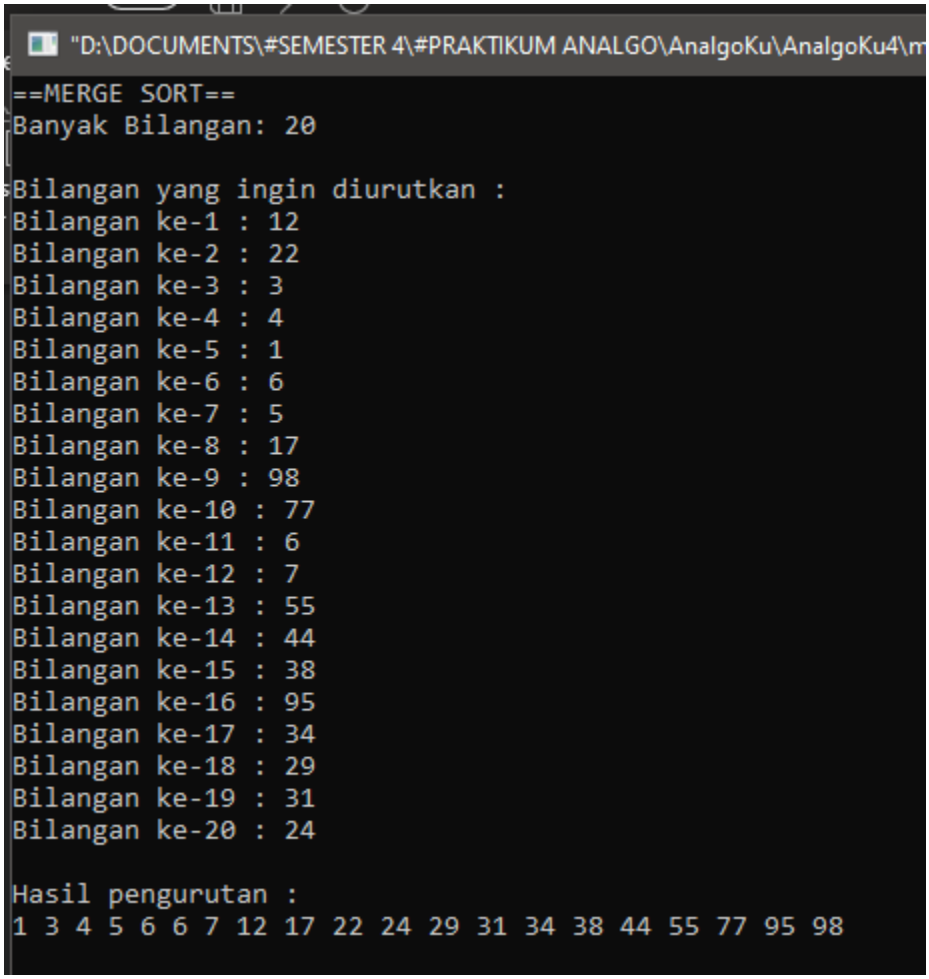
```
/*
Nama    : Shania Salsabila
Kelas   : B
NPM     : 140810180014
Nama program : merge sort
*/

#include <iostream>
using namespace std;

int a[50];
void merge(int,int,int);
void merge_sort(int low,int high)
{
 int mid;
 if(low<high)
 {
  mid=(low+high)/2;
  merge_sort(low,mid);
  merge_sort(mid+1,high);
  merge(low,mid,high);
 }
}
void merge(int low,int mid,int high)
{
 int h,i,j,b[50],k;
 h=low;
 i=low;
 j=mid+1;
 while((h<=mid)&&(j<=high))
 {
```

```
  if(a[h]<=a[j])
  {
   b[i]=a[h]; h++;
  }
  else
  {
   b[i]=a[j]; j++;
  } i++;
 }
 if(h>mid)
 {
  for(k=j;k<=high;k++)
  {
   b[i]=a[k]; i++;
  }
 }
 else
 {
  for(k=h;k<=mid;k++)
  {
   b[i]=a[k]; i++;
  }
 }
 for(k=low;k<=high;k++)
  a[k]=b[k];
}
main()
{
    int num,i;
    cout<<"==MERGE SORT=="<<endl;
    cout<<"Banyak Bilangan: ";cin>>num;
    cout<<endl;
    cout<<"Bilangan yang ingin diurutkan :"<<endl;
    for(i=1;i<=num;i++)
    {
    cout<<"Bilangan ke-"<<i<<" : ";cin>>a[i] ;
    }
    merge_sort(1,num);
    cout<<endl;
    cout<<"Hasil pengurutan :"<<endl;
    for(i=1;i<=num;i++)
    cout<<a[i]<<" ";
    cout<<endl<<endl<<endl<<endl;
}
```

SS



Kompleksitas waktu algoritma merge sort O(n lg n). $T(20 \log_{10} 20)=26$

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:

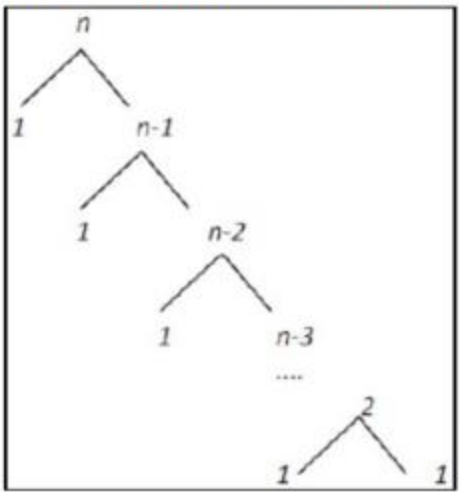
$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

- 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++

```
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

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



$T(n) = cn + cn\text{-}c + cn\text{-}2c + \ldots\ldots + 2c + cn$
$= c((n-1)(n-2)/2) + cn$
$= c((n^2-3n+2)/2) + cn$
$= c(n^2/2)-(3n/2)+1 + cn$
$= O(n^2)$

$T(n) = cn + cn\text{-}c + cn\text{-}2c + \ldots\ldots + 2c + cn$
$= c((n-1)(n-2)/2) + cn$
$= c((n^2-3n+2)/2) + cn$
$= c(n^2/2)-(3n/2)+1 + cn$
$= \Omega(n^2)$

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

SOURCE CODE

```
/*
Nama    : Shania Salsabila
Kelas   : B
NPM     : 140810180014
Nama program : selection sort
*/

#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<<"==SELECTION SORT==";
    cout<<"Banyak Data : ";cin>>n;
    for(int i=1;i<=n;i++)
    {
        cout<<"Data ke-"<<i<<" : ";
        cin>>data[i];
        data2[i]=data[i];
    }

    selection_sort();
    cout<<"Hasil : "<<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++

```
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

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

$T(n) = cn + cn\text{-}c + cn\text{-}2c + \ldots\ldots + 2c + cn <= 2cn^2 + cn^2$
$= c((n\text{-}1)(n\text{-}2)/2) + cn <= 2cn^2 + cn^2$
$= c((n^2\text{-}3n+2)/2) + cn <= 2cn^2 + cn^2$
$= c(n^2/2)\text{-}c(3n/2)+c+cn <= 2cn^2 + cn^2$
$= O(n^2)$

$T(n) = cn <= cn$
$= \Omega(n)$

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

SOURCE CODE

```
      /*
Nama    : Shania Salsabila
Kelas   : B
NPM     : 140810180014
Nama program : insertion sort
*/

#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<<"==INSERTION SORT==";
    cout<<"Banyak Data : "; cin>>n;
    cout<<endl;
    for(int i=1;i<=n;i++)
    {
      cout<<"Data ke-"<<i<<" : ";
      cin>>data[i];
      data2[i]=data[i];
    }
    insertion_sort();
    cout<<"\nHasil : "<<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++

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

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

$= c((n\text{-}1)(n\text{-}2)/2) + c <= 2cn^2 + cn^2$

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

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

$= O(n^2)$

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

$= c((n\text{-}1)(n\text{-}2)/2) + c <= 2cn^2 + cn^2$

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

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

$= \Omega\ (n^2)$

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

$= \Theta(n^2)$

SOURCE CODE

```
/*
Nama    : Shania Salsabila
Kelas   : B
NPM     : 140810180014
Nama program : bubble sort
*/

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

```cpp
using namespace std;


int main(){
    int arr[100],n,temp;
    cout<<"Banyak nilai : ";cin>>n;

    for(int i=0;i<n;++i){
        cout<<"Nilai ke-"<<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<<"\nHasil : "<<endl;
    for(int i=0;i<n;i++){
        cout<<" "<<arr[i];
    }
}
```