

**LAPORAN PRAKTIKUM ANALISIS ALGORITMA
PARADIGMA DIVIDE AND CONQUER**



Oleh

Shalvina Zahwa Aulia – 140810180052

**PROGRAM STUDI S-1 TEKNIK INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PADJAJARAN**

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 : Shalvina Zahwa Aulia

NPM : 140810180052

Kelas : B

Deskripsi : Program C++ merge sort dari terkecil ke terbesar

*/

#include <iostream>

#include<chrono>

using namespace std;

void gabung(int* A, int p, int q, int r){

int n1 = q-p+1;

int n2 = r-q;

int kiri[n1+1];

int kanan[n2+1];

for(int i=1; i<=n1; i++){

kiri[i]=A[(p-1)+i-1];

}

for(int j=1; j<=n2; j++){

kanan[j]=A[(q-1)+j];

}

int i=0, j=0;

kiri[n1]=2147483647;

kanan[n2]=2147483647;

for(int k=(p-1); k<r; k++){

if(kiri[i]<=kanan[j]){

```

        A[k]=kiri[i];
        i+=1;
    }
    else{
        A[k]=kanan[j];
        j+=1;
    }
}
}

void mergeSort(int* A, int p, int r){
    int q;
    if(p<r){
        q=(p+r)/2;
        mergeSort(A,p,q);
        mergeSort(A,q+1,r);
        gabung(A,p,q,r);
    }
}

int main(){
    int A[100];
    int n;
    cout << "Banyak data\t : "; cin >> n;
    for(int i=0; i<n; i++){
        cout << "Angka ke-"<<i+1<<" : "; cin >>A[i];
    }
    auto start = chrono::steady_clock::now();
    mergeSort(A,1,n);
    auto end = chrono::steady_clock::now();
    cout << "Hasil merge sort: ";
    for(int i=0; i<n; i++){
        cout << A[i] << " ";
    }
}

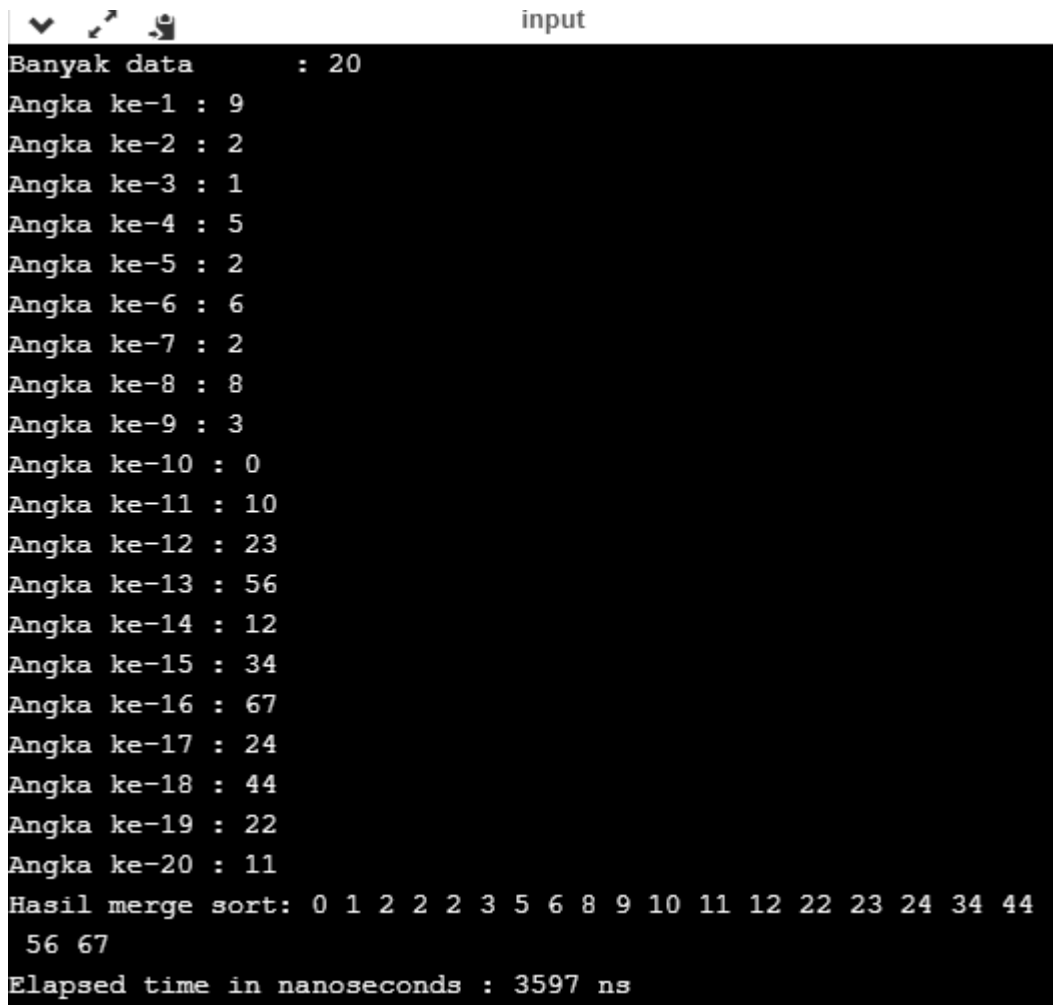
```

```

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

    return 0;
}

```



```

input
Banyak data      : 20
Angka ke-1 : 9
Angka ke-2 : 2
Angka ke-3 : 1
Angka ke-4 : 5
Angka ke-5 : 2
Angka ke-6 : 6
Angka ke-7 : 2
Angka ke-8 : 8
Angka ke-9 : 3
Angka ke-10 : 0
Angka ke-11 : 10
Angka ke-12 : 23
Angka ke-13 : 56
Angka ke-14 : 12
Angka ke-15 : 34
Angka ke-16 : 67
Angka ke-17 : 24
Angka ke-18 : 44
Angka ke-19 : 22
Angka ke-20 : 11
Hasil merge sort: 0 1 2 2 2 3 5 6 8 9 10 11 12 22 23 24 34 44
56 67
Elapsed time in nanoseconds : 3597 ns

```

Kompleksitas waktu yang terhitung oleh program adalah : 3597 ns

$O(10 \log 20) = 26,02$

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++

/*

Nama : Shalvina Zahwa Aulia

NPM : 140810180052

Kelas : B

Deskripsi : Program C++ selection sort dari terkecil ke terbesar

*/

```
#include<iostream>
```

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

```
using namespace std;
```

```
int x1[100], x2[100];
```

```
int n;
```

```
void tukar(int p, int q){
```

```
    int temp;
```

```
    temp = x1[q];
```

```
    x1[q] = x1[p];
```

```
    x1[p] = temp;
```

```
}
```

```
void selectionSort(){
```

```
    int pos, i, j;
```

```
    for(int i=1; i<=n-1; i++){
```

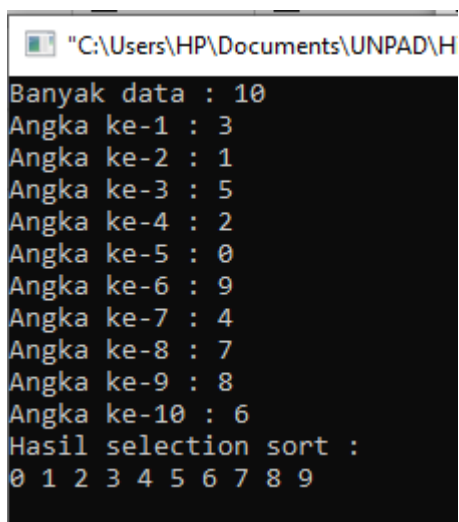
```
        pos=i;
```

```

        for(int j=i+1; j<=n; j++){
            if(x1[j]<x1[pos])
                pos = j;
        }
        if(pos!=i)
            tukar(pos,i);
    }
}

int main(){
    cout << "Banyak data : "; cin >> n;
    for(int i=1; i<=n; i++){
        cout << "Angka ke-"<<i<<" : "; cin>>x1[i];
        x2[i]=x1[i];
    }
    selectionSort();
    cout << "Hasil selection sort : \n";
    for(int i=1; i<=n; i++){
        cout<<x1[i]<< " ";
    }
    getch();
}

```



```

"C:\Users\HP\Documents\UNPAD\H
Banyak data : 10
Angka ke-1 : 3
Angka ke-2 : 1
Angka ke-3 : 5
Angka ke-4 : 2
Angka ke-5 : 0
Angka ke-6 : 9
Angka ke-7 : 4
Angka ke-8 : 7
Angka ke-9 : 8
Angka ke-10 : 6
Hasil selection sort :
0 1 2 3 4 5 6 7 8 9

```

for i <- n downto 2 do assignment n-1 kali

 imaks <- 1

 for j <- 2 to i do

 if $X_j > X_{maks}$ then

 imaks <- j

 endif

 endfor

 temp <- X_i

X_i <- X_{maks}

X_{maks} <- temp

endfor

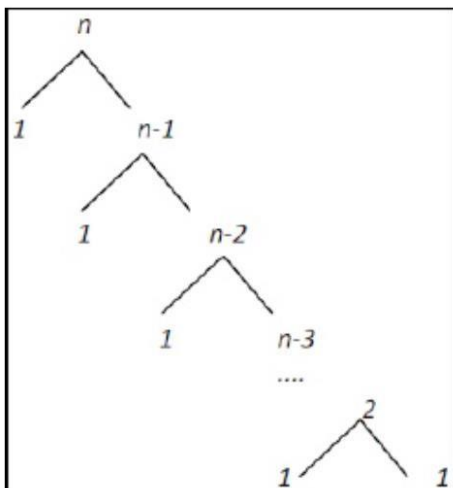
Subproblem = 1

Masalah per subproblem = n-1

Waktu proses pembagian = n

Waktu proses penggabungan = n

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



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

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

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

$$= c(n^2/2) - (3n/2) - 1_cn$$

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

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

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 substitusi 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++

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
```

/*

Nama : Shalvina Zahwa Aulia

NPM : 140810180052

Kelas : B

Deskripsi : Program C++ selection sort dari terkecil ke terbesar

*/

```
#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<<"Jumlah Data : "; cin>>n;
    cout<<endl;
    for(int i=1;i<=n;i++)
    {
        cout<<"Angka ke-"<<i<<" : ";
        cin>>data[i];
        data2[i]=data[i];
    }
    insertion_sort();
    cout<<"\nHasil insertion sort : "<<endl;
    for(int i=1; i<=n; i++)
    {
        cout<<data[i]<<" ";
    }
    getch();
}

```

```
"C:\Users\HP\Documents\UNPAD\Himatif\K
Jumlah Data : 10
Angka ke-1 : 99
Angka ke-2 : 2
Angka ke-3 : 1
Angka ke-4 : 33
Angka ke-5 : 12
Angka ke-6 : 65
Angka ke-7 : 23
Angka ke-8 : 78
Angka ke-9 : 32
Angka ke-10 : 13

Hasil insertion sort :
1 2 12 13 23 32 33 65 78 99
```

Subproblem = 1

Masalah setiap subproblem = $n-1$

Waktu proses penggabungan = n

Waktu proses pembagian = n

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

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

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

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

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

$$= O(n^2)$$

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

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

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++

/*

Nama : Shalvina Zahwa Aulia

NPM : 140810180052

Kelas : B

Deskripsi : Program C++ bubble sort dari terkecil ke terbesar

*/

```
#include <iostream>
```

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

```
using namespace std;
```

```
int main(){
```

```
int arr[100],n,temp;
```

```
cout<<"Banyak data : ";cin>>n;
```

```
for(int i=0;i<n;++i){
```

```
    cout<<"Angka 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 Bubble Sort : "<<endl;
for(int i=0;i<n;i++){
    cout<<" "<<arr[i];
}
}

```

```

"C:\Users\HP\Documents\UNPAD\Himatif\K U L I A H\Semester4\analisisAlgoritma
Banyak data : 10
Angka ke-1 : 99
Angka ke-2 : 77
Angka ke-3 : 54
Angka ke-4 : 85
Angka ke-5 : 12
Angka ke-6 : 34
Angka ke-7 : 22
Angka ke-8 : 67
Angka ke-9 : 86
Angka ke-10 : 32

Hasil Bubble Sort :
12 22 32 34 54 67 77 85 86 99
Process returned 0 (0x0)   execution time : 20.410 s

```

Subproblem = 1

Masalah setiap subproblem = $n-1$

Waktu proses pembagian = n

Waktu proses penggabungan = n

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

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

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

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

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

$$= O(n^2)$$

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

$$\begin{aligned}
T(n) &= cn^2 + cn^2 \\
&= \Theta(n^2)
\end{aligned}$$