

LAPORAN PRAKTIKUM 4

ANALISIS ALGORITMA

Paradigma Algoritma Divide and Conquer



Disusun oleh :

NAMA : Sharashena Chairani
Kelas : B
NPM : 140810180022

Program Studi S-1 Teknik Informatika
Departemen Ilmu Komputer
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++
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?

1. Implementasi Koding dalam C++

/*

Nama : Sharashena Chairani

Kelas : B

NPM : 140810180022

Deskripsi : Mengurutkan bilangan dalam sebuah array dengan menggunakan algoritma merge sort

*/

```
#include <iostream>
```

```
#include <chrono>
```

```
using namespace std;
```

```
void merge(int* in, int p, int q,int r){
```

```
    int n1 = q-p+1;
```

```
    int n2 = r-q;
```

```
    int L[n1+1];
```

```
    int R[n2+1];
```

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

```
        L[i-1] = in[(p-1)+i-1];
```

```
    }
```

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

```
        R[j-1] = in[(q-1)+j];
```

```
    }
```

```

int i=0;
int j=0;
L[n1]=2147483647;
R[n2]=2147483647;

for (int k=(p-1); k<r; k++){
    if(L[i]<=R[j]){
        in[k]=L[i];
        i = i+1;
    }
    else{
        in[k]=R[j];
        j = j+1;
    }
}
}

```

```

void msort(int* in, int p, int r){
    int q;
    if(p<r){
        q = (p+r)/2;
        msort(in, p, q);
        msort(in, q+1, r);

        merge(in, p, q, r);
    }
}

```

```

void data(int* a, int& n){

```

```

        cout
        <<"~~~~~"<<endl;
        cout << "Masukkan banyak data: ";
        cin >> n;

        for (int i=0; i<n; i++){
            cout << "Masukkan angka: ";
            cin >> a[i];
        }
    }

    int main(){
        int in[20];
        int n;
        data(in,n);

        cout<<"~~~~~ Merge Sort ~~~~~"<<endl;
        auto start = chrono::steady_clock::now();
        msort(in,1,n);
        auto end = chrono::steady_clock::now();

        cout <<endl << "Hasil array setelah disorting : ";
        for(int i=0; i<n; i++){
            cout << in[i] << " ";
        }

        cout<<endl;
        cout<< "Waktu yang dibutuhkan komputer dalam Nanosekon: "
            << chrono::duration_cast<chrono::nanoseconds>(end - start).count()
            << " ns" << endl;
    }
}

```

```

        cout
    <<" ~~~~~" <<endl;

    return 0;
}

```

```

D:\Semester 4\Prak. Analgo\Pertemuan 4\mergesort.exe
Masukkan banyak data: 20
Masukkan angka: 6
Masukkan angka: 8
Masukkan angka: 4
Masukkan angka: 8
Masukkan angka: 6
Masukkan angka: 4
Masukkan angka: 3
Masukkan angka: 2
Masukkan angka: 12
Masukkan angka: 13
Masukkan angka: 43
Masukkan angka: 23
Masukkan angka: 54
Masukkan angka: 66
Masukkan angka: 68
Masukkan angka: 76
Masukkan angka: 98
Masukkan angka: 100
Masukkan angka: 54
Masukkan angka: 34
~~~~~ Merge Sort ~~~~~
Hasil array setelah disorting : 2 3 4 4 6 6 8 8 12 13 23 34 43 54 54 66 68 76 98 100
Waktu yang dibutuhkan komputer dalam Nanosekon: 4800 ns
~~~~~
-----
Process exited after 22.74 seconds with return value 0
Press any key to continue . . .

```

2. Proses pada program = 4800 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++

T(n) selection sort

```
for i  $\leftarrow$  n downto 2 do {pass sebanyak n-1 kali}  
    imaks  $\leftarrow$  1  
    for j  $\leftarrow$  2 to i do  
        if  $x_j > x_{\text{imaks}}$  then  
            imaks  $\leftarrow$  j  
        endif  
    endfor  
    {pertukarkan  $x_{\text{imaks}}$  dengan  $x_i$ }  
    temp  $\leftarrow$   $x_i$   
     $x_i \leftarrow x_{\text{imaks}}$   
     $x_{\text{imaks}} \leftarrow$  temp  
endfor
```

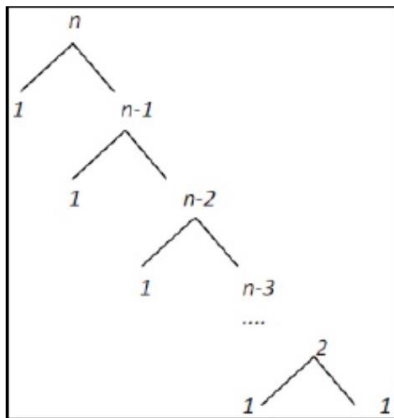
Subproblem = 1

Masalah setiap subproblem = n-1

Waktu proses pembagian = n

Waktu proses penggabungan = n

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



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

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

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

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

$$= O(n^2)$$

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

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

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

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

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

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

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

Implementasi Koding dalam C++

/*

Nama : Sharashena Chairani

Kelas : B

NPM : 140810180022

Deskripsi : Mengurutkan bilangan dalam sebuah array dengan menggunakan algoritma 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 temp,i,j;
    for(i=1;i<=n-1;i++)
    {
        temp = i;
        for(j = i+1;j<=n;j++)
        {
            if(data[j] < data[temp]) temp = j;
        }
        if(temp != i) tukar(temp,i);
    }
}

int main()
{
    cout<<"~~~~~Selection Sort~~~~~" <<endl;
    cout<<"Masukkan Jumlah Array : ";cin>>n;
    cout << "\n~~~~~" << endl;
    for(int i=1;i<=n;i++)
    {
        cout<<"Masukkan data ke-"<<i<<" : ";
        cin>>data[i];
        data2[i]=data[i];
    }

    selection_sort();

    cout << "\n~~~~~" << endl;

```



```

        cout<<"Data array setelah disorting : "<<endl;
        for(int i=1; i<=n; i++)
        {
            cout<<" "<<data[i];

        }

        getch();
    }
}

```

```

D:\Semester 4\Prak. Analgo\Pertemuan 4\selectionsort.exe
~~~~~Selection Sort~~~~~
Masukkan Jumlah Array : 5

~~~~~
Masukkan data ke-1 : 6
Masukkan data ke-2 : 4
Masukkan data ke-3 : 5
Masukkan data ke-4 : 2
Masukkan data ke-5 : 3

~~~~~
Data array setelah disorting :
2 3 4 5 6
-----
Process exited after 18.68 seconds with return value 0
Press any key to continue . . .

```

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++
-

T(n) Insertion Sort

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

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

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

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

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

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

$$= O(n^2)$$

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

$$= \Omega(n)$$

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

$$= \Theta(n)$$

Implementasi Insertion Sort Koding dalam C++

/*

Nama : Sharashena Chairani

Kelas : B

NPM : 140810180022

Deskripsi : Mengurutkan bilangan dalam sebuah array dengan menggunakan algoritma insertion sort

*/

```
#include <iostream>
```

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

```
using namespace std;
```

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

```
void insertionSort()
```

```
{
```

```
    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()
```

```
{
```

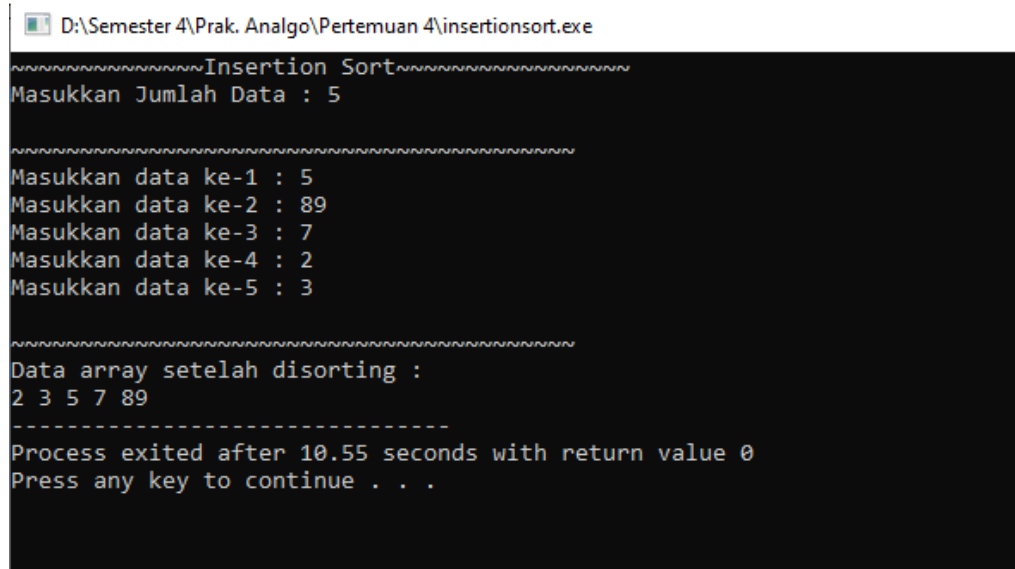
```

cout<<"~~~~~Insertion Sort~~~~~" <<endl;
cout<<"Masukkan Jumlah Data : "; cin>>n;

cout << "\n~~~~~" << endl;
for(int i=1;i<=n;i++)
{
    cout<<"Masukkan data ke-"<<i<<" : ";
    cin>>data[i];
    data2[i]=data[i];
}
cout << "\n~~~~~" << endl;
insertionSort();
cout<<"Data array setelah disorting : "<<endl;
for(int i=1; i<=n; i++)
{
    cout<<data[i]<<" ";
}

getch();
}

```



```

D:\Semester 4\Prak. Analgo\Pertemuan 4\insertionsort.exe
~~~~~Insertion Sort~~~~~
Masukkan Jumlah Data : 5

~~~~~
Masukkan data ke-1 : 5
Masukkan data ke-2 : 89
Masukkan data ke-3 : 7
Masukkan data ke-4 : 2
Masukkan data ke-5 : 3

~~~~~
Data array setelah disorting :
2 3 5 7 89
-----
Process exited after 10.55 seconds with return value 0
Press any key to continue . . .

```

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++

$T(n)$ Bubble Sort

Subproblem = 1

Masalah setiap subproblem = $n-1$

Waktu proses pembagian = n

Waktu proses penggabungan

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

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

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

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

Implementasi Bubble Sort Koding dalam C++

/*

Nama : Sharashena Chairani

Kelas : B

NPM : 140810180022

Deskripsi : Mengurutkan bilangan dalam sebuah array dengan menggunakan algoritma bubble sort

*/

```
#include <iostream>
```

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

```
using namespace std;
```

```
int main(){
```

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

```
    cout << "~~~~~Bubble Sort~~~~~" << endl;
```

```
    cout<<"Masukkan banyak Elemen : ";cin>>size;
```

```
    cout << "\n~~~~~" << endl;
```

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

```
        cout<<"Masukkan Data ke-"<<i+1<<" : ";cin>>arr[i];
```

```
    }
```

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

```
        for( int j = 0; j < (size-1); j++){
```

```
            if(arr[j]>arr[j+1]){
```

```
                temp = arr[j];
```

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

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

```
            }
```


```

        }
    }
    cout << "\n~~~~~" <<
endl;

    cout<<"Array setelah bubble sort : "<<endl;
    for(int i = 0; i < size; i++){
        cout<<" "<<arr[i];

    }
}

```

 D:\Semester 4\Prak. Analgo\Pertemuan 4\bubblesort.exe

```

~~~~~Bubble Sort~~~~~
Masukkan banyak Elemen : 5

~~~~~
Masukkan Data ke-1 : 4
Masukkan Data ke-2 : 7
Masukkan Data ke-3 : 9
Masukkan Data ke-4 : 1
Masukkan Data ke-5 : 78

~~~~~
Array setelah bubble sort :
1 4 7 9 78
-----
Process exited after 7.768 seconds with return value 0
Press any key to continue . . .

```