**LAPORAN PRAKTIKUM ANALGO**

**MODUL 4 Rekurensi dan Paradigma Algoritma Divide & Conquer**

**Karimah Azzuhdu T.M. 140810170027**

**Studi Kasus 1 : MERGE SORT**

1. Buat Program C++

Source Code :

/\* Karimah A.T.M. 140810170027 \*/

#include <iostream>

using namespace std;

// Fungsi untuk melakukan penggabungan kedua bagian menjadi data yang terurut.

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

{

// Menyortir rendah ke sedang dan menengah +1 ke tinggi.

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

i = low;

k = 0;

j = mid + 1;

// Gabungkan kedua bagian menjadi temp [].

while (i <= mid && j <= high)

{

if (a[i] < a[j])

{

temp[k] = a[i];

k++;

i++;

}

else

{

temp[k] = a[j];

k++;

j++;

}

}

// Masukkan semua nilai yang tersisa dari i hingga pertengahan ke temp [].

while (i <= mid)

{

temp[k] = a[i];

k++;

i++;

}

// Masukkan semua nilai yang tersisa dari j ke tinggi ke temp [].

while (j <= high)

{

temp[k] = a[j];

k++;

j++;

}

// Tetapkan data yang diurutkan disimpan dalam temp [] ke a [].

for (i = low; i <= high; i++)

{

a[i] = temp[i-low];

}

}

// Suatu fungsi untuk membagi array menjadi dua bagian.

void MergeSort(int \*a, int low, int high)

{

int mid;

if (low < high)

{

mid=(low+high)/2;

// Membagi data menjadi dua bagian

MergeSort(a, low, mid);

MergeSort(a, mid+1, high);

// Gabungkan mereka untuk mendapatkan hasil yang diurutkan.

Merge(a, low, high, mid);

}

}

int main()

{

int n, i;

n=20;

int arr[] = {23,52,51,5,6,2,64,1,8,9,3,45,67,23,80,13,35,88,7,21};

MergeSort(arr, 0, n-1);

// Mencetak data terurut.

cout<<"\nData Terurut: ";

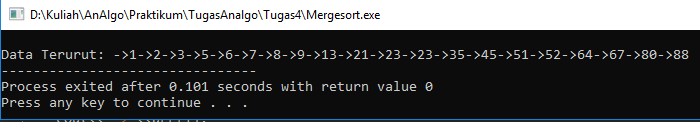
for (i = 0; i < n; i++)

cout<<"->"<<arr[i];

return 0;

}

Screenshot :



1. Kompleksitas waktu mergesort adalah O(n log n).

Cari tau kecepatan Komputer Anda! Ada pokoknya.. males nyari..

Hitung running time yg dibutuhkan apabila input nya 20!

T(26

**Studi Kasus 2 : SELECTION SORT**

* Pelajari cara kerja algoritma selection sort
* Tentukan T(n) dari rekurensi (pengulangan) selection sort berdasarkan penentuan rekurensi divide & conquer DAN Selesaikan persamaan rekurensi T(n) dengan **metode recursion-tree** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ

Menentukan T(n) :

Oleh karena itu :

*Karena*

* Lakukan implementasi koding program untuk algoritma selection sort dengan menggunakan bahasa C++

Source Code :

/\* Karimah A.T.M. 140810170027 \*/

#include <iostream>

using namespace std;

int data[10],data2[10];

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);

}

}

main()

{

cout<<"Masukkan Jumlah Data : ";

cin>>n;

for(int i=1;i<=n;i++)

{

cout<<"Masukkan data ke "<<i<<" : ";

cin>>data[i];

data2[i]=data[i];

}

selection\_sort();

cout<<"Data Setelah di Sort : ";

for(int i=1; i<=n; i++)

{

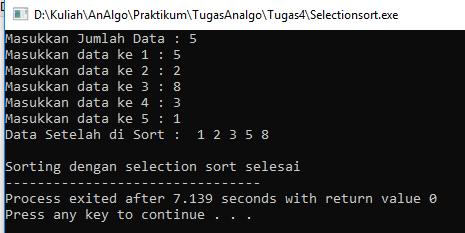
cout<<" "<<data[i];

}

cout<<"\n\nSorting dengan selection sort selesai";

}

Screenshot :



**Studi Kasus 3 : INSERTION SORT**

* Pelajari cara kerja algoritma selection sort
* Tentukan T(n) dari rekurensi (pengulangan) selection sort berdasarkan penentuan rekurensi divide & conquer DAN Selesaikan persamaan rekurensi T(n) dengan **metode recursion-tree** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ

Menentukan T(n) :

T(n) =

* Lakukan implementasi koding program untuk algoritma selection sort dengan menggunakan bahasa C++

Source Code :

/\* Karimah A.T.M. 140810170027 \*/

#include <iostream>

using namespace std;

int data[10],data2[10];

int n;

void tukar(int a, int b)

{

int t;

t = data[b];

data[b] = data[a];

data[a] = t;

}

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;

}

}

main()

{

cout<<"Masukkan Jumlah Data : ";

cin>>n;

for(int i=1;i<=n;i++)

{

cout<<"Masukkan data ke "<<i<<" : ";

cin>>data[i];

data2[i]=data[i];

}

insertion\_sort();

cout<<"Data Setelah di Sort : ";

for(int i=1; i<=n; i++)

{

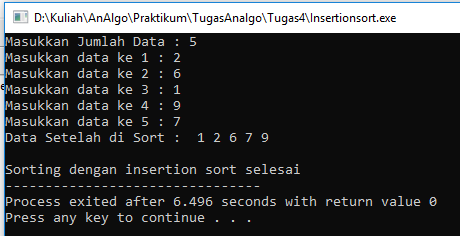
cout<<" "<<data[i];

}

cout<<"\n\nSorting dengan insertion sort selesai";

}

Screenshot :



**Studi Kasus 4 : BUBBLE SORT**

* Pelajari cara kerja algoritma selection sort
* Tentukan T(n) dari rekurensi (pengulangan) selection sort berdasarkan penentuan rekurensi divide & conquer DAN Selesaikan persamaan rekurensi T(n) dengan **metode recursion-tree** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ

Menentukan T(n) :

* Lakukan implementasi koding program untuk algoritma selection sort dengan menggunakan bahasa C++

Source Code :

/\* Karimah A.T.M. 140810170027 \*/

#include<iostream>

using namespace std;

main()

{

int n, i, arr[50], j, temp;

cout<<"Masukkan total elemen yang akan diurutkan: ";

cin>>n;

cout<<"Masukan "<<n<<" angka:\n";

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

cout<<"Masukan angka ke-"<<i+1<<": ";

cin>>arr[i];

}

for(i=0; i<(n-1); i++) {

for(j=0; j<(n-i-1); j++)

{

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

{

temp=arr[j];

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

arr[j+1]=temp;

}

}

}

cout<<"Data terurut dari hasil Bubble Sort::\n";

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

cout<<arr[i]<<" ";

}

}

Screenshot :

