Tugas Praktikum Analisis Algoritma

ALGORITMA DIVIDE & CONQUER



Dibuat Oleh :

Adryan Luthfi Faiz 140810160049

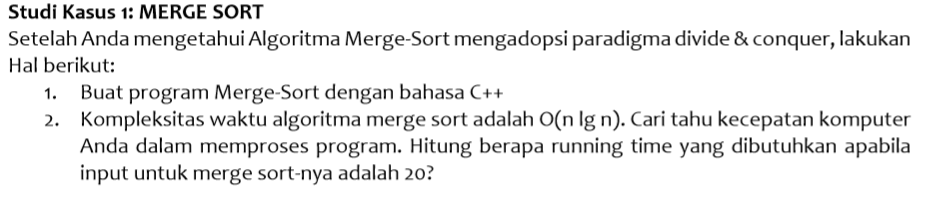
**UNIVERSITAS PADJADJARAN**

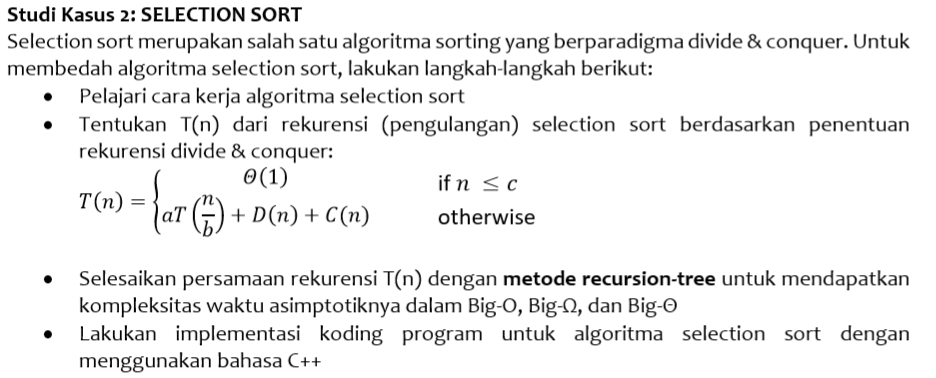
**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

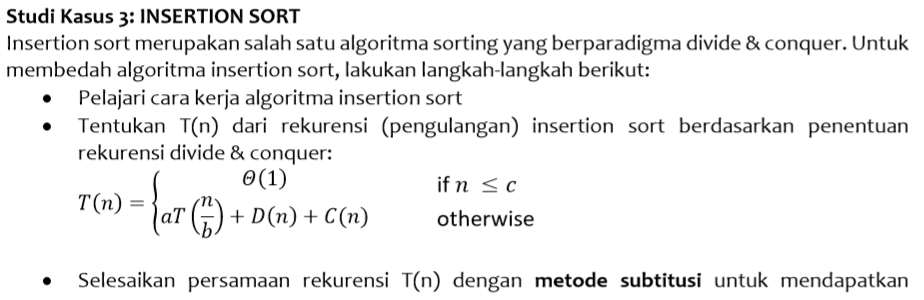
**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

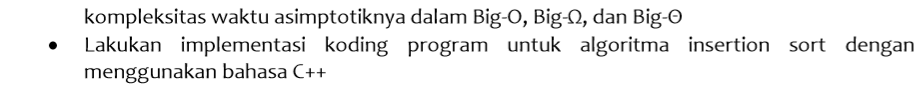
**2019**

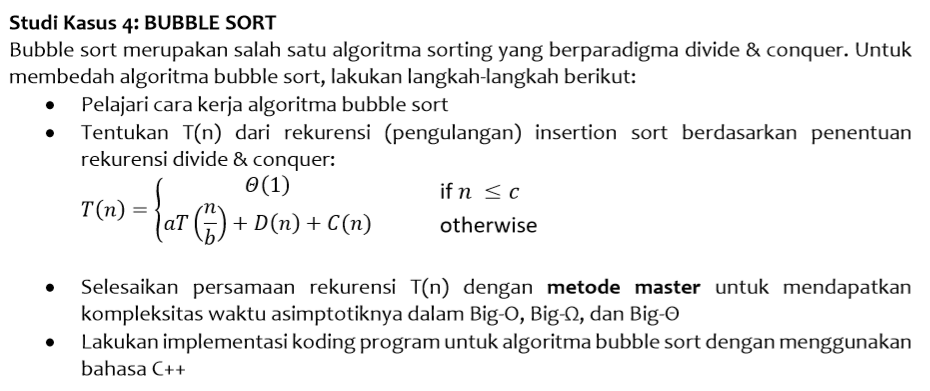
STUDI KASUS











PROGRAM C++

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Semua Sorting (Merge , Select , Insert , Bubble)

Disatuin jadi 1 program "MergeSort.cpp"

Inputnya random sekitar 1 - 100 jadi gak perlu diinput satu-satu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Adryan Luthfi Faiz

140810160049

IYON AdryanLF

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <chrono>

#include <ctime> // For time()

#include <cstdlib> // For srand() and rand()

#include <stdlib.h> // for system cls clear screen

using namespace std;

using namespace std::chrono;

// --- Merge Sorting --- //

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);

}

}

// --- Selection Sorting --- //

void selectionSort (int arr[], int n){

int i, j;

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

for (j = i+1; j < n; ++j){

if (arr[i] > arr[j]){

arr[i] = arr[i]+arr[j];

arr[j] = arr[i]-arr[j];

arr[i] = arr[i]-arr[j];

}

}

}

}

// --- Insertion Sorting --- //

struct list {

int data;

list \*next;

};

list\* InsertinList(list \*head, int n){

list \*newnode = new list;

list \*temp = new list;

newnode->data = n;

newnode->next = NULL;

if(head == NULL){

head = newnode;

return head;

}

else {

temp = head;

if(newnode->data < head->data){

newnode->next = head;

head = newnode;

return head;

}

while(temp->next != NULL){

if(newnode->data < (temp->next)->data)

break;

temp=temp->next;

}

newnode->next = temp->next;

temp->next = newnode;

return head;

}

}

// --- Bubble Sorting --- //

void bubbleSort (int arr[], int n){

int i, j;

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

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

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

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

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

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

}

}

}

}

void listSort (){

cout<<" --- List Sorting --- "<<endl<<endl;

cout<<"1. Merge Sorting"<<endl;

cout<<"2. Selection Sorting"<<endl;

cout<<"3. Insertion Sorting"<<endl;

cout<<"4. Bubble Sorting"<<endl<<endl;

cout<<"Pilih Sorting yang ingin digunakan : ";

}

int main(){

int n, i , num , pilih ;

int temprand;

char coba;

list \*head = new list;

head = NULL;

srand(time(0)); // Initialize random number generator.

start:

pilih = 0;

listSort();

while(pilih < 1 || pilih > 4){

cin>>pilih;

if (pilih < 1 || pilih > 4) cout<<"ERROR , Masukkan angka diantara 1 - 4 : ";

}

cout<<endl<<"Masukkan jumlah elemen data yang ingin diurutkan: ";

cin>>n;

int arr[n];

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

switch (pilih){

case 1 :

cout <<"Berikut elemen random :";

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

temprand = (rand() % 100) + 1;

arr[i] = temprand;

cout<<temprand<<" ";

}

mergeSort(arr, 0, n-1);

break;

case 2 :

cout <<"Berikut elemen random :";

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

temprand = (rand() % 100) + 1;

arr[i] = temprand;

cout<<temprand<<" ";

}

selectionSort(arr, n);

break;

case 3 :

cout <<"Berikut elemen random :";

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

num = (rand() % 100) + 1;

head = InsertinList(head, num);

cout<<num<<" ";

}

cout<<endl<<endl;

cout<<"\nArray yang telah diurutkan: ";

while(head != NULL){

cout<<" "<<head->data;

head = head->next;

}

break;

case 4 :

cout <<"Berikut elemen random :";

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

temprand = (rand() % 100) + 1;

arr[i] = temprand;

cout<<temprand<<" ";

}

bubbleSort(arr, n);

break;

}

cout<<endl<<endl;

if (pilih !=3){ // Selain insertion karena insertion make list

cout<<"\nArray yang telah diurutkan: ";

for (i = 0; i < n; i++) cout<<" "<<arr[i];

}

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

auto duration = duration\_cast<microseconds>( t2 - t1 ).count();

cout<<endl<<duration<<" microseconds"<<endl;

cout<<endl<<endl<<"Ingin mencoba lagi (Y/N) ? ";

cin>>coba;

if (coba == 'y' || coba == 'Y') {

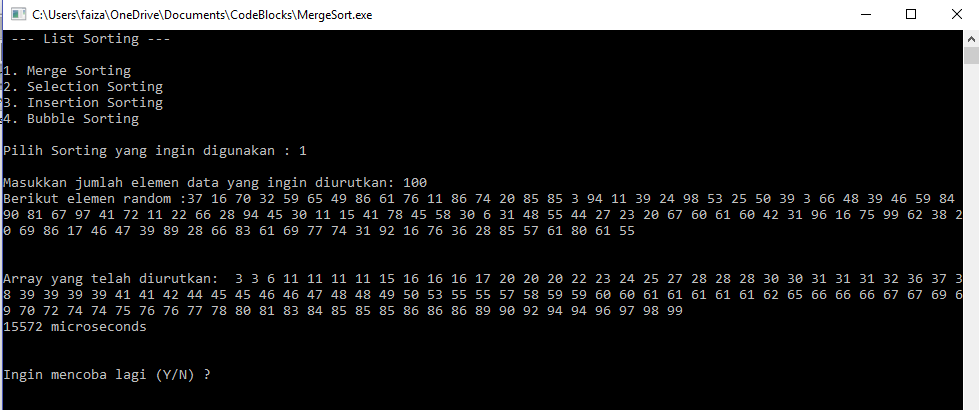
system("CLS");

goto start;

}

}

MERGE SORT

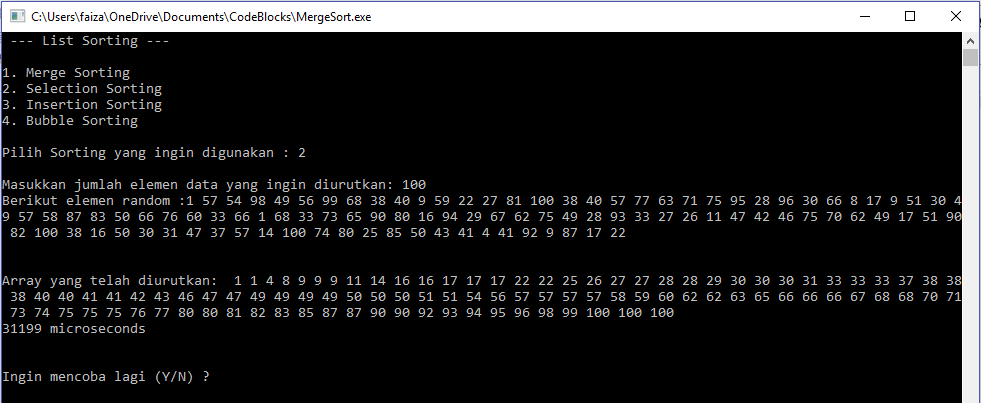


Kompleksitas waktu:

Durasi waktu yang dibutuhkan untuk 100 input: 15572 micro s = 0.015572 s

Big-O = Big-Ω = Big-θ = n \* log n

SELECTION SORT

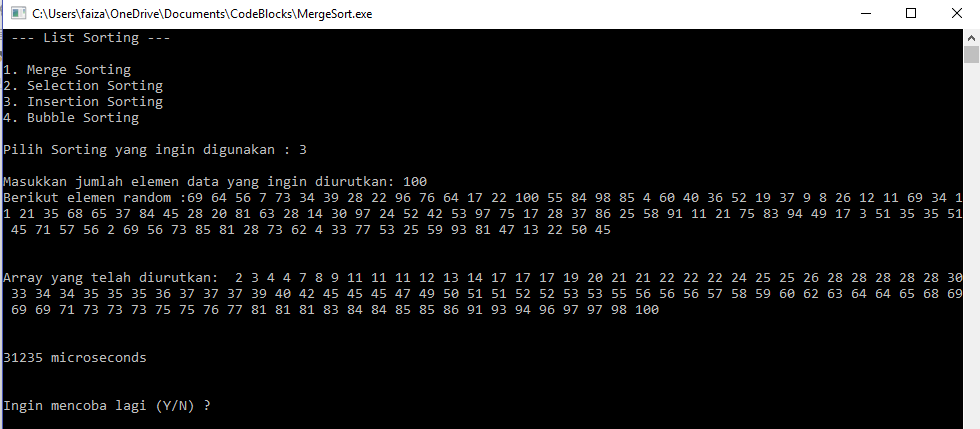


Kompleksitas waktu:

Durasi waktu yang dibutuhkan untuk 100 input: 31199 microseconds = 0.031199 s

Big-O = Big-Ω = Big-θ = n2

INSERTION SORT



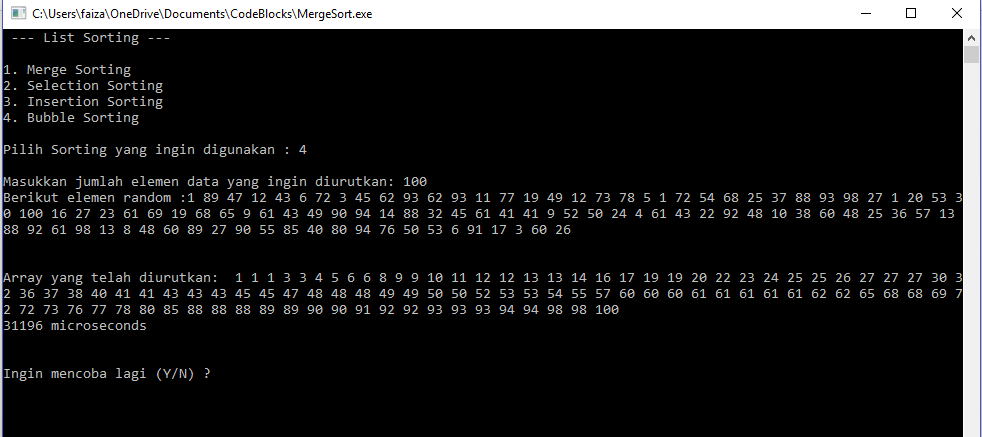
Kompleksitas waktu:

Durasi waktu yang dibutuhkan untuk 100 input: 31235 microseconds = 0.031235 s

Big-O = n

Big-Ω = Big-θ = n2

BUBBLE SORT



Kompleksitas waktu:

Durasi waktu yang dibutuhkan untuk 100 input: 31196 microseconds = 0.031196 s

Big-O = n

Big-Ω = Big-θ = n2