**TUGAS 4**

**MODUL PRAKTIKUM 4**

**MATA KULIAH ANALISIS ALGORITMA**

****

**DISUSUN OLEH**

**TIVANI SHAKILLA ERVI**

**140810170014**

**KELAS B**

**PROGRAM STUDI S-1 TETKNIK INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS PADJADJARAN**

**JATINANGOR**

**2019**

**Pendahuluan**

**PARADIGMA DIVIDE & CONQUER**

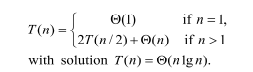
Divide & Conquer merupakan teknik algoritmik dengan cara memecah input menjadi beberapa bagian, memecahkan masalah di setiap bagian secara rekursif, dan kemudian menggabungkan solusi untuk subproblem ini menjadi solusi keseluruhan. Menganalisis running time dari algoritma divide & conquer umumnya melibatkan penyelesaian rekurensi yang membatasi running time secara rekursif pada instance yang lebih kecil

**PENGENALAN REKURENSI**

• Rekurensi adalah persamaan atau ketidaksetaraan yang menggambarkan fungsi terkait nilainya pada input yang lebih kecil. Ini adalah fungsi yang diekspresikan secara rekursif

• Ketika suatu algoritma berisi panggilan rekursif untuk dirinya sendiri, running time-nya sering dapat dijelaskan dengan perulangan

• Sebagai contoh, running time worst case T(*n*) dari algoritma merge-sort dapat dideskripsikan dengan perulangan :



**BEDAH ALGORITMA MERGE-SORT**

 Merupakan algoritma sorting dengan paradigma divide & conquer

 Running time worst case-nya mempunyai laju pertumbuhan yang lebih rendah dibandingkan insertion sort

 Karena kita berhadapan dengan banyak subproblem, kita notasikan setiap subproblem sebagai sorting sebuah subarray A[p..r]

 Inisialisasi, p=1 dan r=n, tetapi nilai ini berubah selama kita melakukan perulangan subproblem

Untuk mengurutkan A[p..r]:

• **Divide** dengan membagi input menjadi 2 subarray A[p..q] dan A[q+1 .. r]

• **Conquer** dengan secara rekursif mengurutkan subarray A[p..q] dan A[q+1 .. r]

• **Combine** dengan menggabungkan 2 subarray terurut A[p..q] dan A[q+1 .. r] untuk

menghasilkan 1 subarray terurut A[p..r]

• Untuk menyelesaikan langkah ini, kita membuat prosedur MERGE(A, p, q, r)

• Rekursi berhenti apabila subarray hanya memiliki 1 elemen (secara trivial terurut)

**Studi Kasus**

**Studi Kasus 1: MERGE SORT**

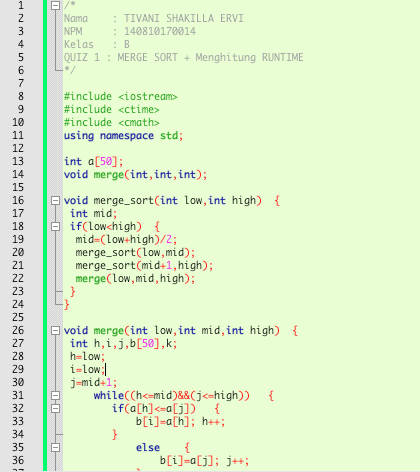
Setelah Anda mengetahui Algoritma Merge-Sort mengadopsi paradigma divide & conquer, lakukan hal berikut:

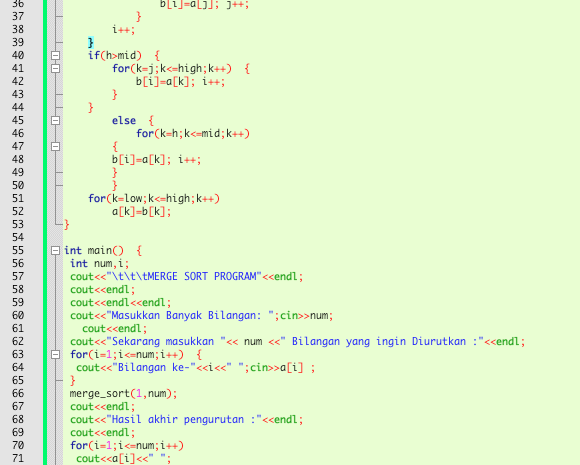
1. Buat program Merge-Sort dengan bahasa C++

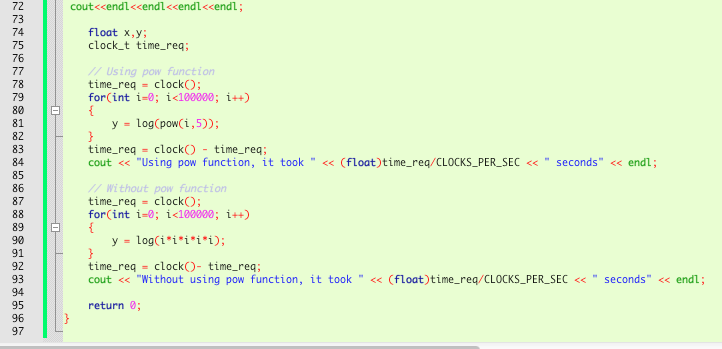
2. Kompleksitas waktu algoritma merge sort adalah O(n lg n). Cari tahu kecepatan komputer Anda dalam memproses program. Hitung berapa running time yang dibutuhkan apabila input untuk merge sort-nya adalah 20?

**Jawab :**

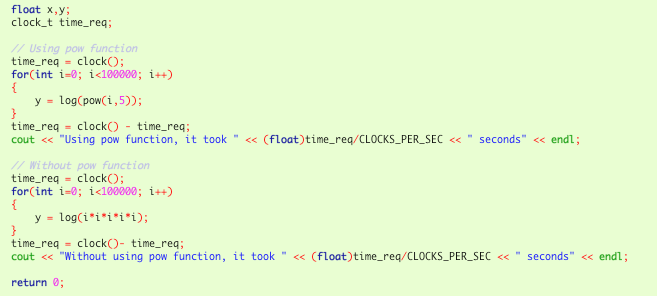
1. **Program Merge Sort**

****

****

****

**2. Kecepatan komputer dan running time saat input data sebanyak 20**

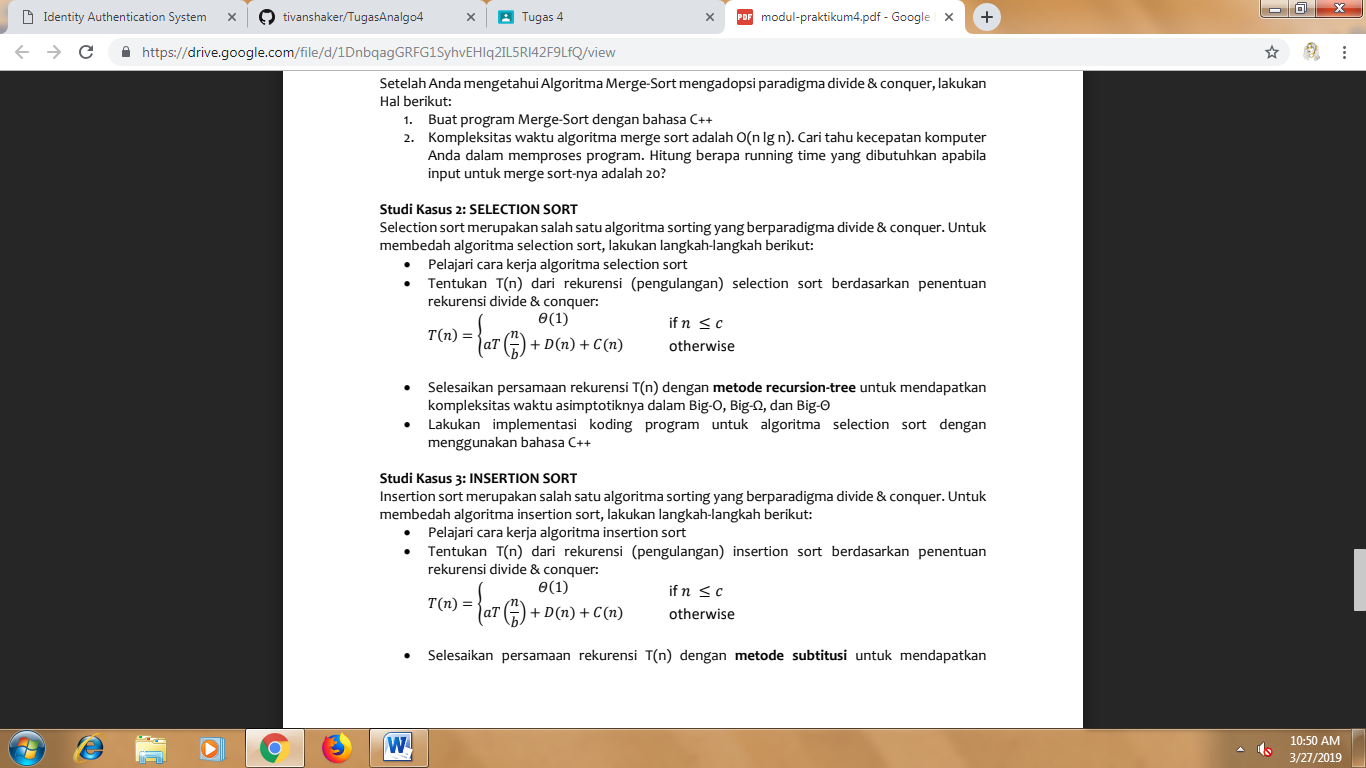


**Studi Kasus 2: SELECTION SORT**

Selection sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untukmmembedah 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:



* 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

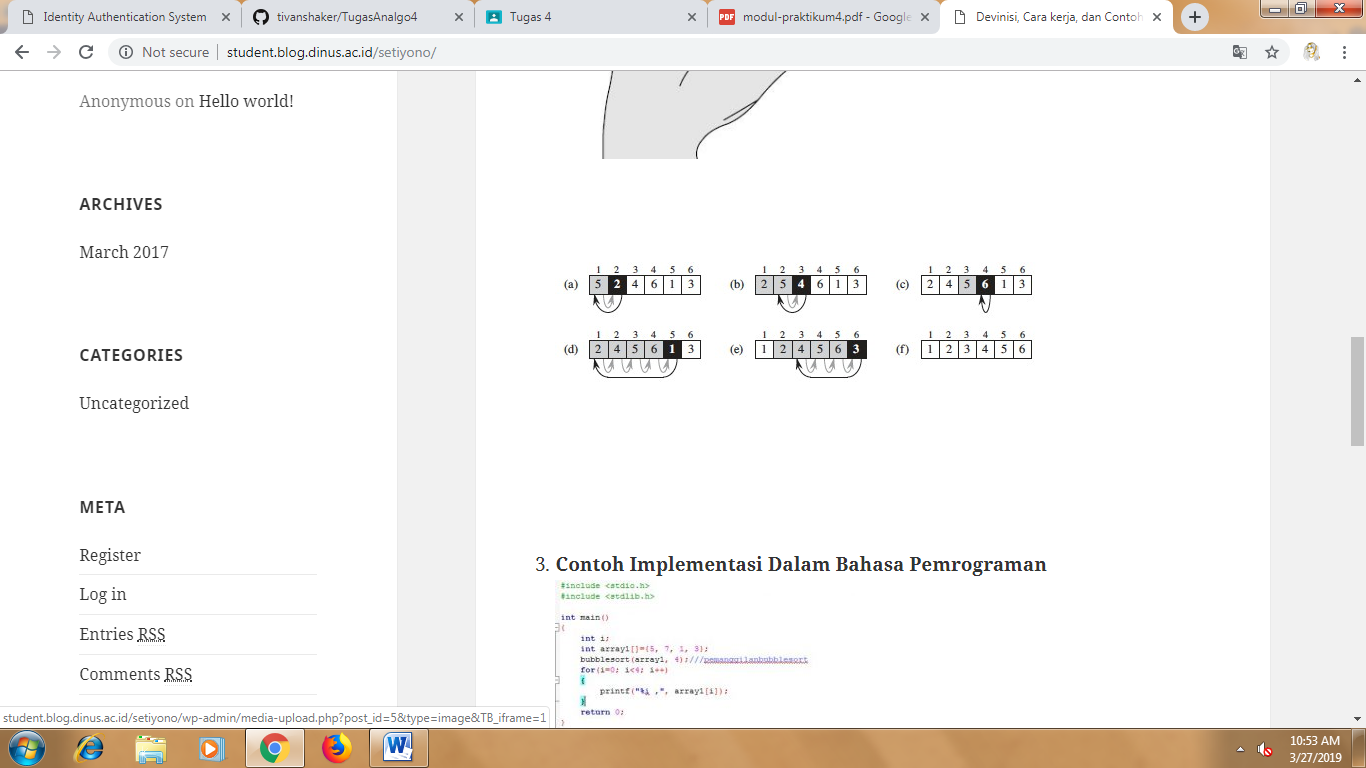
menggunakan bahasa C++

**Jawab :**

* Cara kerja insertion sort :

Contoh ilustrasi cara kerja selection sort adalah sebagai berikut:

* Pertama jika menggunakan cara Ascending maka, program akan mencari index dengan nilai paling kecil dari semua data yang acak lalu ditukar dengan index paling depan.
* Setelah itu program mencari lagi index dengan nilai kecil kedua, disimpan pada variabel temporary lalu ditukar dengan index kedua.
* Begitu seterusnya sampai semua data terurut.



* Menentukan T(n):

Oleh Karena itu:

Karena ,Maka

* Program :

/\*

Nama : Tivani Shakilla Ervi

NPM : 140810170014

Kelas : B

\*/

#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 << "\n=====================================";

cout<<"\nMasukkan 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];

}

selection\_sort();

cout << "\n-------------------------------------" << endl;

cout<<"Data Setelah di Sort : ";

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

{

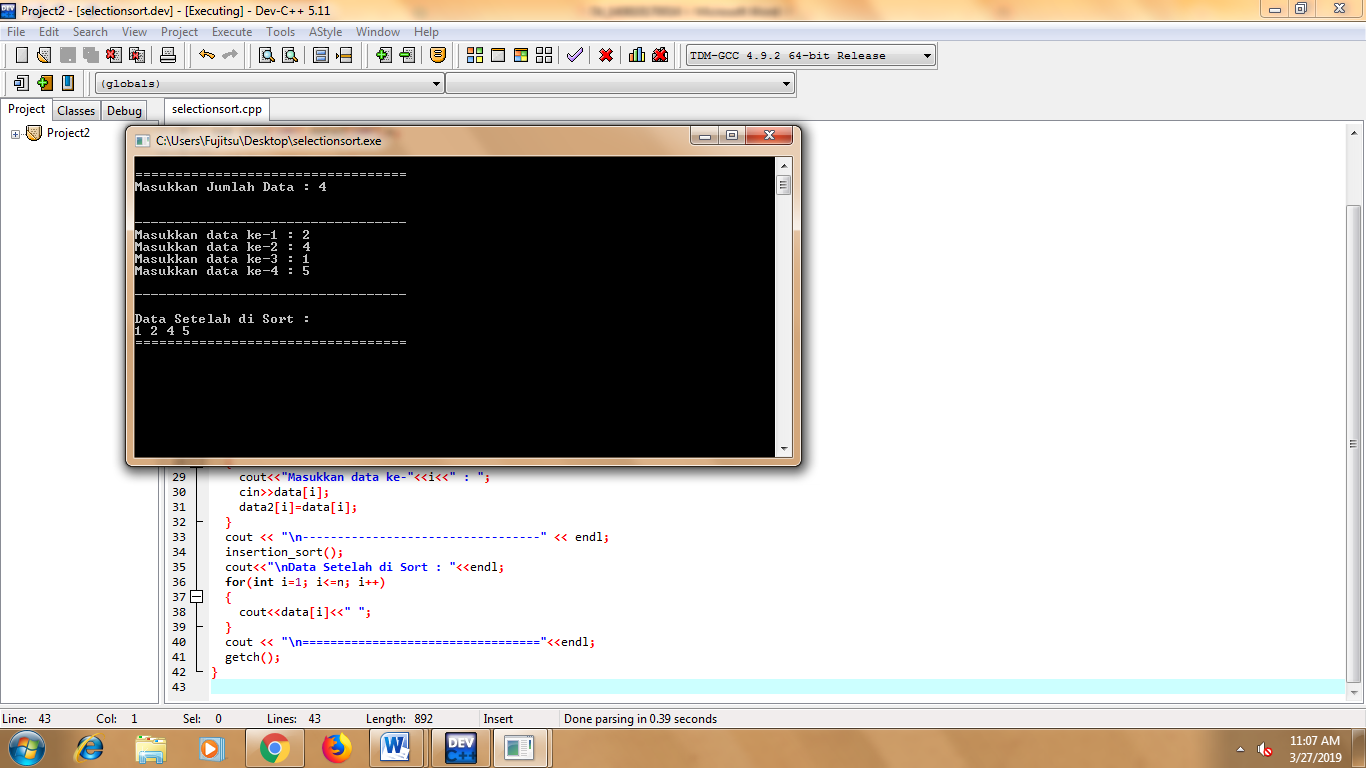
cout<<" "<<data[i];

}

cout << "\n=====================================\n";

getch();

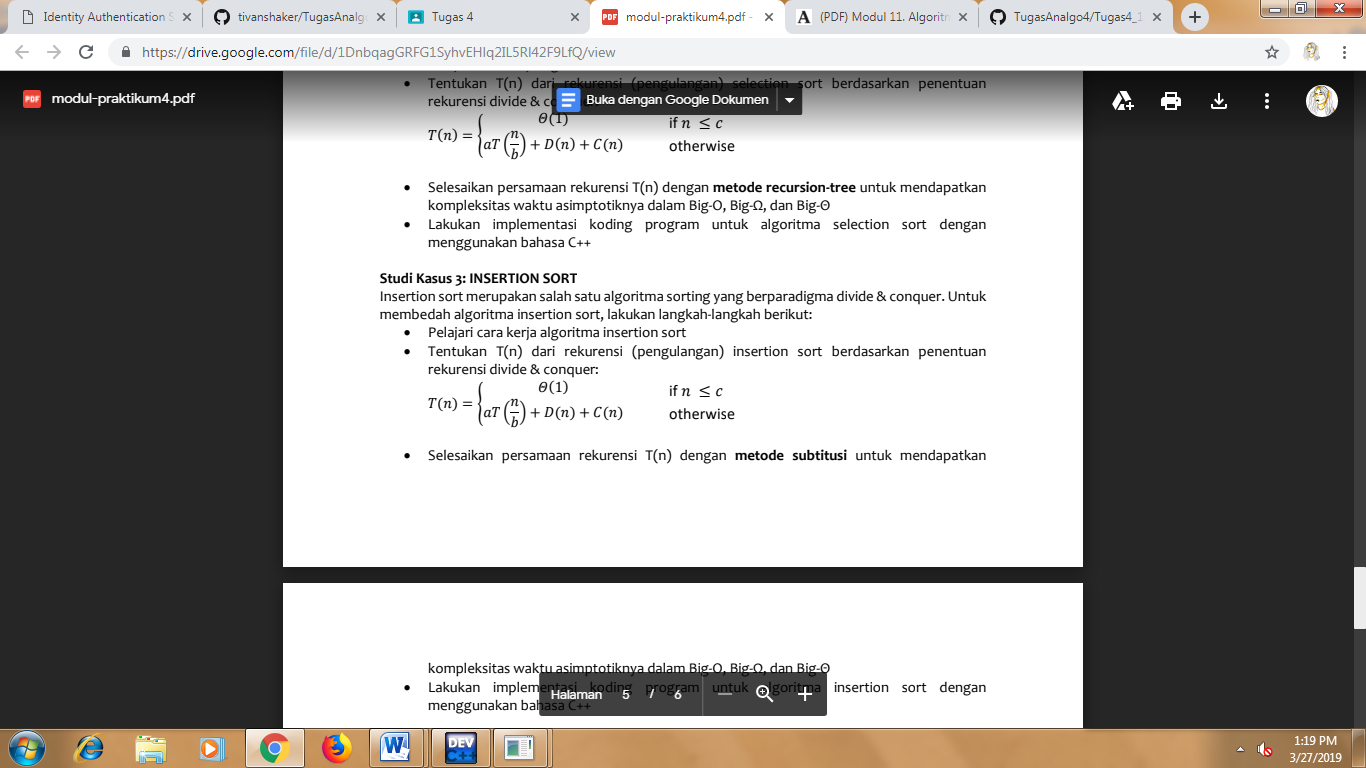
}



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:



* Selesaikan persamaan rekurensi T(n) dengan metode subtitusi untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ

**Jawab :**

* Program C++

#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 << "\n=================================="<<endl;

cout<<"Masukkan Jumlah Data : "; cin>>n;

cout<<endl;

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;

insertion\_sort();

cout<<"\nData Setelah di Sort : "<<endl;

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

{

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

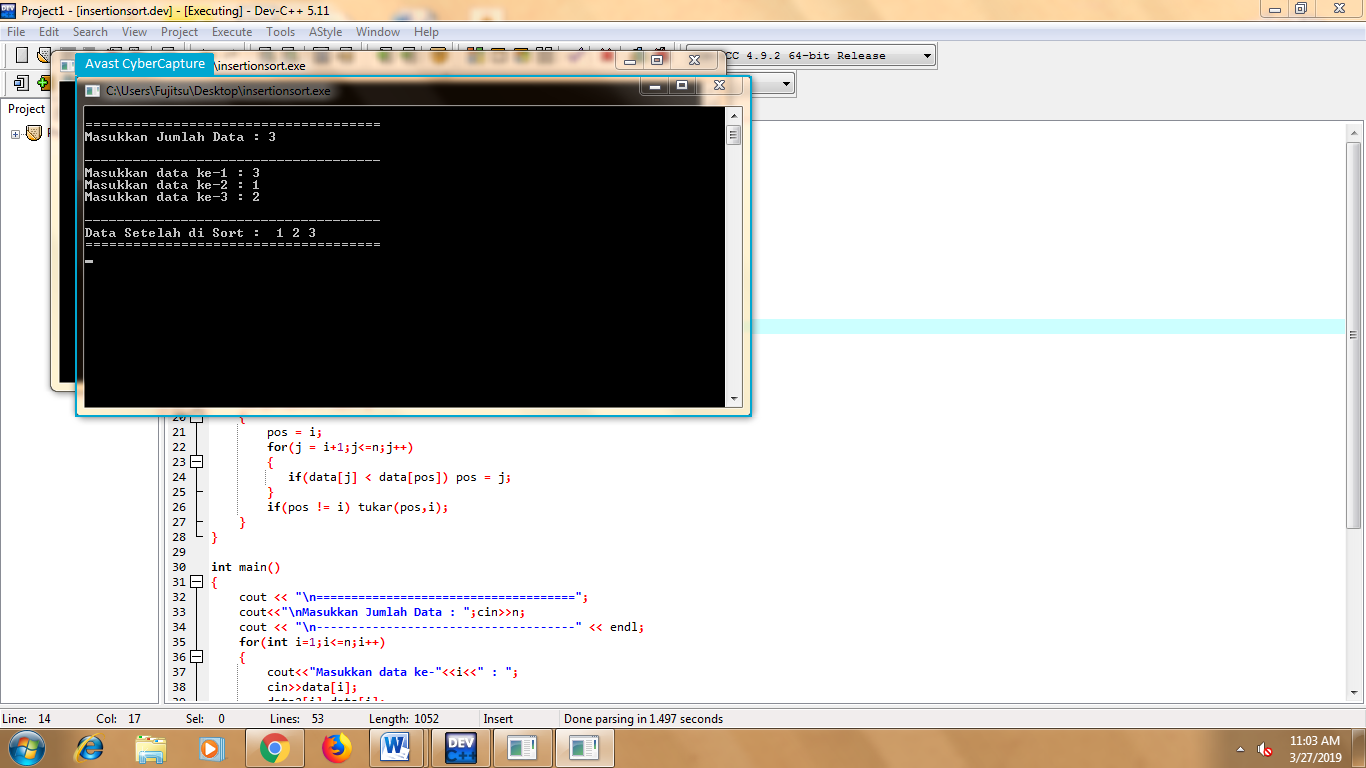
}

cout << "\n=================================="<<endl;

getch();

}

* Screeshoot



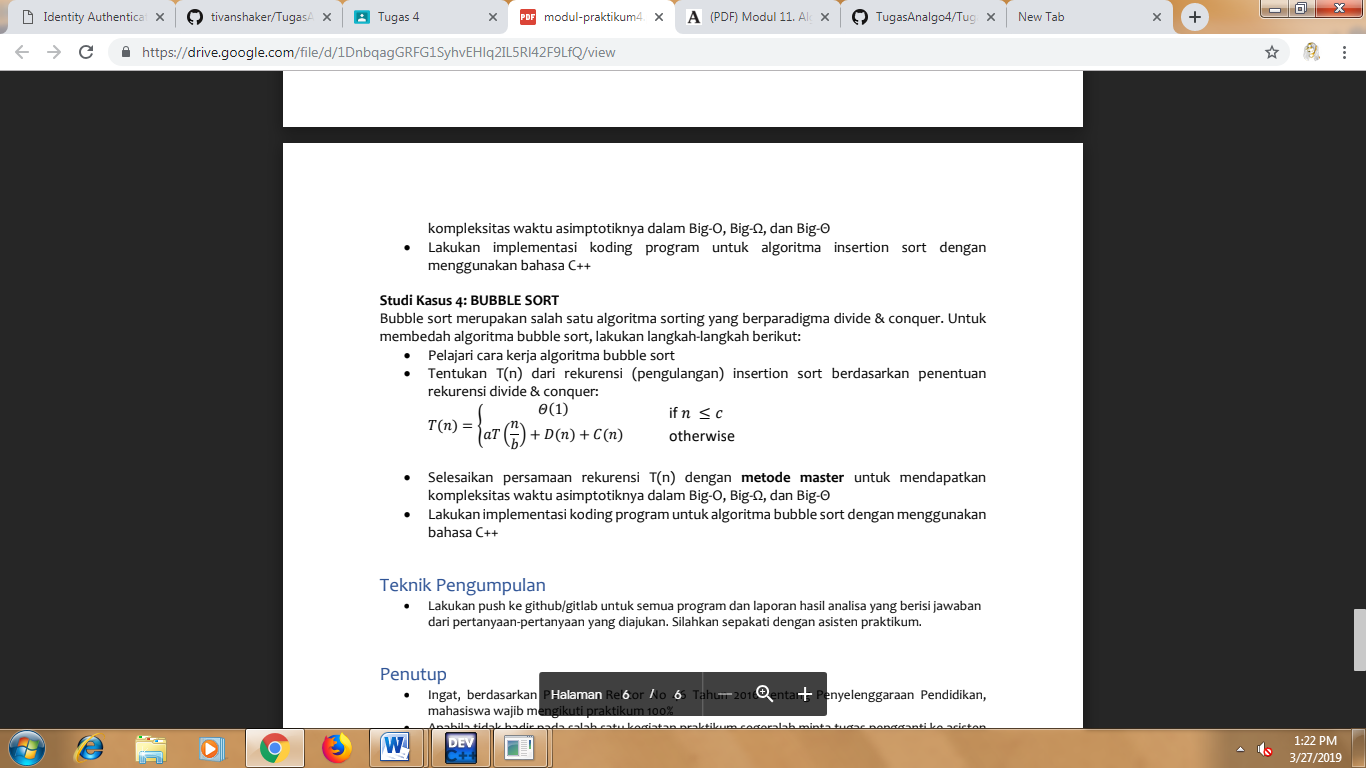
* Menentukan T(n):

T(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:



* 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++

**Jawab :**

* Cara kerja Bubble Sort

Algoritma Bubble Sort ini merupakan proses pengurutan yang secara berangsur-angsur berpindah ke posisi yang tepat karena itulah dinamakan Bubble yang artinya gelembung. Algoritma ini akan mengurutkan data dari yang terbesar ke yang terkecil (ascending) atau sebaliknya (descending).

Secara sederhana, bisa didefenisikan algoritma Bubble Sort adalah pengurutan dengan cara pertukaran data dengan data disebelahnya secara terus menerus sampai dalam satu iterasi tertentu tidak ada lagi perubahan.

* Menentukan T(n)

T(n)=n^2+n=O(n^2)

T(n)=n-1=Ω(n)

* Program Bubble Sort

#include <iostream>

#include <conio.h>

using namespace std;

int main(){

int arr[100],n,temp;

cout << "\n================================================"<<endl;

cout<<"Massukan banyak elemen yang akan diinputkan : ";cin>>n;

cout << "\n------------------------------------------------" << endl;

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

cout<<"Masukkan Elemen 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 << "------------------------------------------------" << endl;

cout<<"\nHasil dari Bubble Sort : "<<endl;

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

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

}

}

