TUGAS 2 KELAS

RUNNING TIME

Disusun sebagai salah satu tugas

mata kuliah Analisis Algoritma



Tasya Amanda Adinegara – 140810160003

Shofiyyah Nadhiroh – 14018160057

Patricia Joanne – 140810160065

Dikumpulkan tanggal

25 Maret 2019

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

2019

**Soal**

1. Program Linear Search
2. Program Binary Search
3. Program Quick Sort
4. Analisis running time dan kompleksitasnya!

**Jawaban**

1. Pencarian linear

Algoritma:

Deklarasikan isi array, jumlah isi array, dan angka yang ingin dicari.

Telusuri isi array dengan mengecek satu persatu urut arraynya hingga urut terakhir untuk menemukan angka yang ingin dicari.

Jika angka yang ingin dicari = isi array tersebut, maka jawabnya ada.

Program: <http://cpp.sh/2xzk3>

#include <iostream>

#include <chrono>

using namespace std;

using namespace std::chrono;

int main(){

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

int arr[] = { 21, 12, 34, 56, 67, 80, 3, 100, 96, 49 };

int size = sizeof(arr) / sizeof(arr[0]); //array 10 angka

int n = 50; //masukkan input manual

cout<<"Melakukan pencarian linear"<<endl;

cout<<"Apakah angka "<<n<<" ada?"<<endl;

bool ketemu = false;

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

if (n == arr[i])

ketemu = true;

}

cout<<"Jawab: "<<(ketemu == true ? "Ada!":"Tidak ada!")<<endl;

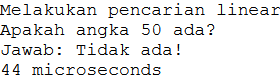
high\_resolution\_clock::time\_point t2 = high\_resolution\_clock ::now();

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

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

}

Output:



1. Pencarian biner

Algoritma:

Deklarasikan isi array, jumlah isi array, dan angka yang ingin dicari.

Telusuri isi array dengan mengecek dari urut tengah ke urut pertama dan urut terakhir hingga ditemukan angka yang ingin dicari.

Jika angka yang ingin dicari = isi array tersebut, maka jawabnya ada.

Program: <http://cpp.sh/9qbc>

#include <iostream>

#include <chrono>

using namespace std;

using namespace std::chrono;

int main(){

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

int arr[] = { 21, 12, 34, 56, 67, 80, 3, 100, 96, 49 };

int size = sizeof(arr) / sizeof(arr[0]); //array 10 angka

int n = 50; //masukkan input manual

cout<<"Melakukan pencarian biner"<<endl;

cout<<"Apakah angka "<<n<<" ada?"<<endl;

bool ketemu = false;

int left = 0, right = size-1;

while (left <= right){

int mid = left + (right - left) / 2;

if (arr[mid] == n)

ketemu = true;

if (arr[mid] < n)

left = mid + 1;

else

right = mid - 1;

}

cout<<"Jawab: "<<(ketemu == true ? "Ada!":"Tidak ada!")<<endl;

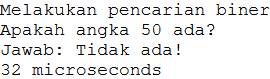
high\_resolution\_clock::time\_point t2 = high\_resolution\_clock ::now();

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

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

}

Output:



1. Quick Sort

Seperti merge sort, quick sort adalah algoritma Divide and Conquer yang mengambil elemen sebagai pivot dan mempartisi array yang diberikan di sekitar pivot yang dipilih. Ada banyak versi quick sort yang memilih pivot dengan berbagai cara.

* Selalu pilih elemen pertama sebagai pivot.
* Selalu pilih elemen terakhir sebagai pivot (diterapkan di bawah)
* Pilih elemen acak sebagai pivot.
* Pilih median sebagai poros.

Proses utama dalam quick sort adalah partisi. Target dari partisi adalah, diberikan array dan elemen x array sebagai pivot, letakkan x pada posisi yang benar dalam array yang diurutkan dan letakkan semua elemen yang lebih kecil (lebih kecil dari x) sebelum x, dan letakkan semua elemen yang lebih besar (lebih besar dari x) setelah x. Semua ini harus dilakukan dalam waktu linier.

Algoritma:

/ \* rendah -> Indeks awal, tinggi -> Indeks akhir \* /

quickSort (arr [], rendah, tinggi){

if (rendah <tinggi){

pi = partisi (arr, rendah, tinggi);

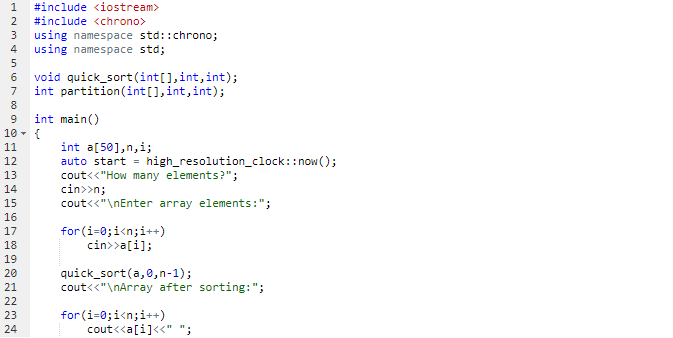
quickSort (arr, low, pi - 1); // Sebelum pi

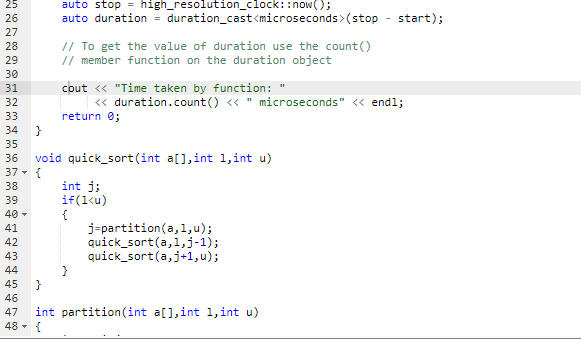
quickSort (arr, pi + 1, tinggi); // Setelah pi

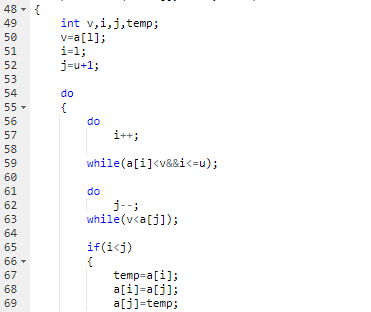
}

}

Program:

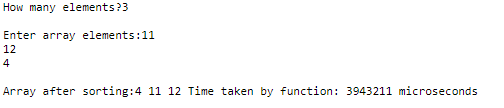








Output:



**Analisis running time**

Running time dalam program-program di atas didapatkan dari library chrono pada C++. Running time selalu berubah-ubah bahkan untuk program dengan code dan data yang sama sekalipun. Running time dipengaruhi oleh kerja CPU dan kita tidak pernah tahu apa saja proses yang sedang dikerjakan oleh CPU kita. Untuk spesifikasi dari PC yang digunakan untuk menjalankan program di atas adalah sebagai berikut:

