**TUGAS 1**

**ANALISIS ALGORITMA**

****

Disusun Oleh:

Satrio Sadrakh Allesandro

140810160038

**PRODI S-1 TEKNIK INFORMATIKA**

**DEPARTEMEN ILMU KOMPUTER**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS PADJADJARAN**

**2018**

**Spesifikasi Komputer:**

Model & Manufacturer : LENOVO G50-45 (80E3)

OS : Windows 10 Enterprise 64-bit (10.0, Build 17134)

Processor : AMD A8-6410 APU with AMD Radeon R5 Graphics (4CPUs), 2.0 GHz

Memory : 4096 MB RAM

Compiler : [https://www.onlinegdb.com/online\_c++\_compiler#](https://www.onlinegdb.com/online_c++_compiler)

1. **Algoritma Pencarian Linier**

Source Code:

#include <iostream>

#include <math.h>

using namespace std;

int search(int arr[], int n, int x)

{

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

if (arr[i] == x){

return i;

}

}

return -1;

}

int main()

{

int n, x;

cout<<"Masukkan jumlah bilangan : ";

cin>>n;

cout<<"Masukkan angka yang dicari : ";

cin>>x;

int arr[n];

cout<<"Elemen array: "<<endl;

for(int x=1; x<=n; x++){

arr[x]=x;

cout<<arr[x]<<" ";

}

cout<<endl<<endl;

clock\_t t1,t2;

t1=clock();

cout<<"Jumlah bilangan: "<<n<<endl;

cout<<"Angka yang dicari: "<<x<<endl;

int result = search(arr, n, x);

if (result == -1)

cout<<"Elemen Tidak Ditemukan";

else

cout<<"Elemen berada di urutan ke: " <<result;

t2=clock();

float diff ((float)t2-(float)t1);

cout<<endl<<"Waktu = "<<diff<<"ms"<<endl;

system ("pause");

return 0;

}

Analisis Kompleksitas:

int i = 0 1 kali

i<=n n + 1 kali

i = i + 1 n kali

i + 1 n kali

return i 1 kali

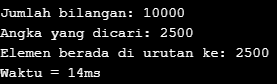
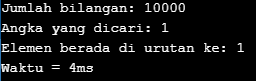
return -1 1 kali

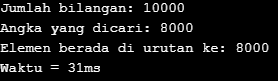
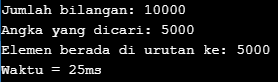
Kompleksitas waktu algoritma adalah:

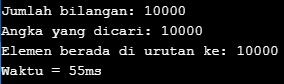
T(n) = 1 + (n + 1) + n + n + 1 + 1 = 3n + 4

Atau T(n) = A n + B 🡪 T(n) = O(n)

Hasil:







1. **Algoritma Pencarian Biner**

Source Code:

#include <iostream>

#include <math.h>

using namespace std;

int binarySearch(int arr[], int l, int n, int x)

{

if (n >= l) {

int mid = l + (n - l) / 2;

if (arr[mid] == x)

return mid;

if (arr[mid] > x)

return binarySearch(arr, l, mid - 1, x);

return binarySearch(arr, mid + 1, n, x);

}

return -1;

}

int main(void)

{

int n, x;

cout<<"Masukkan jumlah bilangan : ";

cin>>n;

cout<<"Masukkan angka yang dicari : ";

cin>>x;

int arr[n];

cout<<"Elemen array: "<<endl;

for(int x=1; x<=n; x++){

arr[x]=x;

cout<<arr[x]<<" ";

}

cout<<endl<<endl;

clock\_t t1,t2;

t1=clock();

cout<<"Jumlah bilangan: "<<n<<endl;

cout<<"Angka yang dicari: "<<x<<endl;

int result = binarySearch(arr, 0, n, x);

if (result == -1)

cout<<"Elemen Tidak Ditemukan";

else

cout<<"Elemen berada di urutan ke: " <<result;

t2=clock();

float diff ((float)t2-(float)t1);

cout<<endl<<"Waktu = "<<diff<<"ms"<<endl;

system ("pause");

return 0;

}

Analisis Kompleksitas:

int mid = l + (n - l) / 2 1 kali

l + (n - l) / 2 1 + 1 + 1 kali

arr[mid] == x 1 kali

return mid 1 kali

arr[mid] > x 1 kali

return binarySearch(arr, l, mid - 1, x) log2n kali

return binarySearch(arr, mid + 1, n, x) log2n kali

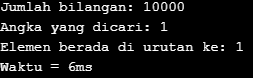
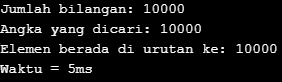
return -1 1 kali

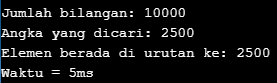
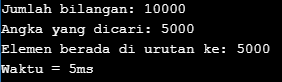
Kompleksitas waktu algoritma adalah:

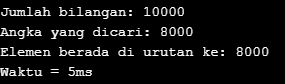
T(n) = 1 + (1 + 1 + 1) + 1 + 1 + 1 + log2n + log2n + 1 = log22n + 8

Atau T(n) = A log n + B 🡪 T(n) = O(log n)

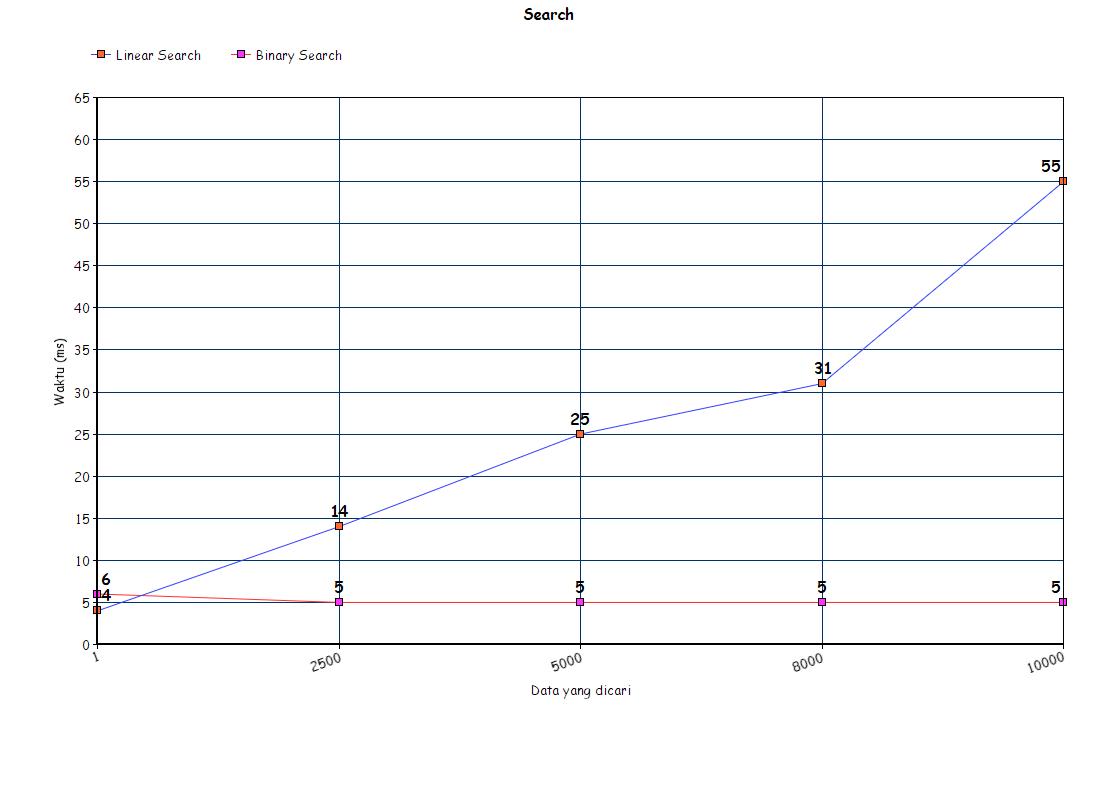
Hasil:





**Kesimpulan:**

****

Saat membandingkan *running time* antara algoritma *linear search* dengan *binary search*, terlihat bahwa algoritma *linear search* akan memiliki *running time* yang lebih cepat dibanding dengan *binary search* jika dan hanya jika data yang dicari berada di awal *list* (di gambar diwakilkan dengan data 1). Selain itu, *binary search* memiliki *running time* yang lebih cepat.