**Tugas 2**

**Analisis Algoritma**



Disusun oleh :

Afifah Kho’eriah (140810160008)

Baby Cattleya Gustina Permatagama (140810160048)

Muhammad Islam Taufikurahman (140810160062)

S-1 Teknik Informatika   
Fakultas Matematika & Ilmu Pengetahuan Alam   
Universitas Padjadjaran  
Jalan Raya Bandung - Sumedang Km. 21 Jatinangor 45363

1. **Program *Searching Linear***

/\*

Nama Program : Program Linear Search

Oleh : Baby, Afifah, Islam

Dibuat : Senin, 25 Maret 2018

\*/

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <chrono>

using namespace std;

int linearSearch(const int [], int, int);

int SIZE;

int main() {

cout << "Masukkan Banyak Elemen : " ;

cin >> SIZE;

int array[SIZE];

unsigned seed = time(0);

srand(seed);

cout << "Array Angka Random: " <<endl;

for(int i = 0; i<100;i++)

{

array[i]=rand()%100+1;

cout << array[i]<<" ";

}

int results;

int input;

cout << endl << endl <<"Masukkan Angka yang Dicari: ";

cin >> input;

auto start = chrono::steady\_clock::now();

results = linearSearch(array, SIZE, input);

if (results == -1){

cout << "Angka Tidak Ditemukan di Array\n";

}

else {

cout << "Angka Ditemukan di indeks ke " << results;

cout << " pada array.\n" <<endl;

}

auto end = chrono::steady\_clock::now();

auto diff = end - start;

cout << "Running Time: "<<chrono::duration <double, milli> (diff).count() << " ms" << endl;

return 0;

}

int linearSearch(const int array[], int size, int value){

int lokasi = 0;

bool found = false;

while (!found && lokasi < size) {

if (array[lokasi] == value){

found = true;

}

else {

lokasi=lokasi+1;

}

}

return lokasi;

}

1. **Buat Program *Binary Search***

/\*

Nama Program : Program Binary Seacrh

Oleh : Baby, Afifah, Islam

Dibuat : 6 April 2018

\*/

#include <iostream>

#include <chrono>

using namespace std;

main () {

int n, i, search, first, last, middle;

cout << "Masukkan Jumlah Elemen: ";

cin >> n;

int arr[n];

unsigned seed = time(0);

srand(seed);

cout << "Array Angka Random: " <<endl;

for(int i = 0; i<n;i++) //to get random numbers inside the array.

{

arr[i]=rand()%1000+1;

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

}

cout<<endl<<"Masukkan angka yang akan dicari :";

cin>>search;

auto start = chrono::steady\_clock::now();

int posisi;

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

posisi=i;

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

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

posisi=j;

}

}

swap(arr[i], arr[posisi]);

}

cout << endl << "Array Angka Sorted: " <<endl;

for(int i = 0; i<n;i++) //to get random numbers inside the array.

{

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

}

first = 0;

last = n-1;

middle = (first+last)/2;

while (first <= last)

{

if(arr[middle] < search)

{

first = middle + 1;

}

else if(arr[middle] == search)

{

cout<< endl <<"Angka "<< search<<" ditemukan"<<endl;

break;

}

else

{

last = middle - 1;

}

middle = (first + last)/2;

}

if(first > last)

{

cout<<endl<<"Error! "<<search<<" tidak ditemukan dalam Array" <<endl;

}

auto end = chrono::steady\_clock::now();

auto diff = end - start;

cout << "waktu program :" <<chrono::duration <double, milli> (diff).count() << " ms" << endl;

}

1. **Program Maksimum Minimum Sort**

// Nama: Maksimum Minimum Sort (C++)

// Kelompok: Afifah 140810160008, Baby 140810160048, Muhammad Islam 140810160062

// Mata Kuliah: Analisis Algoritma

#include<iostream>

#include<conio.h>

#include <ctime>

#include <cstdlib>

#include <chrono>

using namespace std;

int main()

{

int n;

int i,temp,j;

cout << "Masukan panjang array :";

cin >> n;

int a[n];

unsigned seed = time(0);

srand(seed);

// Buat Random Array

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

{

a[i]=rand()%1000+1;

}

cout << "Data array sebelum sort :";

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

{

cout<<a[j]<<" ";

}

auto start = chrono::steady\_clock::now();

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

{

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

{

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

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

auto end = chrono::steady\_clock::now();

auto diff = end - start;

cout<<endl;

cout<<"\nData setelah sorting (ASC): ";

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

{

cout<<a[j]<<" ";

}

cout<<"\nData setelah sorting (DESC): ";

for(j=n-1;j>=0;j--)

{

cout<<a[j]<<" ";

}

cout << endl <<endl <<"Runtime : " << chrono::duration <double, milli> (diff).count() << " ms" << endl;

}

1. **Quick Sort (menggunakan pointer linked list)**

/\*

Nama Program : C++ Quick Sort Singly Linked List

Oleh : Baby, Afifah, Islam

Dibuat : 6 April 2018

\*/

#include <iostream>

#include <cstdio>

#include <ctime>

#include <cstdlib>

#include <chrono>

using namespace std;

/\* node singly linked list \*/

struct Node

{

int data;

struct Node \*next;

};

/\* function insert first linked list \*/

void push(struct Node\*\* head\_ref, int new\_data)

{

/\* buat node baru \*/

struct Node\* new\_node = new Node;

/\* masukan node di data \*/

new\_node->data = new\_data;

/\* link node baru ke head \*/

new\_node->next = (\*head\_ref);

/\* pindahkan head ke node baru \*/

(\*head\_ref) = new\_node;

}

/\* function cetak linked list \*/

void printList(struct Node \*node)

{

while (node != NULL)

{

printf("%d ", node->data);

node = node->next;

}

printf("\n");

}

// mengambil nilai terakhir dari list

struct Node \*getTail(struct Node \*cur)

{

while (cur != NULL && cur->next != NULL)

cur = cur->next;

return cur;

}

// Partisi list mengambil last element sebagai pivot

struct Node \*partition(struct Node \*head, struct Node \*end,

struct Node \*\*newHead, struct Node \*\*newEnd)

{

struct Node \*pivot = end;

struct Node \*prev = NULL, \*cur = head, \*tail = pivot;

// Saat Partisi, head dan tail mengalami perubahan jadi newHead dan newEnd

while (cur != pivot)

{

if (cur->data < pivot->data)

{

// Node pertama yang memiliki value kurang dari pivot menjadi head baru

if ((\*newHead) == NULL)

(\*newHead) = cur;

prev = cur;

cur = cur->next;

}

else // Jika node lebih besar dari pivot

{

// Pindahkan node ke next dari tail, dan ubah tail

if (prev)

prev->next = cur->next;

struct Node \*tmp = cur->next;

cur->next = NULL;

tail->next = cur;

tail = cur;

cur = tmp;

}

}

// Jika pivot adalah element terkecil di list,

// pivot jadi head

if ((\*newHead) == NULL)

(\*newHead) = pivot;

// Update newEnd

(\*newEnd) = tail;

// Return node pivot

return pivot;

}

//here the sorting happens exclusive of the end node

struct Node \*quickSortRecur(struct Node \*head, struct Node \*end)

{

// base condition

if (!head || head == end)

return head;

Node \*newHead = NULL, \*newEnd = NULL;

// Partis list, newHead dan newEnd akan diupdate berdasarkan function partisi

struct Node \*pivot = partition(head, end, &newHead, &newEnd);

// Jika pivot adalah element terkecil - tidak perlu recur bagian kiri.

if (newHead != pivot)

{

// Set node sebelum pivot = NULL

struct Node \*tmp = newHead;

while (tmp->next != pivot)

tmp = tmp->next;

tmp->next = NULL;

// Recur for the list before pivot

newHead = quickSortRecur(newHead, tmp);

// Ubah next dari node terakhir dari bagian kiri pivot

tmp = getTail(newHead);

tmp->next = pivot;

}

// Recur for the list after the pivot element

pivot->next = quickSortRecur(pivot->next, newEnd);

return newHead;

}

// main function untuk quick sort.

// function quickSortRecur()

void quickSort(struct Node \*\*headRef)

{

(\*headRef) = quickSortRecur(\*headRef, getTail(\*headRef));

return;

}

int main()

{

struct Node \*a = NULL;

int n;

cout << "Masukan panjang array :";

cin >> n;

int A[n];

unsigned seed = time(0);

srand(seed);

// Buat Random Array

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

{

A[i]=rand()%1000+1;

}

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

{

push(&a, A[i]);

}

cout << endl << "List sebelum sorting \n";

printList(a);

auto start = chrono::steady\_clock::now();

quickSort(&a);

auto end = chrono::steady\_clock::now();

auto diff = end - start;

cout << "List setelah sorting \n";

printList(a);

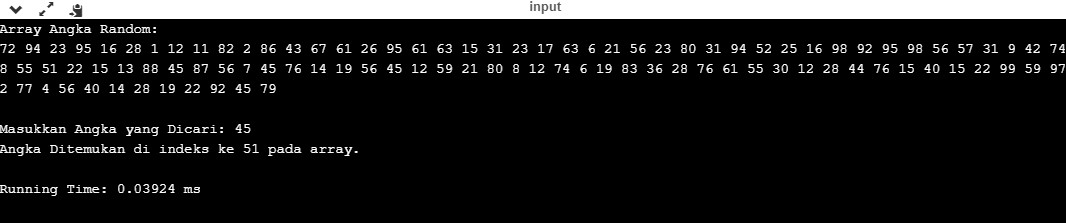
cout << endl <<"Runtime : " << chrono::duration <double, milli> (diff).count() << " ms" << endl;

return 0;

}

1. **Analisis kompleksitas dan perbandingan waktu running time tiap metode dan jumlah data**
2. **Linear Search**

Running Program :

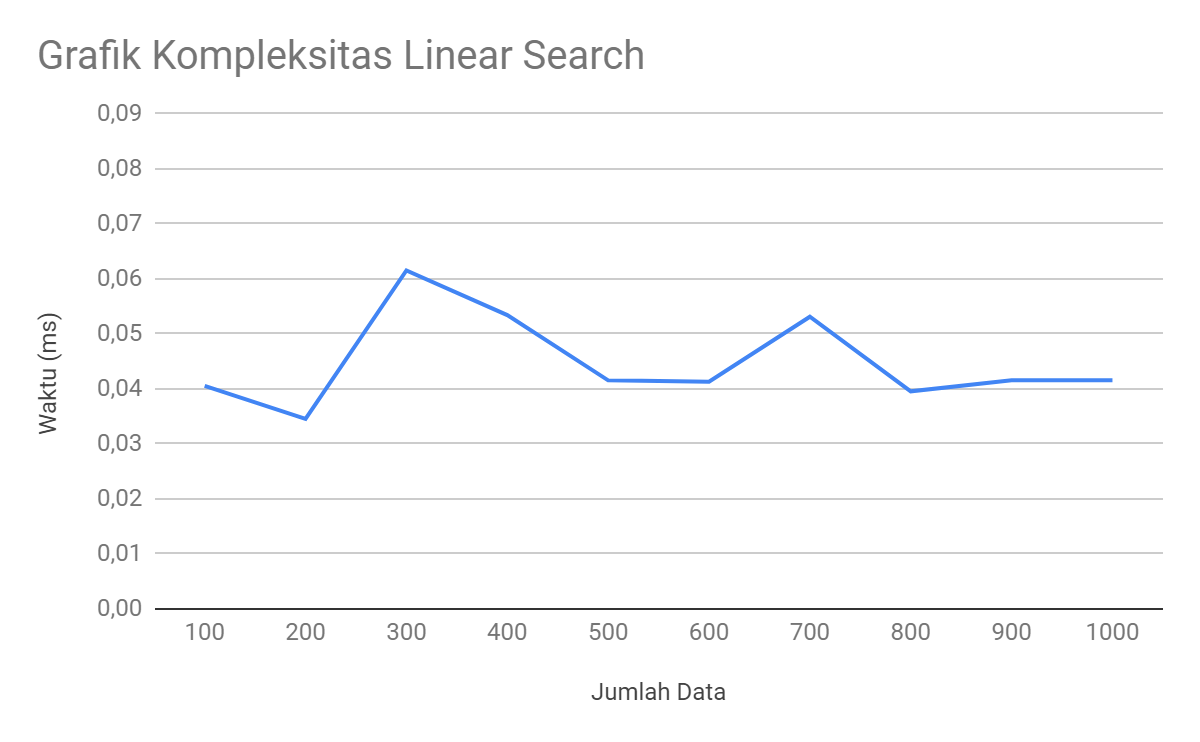


Waktu Running Program :

|  |  |
| --- | --- |
| Jumlah Data | Waktu (ms) |
| 100 | 0,04053 |
| 200 | 0,03453 |
| 300 | 0,06153 |
| 400 | 0,05341 |
| 500 | 0,04155 |
| 600 | 0,04130 |
| 700 | 0,05311 |
| 800 | 0,03955 |
| 900 | 0,04155 |
| 1000 | 0,04155 |

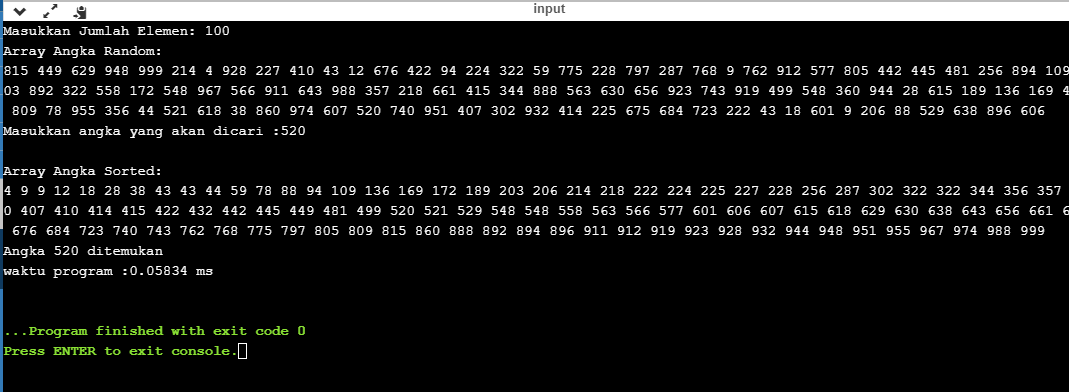
Kompleksitas :

Grafik :



1. **Binary Search**

Running Program :

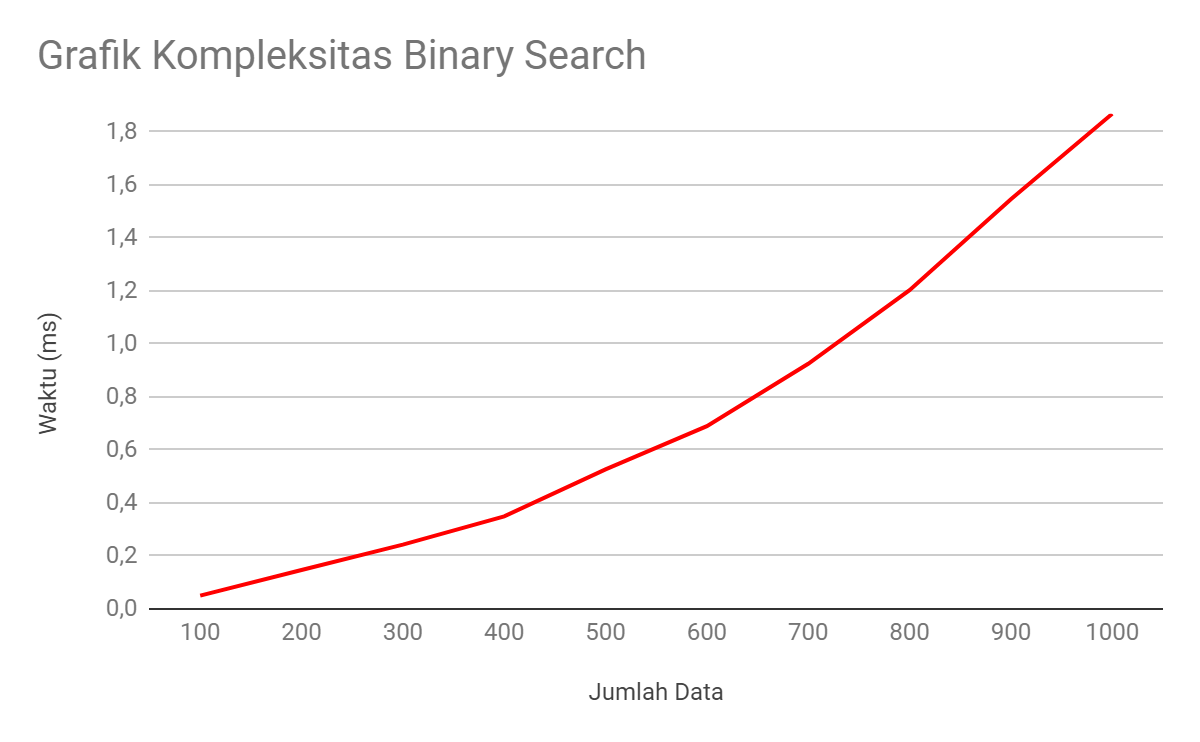


Waktu Running Program :

|  |  |
| --- | --- |
| Jumlah Data | Waktu (ms) |
| 100 | 0,05017 |
| 200 | 0,14670 |
| 300 | 0,24231 |
| 400 | 0,3491 |
| 500 | 0,52653 |
| 600 | 0,6893 |
| 700 | 0,9247 |
| 800 | 1,2024 |
| 900 | 1,54560 |
| 1000 | 1,86750 |

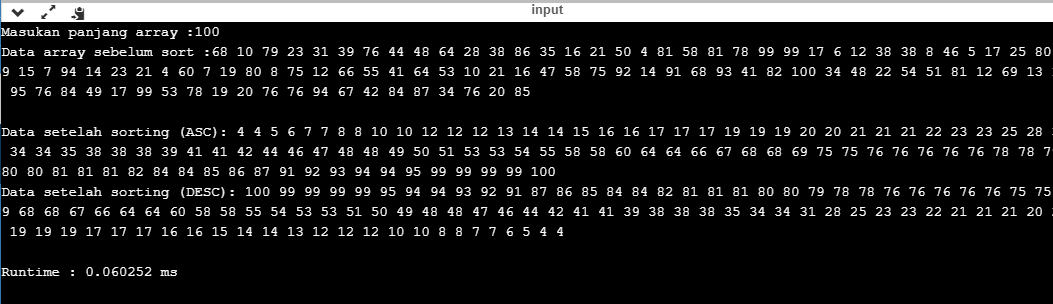
Kompleksitas :

Grafik :



1. **Max Min Search**

Running Program :

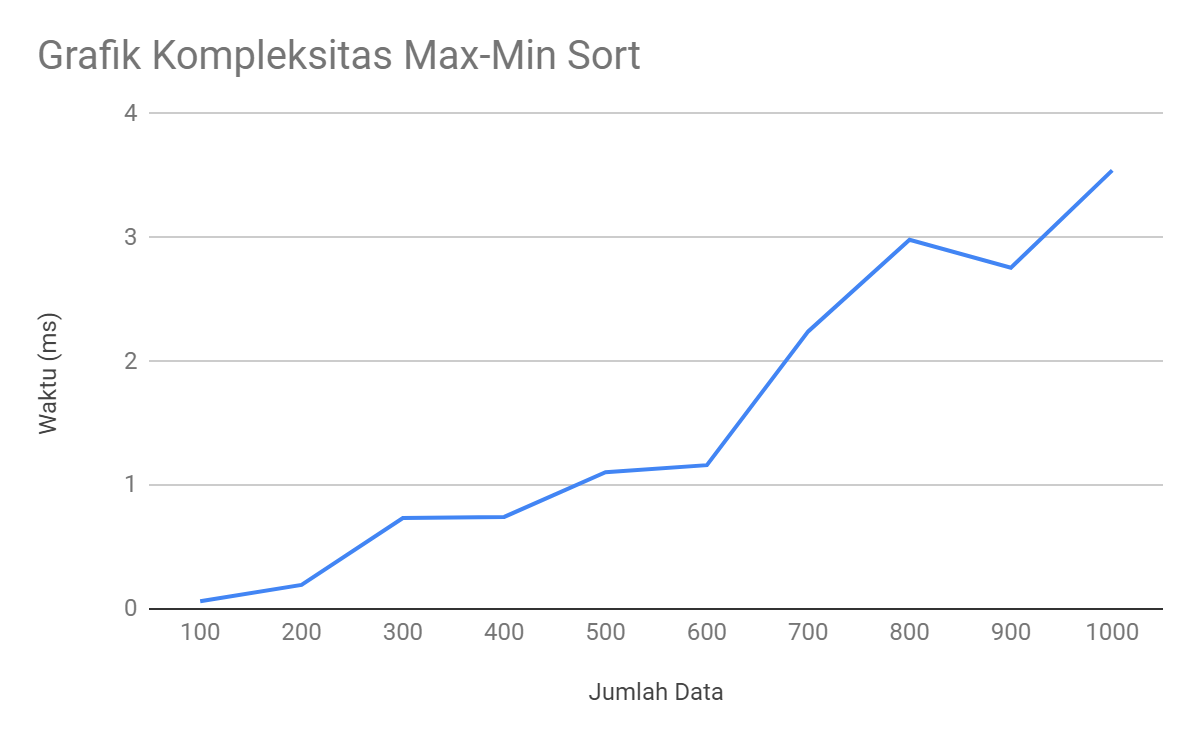


Waktu Running Program :

|  |  |
| --- | --- |
| Jumlah Data | Waktu (ms) |
| 100 | 0,06202 |
| 200 | 0,19345 |
| 300 | 0,7345 |
| 400 | 0,7431 |
| 500 | 1,1045 |
| 600 | 1,1613 |
| 700 | 2,242 |
| 800 | 2,982 |
| 900 | 2,7564 |
| 1000 | 3,5432 |

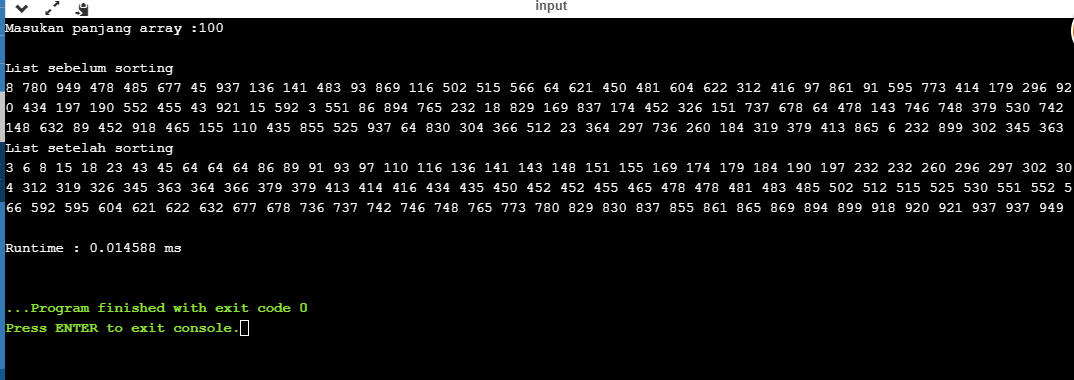
Kompleksitas :

Grafik :



1. **Quick Sort**

Running Program :



Waktu Running Program :

|  |  |
| --- | --- |
| Jumlah Data | Waktu (ms) |
| 100 | 0,0145 |
| 200 | 0,0307 |
| 300 | 0,0453 |
| 400 | 0,0642 |
| 500 | 0,0845 |
| 600 | 0,1653 |
| 700 | 0,1199 |
| 800 | 0,1456 |
| 900 | 0,1705 |
| 1000 | 0,1785 |

Kompleksitas :

Grafik :

