## **LAPORAN PRAKTIKUM 13.1**

“tugas teori”

Mata kuliah Praktikum Algoritma Struktur Data



Nama dosen pengampu:

Ibu Umi Sa'adah S.Kom, M.Kom

Dikerjakan oleh

Nama : Adrian Mendienta Tumanggor

NRP : 3123600012

Kelas :  1 D4 IT A

**Source code:**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <math.h>

#define max 100000

void swap(int \*, int \*);

void generate(int[], int);

int partition(int[], int, int);

void quickSort(int[], int, int);

void sequentialSearch(int[], int, int);

void binarySearch(int[], int, int, int);

void searchInArray(int[], int, int, int);

void displayMenu(int[], int, int);

int main()

{

int n = max;

int arr[n];

int x;

generate(arr, n);

printf("Array 100k.\n");

printf("Masukkan elemen yang ingin dicari: ");

scanf("%d", &x);

displayMenu(arr, n, x);

return 0;

}

void swap(int \*a, int \*b)

{

int t = \*a;

\*a = \*b;

\*b = t;

}

void generate(int arr[], int n)

{

srand(time(NULL));

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

{

arr[i] = rand() % 1000;

}

}

int partition(int arr[], int low, int high)

{

int pivot = arr[high];

int i = low - 1;

for (int j = low; j <= high - 1; j++)

{

if (arr[j] < pivot)

{

i++;

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i + 1], &arr[high]);

return i + 1;

}

void quickSort(int arr[], int low, int high)

{

if (low < high)

{

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

void sequentialSearch(int arr[], int n, int x)

{

int jumlah = 0;

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

{

if (arr[i] == x)

{

jumlah++;

}

}

printf("Elemen %d ditemukan %d kali.\n", x, jumlah);

}

void binarySearch(int arr[], int low, int high, int x)

{

int jumlah = 0;

while (low <= high)

{

int mid = low + (high - low) / 2;

if (arr[mid] == x)

{

jumlah++;

int temp = mid;

while (arr[--temp] == x && temp >= low)

jumlah++;

temp = mid;

while (arr[++temp] == x && temp <= high)

jumlah++;

break;

}

if (arr[mid] < x)

low = mid + 1;

else

high = mid - 1;

}

printf("Elemen %d ditemukan %d kali.\n", x, jumlah);

}

void searchInArray(int arr[], int n, int x, int pilih)

{

clock\_t start, end;

double waktu;

start = clock();

switch (pilih)

{

case 1:

sequentialSearch(arr, n, x);

break;

case 2:

quickSort(arr, 0, n - 1);

sequentialSearch(arr, n, x);

break;

case 3:

quickSort(arr, 0, n - 1);

binarySearch(arr, 0, n - 1, x);

break;

case 4:

printf("Keluar dari program.\n");

exit(0);

default:

printf("Pilihan tidak valid.\n");

}

end = clock();

waktu = ((double)(end - start)) / CLOCKS\_PER\_SEC;

printf("Waktu Komputasi: %lf detik.\n", waktu);

}

void displayMenu(int arr[], int n, int x)

{

int pilih;

do

{

printf("\nPilih metode pencarian:\n");

printf("1. Unsorted Sequential Search\n");

printf("2. Sequential Search setelah Quick Sort\n");

printf("3. Binary Search\n");

printf("4. Keluar\n");

printf("Masukkan pilihan: ");

scanf("%d", &pilih);

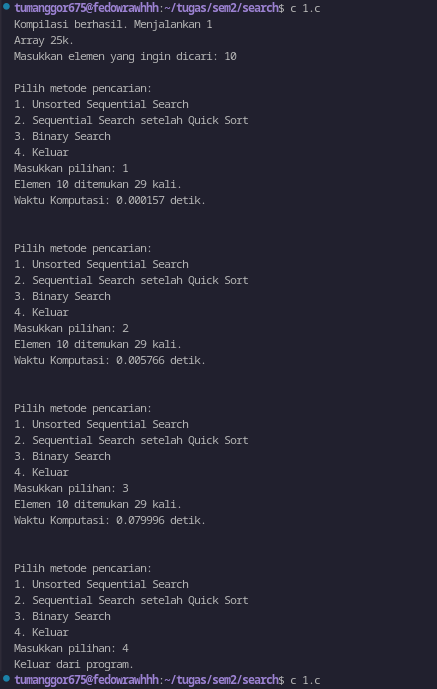
searchInArray(arr, n, x, pilih);

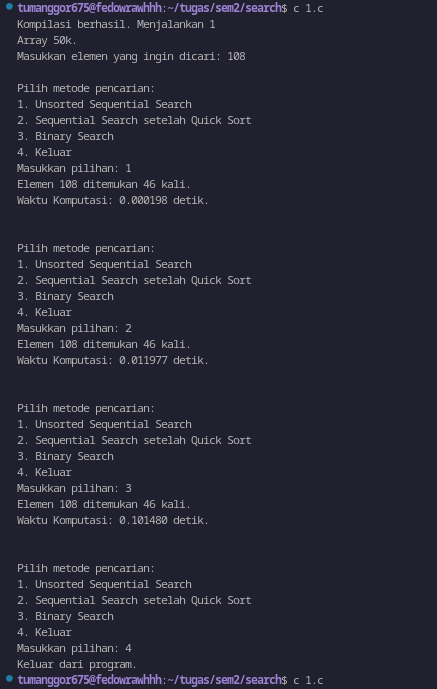
puts("");

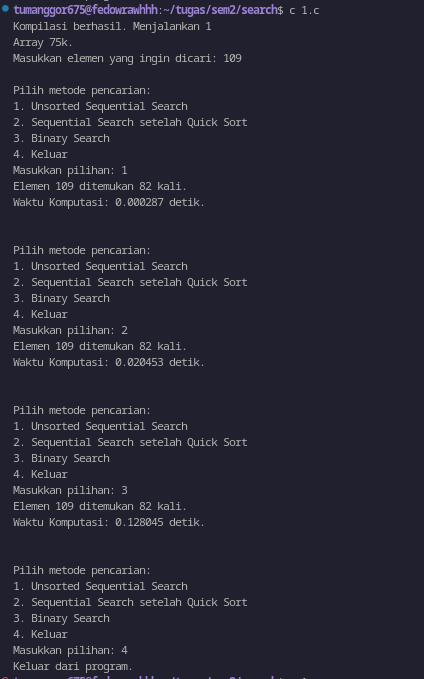
} while (pilih != 4);

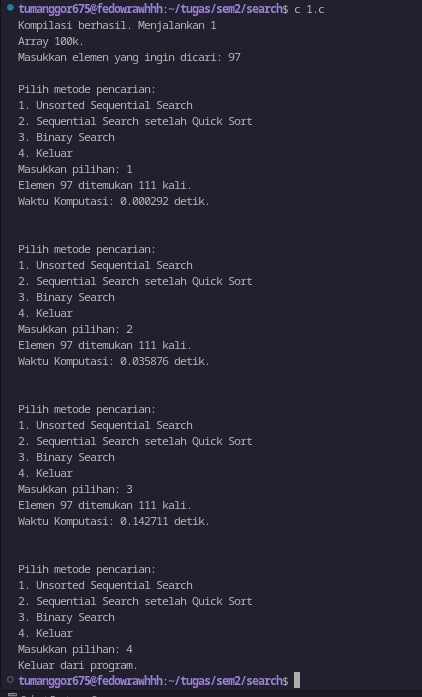
}

**Output:**

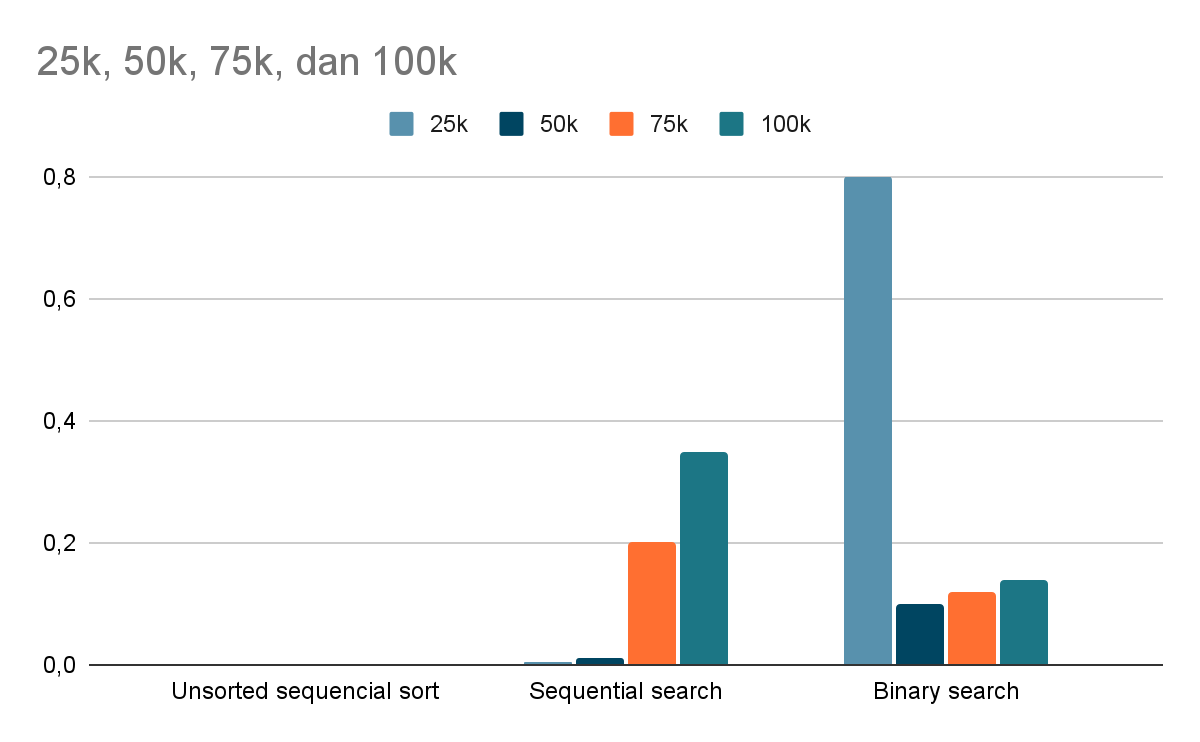
****

****

****

****

**Diagram:**

****

**Kesimpulan:**

Metode pencarian tercepat adalah unsorted sequential search, diikuti oleh sorted sequential search, dan yang terakhir adalah binary search. Meskipun binary search umumnya dianggap sebagai metode pencarian yang paling efisien, dalam kasus ini, unsorted sequential search menunjukkan kinerja yang lebih baik untuk mencari elemen dalam array.