**JOOBSHET 10**

****

**Disusun Oleh :**

**Shadafi fastiyan**

**BP/NIM: 2023/23343084**

**Hari/Tanggal : Selasa/ 20 april 2024**

**Sesi/Jam : 202323430164 / 07:00-08:40 WIB**

**Dosen Pengampu :**

**Randi Proska Sandra, M.Sc**

**Kode kelas;202323430158**

**FAKULTAS TEKNIK**

**DEPARTEMEN TEKNIK ELEKTRONIKA**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**UNIVERSITAS NEGERI PADANG**

**2024**

1.source code

a.sell sort

#include <stdio.h>

void shellSort(int arr[], int n) {

    for (int gap = n / 2; gap > 0; gap /= 2) {

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

            int temp = arr[i];

            int j;

            for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) {

                arr[j] = arr[j - gap];

            }

            arr[j] = temp;

        }

    }

}

void printArray(int arr[], int n) {

    for (int i = 0; i < n; i++) {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

int main() {

    int times[] = {120, 110, 105, 125, 101, 100, 130, 109};

    int n = sizeof(times) / sizeof(times[0]);

    printf("Original array: \n");

    printArray(times, n);

    shellSort(times, n);

    printf("Sorted array: \n");

    printArray(times, n);

    return 0;

}

b.quick sort

#include <stdio.h>

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

    int t = \*a;

    \*a = \*b;

    \*b = t;

}

int partition(int array[], int low, int high) {

    int pivot = array[high];

    int i = (low - 1);

    for (int j = low; j < high; j++) {

        if (array[j] < pivot) {

            i++;

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

        }

    }

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

    return (i + 1);

}

void quickSort(int array[], int low, int high) {

    if (low < high) {

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

        quickSort(array, low, pi - 1);

        quickSort(array, pi + 1, high);

    }

}

void printArray(int arr[], int size) {

    for (int i = 0; i < size; ++i) {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

int main() {

    int times[] = {120, 110, 105, 125, 101, 100, 130, 109};

    int n = sizeof(times) / sizeof(times[0]);

    printf("Original array: \n");

    printArray(times, n);

    quickSort(times, 0, n - 1);

    printf("Sorted array in ascending order: \n");

    printArray(times, n);

    return 0;

}

Kedua program di atas melakukan hal berikut:

1. **Shell Sort**: Memulai dengan gap yang besar dan secara bertahap mengecil gap tersebut, mengurutkan elemen berdasarkan perbandingan dengan elemen yang jaraknya beberapa posisi lebih dekat.
2. **Quick Sort**: Menggunakan pendekatan "divide and conquer", memilih pivot dan mengurutkan elemen berdasarkan apakah mereka lebih kecil atau lebih besar dari pivot.