LAPORAN PRAKTIKUM STRUKTUR DATA

Jobsheet 10 Shell Sort and Quick Sort

Disusun Oleh:

Fitri Waldi: 22343021

Dosen Pengampu:

Randi Proska Sandra S.Pd., M.Sc.

PROGRAM STUDI S1 INFORMATIKA (NK)

JURUSAN TEKNIK ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2023

Tugas

Carilah contoh aplikasi yang mengimplementasikan shell sort dan quick sort serta jelaskan bagaimana aplikasi tersebut bekerja sesuai dengan prinsip kedua metode sorting tersebut!

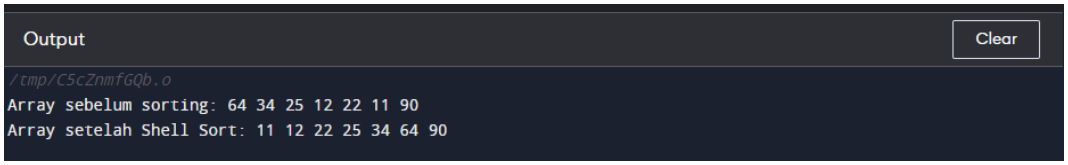
1. Shell Sort

Kode:

```cpp
//dibuat-oleh_22343021_Fitri-Waldi
#include <iostream>
using namespace std;

void shellSort(int arr[], int n) {
    // Menentukan jarak awal
    for (int gap = n / 2; gap > 0; gap /= 2) {
        // Lakukan insertion sort dengan jarak gap
        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++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Array sebelum sorting: ";
    printArray(arr, n);

    shellSort(arr, n);
```

```
    cout << "Array setelah Shell Sort: ";
    printArray(arr, n);

    return 0;
}
```

Output:



Penjelasan:

Algoritma Shell Sort adalah algoritma pengurutan yang membagi array menjadi beberapa subarray yang lebih kecil dan kemudian mengurutkan subarray-subarray tersebut menggunakan algoritma Insertion Sort.

Berikut adalah penjelasan singkat mengenai setiap bagian program:

1. Fungsi **shellSort**:
    - Fungsi ini mengimplementasikan algoritma Shell Sort.
    - Menerima dua parameter: array **arr** dan ukuran **n**.
    - Pada algoritma Shell Sort, kita menggunakan konsep jarak untuk membagi array menjadi subarray.
    - Pertama, kita menentukan jarak awal **gap** dengan menginisialisasi dengan **n/2**.
    - Selanjutnya, kita melakukan insertion sort pada setiap subarray dengan menggunakan jarak **gap**.
    - Dalam insertion sort, kita membandingkan elemen pada jarak **gap** dan menukarnya jika diperlukan untuk memastikan bahwa elemen pada subarray tersebut terurut dengan benar.
    - Kemudian, kita mengurangi jarak **gap** menjadi setengahnya dan mengulangi proses insertion sort dengan jarak yang lebih kecil.

- Proses ini terus diulang hingga jarak **gap** menjadi 1, sehingga pada akhirnya seluruh elemen array akan terurut secara keseluruhan.

2. Fungsi **printArray**:
    - Fungsi ini digunakan untuk mencetak elemen-elemen dalam array.
    - Menerima dua parameter: array **arr** dan ukuran **n**.
    - Iterasi dilakukan untuk mencetak setiap elemen array.
3. Di dalam fungsi **main**, terdapat:
    - Array yang akan diurutkan, yaitu **arr**.
    - Menghitung jumlah elemen dalam array dengan menggunakan **sizeof(arr) / sizeof(arr[0])**.
    - Mencetak array sebelum sorting menggunakan fungsi **printArray**.
    - Memanggil fungsi **shellSort** untuk mengurutkan array.
    - Mencetak array setelah sorting menggunakan fungsi **printArray**.

Hasil program ini adalah array yang terurut secara ascending menggunakan algoritma Shell Sort.

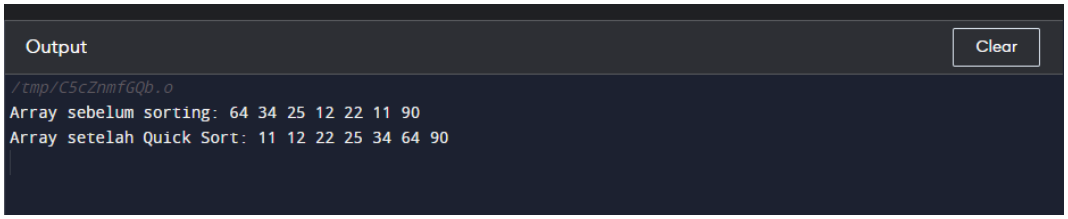
2. Quick Sort

Kode:

```
//dibuat-oleh_22343021_Fitri-Waldi

#include <iostream>
using namespace std;

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 pivotIndex = partition(arr, low, high);

        quickSort(arr, low, pivotIndex - 1);
        quickSort(arr, pivotIndex + 1, high);
    }
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Array sebelum sorting: ";
    printArray(arr, n);

    quickSort(arr, 0, n - 1);

    cout << "Array setelah Quick Sort: ";
    printArray(arr, n);

    return 0;
}
```

Output:



Penjelasan:

Algoritma Quick Sort adalah algoritma pengurutan yang menggunakan pendekatan "divide and conquer" (bagi dan taklukkan) untuk mengurutkan elemen dalam array.

Berikut adalah penjelasan singkat mengenai setiap bagian program:

1. Fungsi **partition**:
    - Fungsi ini digunakan untuk mempartisi array berdasarkan elemen pivot.
    - Menerima tiga parameter: array **arr**, indeks **low**, dan indeks **high**.
    - Pada awalnya, fungsi ini memilih elemen pivot dari array (di sini elemen pivot adalah elemen terakhir, **arr[high]**).
    - Selanjutnya, menggunakan dua indeks **i** dan **j**, fungsi ini melakukan iterasi dari **low** hingga **high-1** untuk membandingkan setiap elemen dengan pivot.
    - Jika elemen **arr[j]** lebih kecil dari pivot, maka elemen tersebut dipindahkan ke sebelah kiri pivot dengan menukar elemen **arr[j]** dengan elemen di sebelah kanan **arr[i]** (dalam hal ini, **arr[i+1]**).
    - Setelah selesai melakukan partisi, elemen pivot ditempatkan di posisi yang benar dengan menukar elemen di sebelah kanan **arr[i]** dengan pivot.
    - Fungsi mengembalikan indeks pivot setelah partisi.
2. Fungsi **quickSort**:
    - Fungsi ini adalah implementasi rekursif dari algoritma Quick Sort.
    - Menerima tiga parameter: array **arr**, indeks **low**, dan indeks **high**.
    - Pada setiap rekursi, fungsi ini memanggil fungsi **partition** untuk mempartisi array dengan menggunakan pivot yang baru.

- Setelah melakukan partisi, fungsi melakukan rekursi pada dua subarray yang dihasilkan sebelum dan setelah indeks pivot.

3. Fungsi **printArray**:
    - Fungsi ini digunakan untuk mencetak elemen-elemen dalam array.
    - Menerima dua parameter: array **arr** dan ukuran **n**.
    - Iterasi dilakukan untuk mencetak setiap elemen array.
4. Di dalam fungsi **main**, terdapat:
    - Array yang akan diurutkan, yaitu **arr**.
    - Menghitung jumlah elemen dalam array dengan menggunakan **sizeof(arr) / sizeof(arr[0])**.
    - Mencetak array sebelum sorting menggunakan fungsi **printArray**.
    - Memanggil fungsi **quickSort** untuk mengurutkan array.
    - Mencetak array setelah sorting menggunakan fungsi **printArray**.

Hasil program ini adalah array yang terurut secara ascending menggunakan algoritma Quick Sort.