LAPORAN PRAKTIKUM STRUKTUR DATA

Jobsheet 9 Selection Sort and Merge 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 selection sort dan merge sort serta jelaskan bagaimana aplikasi tersebut bekerja sesuai dengan prinsip kedua metode sorting tersebut!

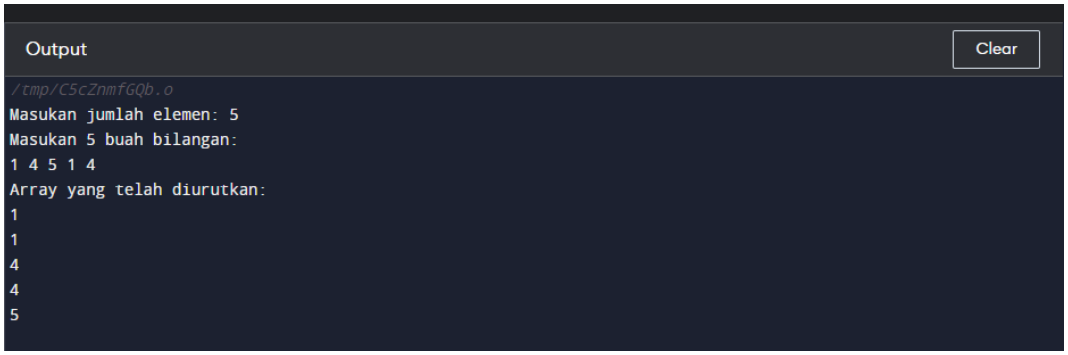
1. Selection Sort

Kode:

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

void selectionSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int min_idx = i;
        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
                min_idx = j;
            }
        }
        swap(arr[i], arr[min_idx]);
    }
}

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);

    selectionSort(arr, n);

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

```
    return 0;
}
```

Output:



Penjelasan:

Pada program tersebut, terdapat beberapa fungsi yang digunakan:

1. **void selectionSort(int arr[], int n)**: Fungsi ini digunakan untuk mengurutkan array **arr** dengan ukuran **n** menggunakan algoritma Selection Sort. Pada setiap iterasi, elemen terkecil akan dicari dan ditukar dengan elemen pada indeks **i**, di mana **i** adalah posisi saat ini dalam iterasi.
2. **void printArray(int arr[], int n)**: Fungsi ini digunakan untuk mencetak elemen-elemen array **arr** dengan ukuran **n**. Fungsi ini hanya melakukan iterasi melalui array dan mencetak setiap elemen.

Selanjutnya, di dalam fungsi **main**, program melakukan hal berikut:

1. Deklarasi dan inisialisasi array **arr** dengan elemen **{64, 34, 25, 12, 22, 11, 90}**.
2. Menghitung ukuran array dengan menggunakan rumus **sizeof(arr) / sizeof(arr[0])** dan menyimpannya dalam variabel **n**.
3. Mencetak array sebelum diurutkan menggunakan fungsi **printArray(arr, n)**.
4. Memanggil fungsi **selectionSort(arr, n)** untuk mengurutkan array.
5. Mencetak array setelah diurutkan menggunakan fungsi **printArray(arr, n)**.

Algoritma Selection Sort bekerja dengan cara memilih elemen terkecil pada setiap iterasi dan menukar posisinya dengan elemen pada posisi saat ini. Iterasi dilakukan untuk seluruh elemen dalam array, sehingga secara bertahap elemen-elemen terkecil akan naik ke awal array, menghasilkan array yang terurut secara ascending.

2. Merge Sort

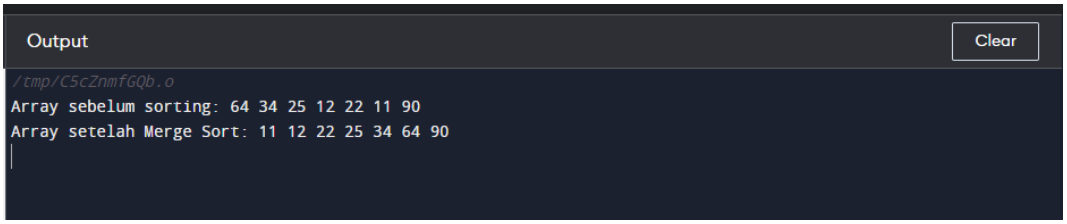
Kode:

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

// Fungsi untuk menggabungkan dua subarray yang terurut menjadi satu subarray terurut
void merge(int arr[], int left, int mid, int right) {
    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;

    // Buat subarray sementara
    int L[n1], R[n2];

    // Salin data ke subarray sementara L[] dan R[]
    for (i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    // Gabungkan subarray sementara L[] dan R[] menjadi arr[]
    i = 0;  // Indeks awal subarray L[]
    j = 0;  // Indeks awal subarray R[]
    k = left;  // Indeks awal subarray gabungan arr[]

    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
```

```
    // Salin elemen yang tersisa dari L[], jika ada
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    // Salin elemen yang tersisa dari R[], jika ada
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

// Fungsi rekursif untuk melakukan Merge Sort pada subarray
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        // Panggil rekursif untuk subarray kiri dan kanan
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

        // Gabungkan dua subarray yang terurut
        merge(arr, left, mid, right);
    }
}

// Fungsi untuk mencetak elemen array
void printArray(int arr[], int size) {
    for (int i = 0; i < size; 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);

    mergeSort(arr, 0, n - 1);
```

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

    return 0;
}
```

Output:



Penjelasan:

Program di atas mengimplementasikan algoritma Merge Sort menggunakan bahasa C++.

Pada awalnya, program mendefinisikan fungsi **merge** yang digunakan untuk menggabungkan dua subarray yang terurut menjadi satu subarray terurut. Fungsi ini menerima empat parameter: array **arr**, indeks awal **left**, indeks tengah **mid**, dan indeks akhir **right**. Fungsi **merge** menggunakan dua subarray sementara **L[]** dan **R[]** untuk menyimpan data dari subarray kiri dan kanan. Selanjutnya, elemen-elemen dari subarray sementara tersebut dibandingkan dan digabungkan kembali ke dalam array **arr**.

Selanjutnya, program mendefinisikan fungsi **mergeSort** yang merupakan fungsi rekursif untuk melakukan Merge Sort pada subarray. Fungsi ini menerima tiga parameter: array **arr**, indeks awal **left**, dan indeks akhir **right**. Pada setiap rekursi, fungsi **mergeSort** membagi array menjadi dua bagian dengan menghitung indeks tengah **mid**. Kemudian, fungsi ini memanggil dirinya sendiri pada kedua subarray tersebut untuk melakukan pengurutan rekursif. Setelah itu, subarray yang telah terurut dipanggil fungsi **merge** untuk digabungkan kembali menjadi satu subarray terurut.

Selanjutnya, program mendefinisikan fungsi **printArray** yang digunakan untuk mencetak elemen-elemen array.

Di dalam fungsi **main**, terdapat inisialisasi array **arr** dengan beberapa nilai. Kemudian, program mencetak array sebelum sorting dengan memanggil fungsi **printArray**. Selanjutnya, fungsi **mergeSort** dipanggil untuk melakukan pengurutan Merge Sort pada array **arr**. Setelah proses pengurutan selesai, program mencetak array setelah Merge Sort dengan memanggil fungsi **printArray**.

Hasil yang diharapkan adalah mencetak array sebelum sorting dan array setelah Merge Sort, sehingga kita dapat melihat hasil pengurutan array menggunakan algoritma Merge Sort.