Pengaplikasian merge sort dan selection sort

Disusun Oleh :

Shadafi fastiyan

BP/NIM: 2023/23343084

Hari/Tanggal **: Selasa/ 19 Maret 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

a.kodingan

```c
//created by shadafi fastiyan
//nim 23343084
#include <stdio.h>
#include <string.h>

// Fungsi untuk menukar dua elemen
void swap(char *str1[], char *str2[]) {
    char *temp = *str1;
    *str1 = *str2;
    *str2 = temp;
}

// Selection Sort untuk menyortir array string
void selectionSort(char *arr[], int n) {
    int i, j, min_idx;

    for (i = 0; i < n-1; i++) {
        min_idx = i;
        for (j = i+1; j < n; j++)
          if (strcmp(arr[j], arr[min_idx]) < 0)
            min_idx = j;

        swap(&arr[min_idx], &arr[i]);
    }
}

// Gabungkan dua subarray dari arr[]
void merge(char *arr[], int l, int m, int r) {
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    char *L[n1], *R[n2];

    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2) {
```

```
        if (strcmp(L[i], R[j]) <= 0) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

// Merge sort untuk menyortir arr[] dari l ke r
void mergeSort(char *arr[], int l, int r) {
    if (l < r) {
        int m = l + (r-l)/2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}

int main() {
    char *list1[] = {"Book D", "Book A", "Book C"};
    int n1 = sizeof(list1) / sizeof(list1[0]);
    char *list2[] = {"Book B", "Book E", "Book F"};
    int n2 = sizeof(list2) / sizeof(list2[0]);

    // Sort list1 using Selection Sort
    selectionSort(list1, n1);

    // Sort list2 using Selection Sort
    selectionSort(list2, n2);
```

```
    // Combine list1 and list2 into list3 and sort using Merge Sort
    char *combinedList[n1 + n2];
    memcpy(combinedList, list1, sizeof(list1));
    memcpy(combinedList + n1, list2, sizeof(list2));

    mergeSort(combinedList, 0, n1 + n2 - 1);

    printf("Sorted combined list of books:\n");
    for (int i = 0; i < n1 + n2; i++) {
        printf("%s\n", combinedList[i]);
    }

    return 0;
}
```

Cara Kerja Aplikasi

1. **Selection Sort:**
    - Pertama, list1 disortir menggunakan selection sort. Di sini, elemen minimum dari daftar dipilih dan dipindahkan ke posisi yang benar. Ini diulang untuk semua elemen dalam array, sehingga setiap langkah memastikan satu elemen lagi sudah berada di posisi terurutnya.
2. **Merge Sort:**
    - Setelah kedua daftar disortir secara individu (kedua menggunakan selection sort di contoh ini untuk kesederhanaan), kedua daftar tersebut digabungkan ke dalam satu array baru.
    - Array gabungan ini kemudian disortir menggunakan merge sort, yang efektif untuk menggabungkan dua daftar yang telah disortir. Merge sort bekerja dengan memecah array menjadi dua, menyortir bagian-bagian tersebut secara rekursif, dan kemudian menggabungkan mereka kembali dengan cara yang terurut.

Kode ini menunjukkan bagaimana mengimplementasikan kedua algoritma sorting dalam C untuk mengelola data secara efisien dan menunjukkan penggunaan praktis dari algoritma dalam konteks seperti sistem manajemen perpustakaan atau toko buku.