PRAKTIKUM STRUKTUR DATA

INF11198

FACHREZI BACHRI – 2401020010

1. Pembuatan codingan untuk metode pengurutan yang dimana mengurutkan dari angka terkecil ke terbesar (ascending order), menggunakan algoritma insertion sort

   Sourch code algoritma insertion sort :

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void insertion_sort(int larik[], int jumlah) {
    for (int i = 1; i < jumlah; i++) {
        int kunci = larik[i];
        int j = i - 1;

        while (j >= 0) {
            if (larik[j] > kunci) {
                larik[j + 1] = larik[j];
                j--;
            } else {
                break;
            }
        }
        larik[j + 1] = kunci;
    }
}
```

```
int memuat_data(const char *nama_file, int **larik) {
    FILE *berkas = fopen(nama_file, "r");
    if (!berkas) {
        perror("Gagal membuka file");
        exit(1);
    }

    int jumlah = 0;
    int sementara;
    while (fscanf(berkas, "%d", &sementara) == 1) jumlah++;

    rewind(berkas);
    *larik = malloc(jumlah * sizeof(int));
    for (int i = 0; i < jumlah; i++) fscanf(berkas, "%d",
&(*larik)[i]);

    fclose(berkas);
    return jumlah;
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("File tidak ditemukan");
        return 1;
    }

    int *data = NULL;
    int jumlah = memuat_data(argv[1], &data);

    clock_t mulai = clock();
    insertion_sort(data, jumlah);
```

```
    clock_t selesai = clock();

    double waktu = (double)(selesai - mulai) / CLOCKS_PER_SEC;
    double mikro = waktu * 1000000;

    printf("Jumlah data: %d\n", jumlah);
    printf("Waktu eksekusi: %.6f detik (%.0f mikrodetik)\n",
waktu, mikro);

    free(data);
    return 0;
}
```

2. Pembuatan codingan untuk metode pengurutan yang dimana mengurutkan dari angka terkecil ke terbesar (ascending order), menggunakan algoritma selection sort

   Sourch code algoritma selection sort:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void selection_sort(int larik[], int jumlah) {
    for (int i = 0; i < jumlah - 1; i++) {
        int indeks_min = i;
        for (int j = i + 1; j < jumlah; j++) {
            if (larik[j] < larik[indeks_min])
                indeks_min = j;
        }
        int sementara = larik[indeks_min];
        larik[indeks_min] = larik[i];
        larik[i] = sementara;
```

```
    }
}

int memuat_data(const char *nama_file, int **larik) {
    FILE *berkas = fopen(nama_file, "r");
    if (!berkas) {
        perror("Gagal membuka file");
        exit(1);
    }

    int jumlah = 0, sementara;
    while (fscanf(berkas, "%d", &sementara) == 1) jumlah++;
    rewind(berkas);

    *larik = malloc(jumlah * sizeof(int));
     for (int i = 0; i < jumlah; i++) fscanf(berkas, "%d",
&(*larik)[i]);

    fclose(berkas);
    return jumlah;
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("File tidak ditemukan");
        return 1;
    }

    int *data = NULL;
    int jumlah = memuat_data(argv[1], &data);

    clock_t mulai = clock();
```

```
    selection_sort(data, jumlah);
    clock_t selesai = clock();

    double waktu = (double)(selesai - mulai) / CLOCKS_PER_SEC;
    double mikro = waktu * 1000000;

    printf("Jumlah data: %d\n", jumlah);
    printf("Waktu eksekusi: %.6f detik (%.6f mikrodetik)\n", waktu,
mikro);

    free(data);
    return 0;
}
```

3. Disini kita di minta untuk mengurutkan data yang sudah kita siapkan dengan 3 file yaitu file dengan nama data100.csv, data1000.csv dan data10000.csv dengan isi yang acak, lalu kita di minta untuk mengurutkan data data itu, saya menggurutkan data data itu dengan metode ascending order dengan algoritma insertion sort dan selection sort dan kita juga di mintak untuk menghitung waktu (dalam detik), yang di hitung adalah waktu eksekusi seberapa lama program bisa mengurutkan data yang sudah kita buat.

Table hasil eksekusi menggunakan metode insertion sort :

TABLE DATA 100		
No	Waktu (dalam detik)	Waktu (dalam mikrodetik)
1	0.000000 detik	0.000000 mikrodetik
2	0.000000 detik	0.000000 mikrodetik
3	0.000000 detik	0.000000 mikrodetik

TABLE DATA 1000		
No	Waktu (dalam detik)	Waktu (dalam mikrodetik)
1	0.000000 detik	0.000000 mikrodetik
2	0.001000 detik	1000.000000 mikrodetik
3	0.000000 detik	0.000000 mikrodetik

TABLE DATA 10000		
No	Waktu (dalam detik)	Waktu (dalam mikrodetik)
1	0.034000 detik	34000.000000 mikrodetik
2	0.034000 detik	34000.000000 mikrodetik
3	0.033000 detik	33000.000000 mikrodetik

Saya melalukan eksekusi sebanyak tiga kali untuk setiap data suapaya menjadi perbandingan saja.

Table hasil eksekusi menggunakan metode selection sort:

TABLE DATA 100		
No	Waktu (dalam detik)	Waktu (dalam mikrodetik)
1	0.000000 detik	0.000000 mikrodetik
2	0.000000 detik	0.000000 mikrodetik
3	0.000000 detik	0.000000 mikrodetik

TABLE DATA 1000		
No	Waktu (dalam detik)	Waktu (dalam mikrodetik)
1	0.000000 detik	0.000000 mikrodetik
2	0.001000 detik	1000.000000 mikrodetik
3	0.000000 detik	0.000000 mikrodetik

TABLE DATA 10000		
No	Waktu (dalam detik)	Waktu (dalam mikrodetik)
1	0.052000 detik	52000.000000 mikrodetik
2	0.053000 detik	53000.000000 mikrodetik
3	0.052000 detik	52000.000000 mikrodetik

Saya melalukan eksekusi sebanyak tiga kali untuk setiap data suapaya menjadi perbandingan saja.

4. Kesimpulan

Berdasarkan data yang ada kedua algoritma Insertion Sort maupun Selection Sort, membutuhkan waktu eksekusi yang lebih lama seiring dengan peningkatan jumlah data yang diolah. Namun, ketika mengurutkan 10.000 data, Insertion Sort (dengan waktu rata-rata sekitar 0.034 detik) terlihat sedikit lebih cepat daripada Selection Sort (dengan waktu rata-rata sekitar 0.052 detik).