Struktur Data dan Algoritma

disusun untuk memenuhi

Oleh:

FARHAN MUJIBURRAHMAN
2308107010078

JURUSAN INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SYIAH KUALA
DARUSSALAM, BANDA ACEH

A. Deskripsi dan Implementasi Algoritma Sorting

1. Bubble sort

Bubble Sort adalah algoritma pengurutan sederhana yang bekerja dengan cara membandingkan dua elemen bersebelahan dan menukarnya jika urutannya salah. Proses ini diulang-ulang sampai seluruh elemen berada dalam urutan yang benar. Bubble Sort dinamakan demikian karena elemen terbesar "mengambang" ke posisi akhir seperti gelembung.

```
void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
```

2. Selection sort

Selection Sort adalah algoritma pengurutan yang bekerja dengan cara memilih elemen terkecil dari daftar dan menempatkannya di posisi pertama. Kemudian, mencari elemen terkecil dari sisa daftar dan menempatkannya di posisi berikutnya, dan seterusnya, hingga semua elemen terurut.

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

```
        int temp = arr[minIdx];
        arr[minIdx] = arr[i];
        arr[i] = temp;
    }
}
```

3. Insertion Sort

Insertion Sort adalah metode pengurutan di mana setiap elemen dari daftar diambil satu per satu dan disisipkan ke posisi yang sesuai dalam bagian daftar yang sudah terurut. Teknik ini menyerupai cara seseorang mengurutkan kartu dalam tangan saat bermain kartu.

```
void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = key;
    }
}
```

4. Merge Sort

Merge Sort adalah algoritma pengurutan berbasis konsep divide and conquer (membagi dan menaklukkan). Array dibagi menjadi dua bagian yang lebih kecil, masing-masing diurutkan secara rekursif, lalu kedua bagian tersebut digabungkan kembali menjadi satu array yang terurut.

```
void merge(int arr[], int l, int m, int r) {

    int n1 = m - l + 1;
```

```
    int n2 = r - m;



    int *L = (int*) malloc(n1 * sizeof(int));

    int *R = (int*) malloc(n2 * sizeof(int));



    for (int i = 0; i < n1; i++) L[i] = arr[l + i];

    for (int j = 0; j < n2; j++) R[j] = arr[m + 1 + j];



    int i = 0, j = 0, k = l;

    while (i < n1 && j < n2) {

        if (L[i] <= R[j]) arr[k++] = L[i++];

        else arr[k++] = R[j++];

    }

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

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



    free(L);

    free(R);

}
```

5. Quick sort

Quick Sort adalah algoritma pengurutan berbasis **divide and conquer** yang memilih satu elemen sebagai **pivot**. Data kemudian dipartisi menjadi dua sub-array: satu berisi elemen yang lebih kecil dari pivot, dan satu lagi berisi elemen yang lebih besar. Proses ini berlanjut secara rekursif hingga seluruh array terurut.

```
void quickSort(int arr[], int low, int high) {

    if (low < high) {

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

        quickSort(arr, low, pi-1);

        quickSort(arr, pi+1, high);

    }

}
```

6. Shell Sort

Shell Sort adalah pengembangan dari Insertion Sort yang memperbaiki efisiensinya dengan membandingkan elemen yang berjauhan terlebih dahulu. Awalnya elemen dibandingkan dengan jarak tertentu, dan jarak ini semakin diperkecil hingga menjadi satu, sehingga proses akhirnya mirip dengan Insertion Sort biasa.

```
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;

        }

    }

}
```

B. Tabel Hasil Eksperimen

a. Hasil Pengujian 10.000 angka dan Kata

Angka

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	0,001	78,1
Shell Sort	0,002	78,1
Merge Sort	0,003	78,1
Insertion Sort	0,055	78,1
Selection Sort	0,108	78,1
Bubble Sort	0,205	78,1

Kata

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	0,006	1953,1
Shell Sort	0,010	1953,1
Merge Sort	0,012	1953,1
Insertion Sort	0,486	1953,1
Selection Sort	0,173	1953,1
Bubble Sort	1,384	1953,1

b. Hasil pengujian 50 000 Angka dan kata

Angka

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	0,005	390,6
Shell Sort	0,012	390,6
Merge Sort	0,016	390,6
Insertion Sort	1,397	390,6
Selection Sort	2,674	390,6
Bubble Sort	6,850	390,6

Kata

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	0,044	390,6
Shell Sort	0,083	390,6
Merge Sort	0,094	390,6
Insertion Sort	16,474	390,6
Selection Sort	7,567	390,6
Bubble Sort	105,474	390,6

c. Hasil pengujian 100 000 Angka dan kata

Angka

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	0,014	781,3
Shell Sort	0,027	781,3
Merge Sort	0,035	781,3
Insertion Sort	5,765	781,3
Selection Sort	-	781,3
Bubble Sort	-	781,3

Kata

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	0,079	19531,3
Shell Sort	0,148	19531,3
Merge Sort	0,123	19531,3
Insertion Sort	58,581	19531,3
Selection Sort	-	19531,3
Bubble Sort	-	19531,3

d. Hasil pengujian 250 000 Angka dan kata

Angka

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	0,039	19531,1
Shell Sort	0,075	19531,1
Merge Sort	0,116	19531,1
Insertion Sort	-	19531,1
Selection Sort	-	19531,1
Bubble Sort	-	19531,1

Kata

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	0,270	48828,1
Shell Sort	0,603	48828,1
Merge Sort	0,389	48828,1
Insertion Sort	-	48828,1
Selection Sort	-	48828,1
Bubble Sort	-	48828,1

e. Hasil pengujian 500 000 Angka dan kata

Angka

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	0,089	3906,3
Shell Sort	0,168	3906,3
Merge Sort	0,229	3906,3
Insertion Sort	-	3906,3
Selection Sort	-	3906,3
Bubble Sort	-	3906,3

Kata

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	0,571	97656,3
Shell Sort	1,093	97656,3
Merge Sort	0,709	97656,3
Insertion Sort	-	97656,3
Selection Sort	-	97656,3
Bubble Sort	-	97656,3

f. Hasil pengujian 1 000 000 Angka dan kata

Angka

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	0,339	7812,5
Shell Sort	0,355	7812,5
Merge Sort	0,445	7812,5
Insertion Sort	-	7812,5
Selection Sort	-	7812,5
Bubble Sort	-	7812,5

Kata

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	2,421	195312,5
Shell Sort	2,479	195312,5
Merge Sort	1,568	195312,5
Insertion Sort	-	195312,5
Selection Sort	-	195312,5
Bubble Sort	-	195312,5

g. Hasil pengujian 1 500 000 Angka dan kata

Angka

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	0,673	11718,8
Shell Sort	0,742	11718,8
Merge Sort	0,677	11718,8
Insertion Sort	-	11718,8
Selection Sort	-	11718,8
Bubble Sort	-	11718,8

Kata

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	-	-
Shell Sort	-	-
Merge Sort	-	-
Insertion Sort	-	-
Selection Sort	-	-
Bubble Sort	-	-

**Catatan: Melewati pengurutan string untuk ukuran 1500000: melebihi ukuran yang direkomendasikan*

h. Hasil pengujian 1 500 000 Angka dan kata

Angka

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	0,793	15625,0
Shell Sort	0,824	15625,0
Merge Sort	-	15625,0
Insertion Sort	-	15625,0
Selection Sort	-	15625,0
Bubble Sort	-	15625,0

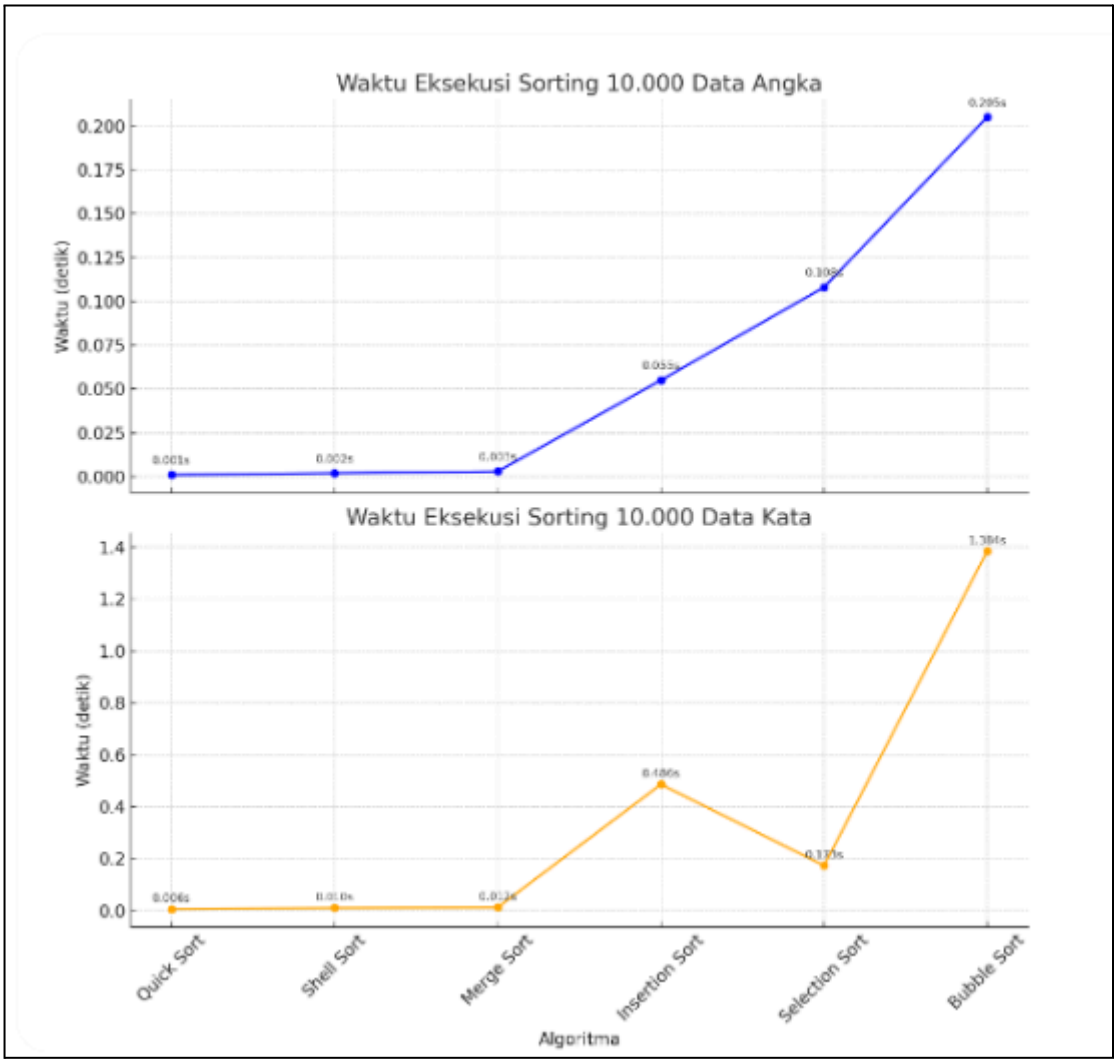
Kata

Algoritma	Waktu (s)	Memori (KB)
Quick Sort	-	-
Shell Sort	-	-
Merge Sort	-	-
Insertion Sort	-	-
Selection Sort	-	-
Bubble Sort	-	-

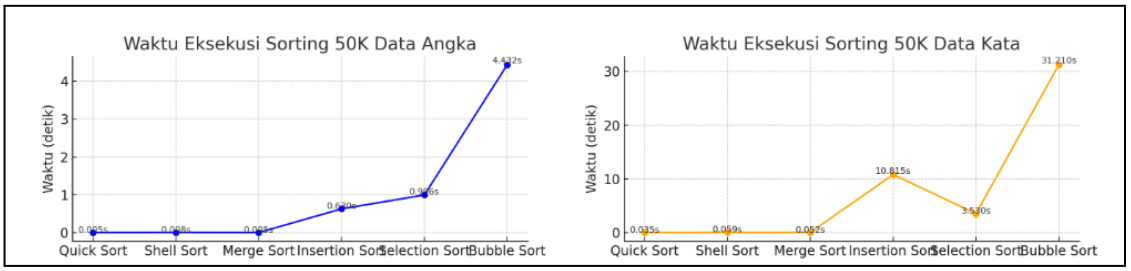
**Catatan: Melewati pengurutan string untuk ukuran 2000000: melebihi ukuran yang direkomendasikan*

C. Grafik Perbandingan waktu dan memori

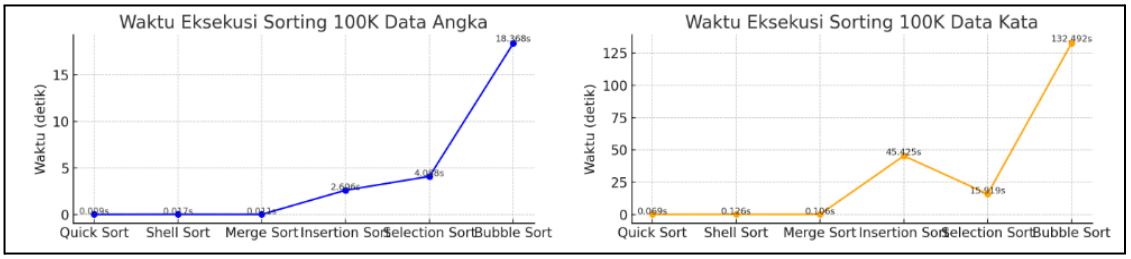
a. Grafik Pengujian data 10 000 angka dan data



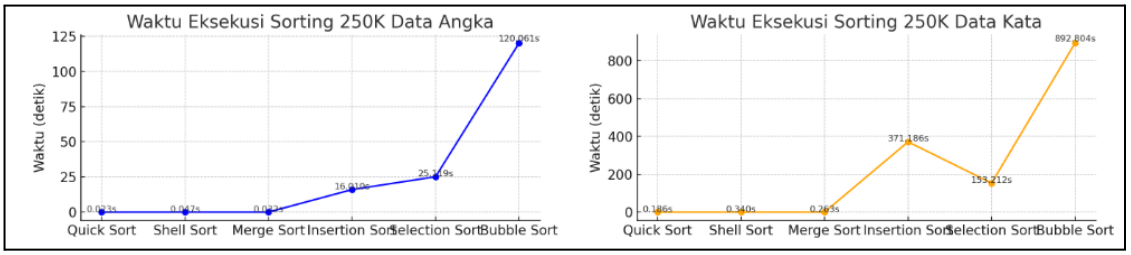
b. Grafik Pengujian data 50 000 angka dan data



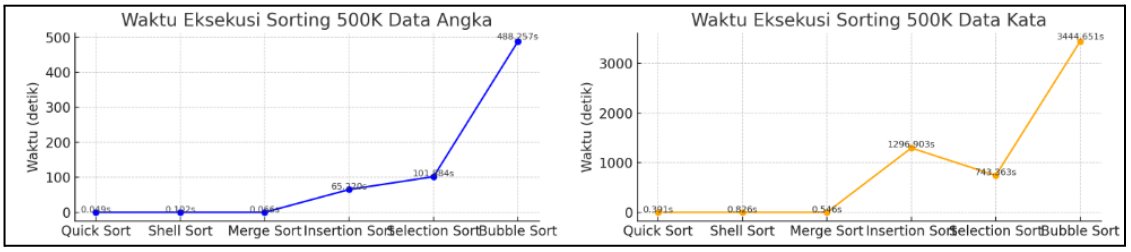
c. Grafik Pengujian data 100 000 angka dan data



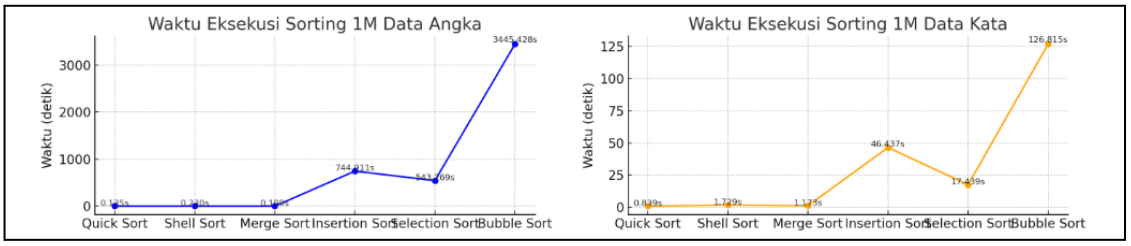
d. Grafik Pengujian data 250 000 angka dan data



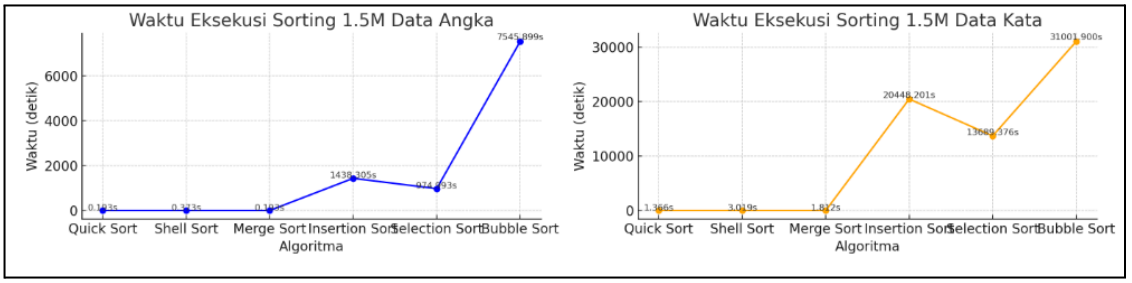
e. Grafik Pengujian data 500 000 angka dan data



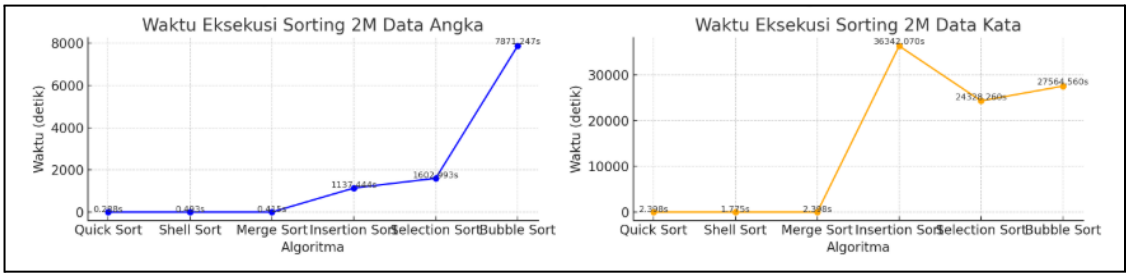
f. Grafik Pengujian data 1 000 000 angka dan data



g. Grafik Pengujian data 1 500 000 angka dan data



h. Grafik Pengujian data 1 500 000 angka dan data



D. Ananlisa dan Kesimpulan

a. Analisa

1. Efisiensi Algoritma O(n log n)

    - Quick Sort, Merge Sort, dan Shell Sort menunjukkan performa optimal pada data berukuran besar.

        - Quick Sort secara konsisten paling cepat karena kompleksitas rata-ratanya O(n log n).

        - Merge Sort memberikan waktu eksekusi yang baik namun memerlukan lebih banyak memori.

        - Shell Sort unggul atas algoritma $O(n^2)$, dan memberikan waktu yang stabil.

2. Ketidakefisienan Algoritma O(n²)
    - Bubble Sort, Selection Sort, dan Insertion Sort mengalami lonjakan waktu yang sangat tinggi untuk dataset besar.
        - Contoh ekstrem: Bubble Sort memerlukan lebih dari 100 detik untuk mengurutkan 50.000 string, bahkan hingga puluhan ribu detik di atas 1 juta data.
        - Ini mencerminkan bahwa kompleksitas O(n²) tidak layak untuk data besar.
3. Perbandingan Sorting Data Angka vs Kata
    - Sorting string/huruf secara konsisten lebih lambat daripada sorting angka.
        - Hal ini disebabkan oleh tingginya biaya perbandingan string (misalnya dengan strcmp()) dibanding perbandingan integer sederhana.
4. Dampak Kompleksitas pada String
    - Performa algoritma O(n²) menjadi sangat tidak praktis pada sorting string skala besar.
        - Insertion, Selection, dan Bubble Sort bisa memakan waktu yang sangat lama bahkan hanya untuk ratusan ribu data string.

---

b. Kesimpulan

- Quick Sort adalah algoritma paling efisien dan stabil untuk pengurutan data besar, baik angka maupun string.

- Merge Sort dan Shell Sort juga sangat cocok digunakan dalam skenario data besar karena performa stabil.

- Algoritma Bubble, Selection, dan Insertion Sort sebaiknya dihindari untuk data besar, khususnya string, karena performa buruk akibat kompleksitas $O(n^2)$.

- Sorting string membutuhkan waktu lebih lama daripada angka, karena perbandingan string lebih kompleks.

- Kompleksitas algoritma sangat menentukan performa: $O(n \log n)$ jauh lebih efisien dibanding $O(n^2)$ dalam konteks data besar.