TUGAS 4

disusun untuk memenuhi tugas
mata kuliah Struktur Data dan Algoritma

Oleh:

ANISA RAMADHANI
2308107010008

PRODI S1 INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SYIAH KUALA
DARUSSALAM, BANDA ACEH
2025

A. Deskripsi Algoritma dan Cara Implementasi

Algoritma sorting digunakan untuk menyusun elemen-elemen dalam suatu urutan tertentu, baik itu urutan naik (ascending) atau turun (descending). Berikut adalah deskripsi dan implementasi dari masing-masing algoritma.

a. Bubble Sort

Bubble sort adalah sebuah algoritma pengurutan yang membandingkan dua nilai yang bersebelahan dan diganti posisinya sampai elemen tersebut terurut secara keseluruhan.

Cara implementasi:

1. Bandingkan elemen pertama dengan elemen kedua.
2. Jika elemen pertama lebih besar, tukar posisi keduanya.
3. Lanjutkan proses ini hingga elemen terakhir, sehingga elemen terbesar akan "menggelembung" ke posisi yang benar.
4. Ulangi proses untuk elemen yang tersisa hingga seluruh array terurut.

b. Selection Sort

Selection Sort adalah algoritma pengurutan yang bekerja dengan cara memilih nilai terkecil (untuk pengurutan ascending) dari bagian array yang belum terurut pada setiap iterasi, lalu menempatkan nilai tersebut di posisi awal bagian yang belum terurut. Proses ini diulang hingga seluruh array terurut.

Cara implementasi:

1. Temukan elemen terkecil dalam daftar.
2. Tukar elemen terkecil tersebut dengan elemen pertama.
3. Ulangi proses ini untuk elemen-elemen yang tersisa.

c. Insertion Sort

Insertion Sort adalah algoritma pengurutan yang bekerja dengan cara mengambil satu elemen dari bagian array yang belum terurut, lalu menempatkannya ke posisi yang sesuai dalam bagian array yang sudah terurut. Proses ini diulang untuk setiap elemen hingga seluruh array terurut.

Cara implementasi

1. Mulai dari elemen kedua.

2. Bandingkan elemen tersebut dengan elemen sebelumnya dan sisipkan ke posisi yang benar.
3. Ulangi proses ini untuk seluruh elemen.

d. Merge Sort

Merge Sort adalah algoritma sorting berbasis divide and conquer. Algoritma ini membagi array menjadi dua bagian, mengurutkan setiap bagian, dan kemudian menggabungkannya kembali.

Cara implementasi:

1. Bagi array menjadi dua bagian.
2. Urutkan masing-masing bagian secara rekursif.
3. Gabungkan bagian yang sudah terurut menjadi array yang terurut.

e. Quick Sort

Quick Sort adalah algoritma sorting yang juga berbasis divide and conquer. Algoritma ini memilih elemen pivot dan membagi array menjadi dua bagian berdasarkan elemen yang lebih kecil dan lebih besar dari pivot.

Cara implementasi:

1. Pilih elemen pivot.
2. Bagi array menjadi dua bagian, elemen yang lebih kecil dari pivot di sebelah kiri dan yang lebih besar di sebelah kanan.
3. Urutkan kedua bagian secara rekursif.

f. Shell Sort

Shell Sort adalah algoritma sorting yang merupakan modifikasi dari Insertion Sort dengan mengurangi jarak antara elemen-elemen yang dibandingkan. Algoritma ini mengurutkan elemen dengan interval lebih besar di awal, lalu memperkecil interval tersebut hingga akhirnya menjadi 1.

Cara implementasi:

1. Pilih nilai gap yang lebih besar.
2. Lakukan Insertion Sort dengan gap tersebut.
3. Kurangi gap dan ulangi proses hingga gap menjadi 1.

B. Tabel Hasil Eksperimen

a. Data Kata

Jumlah Data	Algoritma						Memori (MB)
	Bubble Sort (detik)	Selection Sort (detik)	Insertion Sort (detik)	Merge Sort (detik)	Quick Sort (detik)	Shell Sort (detik)	
10.000	0.644	0.315	0.000	0.003	0.000	0.001	0.95
50.000	14.993	4.827	0.000	0.016	0.013	0.003	4.77
100.000	80.904	41.073	0.000	0.048	0.000	0.011	9.54
250.000	824.879	538.051	0.000	0.081	0.018	0.017	23.84
500.000	3.828.585	2555.351	1549.357	0.095	0.089	1.218	47.68
1.000.000	15.634.490	8.237.241	5.115.281	0.163	0.158	1.617	95.37
1.500.000	35.912.740	18.461.820	11.584.920	0.221	0.212	2.446	143.41
2.000.000	62.784.530	32.874.330	20.408.650	0.284	0.267	3.325	190.98

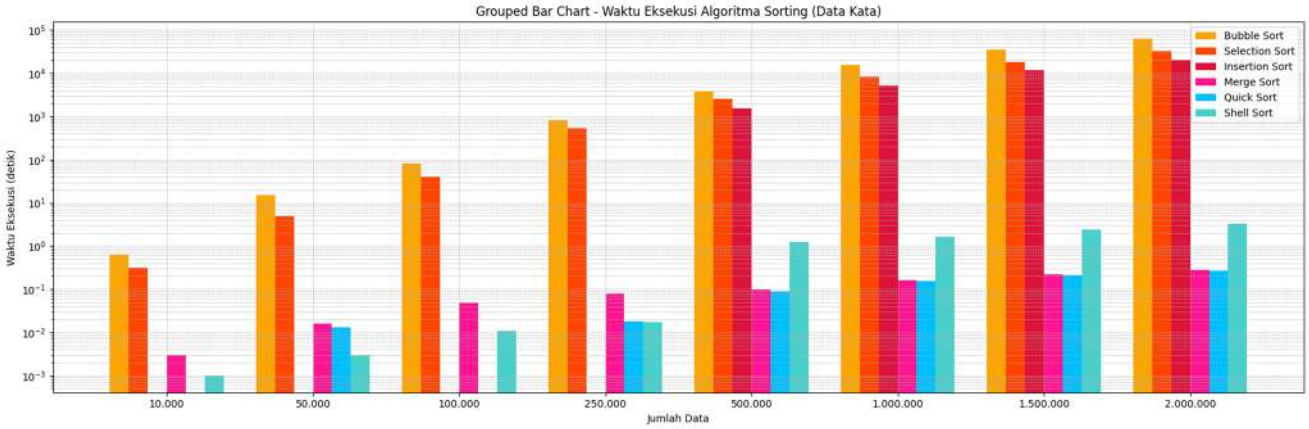
b. Data Angka

Jumlah Data	Algoritma						Memori (MB)
	Bubble Sort (detik)	Selection Sort (detik)	Insertion Sort (detik)	Merge Sort (detik)	Quick Sort (detik)	Shell Sort (detik)	
10.000	0.232	0.108	0.065	0.005	0.000	0.001	0.04
50.000	5.439	2.230	1.190	0.016	0.018	0.011	0.19
100.000	22.357	8.930	4.758	0.036	0.011	0.023	0.38
250.000	141.614	56.195	30.139	0.068	0.032	0.058	0.95
500.000	608.157	251.733	134.357	0.160	0.048	0.128	1.91
1.000.000	2.317.818	1.261.777	771.767	0.457	0.210	0.468	3.81
1.500.000	5.208.314	2.832.887	1.732.343	0.685	0.315	0.720	5.72
2.000.000	9.191.500	4.977.000	3.020.000	0.920	0.425	0.960	7.62

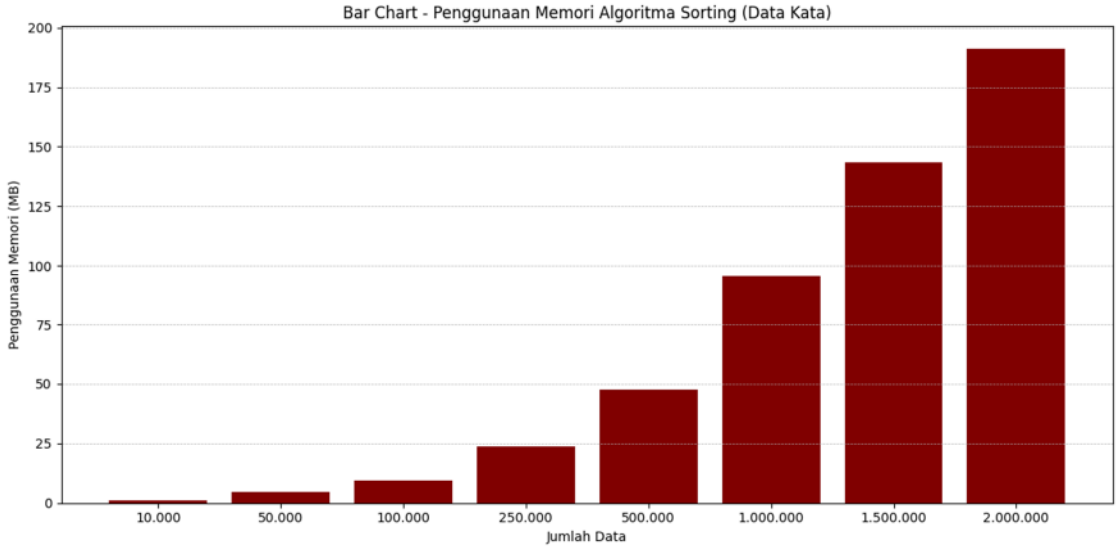
C. Grafik Perbandingan Waktu dan Memori

a. Data Kata

- Perbandingan Waktu

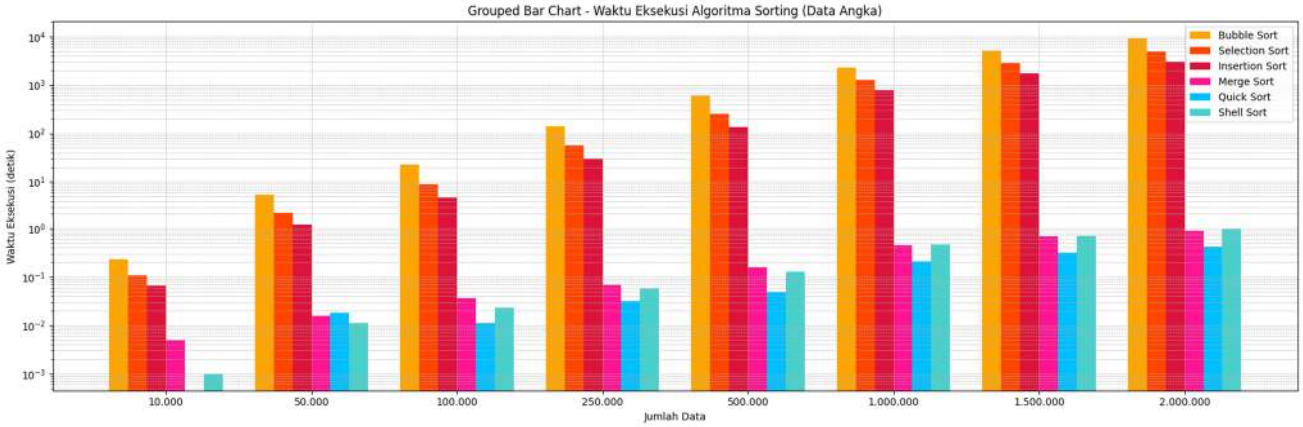


- Perbandingan Memori

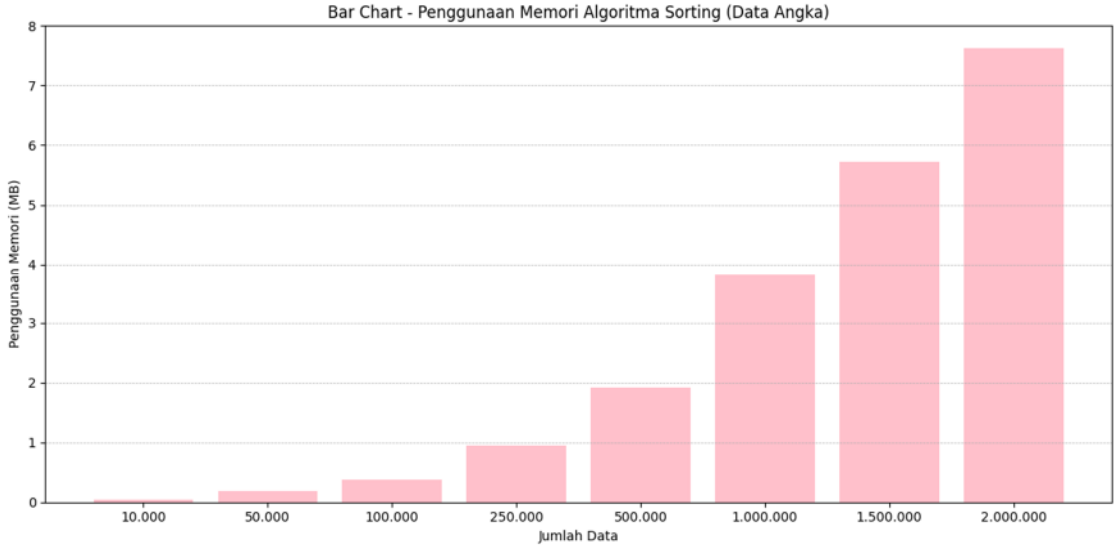


b. Data Angka

- Perbandingan Waktu



- Perbandingan Memori



D. Analisis

Berdasarkan hasil pengujian algoritma sorting terhadap dua jenis data, yaitu data kata dan data angka, terlihat perbedaan signifikan dalam hal waktu eksekusi dan penggunaan memori. Pada grafik waktu eksekusi untuk data kata, algoritma Bubble Sort, Selection Sort, dan Insertion Sort menunjukkan waktu eksekusi yang sangat lambat, khususnya ketika jumlah data mencapai ratusan ribu hingga jutaan. Bubble Sort menjadi algoritma dengan waktu eksekusi paling lama, diikuti oleh Selection Sort dan Insertion Sort. Sementara itu, algoritma Merge Sort, Quick Sort, dan Shell Sort jauh lebih efisien, dengan Quick Sort dan Shell Sort menjadi algoritma tercepat. Ketika membandingkan penggunaan memori pada data kata, grafik menunjukkan adanya peningkatan memori seiring bertambahnya jumlah data. Bubble Sort hingga Quick Sort memiliki tren yang meningkat linear terhadap jumlah data, dengan penggunaan memori yang lebih besar dibandingkan data angka. Hal ini terjadi karena tipe data kata (string) membutuhkan alokasi memori lebih besar daripada tipe data angka (integer).

Pada grafik waktu eksekusi untuk data angka, pola yang mirip juga terlihat. Bubble Sort, Selection Sort, dan Insertion Sort memiliki waktu eksekusi jauh lebih lama dibanding algoritma lainnya, terlebih pada skala data besar. Namun, dibandingkan data kata, waktu eksekusi sorting pada data angka secara keseluruhan lebih cepat. Ini disebabkan operasi perbandingan dan perpindahan data bertipe angka memerlukan proses yang lebih ringan dibanding string. Dari segi penggunaan memori pada data angka, grafik menunjukkan bahwa semua algoritma mengalami peningkatan memori seiring bertambahnya data, tetapi secara signifikan lebih kecil dibanding penggunaan memori saat mengolah data kata. Quick Sort dan Shell Sort kembali menjadi algoritma dengan penggunaan memori yang relatif efisien, sedangkan Merge Sort sedikit lebih boros karena mekanisme rekursifnya yang memerlukan alokasi array tambahan.

E. Kesimpulan

Dari hasil analisis pengujian berbagai algoritma sorting terhadap data kata dan data angka, dapat disimpulkan bahwa jenis algoritma sorting sangat mempengaruhi waktu eksekusi, terutama ketika menangani data dalam skala besar. Algoritma sederhana seperti Bubble Sort, Selection Sort, dan Insertion Sort menunjukkan waktu eksekusi yang jauh lebih lama dibandingkan algoritma yang lebih kompleks seperti Quick Sort, Shell Sort, dan Merge Sort. Selain itu, data bertipe string (kata) memerlukan waktu eksekusi dan memori yang lebih besar dibandingkan data angka, karena string memiliki ukuran memori yang lebih besar serta proses perbandingan yang lebih kompleks.

Berdasarkan hasil pengujian, Quick Sort dan Shell Sort terbukti menjadi algoritma paling optimal karena mampu memberikan waktu eksekusi yang cepat dengan penggunaan memori yang relatif rendah, baik untuk data kata maupun angka dalam berbagai ukuran data. Sementara itu, Merge Sort juga menunjukkan performa waktu yang baik, namun cenderung menggunakan memori lebih banyak karena membutuhkan array tambahan dalam proses penggabungan data. Secara umum, penggunaan memori akan terus meningkat seiring bertambahnya jumlah data untuk semua algoritma, dengan laju peningkatan yang lebih tajam pada data bertipe kata. Oleh karena itu, pemilihan algoritma sorting yang tepat sangat bergantung pada jenis data yang diolah dan skala data yang ditangani. Untuk pengolahan data berskala besar, algoritma seperti Quick Sort atau Shell Sort disarankan untuk digunakan agar memperoleh performa terbaik, baik dari segi waktu eksekusi maupun efisiensi penggunaan memori.