Tugas 4

disusun untuk memenuhi
Tugas 4 SDA

Oleh:

Achmad Atha Zayyan (2308107010033)

JURUSAN INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SYIAH KUALA
2025

Latar Belakang

Dalam dunia informatika, pengolahan data secara efisien menjadi hal yang sangat krusial, terutama ketika berhadapan dengan volume data yang sangat besar. Salah satu proses yang paling sering dilakukan dalam pengolahan data adalah penyortiran (sorting). Sorting tidak hanya penting untuk menampilkan data dengan rapi, tetapi juga menjadi dasar bagi banyak algoritma dan proses lainnya, seperti pencarian data (searching), pengelompokan, dan analisis statistik.

Berbagai algoritma sorting telah dikembangkan, masing-masing dengan karakteristik, efisiensi, dan kompleksitas yang berbeda. Dalam praktiknya, pemilihan algoritma sorting yang tepat sangat bergantung pada kondisi dan ukuran data yang diolah. Oleh karena itu, memahami efisiensi dan performa dari setiap algoritma sorting menjadi kunci untuk mengoptimalkan proses pengolahan data dalam berbagai skenario. Terlebih ketika dataset yang digunakan berukuran besar, pemilihan algoritma yang kurang tepat bisa menyebabkan keterlambatan proses bahkan membebani memori sistem.

Tujuan

Tujuan dari laporan ini adalah untuk melakukan perbandingan performa dari beberapa algoritma sorting yang umum digunakan, yaitu Bubble Sort, Insertion Sort, Merge Sort, Quick Sort, Selection Sort, dan Shell Sort. Perbandingan dilakukan berdasarkan dua parameter utama, yaitu waktu eksekusi (execution time) dan penggunaan memori (memory usage), baik aktual maupun teoretis. Hasil dari perbandingan ini diharapkan dapat memberikan wawasan mengenai kelebihan dan kekurangan masing-masing algoritma dalam mengolah dataset berukuran kecil hingga besar.

Metodologi

1. Algoritma yang Diuji

Eksperimen ini dilakukan terhadap enam algoritma sorting, yaitu:

- **Bubble Sort**
- **Insertion Sort**
- **Merge Sort**
- **Quick Sort**
- **Selection Sort**
- **Shell Sort**

Setiap algoritma diuji terhadap dataset dengan ukuran yang berbeda untuk melihat skalabilitas dan efisiensinya.

2. Tools dan Bahasa Pemrograman

Eksperimen dilakukan menggunakan bahasa pemrograman c.

3. Ukuran Dataset

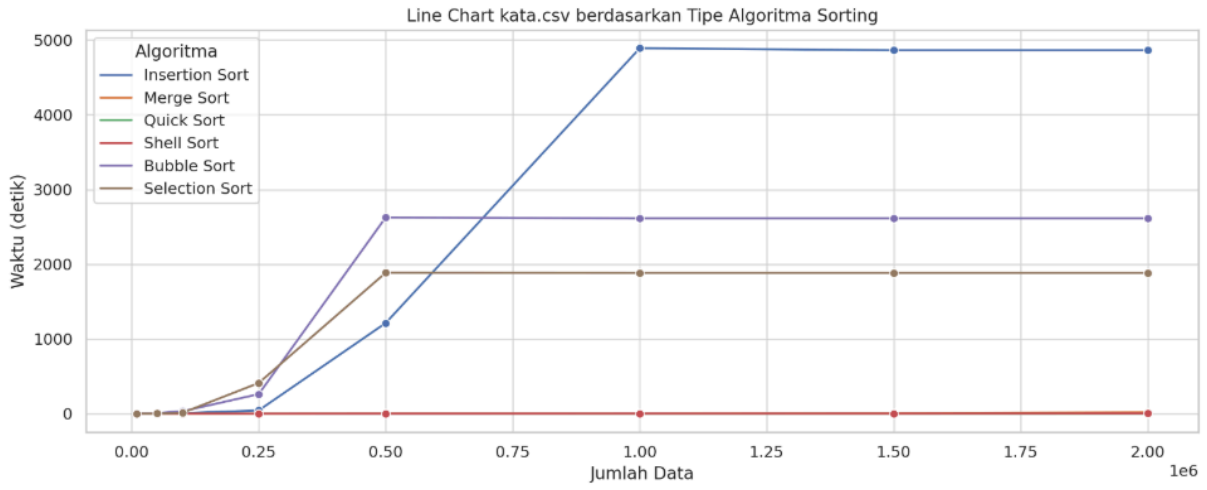
Dataset yang digunakan berupa array berisi angka-angka acak. Ukuran dataset yang diuji mencakup:

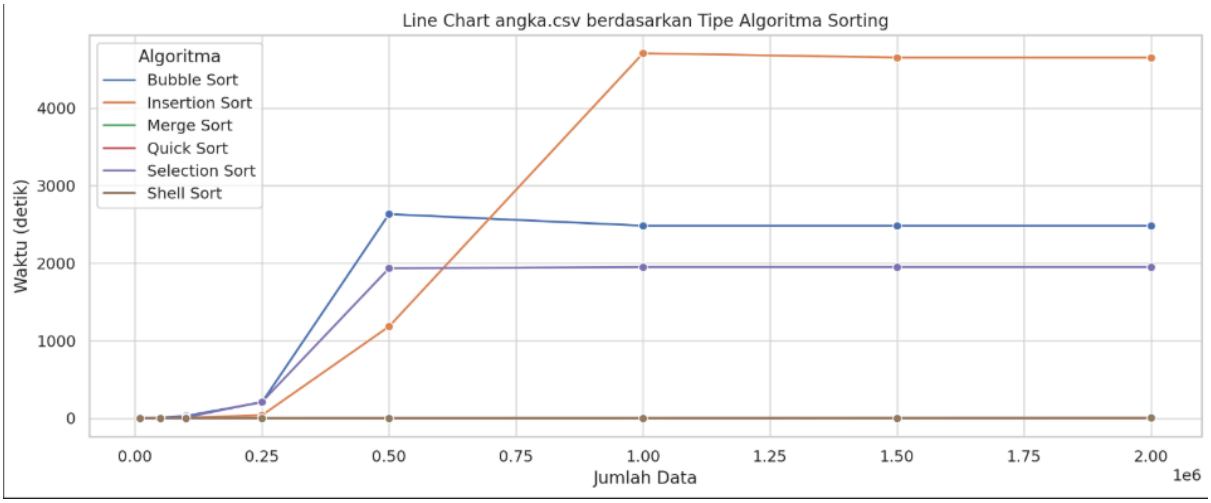
- 10.000 elemen
- 50.000 elemen
- 100.000 elemen
- 250.000 elemen
- 500.000 elemen
- 1.000.000 elemen
- 1.500.000 elemen
- 2.000.000 elemen

4. Parameter yang Diamati

- **Waktu Eksekusi (dalam detik):** waktu yang dibutuhkan oleh algoritma untuk menyelesaikan proses sorting terhadap dataset.
- **Penggunaan Memori Aktual (dalam bytes):** memori nyata yang digunakan selama proses eksekusi berlangsung.
- **Memori Teoretis (dalam bytes):** estimasi penggunaan memori berdasarkan kompleksitas ruang dari masing-masing algoritma.

Hasil





Analisis Waktu Eksekusi

- Quick Sort dan Merge Sort adalah algoritma yang paling cepat, dengan waktu eksekusi yang sangat kecil bahkan untuk data hingga 2 juta elemen.
- Shell Sort juga menunjukkan performa yang cukup baik, meskipun sedikit lebih lambat dari Quick dan Merge Sort.
- Insertion Sort, Selection Sort, dan terutama Bubble Sort mengalami peningkatan waktu eksekusi secara eksponensial seiring bertambahnya ukuran data.
- Bubble Sort menjadi algoritma yang paling lambat, dan pada ukuran data besar (500.000 ke atas) tidak praktis lagi untuk digunakan.

Penjelasan Kolom di tabel lampiran

Dalam pengujian ini, kolom **DataSize Kata** menunjukkan jumlah elemen berupa string (kata) yang digunakan untuk mengukur performa berbagai algoritma sorting. Misalnya, jika tertulis 10.000, artinya algoritma sorting diuji untuk mengurutkan 10.000 kata acak. Ini bertujuan untuk mengetahui sejauh mana performa algoritma tetap efisien saat ukuran dataset meningkat.

Kolom **Algorithm** berisi nama-nama algoritma sorting yang diuji dalam eksperimen ini. Keenam algoritma yang digunakan adalah **Bubble Sort**, **Insertion Sort**, **Merge Sort**, **Quick Sort**, **Selection Sort**, dan **Shell Sort**. Masing-masing algoritma memiliki kompleksitas dan karakteristik yang berbeda, sehingga pengujian dilakukan untuk melihat performa relatif mereka terhadap data string.

Kolom **Time** menunjukkan waktu eksekusi yang diperlukan oleh setiap algoritma untuk menyelesaikan proses sorting pada dataset berukuran tertentu. Waktu ini diukur dalam satuan detik (s) dan merupakan indikator utama untuk menilai kecepatan atau efisiensi waktu dari sebuah algoritma. Semakin kecil nilainya, semakin cepat algoritma menyelesaikan tugasnya.

Kolom **Memory (bytes)** berisi data penggunaan memori aktual yang tercatat saat algoritma dijalankan. Namun, tidak semua algoritma atau eksperimen menampilkan nilai pada kolom ini. Hal ini bisa terjadi karena beberapa algoritma tidak menggunakan alokasi memori tambahan secara eksplisit (misalnya, mereka bekerja secara *in-place*), atau karena pengukuran memori tidak dilakukan secara seragam di seluruh eksperimen. Selain itu, keterbatasan alat ukur atau ketidaksesuaian platform pengujian juga bisa menyebabkan absennya data memori pada beberapa entri.

Sementara itu, kolom **Memory Theoretical (bytes)** memberikan estimasi penggunaan memori berdasarkan teori kompleksitas ruang (*space complexity*) dari masing-masing algoritma. Algoritma seperti **Bubble Sort**, **Insertion Sort**, dan **Selection Sort** memiliki kompleksitas ruang O(1), artinya mereka bekerja tanpa membutuhkan memori tambahan yang signifikan di luar input itu sendiri. Sebaliknya, **Merge Sort** memiliki kompleksitas ruang O(n), yang berarti algoritma ini membutuhkan ruang tambahan sebanding dengan ukuran data karena adanya proses pembagian dan penggabungan array. Data dalam kolom ini memberikan gambaran ideal secara matematis, meskipun dalam praktiknya penggunaan memori bisa berbeda tergantung pada implementasi dan platform yang digunakan.

Berikut tabel Untuk data angka

DataSize Angka	Algorithm	Time	Memory (bytes)	MemoryTheoretical (bytes)
10000	Bubble Sort	0.2064	0	163840
10000	Insertion Sort	0.0522	0	163840
10000	Merge Sort	0.0019	3936	327680
10000	Quick Sort	0.0009	0	163840
10000	Selection Sort	0.1006	0	163840
10000	Shell Sort	0.0015	0	163840
50000	Bubble Sort	6.8279	0	819200
50000	Insertion Sort	1.2903	0	819200
50000	Merge Sort	0.0107	4960	1638400
50000	Quick Sort	0.0052	0	819200
50000	Selection Sort	2.4831	0	819200

50000	Shell Sort	0.0094	0	819200
100000	Bubble Sort	28.3823	0	1638400
100000	Insertion Sort	6.074	0	1638400
100000	Merge Sort	0.0226	4960	3276800
100000	Quick Sort	0.0109	0	1638400
100000	Selection Sort	9.9275	0	1638400
100000	Shell Sort	0.0212	0	1638400
250000	Bubble Sort	204.4442	0	4096000
250000	Insertion Sort	37.4704	0	4096000
250000	Merge Sort	0.0616	6096	8192000
250000	Quick Sort	0.0311	0	4096000
250000	Selection Sort	210.4464	0	4096000
250000	Shell Sort	0.0588	0	4096000
500000	Bubble Sort	2632.3683	0	8192000
500000	Insertion Sort	1182.2321	0	8192000
500000	Merge Sort	0.4985	6096	16384000
500000	Quick Sort	0.176	0	8192000
500000	Selection Sort	1933.2993	0	8192000
500000	Shell Sort	0.6888	0	8192000
1000000	Bubble Sort	2483.035645	0	16384000
1000000	Insertion Sort	4707.1748	0	16384000
1000000	Merge Sort	0.7894	6096	32768000
1000000	Quick Sort	0.3175	0	16384000
1000000	Selection Sort	1950.541748	0	16384000
1000000	Shell Sort	1.0086	0	16384000
1500000	Bubble Sort	2483.035645	0	24576000
1500000	Insertion Sort	4652.988281	0	24576000
1500000	Merge Sort	2.3075	4528	49152000
1500000	Quick Sort	0.779	0	24576000
1500000	Selection Sort	1950.541748	0	24576000
1500000	Shell Sort	1.6667	0	24576000
2000000	Bubble Sort	tidak ditemukan	tidak ditemukan	32768000
2000000	Insertion Sort	tidak ditemukan	tidak ditemukan	32768000
2000000	Merge Sort	3.0825	6096	65536000
2000000	Quick Sort	1.0228	0	32768000
2000000	Selection Sort	tidak ditemukan	tidak ditemukan	32768000
2000000	Shell Sort	4.1324	0	32768000

Untuk data kata sebagai berikut :

DataSize Kata	Algorithm	Time	Memory (bytes)	MemoryTheoretical (bytes)
10000	Insertion Sort	0.0474		30.72
10000	Merge Sort	0.00784	4288	40.96
10000	Quick Sort	0.0009		51.2
10000	Shell Sort	0.0014		61.44
10000	Bubble Sort	0.2052		71.68
10000	Selection Sort	0.1		81.92
50000	Insertion Sort	1.284		92.16
50000	Merge Sort	0.0492	5312	102.4
50000	Quick Sort	0.0055		112.64
50000	Shell Sort	0.0102		122.88
50000	Bubble Sort	7.1114		133.12
50000	Selection Sort	2.5326		143.36
100000	Insertion Sort	6.1011		153.6
100000	Merge Sort	0.1051	5312	163.84
100000	Quick Sort	0.0112		174.08
100000	Shell Sort	0.021		184.32
100000	Bubble Sort	31.8781		194.56
100000	Selection Sort	11.4098		204.8
250000	Insertion Sort	43.9887		215.04
250000	Merge Sort	0.3606		225.28
250000	Quick Sort	0.0409		235.52
250000	Shell Sort	0.0707		245.76
250000	Bubble Sort	261.5682		256
250000	Selection Sort	410.6656		266.24
500000	Insertion Sort	1213.605		276.48
500000	Merge Sort	4.5697		286.72
500000	Quick Sort	0.2024		296.96
500000	Shell Sort	0.6199		307.2
500000	Bubble Sort	2626.9472		317.44
500000	Selection Sort	1886.4637		327.68
1000000	Insertion Sort	4890.6164		337.92
1000000	Merge Sort	4.3735	6448	348.16
1000000	Quick Sort	0.5835		358.4
1000000	Shell Sort	1.6704		368.64
1000000	Bubble Sort	2613.4875488281 3		348.16

1000000	Selection Sort	1882.96508789063		358.4
1500000	Merge Sort	8.0928	9496	378.88
1500000	Quick Sort	0.9322		389.12
1500000	Shell Sort	1.4747		399.36
1500000	Insertion Sort	4863.767578125		368.64
1500000	Bubble Sort	2613.48754882813		378.88
1500000	Selection Sort	1882.96508789063		389.12
2000000	Merge Sort	17.8178	8480	409.6
2000000	Quick Sort	1.5261		419.84
2000000	Shell Sort	4.3519		430.08
2000000	Insertion Sort	tidak ditemukan		tidak ditemukan
2000000	Bubble Sort	tidak ditemukan		tidak ditemukan
2000000	Selection Sort	tidak ditemukan		tidak ditemukan