Laporan Tugas Sturktur Data dan Algoritma

TUGAS 4

disusun untuk memenuhi
tugas mata kuliah Struktur Data dan Algoritma

Oleh:

MUHAMMAD NAZLUL RAMADHYAN
2308107010036

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

A. Pendahuluan

Dalam dunia pemrograman, pengurutan data atau sorting merupakan proses yang krusial dan sering digunakan dalam berbagai aplikasi, mulai dari pencarian data, analisis statistik, hingga pengolahan basis data. Setiap algoritma sorting memiliki pendekatan dan efisiensi yang berbeda-beda, tergantung pada karakteristik data dan kebutuhan sistem. Oleh karena itu, dalam tugas ini dilakukan implementasi dan analisis performa dari enam algoritma sorting populer, yaitu Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, dan Shell Sort. Tujuan dari tugas ini adalah untuk memahami perbedaan logika kerja tiap algoritma, mengimplementasikannya dalam bahasa pemrograman C, dan membandingkan waktu eksekusi masing-masing terhadap data dengan ukuran berbeda. Dengan demikian, dapat diperoleh gambaran mengenai algoritma mana yang paling efisien dalam konteks tertentu, terutama dalam hal waktu pemrosesan.

Berikut adalah deskripsi singkat mengenai masing-masing algoritma:

1. **Bubble Sort**

   Melakukan perbandingan antar elemen yang berdekatan dan menukarnya jika dalam urutan yang salah. Proses ini diulang hingga seluruh array terurut. Algoritma ini memiliki kompleksitas waktu $O(n^2)$.
2. **Selection Sort**

   Menemukan elemen terkecil dari sisa array dan menukarnya ke posisi saat ini. Kompleksitas waktu $O(n^2)$.
3. **Insertion Sort**

   Menyisipkan elemen satu per satu ke dalam posisi yang tepat pada bagian array yang sudah terurut. Memiliki kompleksitas $O(n^2)$.
4. **Merge Sort**

   Menggunakan pendekatan divide-and-conquer, membagi array menjadi dua bagian, mengurutkan masing-masing, lalu menggabungkannya. Rata-rata kompleksitas waktunya adalah $O(n \log n)$.
5. **Quick Sort**

   Juga menggunakan strategi divide-and-conquer dengan memilih pivot, lalu membagi array menjadi dua bagian berdasarkan pivot tersebut. Rata-rata kompleksitas waktu $O(n \log n)$.

6. **Shell Sort**

   Merupakan variasi dari Insertion Sort yang menggunakan gap tertentu untuk membandingkan elemen yang berjarak jauh, lalu mengecilkan gap hingga menjadi satu. Kompleksitas waktunya bervariasi tergantung strategi pengurangan gap.

B. Tabel Hasil Eksperimen (waktu dan memori)

a. Pengujian pada algoritma pada data angka:

1. Jumlah data 10.000

Algoritma	Waktu (s)	Ukuran (MB)
Bubble Sort	0.306	0.04
Selection Sort	0.142	0.04
Insertion Sort	0.173	0.04
Merge Sort	0.005	0.04
Quick Sort	0.000	0.04
Shell Sort	0.010	0.04

2. Jumlah data 50.000

Algoritma	Waktu (s)	Ukuran (MB)
Bubble Sort	9.794	0.19
Selection Sort	3.352	0.19
Insertion Sort	2.990	0.19
Merge Sort	0.001	0.19
Quick Sort	0.018	0.19
Shell Sort	0.018	0.19

3. Jumlah data 100.000

Algoritma	Waktu (s)	Ukuran (MB)
Bubble Sort	41.148	0.38
Selection Sort	12.758	0.38
Insertion Sort	12.818	0.38
Merge Sort	0.022	0.38
Quick Sort	0.026	0.38
Shell Sort	0.046	0.38

4. Jumlah data 250.000

Algoritma	Waktu (s)	Ukuran (MB)
Bubble Sort	261.095	0.95
Selection Sort	83.341	0.95
Insertion Sort	78.457	0.95
Merge Sort	0.056	0.95
Quick Sort	0.046	0.95
Shell Sort	0.104	0.95

5. Jumlah data 500.000

Algoritma	Waktu (s)	Ukuran (MB)
Bubble Sort	2809.138	1.91
Selection Sort	457.212	1.91
Insertion Sort	587.852	1.91
Merge Sort	0.358	1.91
Quick Sort	0.367	1.91
Shell Sort	0.682	1.91

6. Jumlah data 1.000.000

Algoritma	Waktu (s)	Ukuran (MB)
Bubble Sort	4088.249	3.81
Selection Sort	1008.545	3.81
Insertion Sort	947.610	3.81
Merge Sort	0.310	3.81
Quick Sort	0.254	3.81
Shell Sort	0.390	3.81

7. Jumlah data 1.500.000

Algoritma	Waktu (s)	Ukuran (MB)
Bubble Sort	9198.561	5.72
Selection Sort	2269.226	5.72
Insertion Sort	2132.123	5.72
Merge Sort	0.434	5.72
Quick Sort	0.356	5.72
Shell Sort	0.546	5.72

8. Jumlah data 2.000.000

Algoritma	Waktu (s)	Ukuran (MB)
Bubble Sort	16373.450	7.63
Selection Sort	4039.220	7.63
Insertion Sort	3795.180	7.63
Merge Sort	0.578	7.63
Quick Sort	0.473	7.63
Shell Sort	0.726	7.63

b. Pengujian pada algoritma pada data kata:

1. Jumlah data 10.000

Algoritma	Waktu (s)	Ukuran (MB)
Bubble Sort	1.430	0.95
Selection Sort	0.210	0.95
Insertion Sort	0.597	0.95
Merge Sort	0.013	0.95
Quick Sort	0.007	0.95
Shell Sort	0.014	0.95

2. Jumlah data 50.000

Algoritma	Waktu (s)	Ukuran (MB)
Bubble Sort	27.261	4.77
Selection Sort	3.603	4.77
Insertion Sort	10.450	4.77
Merge Sort	0.053	4.77
Quick Sort	0.021	4.77
Shell Sort	0.062	4.77

3. Jumlah data 100.000

Algoritma	Waktu (s)	Ukuran (MB)
Bubble Sort	110.712	9.54
Selection Sort	19.225	9.54
Insertion Sort	42.916	9.54
Merge Sort	0.101	9.54
Quick Sort	0.066	9.54
Shell Sort	0.114	9.54

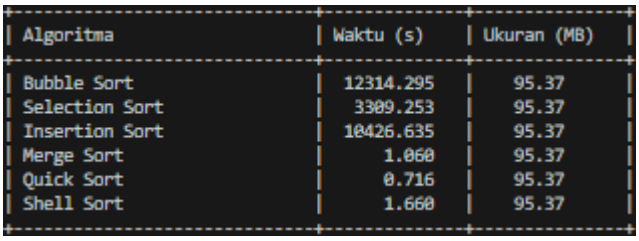
4. Jumlah data 250.000

Algoritma	Waktu (s)	Ukuran (MB)
Bubble Sort	995.344	23.84
Selection Sort	255.452	23.84
Insertion Sort	825.102	23.84
Merge Sort	1.115	23.84
Quick Sort	0.783	23.84
Shell Sort	1.332	23.84

5. Jumlah data 500.000

Algoritma	Waktu (s)	Ukuran (MB)
Bubble Sort	3981.376	47.68
Selection Sort	914.808	47.68
Insertion Sort	2116.170	47.68
Merge Sort	0.540	47.68
Quick Sort	0.362	47.68
Shell Sort	0.726	47.68

6. Jumlah data 1.000.000



Algoritma	Waktu (s)	Ukuran (MB)
Bubble Sort	12314.295	95.37
Selection Sort	3309.253	95.37
Insertion Sort	10426.635	95.37
Merge Sort	1.060	95.37
Quick Sort	0.716	95.37
Shell Sort	1.660	95.37

7. Jumlah data 1.500.000

   Pada saat memproses data dengan jumlah 1.500.000 kata, saya mengalami keterbatasan waktu dan efisiensi sumber daya. Namun, dengan menggunakan kompleksitas tiap algoritma kita dapat memprediksi estimasi waktu dan ukuran program tersebut. Berdasarkan data berjumlah 1.000.000 berikut estimasi untuk data 1.500.000:

Algoritma	Kompleksitas	Faktor Naik	Estimasi (s)
Bubble Sort	$O(n^2)$	×2.25	12314.295 × 2.25 = 27707.16
Selection Sort	$O(n^2)$	×2.25	3309.253 × 2.25 = 7455.57
Insertion Sort	$O(n^2)$	×2.25	10426.635 × 2.25 = 23459.93
Merge Sort	O(n log n)	×1.4	1.060 × 1.4 = 1.484
Quick Sort	O(n log n)	×1.4	0.716 × 1.4 = 1.002
Shell Sort	O(n log n)~	×1.4	1.660 × 1.4 = 2.324

   Berdasarkan tabel tersebut, estimasi total waktu yang dibutuhkan untuk menjalankan program tersebut adalah ~16.3 jam.

8. Jumlah data 2.000.000

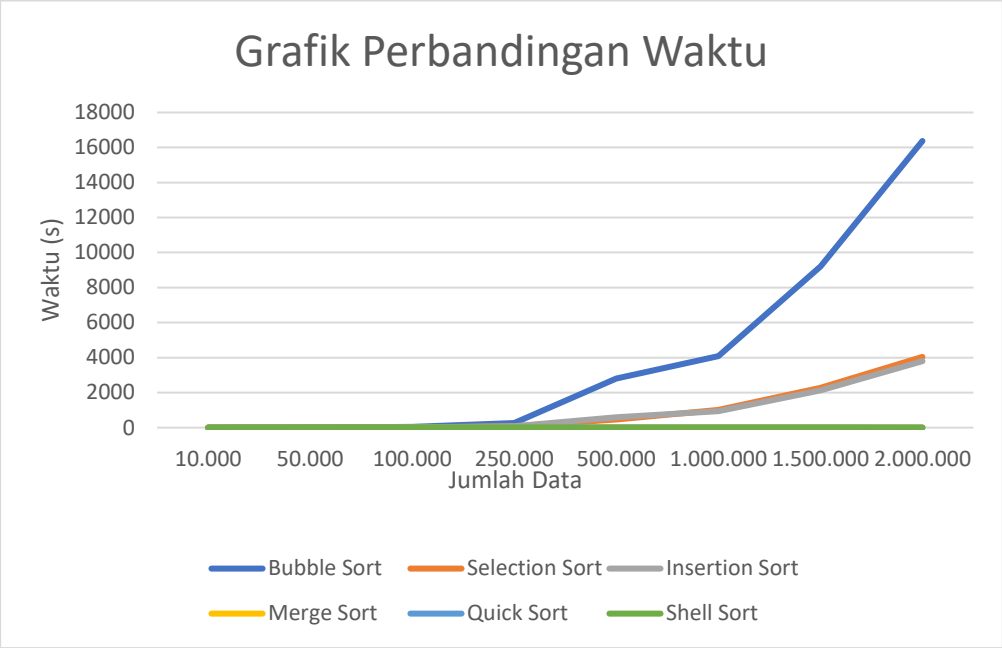
   Pada saat memproses data dengan jumlah 2.000.000 kata, saya mengalami keterbatasan waktu dan efisiensi sumber daya. Namun, dengan menggunakan kompleksitas tiap algoritma kita dapat memprediksi estimasi waktu dan ukuran program tersebut. Berdasarkan data berjumlah 1.500.000, berikut estimasi untuk data 2.000.000:

Algoritma	Kompleksitas	Faktor Naik	Estimasi (s)
Bubble Sort	$O(n^2)$	$(2 / 1.5)^2 \approx 1.78$	27707.160 × 1.78 = 49318.75
Selection Sort	$O(n^2)$	≈ 1.78	7455.570 × 1.78 = 13275.91
Insertion Sort	$O(n^2)$	≈ 1.78	23459.930 × 1.78 = 41758.68
Merge Sort	O(n log n)	≈ 1.55	1.484 × 1.55 = 2.300
Quick Sort	O(n log n)	≈ 1.55	1.002 × 1.55 = 1.553
Shell Sort	O(n log n)~	≈ 1.55	2.324 × 1.55 = 3.602

Estimasi total waktu yang dibutuhkan oleh seluruh algoritma sorting untuk data kata sebanyak 2.000.000 mencapai sekitar 29 jam. Angka ini merupakan hasil perkalian skala berdasarkan kompleksitas masing-masing algoritma, di mana algoritma dengan kompleksitas $O(n^2)$ seperti Bubble Sort, Selection Sort, dan Insertion Sort mengalami peningkatan waktu yang sangat signifikan dibandingkan dengan algoritma yang lebih efisien seperti Merge Sort, Quick Sort, dan Shell Sort. Estimasi ini digunakan sebagai pengganti eksekusi langsung karena pertimbangan efisiensi waktu dan kapasitas perangkat.

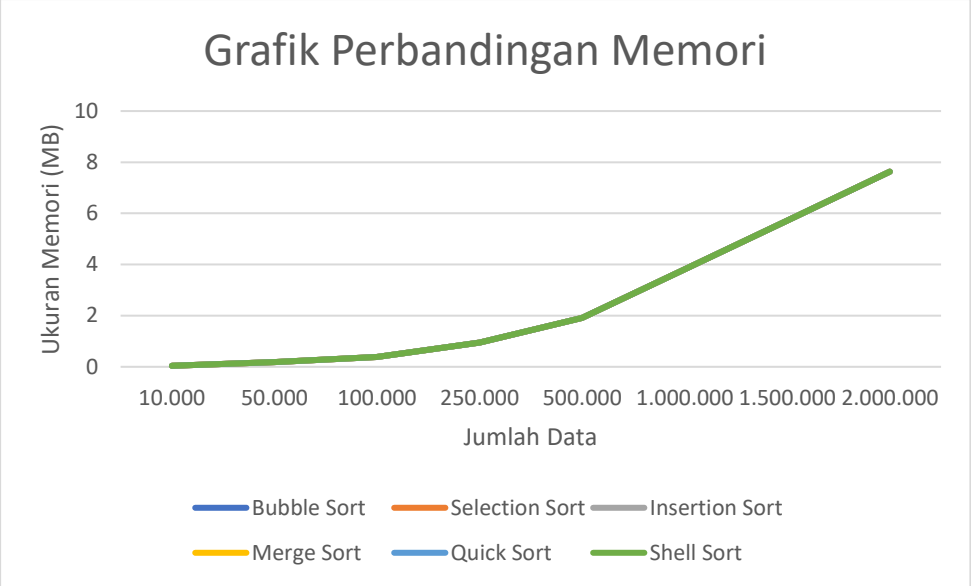
c. Grafik perbandingan waktu dan memori pada data angka

1. Grafik Perbandingan Waktu pada Data Angka



Grafik perbandingan waktu menunjukkan tren peningkatan signifikan pada algoritma dengan kompleksitas tinggi, terutama Bubble Sort yang mengalami lonjakan drastis seiring bertambahnya jumlah data. Pada data berukuran 2.000.000, waktu eksekusi Bubble Sort jauh melampaui algoritma lainnya, mencapai lebih dari 18.000 detik. Sebaliknya, algoritma seperti Merge Sort, Quick Sort, dan Shell Sort menunjukkan performa yang jauh lebih stabil dan efisien, bahkan saat memproses data dalam skala besar. Hal ini mencerminkan efisiensi algoritma berbasis divide-and-conquer atau hybrid dalam menangani kompleksitas waktu, menjadikannya lebih layak digunakan untuk kebutuhan pengolahan data besar.

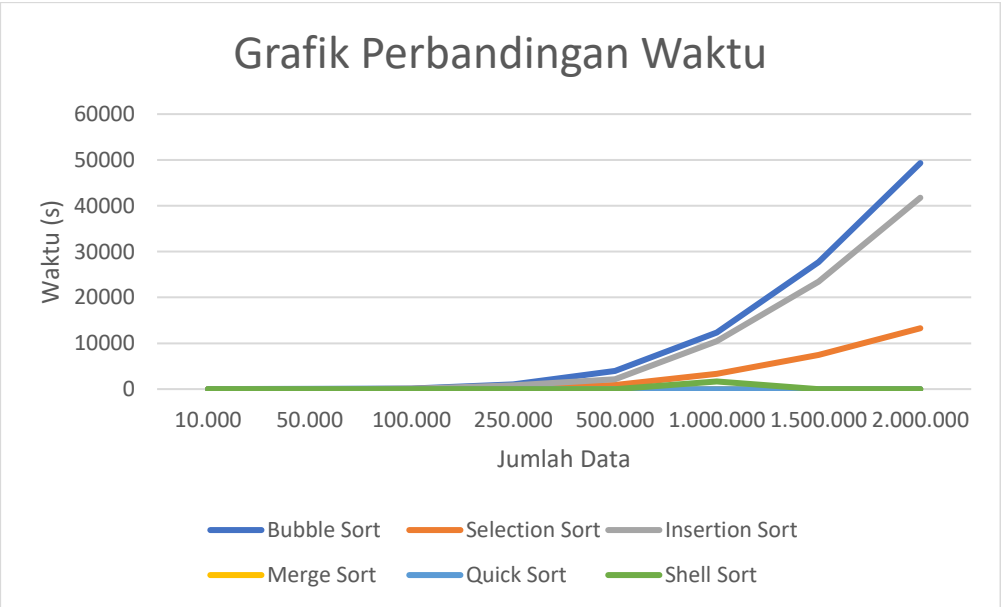
2. Grafik Perbandingan Memori pada Data Angka



Grafik perbandingan memori menunjukkan bahwa semua algoritma sorting menggunakan jumlah memori yang identik pada setiap skala data, mulai dari sekitar 0,19 MB untuk 10.000 data hingga 7,63 MB untuk 2.000.000 data. Kesamaan ini kemungkinan besar disebabkan oleh penggunaan alokasi memori dinamis yang seragam (malloc) untuk setiap salinan array input, sehingga perbedaan algoritma tidak memengaruhi hasil pengukuran memori.

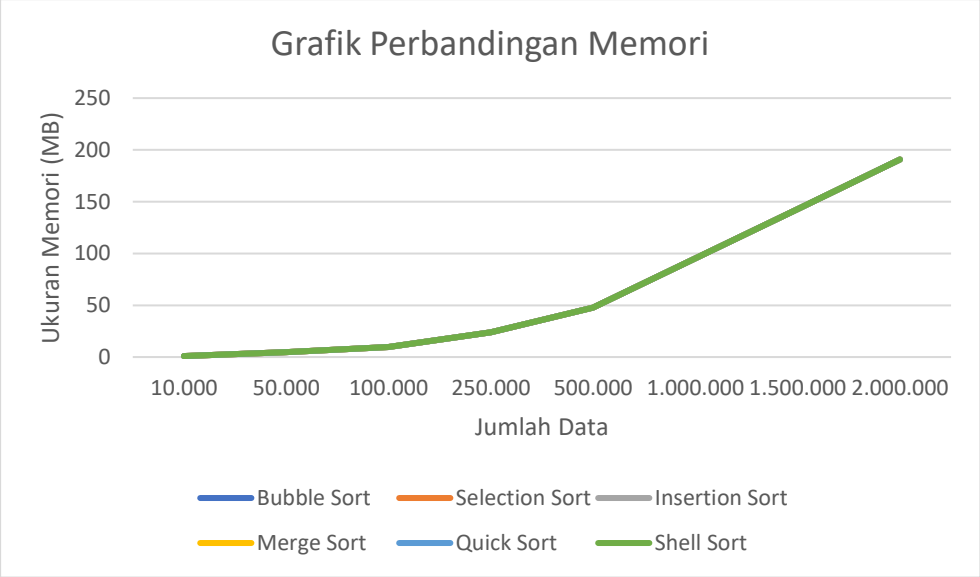
d. Grafik perbandingan waktu dan memori pada data kata

1. Grafik Perbandingan Waktu pada Data Kata



Grafik tersebut memperlihatkan perbandingan waktu eksekusi berbagai algoritma sorting terhadap jumlah data yang diolah. Terlihat jelas bahwa seiring bertambahnya jumlah data, waktu proses Bubble Sort meningkat sangat tajam hingga hampir 50.000 detik, menunjukkan efisiensi yang rendah pada skala besar. Selection Sort dan Insertion Sort juga menunjukkan tren peningkatan signifikan, namun masih lebih baik dibanding Bubble Sort. Sementara itu, Merge Sort, Quick Sort, dan Shell Sort menunjukkan waktu proses yang jauh lebih stabil dan rendah, bahkan ketika data mencapai dua juta. Hal ini menegaskan bahwa algoritma dengan kompleksitas yang lebih efisien (seperti O(n log n)) jauh lebih cocok untuk pengolahan data dalam jumlah besar.

2. Grafik Perbandingan Memori pada Data Kata



Grafik menunjukkan bahwa penggunaan memori meningkat secara konsisten seiring bertambahnya jumlah data yang diolah oleh algoritma sorting. Menariknya, seluruh algoritma menunjukkan pola penggunaan memori yang identik, yang kemungkinan besar disebabkan oleh pendekatan alokasi memori yang seragam, seperti penggunaan malloc untuk membuat salinan array sebelum proses sorting. Hal ini menunjukkan bahwa perbedaan algoritma tidak berdampak signifikan terhadap konsumsi memori dalam konteks implementasi ini, melainkan lebih dipengaruhi oleh ukuran data itu sendiri.

C. Kesimpulan

Proyek ini berhasil membandingkan enam algoritma sorting yaitu Bubble, Selection, Insertion, Merge, Quick, dan Shell dalam hal efisiensi waktu dan penggunaan memori pada dataset angka dan kata. Hasil pengujian menunjukkan bahwa algoritma berbasis $O(n^2)$, seperti Bubble, Selection, dan Insertion Sort, mengalami penurunan performa yang signifikan ketika jumlah data meningkat, dengan waktu eksekusi yang meningkat secara eksponensial. Sebaliknya, algoritma yang lebih efisien dengan kompleksitas $O(n \log n)$, seperti Merge, Quick, dan Shell Sort, menunjukkan performa yang lebih stabil dan cepat meskipun ukuran data sangat besar. Oleh karena itu, untuk pengurusan data dalam jumlah besar, algoritma seperti Merge Sort dan Quick Sort lebih direkomendasikan karena dapat menangani skala data besar dengan lebih efisien baik dari segi waktu maupun memori. Kesimpulannya, pemilihan algoritma sorting yang tepat sangat berpengaruh terhadap kinerja aplikasi, terutama ketika berhadapan dengan dataset yang besar dan kompleks.