TUGAS 4

disusun untuk memenuhi tugas

Mata Kuliah Struktur Data dan Algoritma D

Oleh:

Akrimah Usri
2308107010009

PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SYIAH KUALA

DARUSSALAM, BANDA ACEH

2025

A. Deskripsi Algoritma dan Cara Implementasinya

Dalam eksperimen ini, saya membandingkan enam algoritma sorting: Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, dan Shell Sort. Setiap algoritma diimplementasikan dalam bahasa C dan diuji untuk mengurutkan data bertipe int dan char*.

1. Bubble Sort

Bubble Sort adalah algoritma sederhana yang berulang kali membandingkan pasangan elemen berurutan dan menukar jika dalam urutan yang salah.

- Kompleksitas Waktu:
  - Terburuk: $O(n^2)$
  - Terbaik: O(n) [jika sudah terurut]
- Implementasi: Dua loop bersarang yang menukar elemen jika tidak terurut.

2. Selection Sort

Selection Sort mencari elemen terkecil dalam array dan meletakkannya di posisi pertama, lalu mengulang proses ini untuk elemen berikutnya.

- Kompleksitas Waktu: $O(n^2)$
- Implementasi: Dua loop, satu untuk iterasi posisi, satu untuk mencari nilai minimum.

3. Insertion Sort

Algoritma ini membangun array terurut secara bertahap dengan menyisipkan elemen ke posisi yang sesuai.

- Kompleksitas Waktu:
  - Terburuk: $O(n^2)$
  - Terbaik: O(n)
- Implementasi: Loop utama menempatkan elemen pada posisi yang tepat di bagian array yang telah terurut.

4. Merge Sort

Merge Sort membagi array menjadi dua bagian, menyortir keduanya secara rekursif, lalu menggabungkannya kembali secara terurut.

- Kompleksitas Waktu: O(n log n)
- Implementasi: Pendekatan divide-and-conquer dengan fungsi merge.

5. Quick Sort

Quick Sort memilih pivot, membagi array ke dua bagian berdasar pivot, lalu mengurutkan masing-masing bagian secara rekursif.

- Kompleksitas Waktu:
  - Terburuk: O(n^2)
  - Rata-rata: O(n log n)
- Implementasi: Pendekatan divide-and-conquer menggunakan fungsi partition.

6. Shell Sort

Shell Sort adalah pengembangan Insertion Sort yang membandingkan elemen dengan jarak tertentu (gap) yang menyusut.

- Kompleksitas Waktu: Bervariasi (sekitar O($n \log^2 n$))
- Implementasi: Insertion Sort dengan elemen berjeda.

B. Tabel Hasil Eksperimen

1. Data Integer (data_angka.txt)

1.1 Pengujian 10000 data

Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	0,095	39
Selection Sort	0,038	39
Insertion Sort	0,024	39
Merge Sort	0,000	39
Quick Sort	0,004	39
Shell Sort	0,001	39

1.2 Pengujian 50000 data

Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	4,046	195
Selection Sort	0,931	195
Insertion Sort	0,591	195
Merge Sort	0,003	195
Quick Sort	0,000	195
Shell Sort	0,014	195

1.3 Pengujian 100000 data

Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	16,757	390
Selection Sort	3,879	390
Insertion Sort	2,560	390
Merge Sort	0,014	390
Quick Sort	0,007	390
Shell Sort	0,019	390

1.4 Pengujian 250000 data

Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	220,719	976
Selection Sort	24,491	976
Insertion Sort	35,681	976
Merge Sort	0,079	976
Quick Sort	0,047	976
Shell Sort	0,087	976

1.5 Pengujian 500000 data

Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	468,247	1953
Selection Sort	94,220	1953
Insertion Sort	59,371	1953
Merge Sort	0,060	1953
Quick Sort	0,045	1953
Shell Sort	0,113	1953

1.6 Pengujian 1000000 data

Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	1785,654	3906
Selection Sort	373,955	3906
Insertion Sort	237,108	3906
Merge Sort	0,183	3906
Quick Sort	0,107	3906
Shell Sort	0,215	3906

1.7 Pengujian 1500000 data

Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	6166,050	5859
Selection Sort	2325,213	5859
Insertion Sort	1815,541	5859
Merge Sort	0,702	5859
Quick Sort	0,415	5859
Shell Sort	0,738	5859

1.8 Pengujian 2000000 data

Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	7245,565	7812
Selection Sort	4198,181	7812
Insertion Sort	3146,857	7812
Merge Sort	0,957	7812
Quick Sort	0,667	7812
Shell Sort	1,004	7812

2. Data String (data_kata.txt)

2.1 Pengujian 10000 data

Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	0,391	195
Selection Sort	0,126	195
Insertion Sort	0,050	195
Merge Sort	0,004	195
Quick Sort	0,001	195
Shell Sort	0,002	195

2.2 Pengujian 50000 data

Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	11,189	978
Selection Sort	3,326	978
Insertion Sort	1,464	978
Merge Sort	0,012	978
Quick Sort	0,007	978
Shell Sort	0,018	978

2.3 Pengujian 100000 data

Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	45,636	1954
Selection Sort	14,279	1954
Insertion Sort	6,556	1954
Merge Sort	0,027	1954
Quick Sort	0,018	1954
Shell Sort	0,042	1954

2.4 Pengujian 250000 data

Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	304,911	4882
Selection Sort	91,844	4882
Insertion Sort	45,053	4882
Merge Sort	0,064	4882
Quick Sort	0,056	4882
Shell Sort	0,119	4882

2.5 Pengujian 500000 data

Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	1387,622	9763
Selection Sort	1896,629	9763
Insertion Sort	1132,004	9763
Merge Sort	0,162	9763
Quick Sort	0,119	9763
Shell Sort	0,373	9763

2.6 Pengujian 1000000 data

Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	9963,179	19531
Selection Sort	9461,216	19531
Insertion Sort	2974,680	19531
Merge Sort	0,409	19531
Quick Sort	0,320	19531
Shell Sort	1,265	19531

2.7 Pengujian 1500000 data

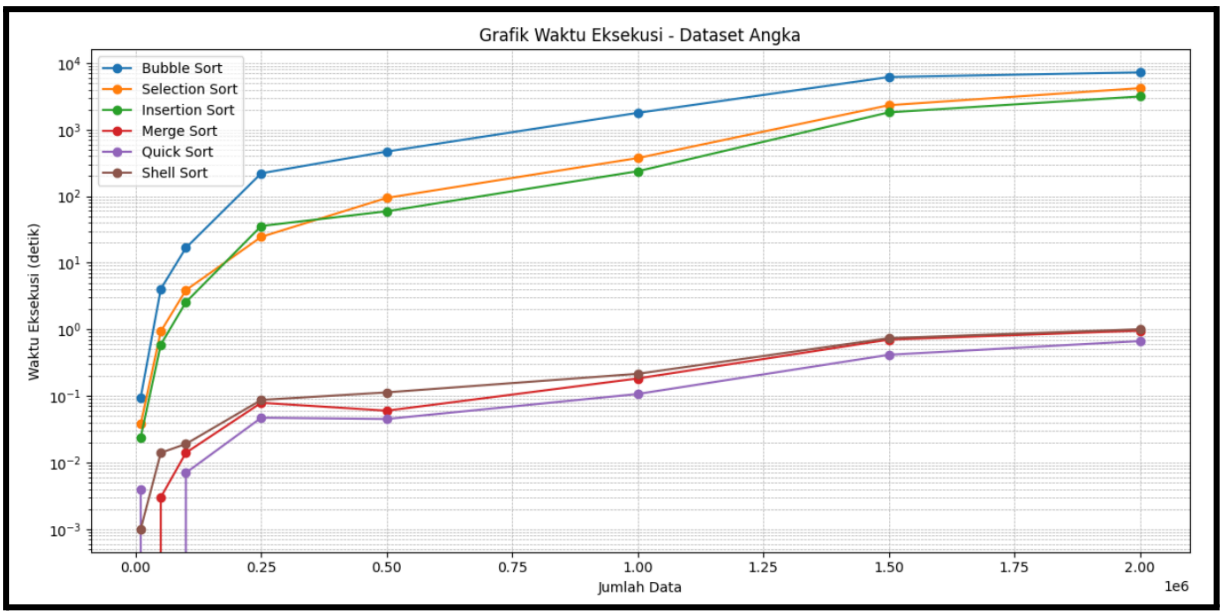
Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	22347,002	29298
Selection Sort	21374,621	29298
Insertion Sort	4885,117	29298
Merge Sort	0,531	29298
Quick Sort	0,489	29298
Shell Sort	1,983	29298

2.8 Pengujian 2000000 data

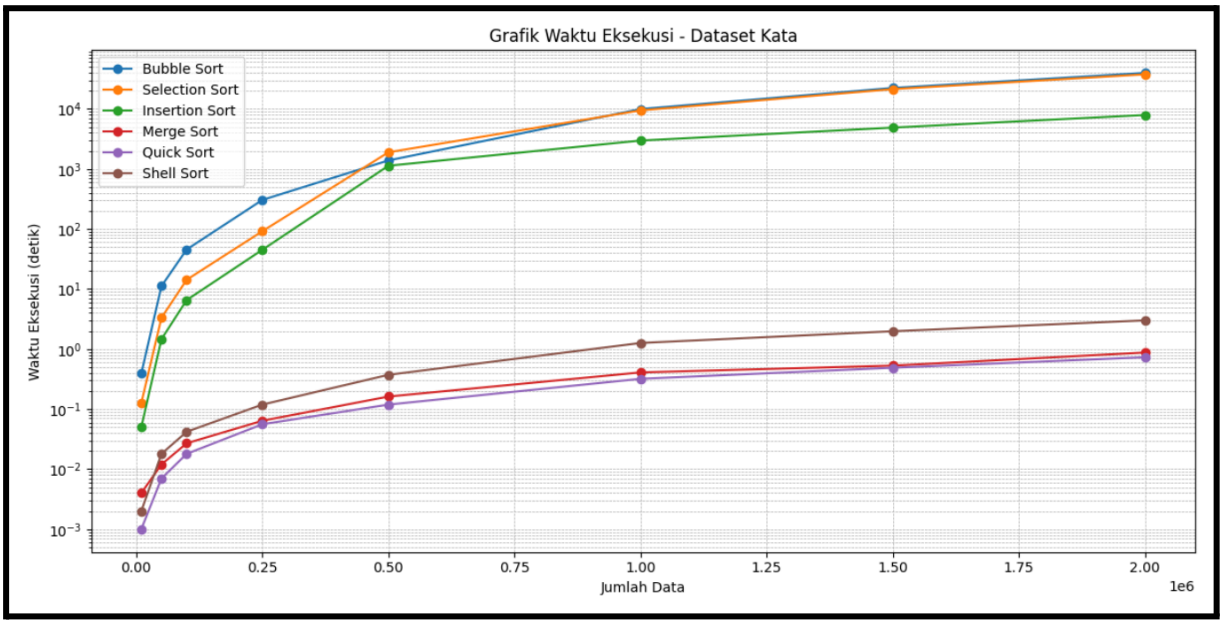
Algoritma	Waktu (detik)	Memori (KB)
Bubble Sort	39750,303	39062
Selection Sort	37650,011	39062
Insertion Sort	7920,748	39062
Merge Sort	0,876	39062
Quick Sort	0,732	39062
Shell Sort	3,006	39062

C. Grafik Perbandingan

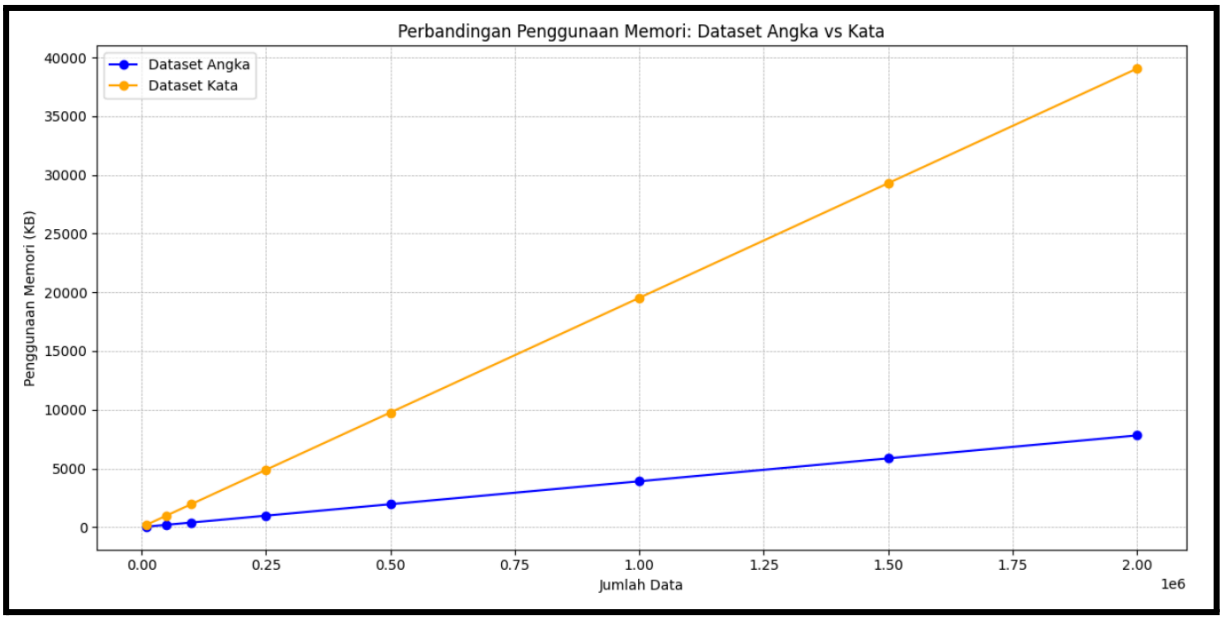
1. Grafik perbandingan waktu eksekusi dataset angka



2. Grafik perbandingan waktu eksekusi dataset kata



3. Grafik penggunaan memori



D. Analisis dan Kesimpulan

1. Analisis

1.1 Waktu Eksekusi

- Bubble Sort, Selection Sort, dan Insertion Sort mengalami peningkatan waktu secara eksponensial. Waktu eksekusi menjadi tidak praktis di atas 250.000 data.
- Merge Sort dan Quick Sort menunjukkan performa sangat baik, dengan peningkatan waktu yang sangat kecil meskipun data membesar hingga 2 juta. Hal ini sesuai dengan kompleksitas O(n log n).
- Shell Sort cukup efisien dan lebih stabil dibanding tiga algoritma sederhana, meskipun masih kalah dari Quick dan Merge untuk dataset besar.

1.2 Penggunaan Memori

Semua algoritma menggunakan memori yang relatif sama (berbanding lurus dengan ukuran data).

2. Kesimpulan

1) Quick Sort adalah algoritma terbaik dalam eksperimen ini berdasarkan kecepatan dan efisiensi memori.
2) Merge Sort memiliki performa sangat cepat dan stabil, cocok untuk data besar.

3) Shell Sort memberikan alternatif yang baik antara kecepatan dan kesederhanaan.
4) Bubble Sort, Selection Sort, dan Insertion Sort sangat lambat untuk data besar dan hanya layak digunakan pada skenario edukatif atau data kecil.