Nama: Yuyun Nailufar
Npm ; 2308107010066
SDA D

TUGAS 4 STRUKTUR DATA DAN ALGORITMA

Algoritma dan Cara Implementasi

Dalam eksperimen ini, diuji enam algoritma sorting populer: Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, dan Shell Sort. Semua algoritma diimplementasikan dalam bahasa pemrograman C untuk dua jenis data, yaitu angka dan kata (string).

1. **Bubble Sort**
   Bubble Sort bekerja dengan membandingkan dua elemen bersebelahan dan menukarnya jika urutannya salah. Proses ini diulang hingga seluruh elemen terurut.
   Implementasi:
   Bubble Sort menggunakan dua loop bersarang. Untuk data string, perbandingan dilakukan menggunakan fungsi strcmp().

2. **Selection Sort**
   Selection Sort mencari elemen terkecil dari array dan menempatkannya di posisi awal. Proses diulang dengan bagian array yang belum terurut.
   Implementasi:
   Algoritma ini menggunakan loop untuk mencari indeks minimum, lalu menukar nilai jika perlu. Versi string juga menggunakan strcmp().

3. **Insertion Sort**
   Insertion Sort menyisipkan elemen ke dalam bagian array yang sudah terurut dengan cara menggeser elemen yang lebih besar ke kanan.
   Implementasi:
   Menggunakan pergeseran elemen hingga menemukan posisi yang sesuai. Untuk string, perbandingan antar elemen dilakukan dengan strcmp().

4. **Merge Sort**
   Merge Sort merupakan algoritma Divide and Conquer. Array dibagi menjadi dua bagian, disortir secara rekursif, lalu digabungkan (merge) menjadi satu array yang terurut.
   Implementasi:
   Menggunakan dua fungsi: merge_sort() untuk pembagian, dan merge() untuk penggabungan. Untuk string, penggabungan mempertimbangkan urutan leksikografis.

5. **Quick Sort**
   Quick Sort juga menggunakan pendekatan Divide and Conquer. Algoritma memilih elemen pivot, mempartisi array menjadi dua bagian berdasarkan pivot, lalu melakukan sorting secara rekursif.
   Implementasi:
   Fungsi partition() membagi array, dan quick_sort() menyortir bagian kiri dan kanan. Pada string, digunakan strcmp() untuk perbandingan.

6. **Shell Sort**
   Shell Sort adalah varian dari Insertion Sort yang membandingkan elemen dengan jarak tertentu (gap), lalu mengurangi gap secara bertahap sampai 1.
   Implementasi:
   Menggunakan loop gap yang dimulai dari n/2, lalu dibagi dua di setiap iterasi. Untuk string, strcmp() digunakan dalam proses penyisipan.

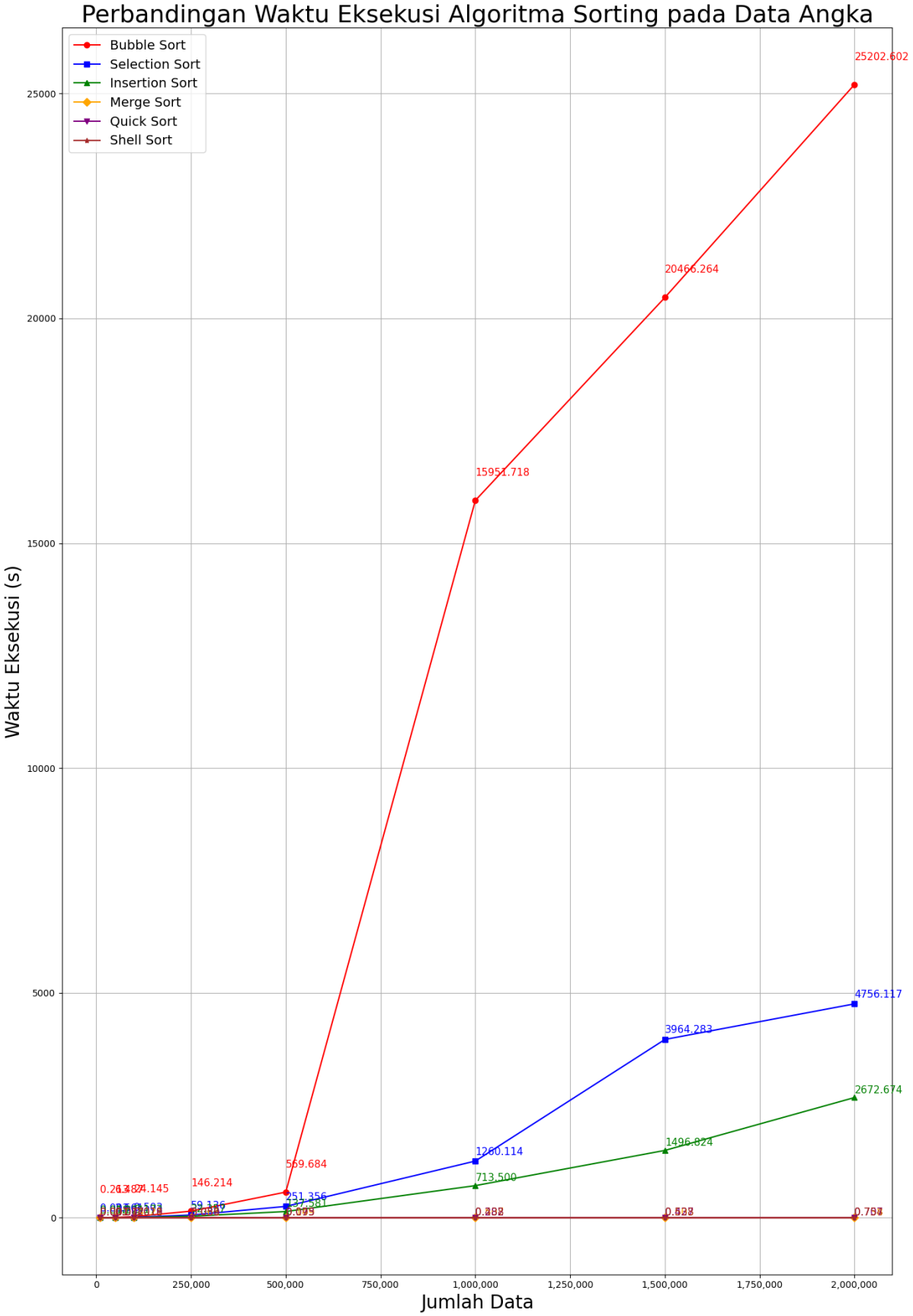
Tabel Hasil Eksperimen

- **Data Angka**

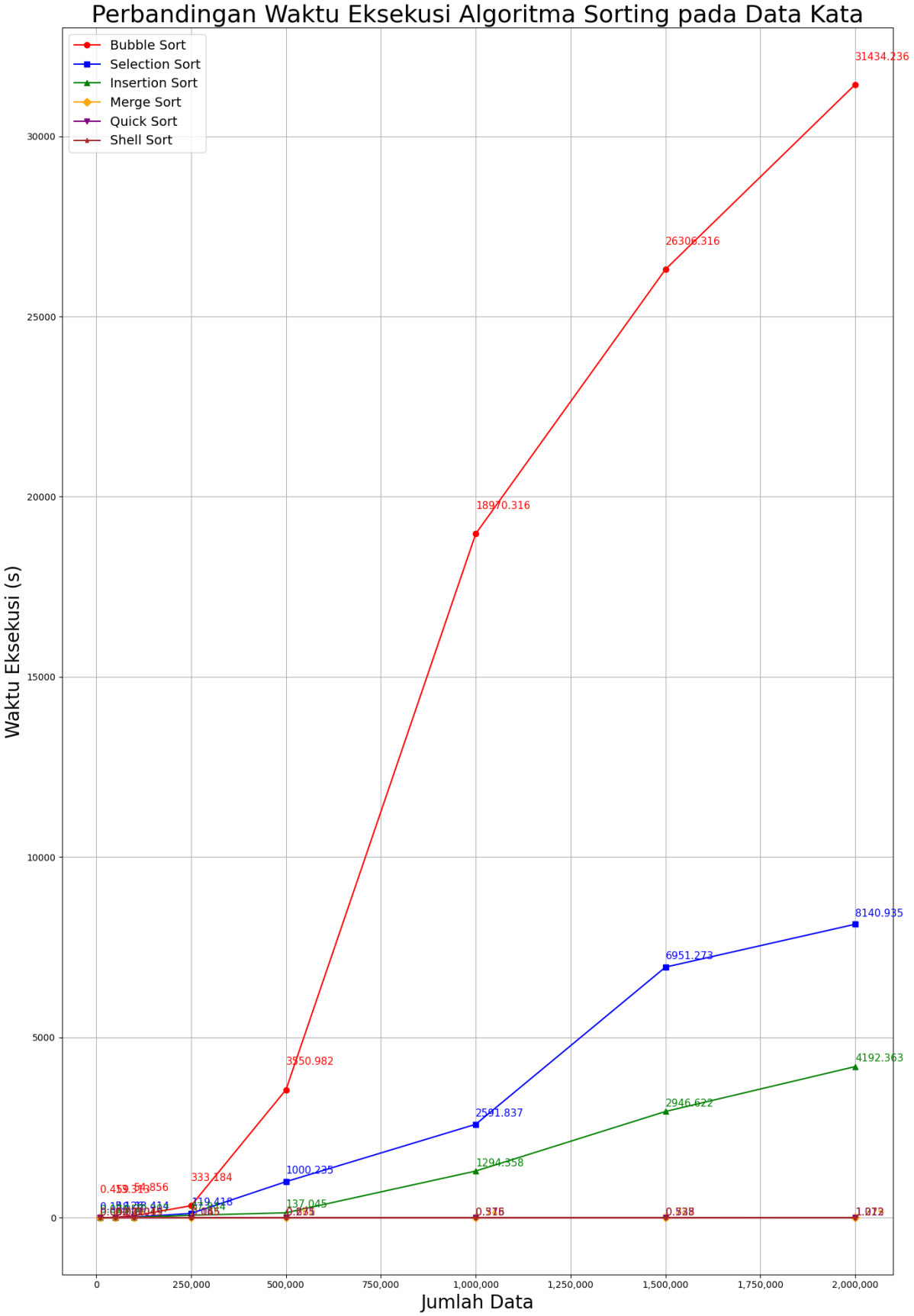
Banyak Angka	Waktu Eksekusi (S)						
	Bubble Sort	Selection Sort	Insertion Sort	Merge Sort	Quick Sort	Shell Sort	Memori (Mb)
10.000	0.213	0.093	0.058	0.007	0.001	0.001	0.04
50.000	6.487	2.561	1.321	0.014	0.007	0.012	0.19
100.000	24.145	9.503	5.173	0.021	0.019	0.018	0.38
250.000	146.214	59.136	32.357	0.061	0.039	0.056	0.95
500.000	569.684	251.356	137.581	0.149	0..073	0.095	1.91
1.000.000	15951.718	1260.114	713.500	0.507	0.282	0.438	3.81
1.500.000	20466.264	3964.283	1496.824	0.603	0.427	0.538	5.72
2.000.000	25202.602	4756.117	2672.674	0.738	0.707	0.753	7.63

- **Data Kata**

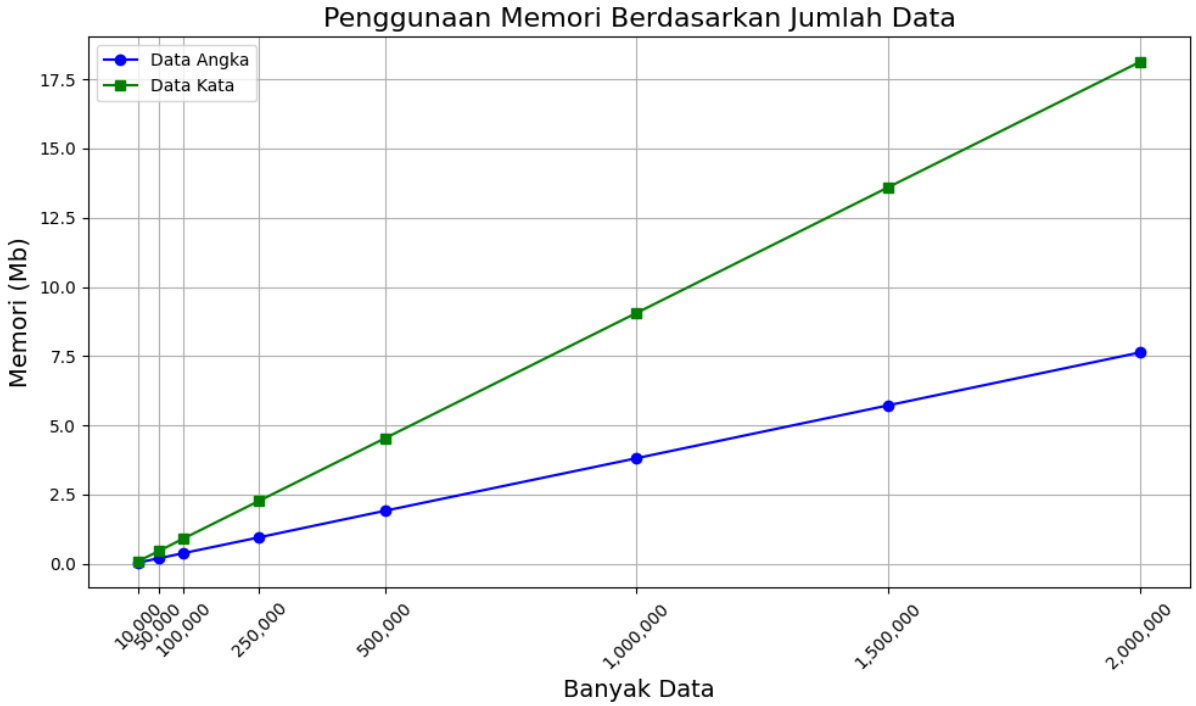
Banyak Kata	Waktu Eksekusi (S)						
	Bubble Sort	Selection Sort	Insertion Sort	Merge Sort	Quick Sort	Shell Sort	Memori (Mb)
10.000	0.459	0.148	0.076	0.000	0.000	0.002	0.09
50.000	13.313	5.126	2.657	0.001	0.001	0.032	0.45
100.000	54.856	18.414	10.207	0.019	0.019	0.045	0.91
250.000	333.184	119.418	67.844	0.076	0.065	0.145	2.27
500.000	3550.982	1000.235	137.045	0.260	0.275	0.631	4.53
1.000.000	18970.316	2591.837	1294.358	0.748	0.376	0.515	9.06
1.500.000	26306.316	6951.273	2946.622	0.847	0.538	0.725	13.60
2.000.000	31434.236	8140.935	4192.363	1.329	1.012	1.273	18.13



Gambar 1. Grafik Perbandingan waktu eksekusi algoritma Sorting pada data Angka



Gambar 2. Grafik Perbandingan waktu eksekusi algoritma Sorting pada data kata



Gambar 3. Grafik Penggunaan Memori Berdasarkan Jumlah Data

Analisis Data

Berdasarkan hasil eksperimen terhadap beberapa algoritma pengurutan data, terlihat perbedaan yang cukup signifikan dalam hal efisiensi waktu eksekusi dan penggunaan memori. Algoritma Bubble Sort, Selection Sort, dan Insertion Sort menunjukkan performa yang buruk saat jumlah data meningkat, baik untuk data angka maupun data kata. Sebagai contoh, untuk 2.000.000 data angka, waktu eksekusi Bubble Sort mencapai lebih dari 25.000 detik, sedangkan Merge Sort hanya membutuhkan kurang dari 1 detik.

Dari sisi penggunaan memori, terlihat bahwa jumlah memori yang digunakan meningkat seiring bertambahnya banyaknya data. Selain itu, data kata (string) menggunakan memori yang lebih besar dibandingkan data angka. Hal ini dikarenakan struktur data string memang membutuhkan ruang penyimpanan lebih banyak dibandingkan bilangan numerik. Performa waktu eksekusi pada data kata juga cenderung lebih lambat dibandingkan data angka untuk algoritma yang sama. Ini terjadi karena operasi perbandingan string lebih kompleks dibandingkan operasi perbandingan angka.

Secara keseluruhan, algoritma Merge Sort, Quick Sort, dan Shell Sort mampu memberikan hasil yang jauh lebih efisien dibandingkan tiga algoritma dasar lainnya, terutama pada dataset yang berukuran besar. Quick Sort tercatat sebagai algoritma tercepat pada hampir semua skenario pengujian.

Kesimpulan

Dari hasil analisis yang telah dilakukan, dapat disimpulkan beberapa hal berikut:

- Bubble Sort, Selection Sort, dan Insertion Sort tidak direkomendasikan untuk pengurutan data berukuran besar karena memiliki waktu eksekusi yang sangat lama.
- Merge Sort dan Quick Sort merupakan algoritma yang paling efisien dalam hal waktu eksekusi untuk dataset besar, baik pada data angka maupun data kata.
- Penggunaan memori bertambah seiring dengan bertambahnya jumlah data. Data bertipe string membutuhkan memori lebih banyak dibandingkan data bertipe angka.
- Untuk pengurutan data dalam skala besar, penggunaan algoritma yang lebih kompleks seperti Quick Sort atau Merge Sort sangat disarankan untuk mendapatkan hasil yang cepat dan efisien.