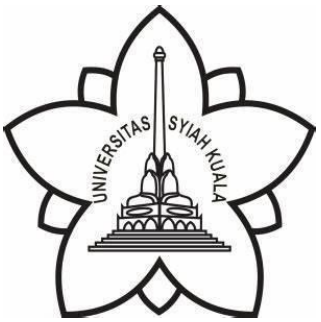
TUGAS 4

Disusun Untuk Memenuhi Tugas
Mata Kuliah Struktur Data dan Algoritma

Oleh :

Muhammad Hizqil Alfi

(2308107010046)



DEPARTEMEN INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SYIAH KUALA

2025

A. Kode dan Algoritma

1. Kode

Kode C ini memiliki tujuan untuk mengukur dan membandingkan performa berbagai algoritma sorting (pengurutan), baik untuk data angka maupun kata acak, dalam hal waktu eksekusi dan penggunaan memori. Kode yang saya buat memisahkan 6 algoritma sorting menjadi 1 file header dan 2 file **main** yang berguna untuk menjalankan kode untuk melakukan generate data acak.

2. Algoritma

Didalam kode terdapat 6 algoritma sorting, diantaranya :

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

Semua algoritma sorting tersebut terletak didalam satu file header yang menggabungkan semua metode sorting dan akan berjalan ketika file utama memanggil fungsi - fungsi yang diinginkan pada file header.

Alur Program :

a. Kode akan meminta input kepada pengguna tentang seberapa banyak data yang ingin di generate.
b. Generate data acak (angka atau kata tergantung file yang dijalankan) sebanyak yang diinginkan pengguna. Contoh : 1.500.000 data
c. Untuk setiap algoritma sorting :
    - Data disalin
    - Sorting dijalankan dalam beberapa thread terpisah (saya mengimplementasikan metode parallel execution)
    - Menggunakan fungsi **QueryPerformanceCounter** untuk menghitung waktu eksekusi dan fungsi **GetProccessMemoryInfo** untuk menghitung memori yang digunakan
d. Tunggu semua thread selesai (menggunakan fungsi **WaitForMultipleObjects**).
e. Tampilkan hasil waktu dan memori untuk setiap metode sorting.
f. Lakukan looping interaktif agar pengguna bisa memilih lagi untuk generate data acak lainnya dengan jumlah data yang berbeda atau pengguna bisa keluar dari program.

Kedua file C utama memiliki alur kerja program yang sama dimana metode sorting akan dijalankan kedalam beberapa thread untuk mempercepat waktu eksekusinya. Yang berbeda hanyalah data acak yang di-*generate* yang mana pada file main_angka.c data yang di-*generate* berupa angka dan main_kata.c data yang di-*generate* berupa kata atau string.

B. Hasil Eksperimen

1. Angka

Jumlah data : 10.000

Metode Sorting	Waktu	Memory
Bubble Sort	0.14s	4.58MB
Selection Sort	0.07s	4.59MB
Insertion Sort	0.038s	4.60MB
Merge Sort	0.005s	4.65MB
Quick Sort	0.002s	4.65MB
Shell Sort	0.002s	4.65MB

Jumlah data : 50.000

Metode Sorting	Waktu	Memory
Bubble Sort	3.51s	4.65MB
Selection Sort	1.80s	4.65MB
Insertion Sort	1.01s	4.65MB
Merge Sort	0.02s	4.65MB
Quick Sort	0.006s	4.65MB
Shell Sort	0.011s	4.65MB

Jumlah data : 100.000

Metode Sorting	Waktu	Memory
Bubble Sort	17.7s	4.65MB
Selection Sort	7.30s	4.65MB
Insertion Sort	4.58s	4.65MB

Merge Sort	0.04s	4.72MB
Quick Sort	0.011s	4.72MB
Shell Sort	0.021s	4.72MB

Jumlah data : 250.000

Metode Sorting	Waktu	Memory
Bubble Sort	142s	5.95MB
Selection Sort	40.18s	5.96MB
Insertion Sort	26.15s	5.96MB
Merge Sort	0.064s	6.41MB
Quick Sort	0.023s	6.41MB
Shell Sort	0.042s	6.41MB

Jumlah data : 500.000

Metode Sorting	Waktu	Memory
Bubble Sort	476.2s	8.35MB
Selection Sort	128.8s	8.35MB
Insertion Sort	96.7s	8.36MB
Merge Sort	0.155s	10.18MB
Quick Sort	0.06s	10.18MB
Shell Sort	0.13s	10.18MB

Jumlah data : 1.000.000

Metode Sorting	Waktu	Memory
Bubble Sort	1881s	12.14MB
Selection Sort	656.3s	12.14MB
Insertion Sort	385s	12.14MB
Merge Sort	0.281s	15.73MB
Quick Sort	0.13s	15.73MB

Shell Sort	0.22s	15.73MB

Jumlah data : 1.500.000

Metode Sorting	Waktu	Memory
Bubble Sort	4218s	15.74MB
Selection Sort	1233s	15.74MB
Insertion Sort	694.5s	15.74MB
Merge Sort	0.314s	17.76MB
Quick Sort	0.198s	17.78MB
Shell Sort	0.34s	17.78MB

Jumlah data : 2.000.000

Metode Sorting	Waktu	Memory
Bubble Sort	6912s	61.42MB
Selection Sort	2557s	61.42MB
Insertion Sort	1640s	61.42MB
Merge Sort	0.55s	61.42MB
Quick Sort	0.347s	61.42MB
Shell Sort	0.554s	61.42MB

2. **Kata**

Jumlah data : 10.000

Metode Sorting	Waktu	Memory
Bubble Sort	0.08s	5.23MB
Selection Sort	0.054s	5.25MB
Insertion Sort	0.027s	5.25MB
Merge Sort	0.001s	5.25MB
Quick Sort	0.002s	5.25MB
Shell Sort	0.001s	5.25MB

Jumlah data : 50.000

Metode Sorting	Waktu	Memory
Bubble Sort	2.16s	8.06MB
Selection Sort	1.4s	8.06MB
Insertion Sort	0.7s	8.06MB
Merge Sort	0.009s	8.14MB
Quick Sort	0.005s	8.14MB
Shell Sort	0.006s	8.14MB

Jumlah data : 100.000

Metode Sorting	Waktu	Memory
Bubble Sort	11.22s	11.39MB
Selection Sort	4.7s	11.42MB
Insertion Sort	2.6s	11.46MB
Merge Sort	0.023s	11.86MB
Quick Sort	0.013s	11.86MB
Shell Sort	0.012s	11.86MB

Jumlah data : 250.000

Metode Sorting	Waktu	Memory
Bubble Sort	92.1s	21.55MB
Selection Sort	26.3s	21.55MB
Insertion Sort	21.5s	21.55MB
Merge Sort	0.055s	21.55MB
Quick Sort	0.057s	21.56MB
Shell Sort	0.031s	21.56MB

Jumlah data : 500.000

Metode Sorting	Waktu	Memory

Bubble Sort	423.7s	33.68MB
Selection Sort	144.2s	33.68MB
Insertion Sort	94.6s	33.64MB
Merge Sort	0.084s	35.83MB
Quick Sort	0.189s	33.83MB
Shell Sort	0.064s	33.83MB

Jumlah data : 1.000.000

Metode Sorting	Waktu	Memory
Bubble Sort	1651s	61.96MB
Selection Sort	615.9s	65.85MB
Insertion Sort	207s	65.91MB
Merge Sort	0.152s	67.24MB
Quick Sort	0.089s	67.24MB
Shell Sort	0.22s	68.51MB

Jumlah data : 1.500.000

Metode Sorting	Waktu	Memory
Bubble Sort	4112s	81.52MB
Selection Sort	937s	81.52MB
Insertion Sort	610.2s	83.68Mb
Merge Sort	0.294s	83.71MB
Quick Sort	0.12s	83.35MB
Shell Sort	0.214s	83.38MB

Jumlah data : 2.000.000

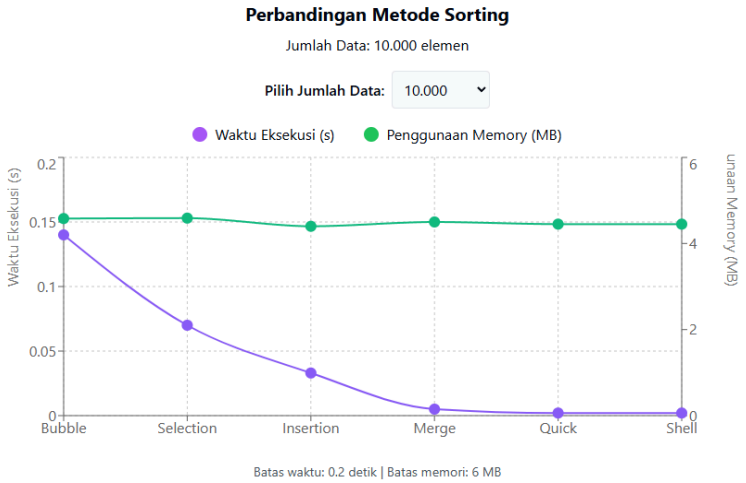
Metode Sorting	Waktu	Memory
Bubble Sort	6394s	104.12MB
Selection Sort	1985s	104.12MB

Insertion Sort	1231.7s	104.12MB
Merge Sort	0.452s	107.17MB
Quick Sort	0.277s	107.17MB
Shell Sort	0.514s	107.17MB

C. Perbandingnan Waktu dan Memory

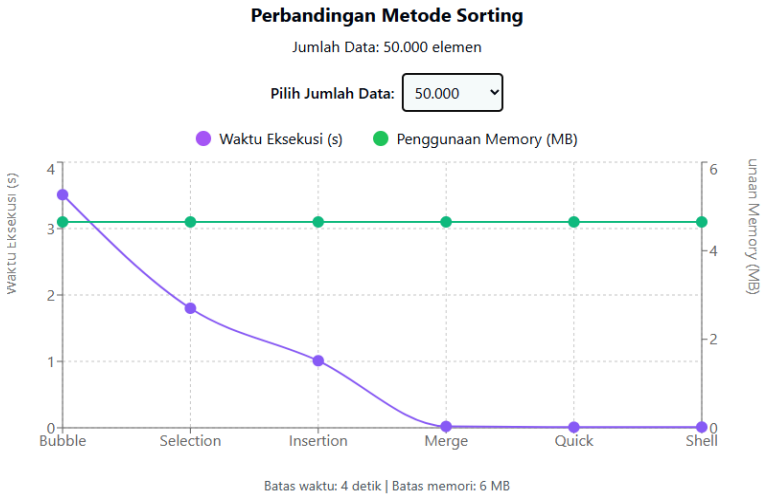
1. Angka

Jumlah Data : 10.000



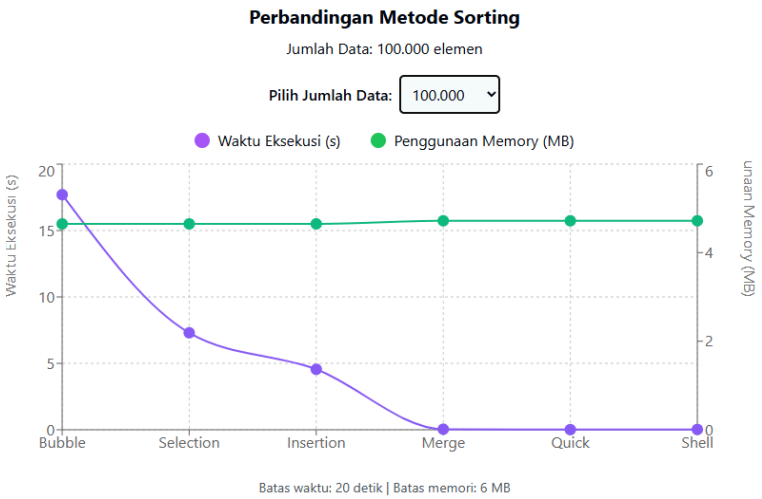
Gambar 1.1. Jumlah Data 10.000

Jumlah Data : 50.000



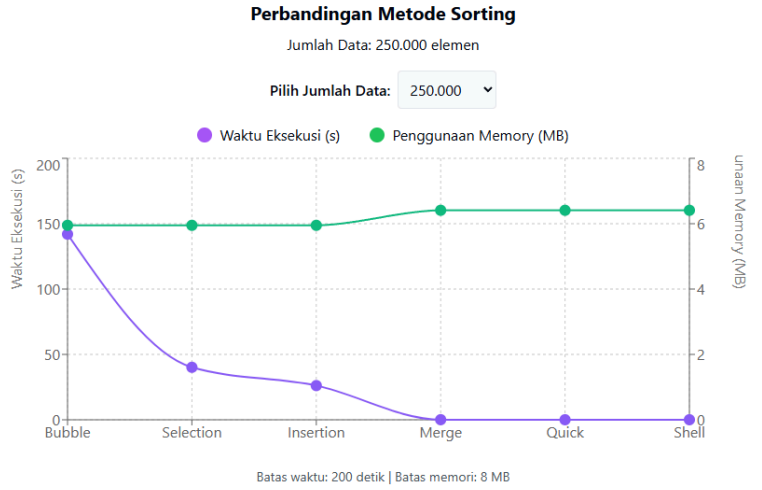
Gambar 1.2. Jumlah Data 50.000

Jumlah Data : 100.000



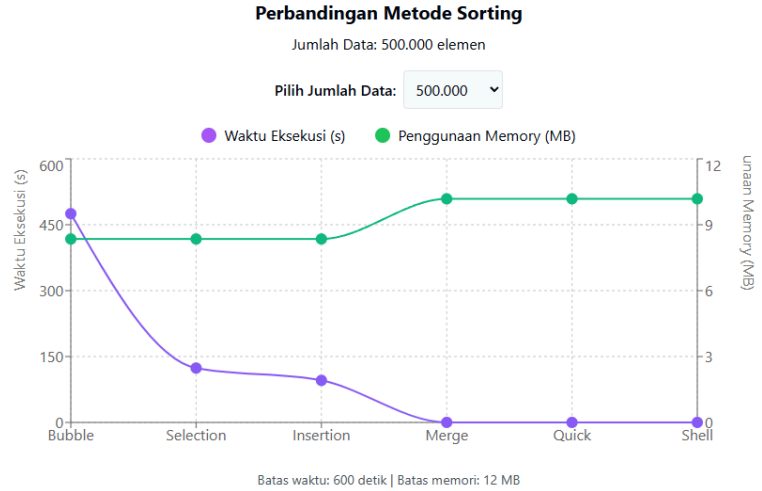
Gambar 1.3. Jumlah Data 100.000

Jumlah Data : 250.000



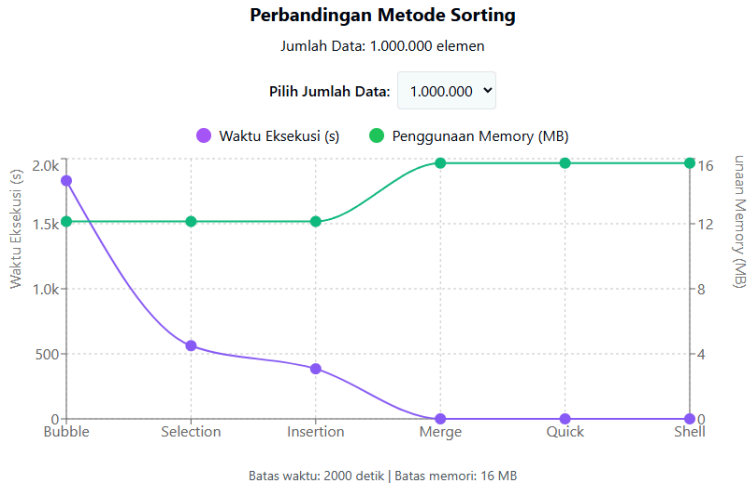
Gambar 1.4. Jumlah Data 250.000

Jumlah Data : 500.000



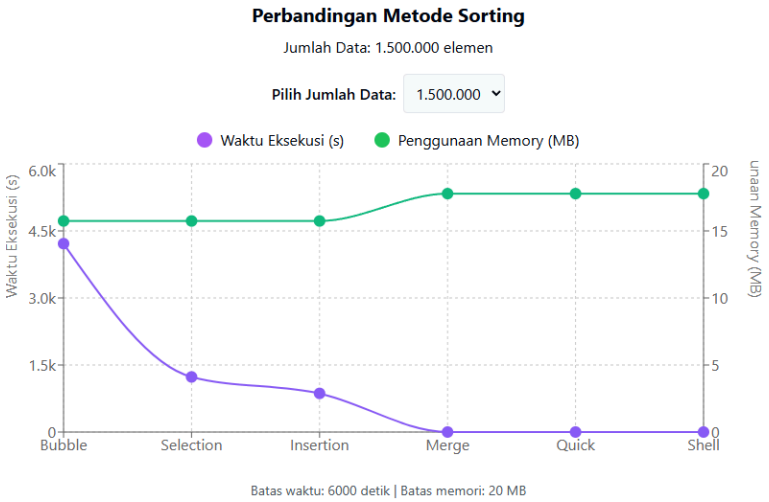
Gambar 1.5. Jumlah Data 500.000

Jumlah Data : 1.000.000



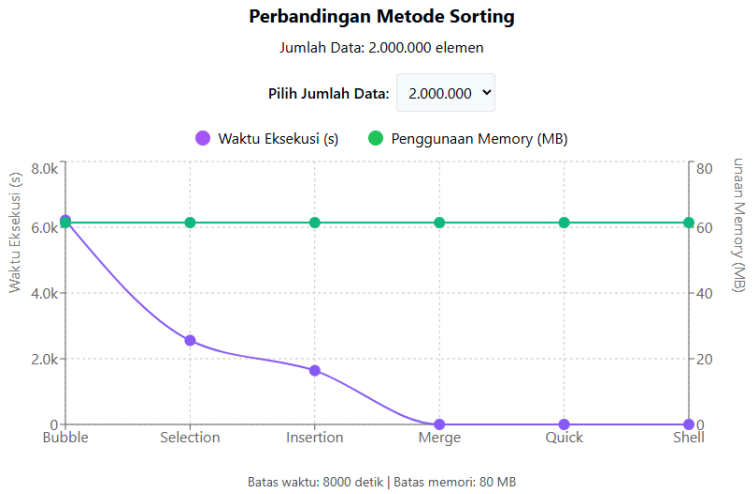
Gambar 1.6. Jumlah Data 1.000.000

Jumlah Data : 1.500.000



Gambar 1.7. Jumlah Data 1.500.000

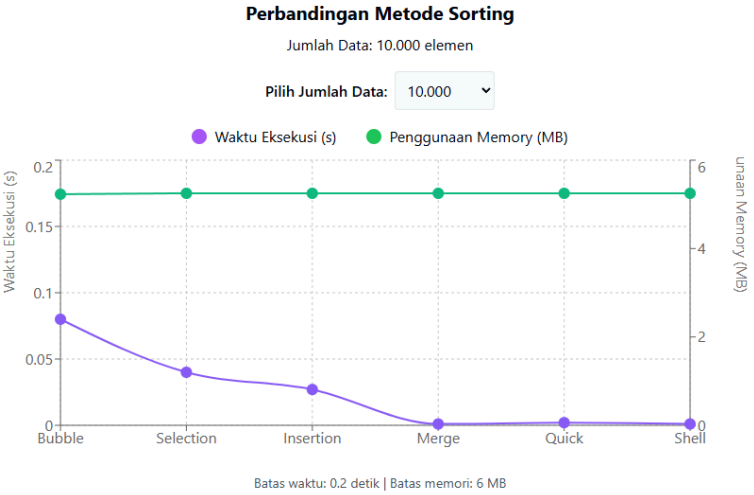
Jumlah Data : 2.000.000



Gambar 1.8. Jumlah Data 2.000.000

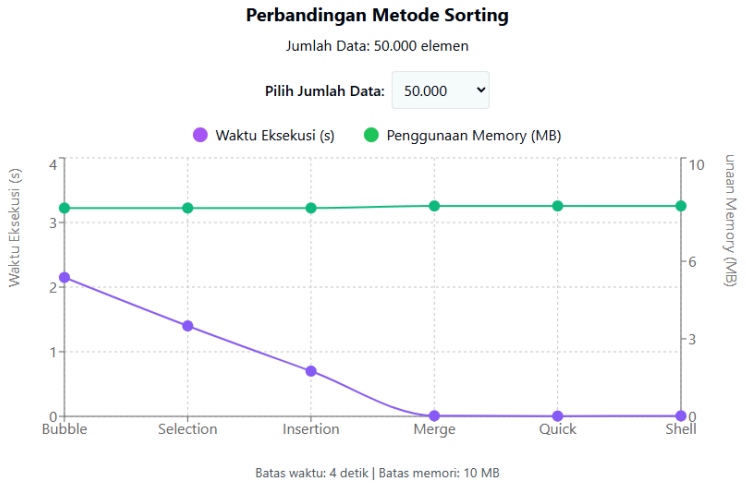
2. Kata

Jumlah Data : 10.000



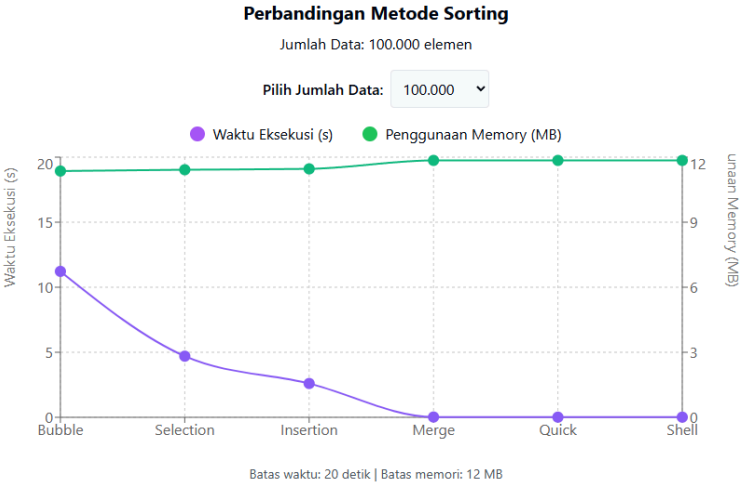
Gambar 2.1. Jumlah Data 10.000

Jumlah Data : 50.000



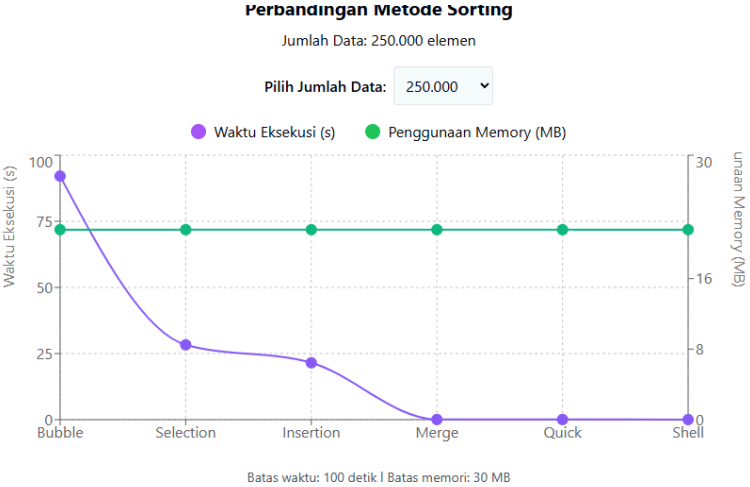
Gambar 2.2. Jumlah Data 50.000

Jumlah Data : 100.000



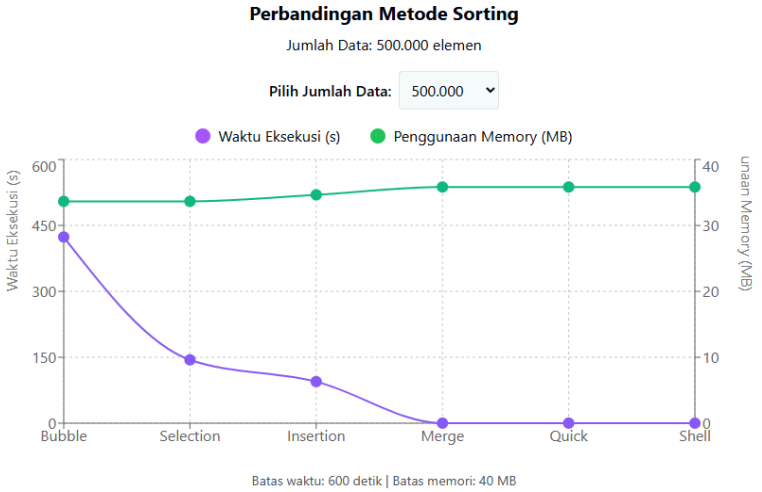
Gambar 2.3. Jumlah Data 100.000

Jumlah Data : 250.000



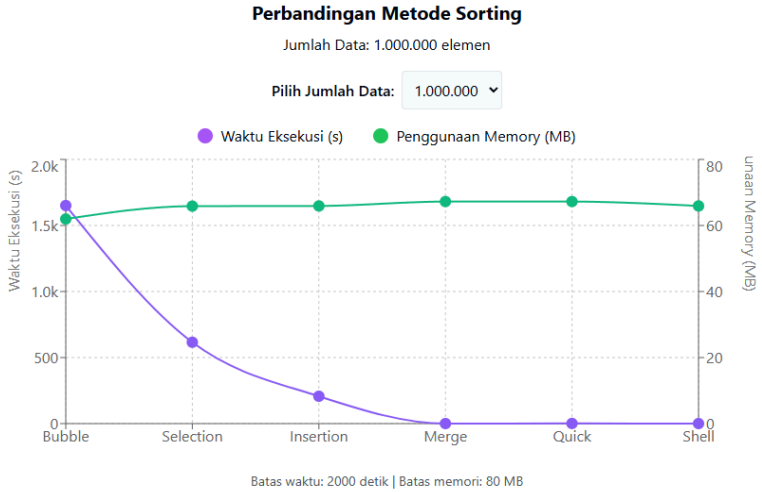
Gambar 2.4. Jumlah Data 250.000

Jumlah Data : 500.000



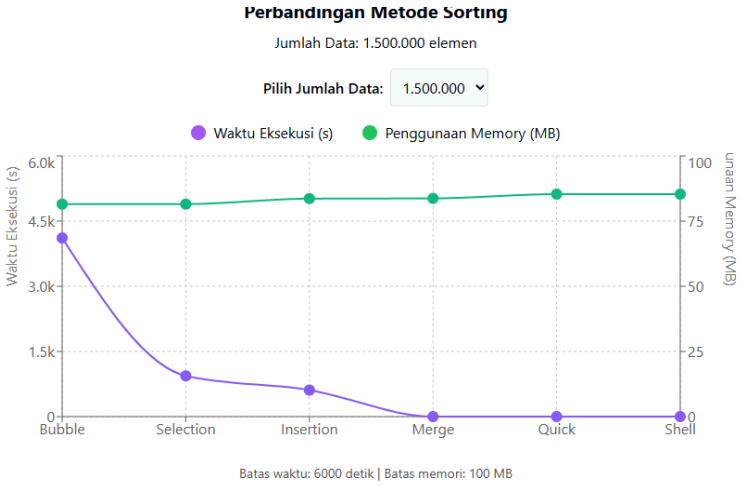
Gambar 2.5. Jumlah Data 500.000

Jumlah Data : 1.000.000



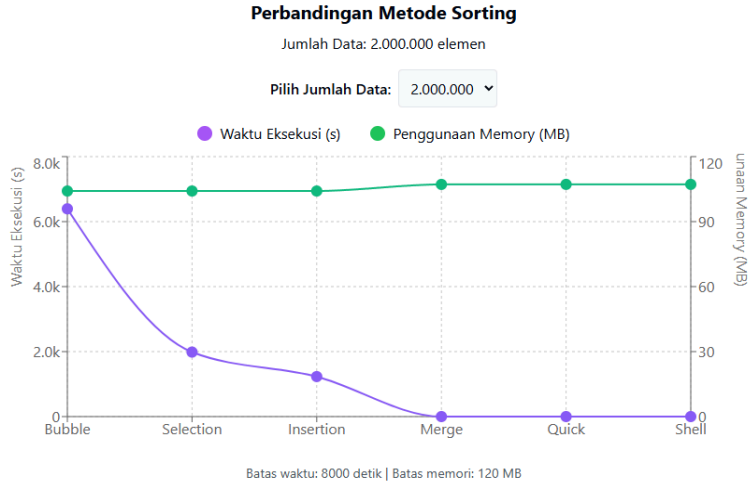
Gambar 2.6. Jumlah Data 1.000.000

Jumlah Data : 1.500.000



Gambar 2.7. Jumlah Data 1.500.000

Jumlah Data : 2.000.000



Gambar 2.8. Jumlah Data 2.000.000

D. Analisis dan Kesimpulan

1. Analisis

- **Performa Bubble Sorting**
  - Bubble Sorting merupakan metode dengan waktu eksekusi paling lama dibandingkan metode sorting lainnya.
  - Meskipun lambat, Bubble Sorting menggunakan memori paling sedikit, sehingga efisien dalam hal penggunaan sumber daya.
- **Performa Quick Sorting**
  - Quick Sorting menjadi metode yang paling cepat dalam proses eksekusi, terutama pada beberapa kondisi tertentu.
  - Namun, metode ini membutuhkan penggunaan memori yang lebih besar dibandingkan metode lainnya.
- **Shell Sorting**

- Dalam beberapa kondisi jumlah data, Shell Sorting menunjukkan waktu eksekusi yang lebih cepat dibandingkan Quick Sorting.

- **Perbandingan Jenis Data (Angka dan Kata)**
  - Metode sorting yang sama cenderung mengeksekusi lebih cepat pada data berupa **kata** dibandingkan dengan **angka**.
  - Hal ini berlaku baik untuk data acak berupa angka maupun kata, yang menunjukkan bahwa tipe data mempengaruhi performa sorting.

2. Kesimpulan

- Dari eksperimen yang telah dilakukan, bisa disimpulkan bahwa kita bisa membagi 6 metode sorting tersebut menjadi 2 kategori, yaitu :
  a. Metode sorting lambat menggunakan memori sedikit (Bubble Sort, Selection Sort, Insertion Sort)
  b. Metode sorting cepat menggunakan memori yang besar (Merge Sort, Quick Sort, Shell Sort)
- Bubble Sort merupakan metode sorting paling lambat tetapi menggunakan memori yang sedikit
- Quick Sort merupakan metode sorting paling cepat tetapi menggunakan memori yang besar
- Penggunaan setiap metode sorting ditentukan dengan case yang didapati atau yang ingin diselesaikan
- Jika ingin tidak terlalu memberatkan memori dan CPU ada baiknya memilih metode sorting lambat tetapi mengorbankan waktu yang lama sebagai gantinya
- Jika ingin pekerjaan diselesaikan dengan cepat maka gunakan metode sorting cepat yang mana mengorbankan memori dan CPU untuk bekerja lebih keras