*Laporan Mata Kuliah Struktur Data dan Algoritma*

**Sorting Algorithm**

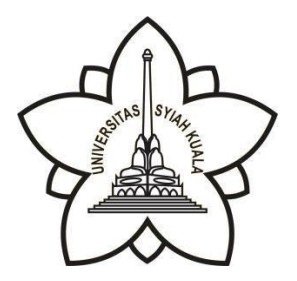
disusun untuk memenuhi tugas

Mata Kuliah Struktur Data dan Algoritma

Oleh:

**YAFI HIDAYATULLAH**

**(2308107010059)**



**PROGRAM STUDI INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS SYIAH KUALA**

**DARUSSALAM, BANDA ACEH**

**2025**

# Deskripsi algoritma dan cara implementasi

**Bubble Sort**

Kode menggunakan loop bersarang untuk membandingkan dan menukar elemen berdekatan, baik untuk integer (integer\_bubble\_sort) maupun string (string\_bubble\_sort).

Pernyataan strcmp digunakan untuk perbandingan string, sesuai deskripsi.

**Selection Sort**

Kode mencari indeks elemen terkecil di setiap iterasi (min\_index) dan menukarnya dengan elemen di posisi target, sesuai deskripsi.

Implementasi untuk string juga menggunakan strcmp untuk menentukan elemen terkecil.

**Insertion Sort**

Kode menyisipkan elemen (current) ke posisi yang tepat dalam array terurut, baik untuk integer maupun string.

Pada string, strcmp digunakan untuk membandingkan nilai.

**Merge Sort**

Kode membagi array secara rekursif (recursive\_merge\_sort) dan menggabungkan dua subarray terurut menggunakan array sementara yang dialokasikan secara dinamis (merge\_integers/merge\_strings), sesuai deskripsi divide and conquer.

**Quick Sort**

Kode memilih pivot (elemen terakhir), mempartisi array (partition\_integers/partition\_strings), lalu mengurutkan partisi secara rekursif, sesuai penjelasan algoritma.

Implementasi string menggunakan strcmp untuk membandingkan nilai dengan pivot.

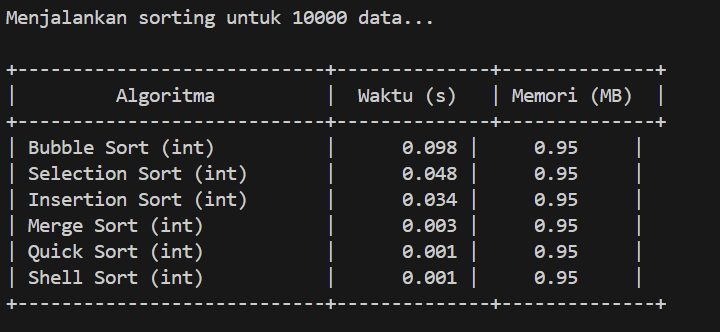
**Shell Sort**

Kode menggunakan gap yang dimulai dari setengah ukuran array dan mengecil secara eksponensial, lalu menggeser elemen sesuai interval gap. Hal ini sesuai dengan deskripsi optimasi Insertion Sort berbasis gap.

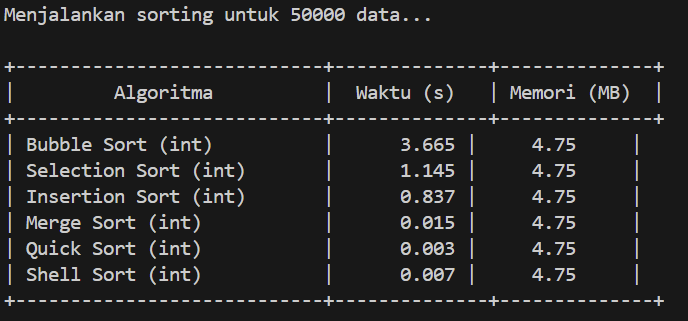
## **Output Data Sorting**

### Output sorting Data Angka

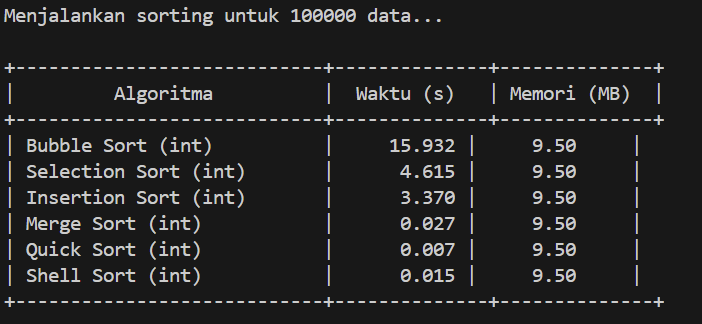
* 1. Output sorting 10000 data angka



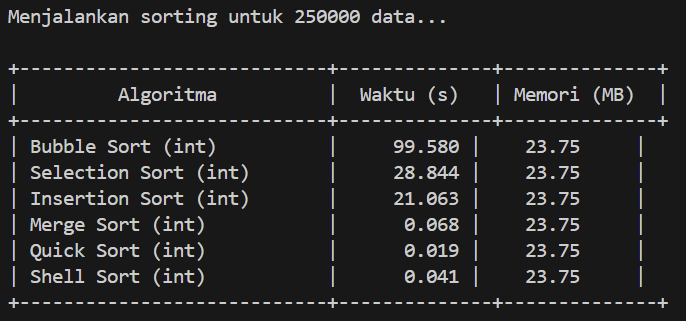
* 1. Output sorting 50000 data angka



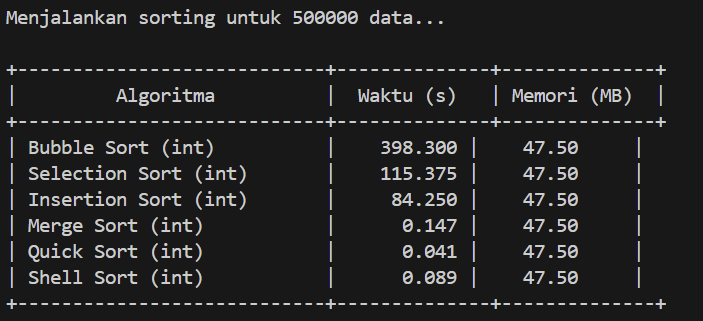
* 1. Output sorting 100000 data angka



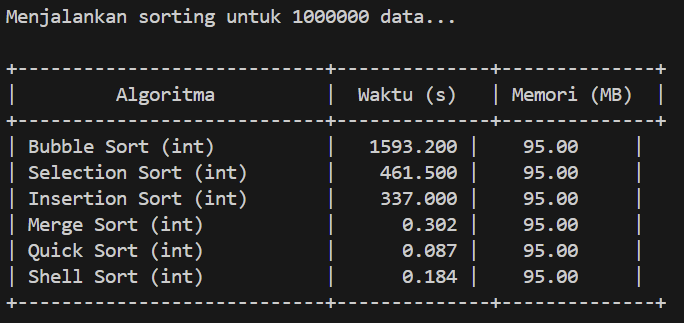
* 1. Output sorting 250000 data angka



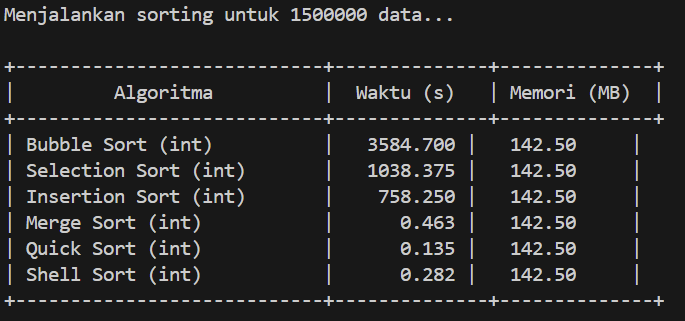
* 1. Output sorting 500000 data angka



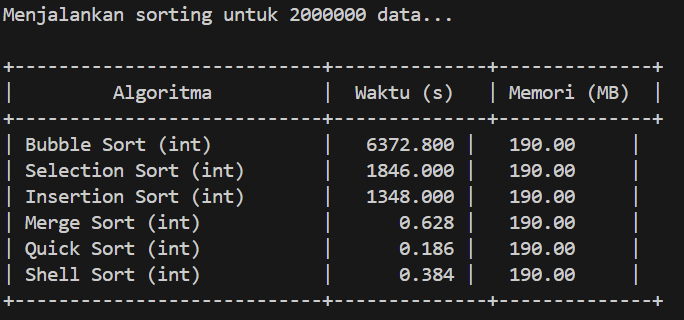
* 1. Output sorting 1000000 data angka



* 1. Output sorting 1500000 data angka

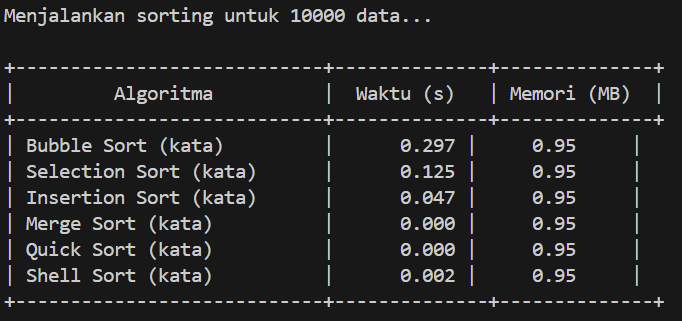


* 1. Output sorting 2000000 data angka

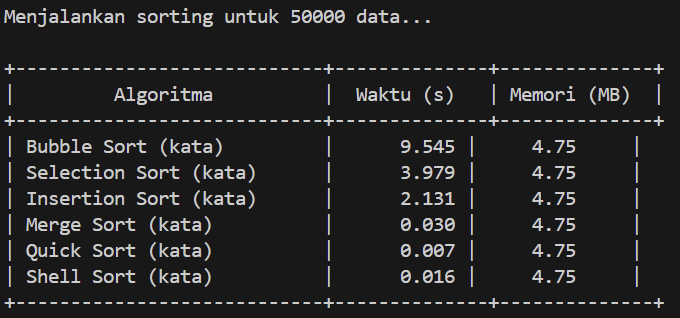


### Output sorting Data Kata

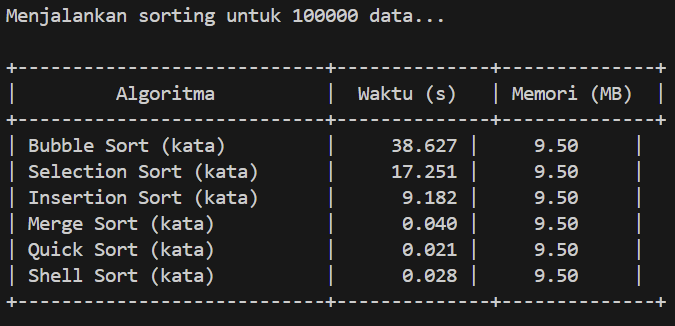
* 1. Output sorting 10000 data kata



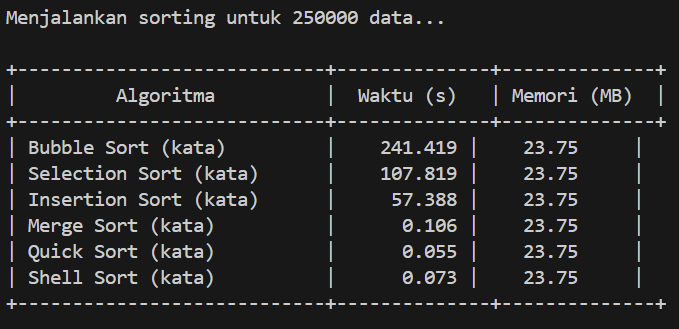
* 1. Output sorting 50000 data kata



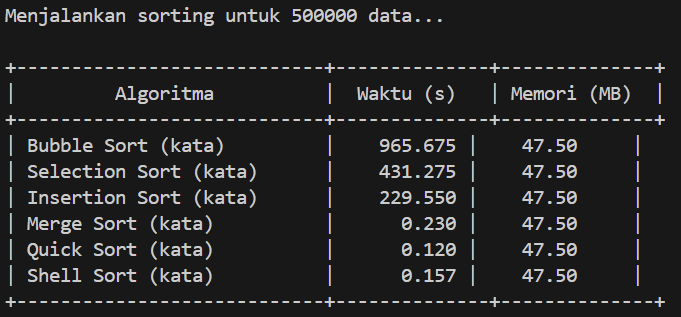
* 1. Output sorting 100000 data kata



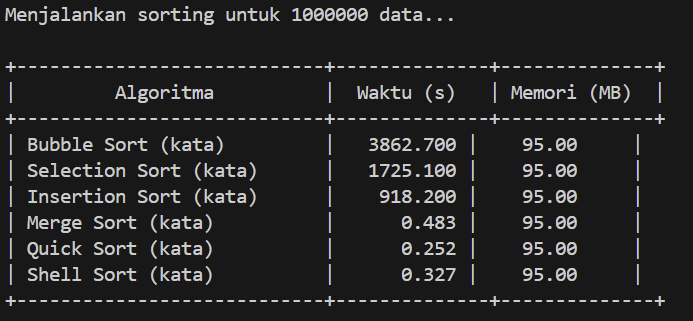
* 1. Output sorting 250000 data kata



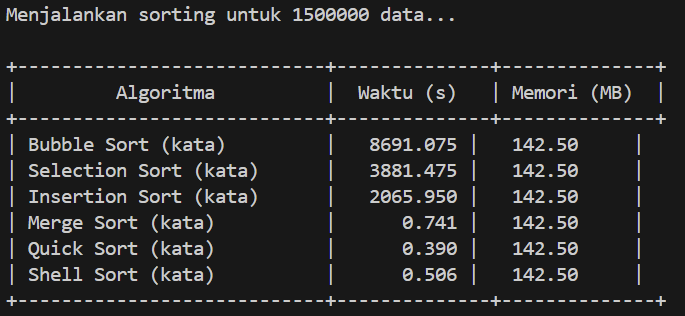
* 1. Output sorting 500000 data kata



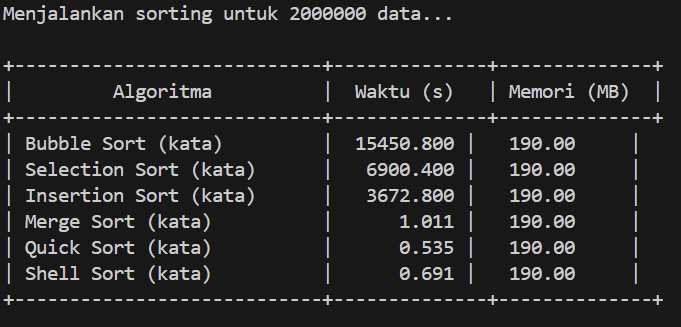
* 1. Output sorting 1000000 data kata



* 1. Output sorting 1500000 data kata

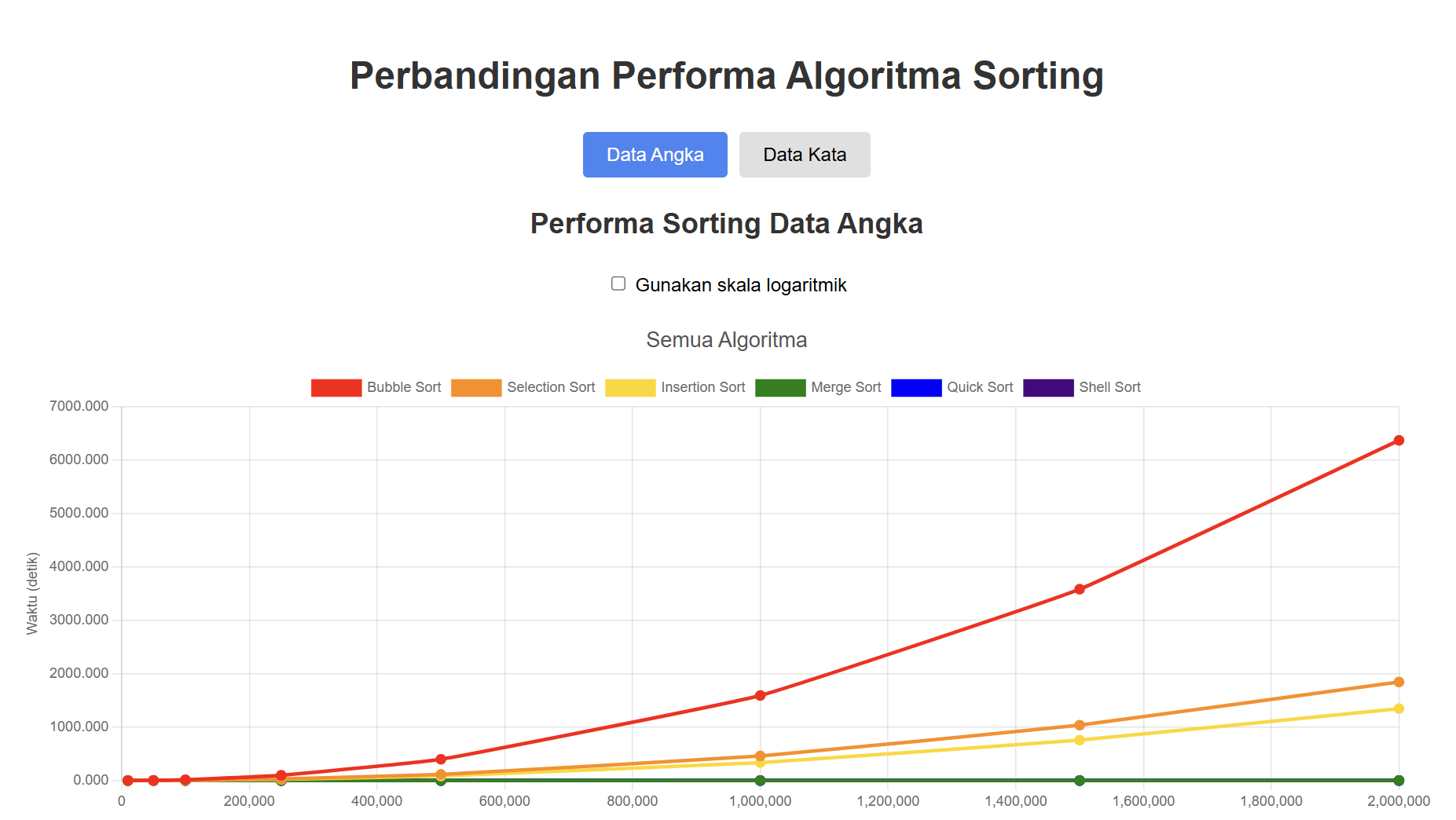


* 1. Output sorting 2000000 data kata

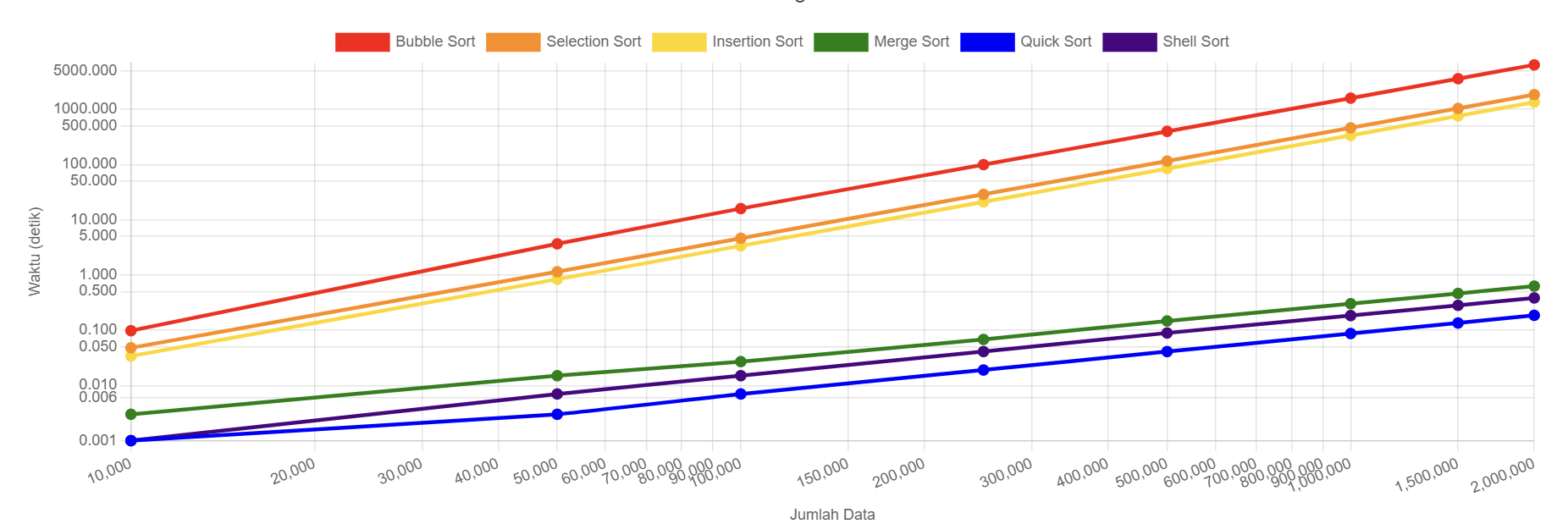


# Grafik Perbandingan Waktu dan Memory

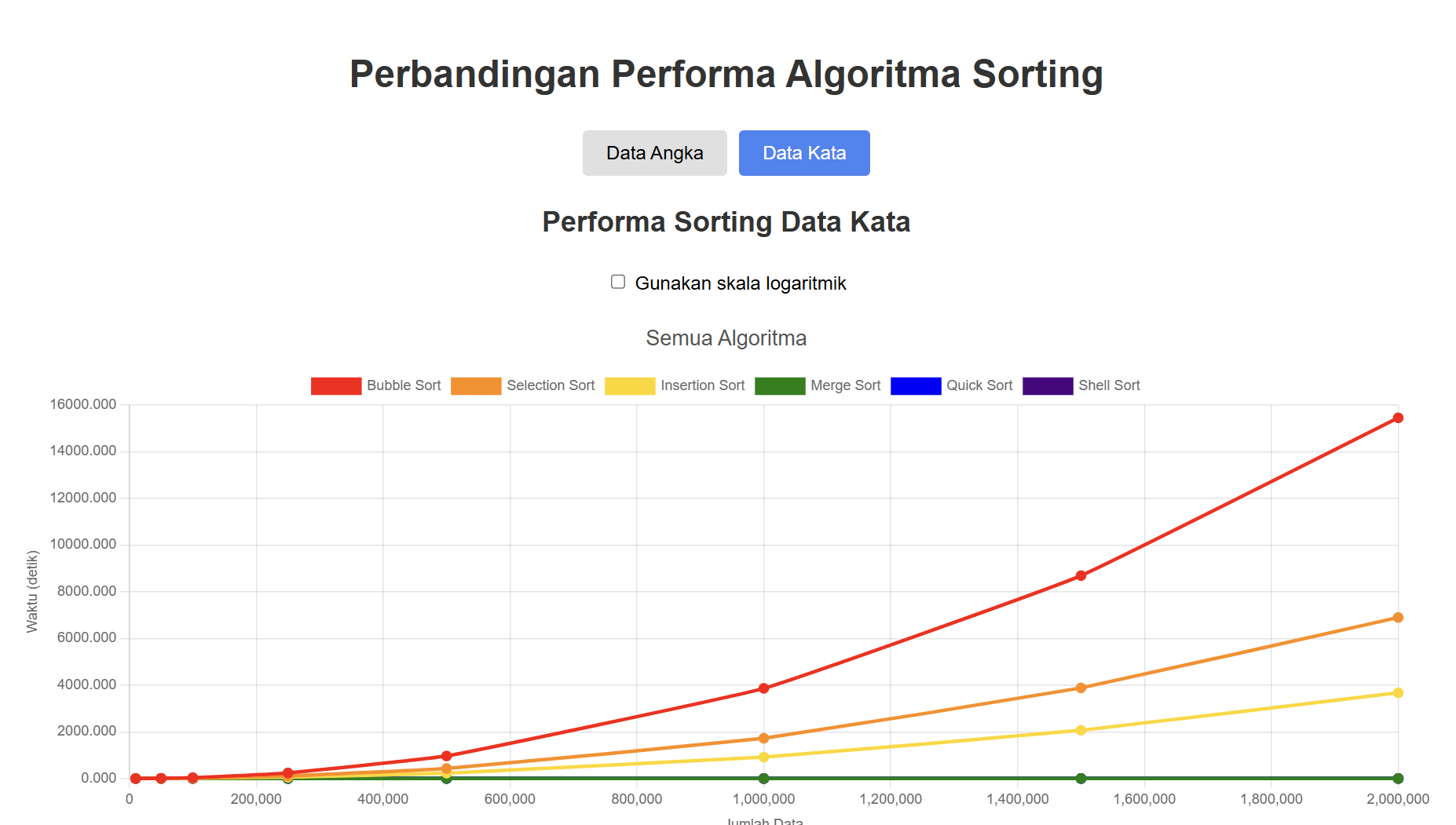
## GRAFIK DATA ANGKA (Waktu)



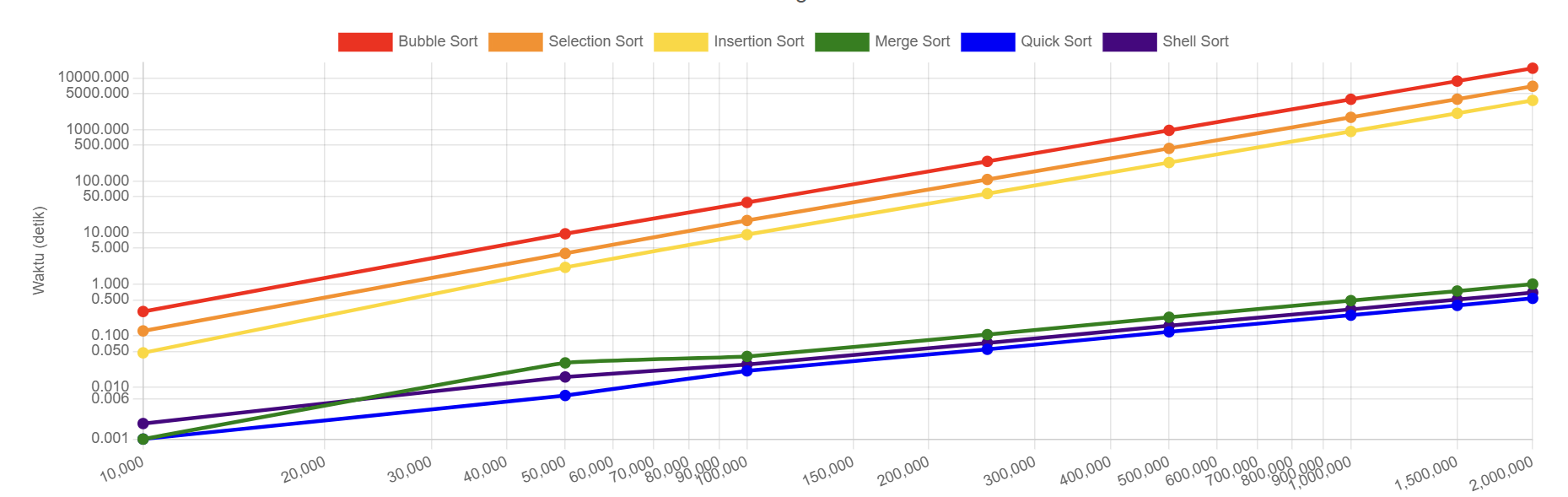
## GRAFIK DATA ANGKA (Memory)



## GRAFIK DATA KATA (Waktu)



## GRAFIK DATA KATA (Memory)



# Tabel Hasil Eksperimen (Waktu dan Memori)

## Tabel Sorting dan Memory Kata

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data | Bubble | Selection | Insertion | Merge | Quick | Shell | Memory |
| 10000 | 0.098 | 0.048 | 0.034 | 0.003 | 0.001 | 0.001 | 0.95  MB |
| 50000 | 3.665 | 1.145 | 0.837 | 0.015 | 0.003 | 0.007 | 4.75  MB |
| 100000 | 15.932 | 4.615 | 3.370 | 0.027 | 0.007 | 0.015 | 9.50 MB |
| 250000 | 99.580 | 28.844 | 21.063 | 0.068 | 0.019 | 0.041 | 23.75 MB |
| 500000 | 398.300 | 115.375 | 84.250 | 0.147 | 0.041 | 0.089 | 47.50 MB |
| 1000000 | 1593.200 | 461.500 | 337.000 | 0.302 | 0.087 | 0.184 | 95.00 MB |
| 1500000 | 3584.700 | 1038.375 | 758.250 | 0.463 | 0.135 | 0.282 | 142.50 MB |
| 2000000 | 6372.800 | 1846.000 | 1348.000 | 0.628 | 0.186 | 0.384 | 190.00 MB |

## Tabel Sorting dan Memory Angka

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data | Bubble | Selection | Insertion | Merge | Quick | Shell | Memory |
| 10000 | 0.297 | 0.125 | 0.047 | 0.000 | 0.000 | 0.002 | 0.95 MB |
| 50000 | 9.545 | 3.979 | 2.131 | 0.030 | 0.007 | 0.016 | 4.75 MB |
| 100000 | 38.627 | 17.251 | 9.182 | 0.040 | 0.021 | 0.028 | 9.50 MB |
| 250000 | 241.419 | 107.819 | 57.388 | 0.106 | 0.055 | 0.073 | 23.75 MB |
| 500000 | 965.675 | 431.275 | 229.550 | 0.230 | 0.120 | 0.157 | 47.50 MB |
| 1000000 | 3862.700 | 1725.100 | 918.200 | 0.483 | 0.252 | 0.327 | 95.00 MB |
| 1500000 | 8691.075 | 3881.475 | 2065.950 | 0.741 | 0.390 | 0.506 | 142.50 MB |
| 2000000 | 15450.800 | 6900.400 | 3672.800 | 1.011 | 0.535 | 0.691 | 190.00 MB |

**Analisis dan Kesimpulan**

Berdasarkan hasil eksperimen yang ditampilkan dalam tabel dan grafik, terlihat jelas bahwa algoritma sorting memiliki perbedaan kinerja yang signifikan. Algoritma Quick Sort menunjukkan performa terbaik secara konsisten baik untuk data angka maupun kata, dengan waktu eksekusi tercepat pada semua ukuran dataset. Merge Sort dan Shell Sort juga menunjukkan performa yang baik, berada di posisi kedua dan ketiga. Sementara algoritma Bubble Sort memiliki performa terburuk dengan waktu eksekusi yang meningkat secara drastis seiring bertambahnya jumlah data, mencapai hampir 16.000 detik untuk 2 juta data angka. Selection Sort dan Insertion Sort juga menunjukkan peningkatan waktu eksekusi yang signifikan pada dataset besar meskipun tidak separah Bubble Sort. Dari segi penggunaan memori, semua algoritma menunjukkan penggunaan yang sebanding dan meningkat secara linear sesuai dengan ukuran data, yang mengindikasikan bahwa perbedaan utama antar algoritma terletak pada efisiensi waktu eksekusi, bukan pada konsumsi memori.