SORTING ALGORITHM

disusun untuk memenuhi
tugas mata kuliah Struktur Data dan Algoritma

Oleh:

Bunga Rasikhah Haya
(2308107010010)

JURUSAN INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SYIAH KUALA
2025

1. **Deskripsi Algoritma dan Implementasi**
   a. **Bubble Sort**

      Bubble Sort bekerja dengan cara membandingkan dua elemen bersebelahan dan menukarnya jika berada dalam urutan yang salah. Proses ini diulang sampai array benar-benar terurut.

      Implementasi:
      - Fungsi **bubble_sort(int arr[], int n) dan bubble_sort_str(char *arr[], int n)** untuk angka dan string.
      - Menggunakan dua for loop bersarang.
      - Kompleksitas waktu: **$O(n^2)$**.

   b. **Selection Sort**

      Selection Sort mencari elemen terkecil dari bagian array yang belum terurut dan menukarnya ke posisi awal. Proses ini diulang untuk seluruh array.

      Implementasi:
      - Fungsi **selection_sort(int arr[], int n) dan selection_sort_str(char *arr[], int n).**
      - Mencari indeks minimum, lalu tukar dengan elemen ke-i.
      - Kompleksitas waktu: **$O(n^2)$**.

   c. **Insertion Sort**

      Insertion Sort menyisipkan setiap elemen ke posisi yang sesuai dalam bagian array yang sudah terurut.

      Implementasi:
      - Fungsi **insertion_sort(int arr[], int n) dan insertion_sort_str(char *arr[], int n).**
      - Menggeser elemen yang lebih besar ke kanan sebelum menyisipkan.
      - Kompleksitas waktu: **$O(n^2)$**.

   d. **Merge Sort**

      Merge Sort adalah algoritma **rekursif berbasis divide and conquer**. Ia membagi array menjadi dua bagian, mengurutkan masing-masing bagian, lalu menggabungkannya secara terurut.

      Implementasi:
      - Fungsi **merge_sort(int arr[], int left, int right) dan merge_sort_str(char *arr[], int left, int right).**
      - Fungsi tambahan merge() atau merge_str() untuk proses penggabungan.
      - Kompleksitas waktu: **$O(n \log n)$**.

e. Quick Sort

Quick Sort menggunakan prinsip divide and conquer dengan memilih **pivot**. Elemen dibagi dua kelompok: lebih kecil dan lebih besar dari pivot, lalu masing-masing diurutkan secara rekursif.

Implementasi:

- Fungsi **quick_sort(int arr[], int low, int high) dan quick_sort_str(char *arr[], int low, int high).**
- Menggunakan partition() untuk mengatur posisi pivot.
- Kompleksitas waktu:
  - Rata-rata: **O(n log n)**
  - Terburuk (pivot buruk): **O(n^2)**

f. Shell Sort

Shell Sort adalah versi efisien dari Insertion Sort yang memulai dengan gap besar antar elemen, lalu menguranginya secara bertahap hingga menjadi 1.

Implementasi:

- Fungsi s**hell_sort(int arr[], int n) dan shell_sort_str(char *arr[], int n).**
- Gap dimulai dari n/2 dan dibagi 2 setiap iterasi.
- Kompleksitas waktu bervariasi: antara **O(n $\log^2$ n)** hingga **O(n)** tergantung strategi gap.

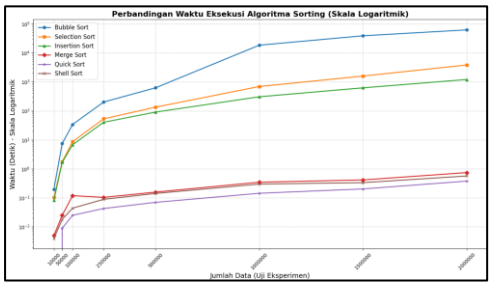
2. Tabel Hasil Eksperimen

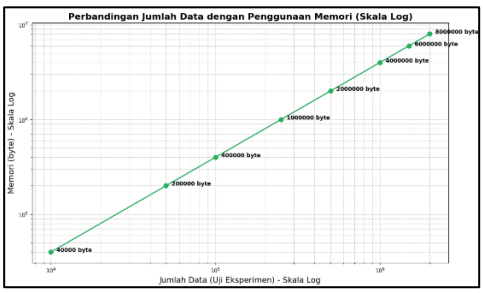
Uji Eksperimen	Tipe Data	Algoritma						Memori
		Bubble Sort	Selection Sort	Insertion Sort	Merge Sort	Quick Sort	Shell Sort	
10000	Angka	0.198 Detik	0.102 Detik	0.083 Detik	0.005 Detik	0.000 Detik	0.004 Detik	40000 byte
50000	Angka	7.443 Detik	1.758 Detik	1.681 Detik	0.025 Detik	0.009 Detik	0.018 Detik	200000 byte
100000	Angka	33.989 Detik	8.678 Detik	6.759 Detik	0.119 Detik	0.025 Detik	0.044 Detik	400000 byte
250000	Angka	202.115 Detik	53.116 Detik	40.487 Detik	0.104 Detik	0.043 Detik	0.089 Detik	1000000 byte
500000	Angka	624.130 Detik	134.310 Detik	90.789 Detik	0.158 Detik	0.070 Detik	0.142 Detik	2000000 byte
1000000	Angka	18439.310 Detik	692.923 Detik	306.449 Detik	0.347 Detik	0.144 Detik	0.299 Detik	4000000 byte
1500000	Angka	38878.620 Detik	1585.846 Detik	622.898 Detik	0.416 Detik	0.203 Detik	0.335 Detik	6000000 byte
2000000	Angka	62020.430 Detik	3832.543 Detik	1207.543 Detik	0.743 Detik	0.378 Detik	0.568 Detik	8000000 byte

Uji Eksperimen	Tipe Data	Algoritma						Memori
		Bubble Sort	Selection Sort	Insertion Sort	Merge Sort	Quick Sort	Shell Sort	
10000	Kata	0.114 Detik	0.083 Detik	0.038 Detik	0.006 Detik	0.000 Detik	0.007 Detik	159664 byte
50000	Kata	3.745 Detik	1.907 Detik	1.361 Detik	0.029 Detik	0.025 Detik	0.057 Detik	801547 byte
100000	Kata	17.211 Detik	7.719 Detik	4.572 Detik	0.079 Detik	0.055 Detik	0.145 Detik	1601413 byte
250000	Kata	48.661 Detik	37.186 Detik	0.414 Detik	0.104 Detik	0.054 Detik	0.070 Detik	4003342 byte
500000	Kata	1977.756 detik	774.745 Detik	370.325 Detik	0.110 Detik	0.038 Detik	1.053 Detik	7999524 byte
1000000	Kata	10198.180 Detik	1265.672 Detik	751.642 Detik	0.125 Detik	0.148 Detik	1.358 Detik	16000083 byte
1500000	Kata	23553.642 Detik	2471.324 Detik	1608.652 Detik	0.187 Detik	0.164 Detik	2.327 Detik	15613221 byte
2000000	Kata	44107.284 Detik	3242.538 Detik	2037.348 Detik	0.244 Detik	0.205 Detik	2.874 Detik	20080196 byte

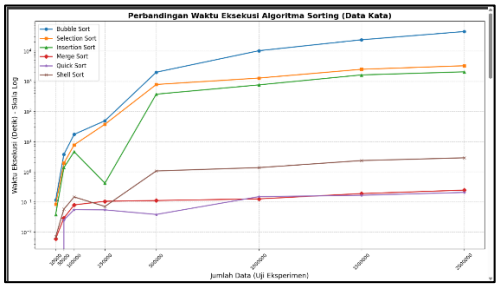
3. Grafik Perbandingan Waktu dan Memori

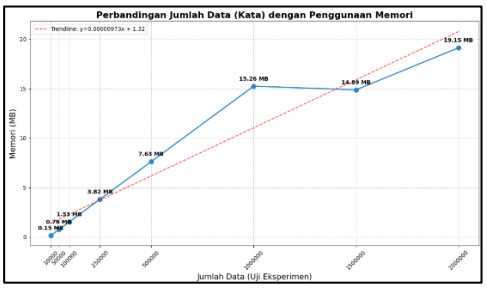
- Angka





- Kata





4. Analisis dan Kesimpulan

- Kinerja Waktu Eksekusi

a. Bubble Sort

Bubble Sort menunjukkan performa paling buruk untuk kedua jenis data.

- Data Kata: dari 0.114 detik (10.000 data) melonjak ke 44.107,284 detik (2 juta data).
- Data Angka: dari 0.198 detik ke 10.020,430 detik. Kompleksitas **$O(n^2)$** menyebabkan waktu eksekusi meningkat drastis seiring jumlah data bertambah, menjadikannya sangat tidak efisien.

b. Selection Sort

Selection Sort sedikit lebih baik dari Bubble Sort, tetapi tetap buruk untuk data besar.

- Data Kata: dari 0.083 detik menjadi 3242.538 detik.
- Data Angka: dari 0.102 detik menjadi 3832.543 detik. Dengan kompleksitas **$O(n^2)$**, algoritma ini juga tidak direkomendasikan untuk dataset skala besar.

c. Insertion Sort

Lebih cepat dibanding Bubble dan Selection Sort, tapi tetap tidak cocok untuk data besar.

- Data Kata: dari 0.038 detik ke 2037.348 detik.
- Data Angka: dari 0.083 detik ke 12207.543 detik. Meskipun sedikit lebih efisien, kompleksitas **$O(n^2)$** membuat performanya kurang optimal pada data besar.

d. Merge Sort

Merge Sort menunjukkan performa yang sangat stabil dan efisien.

- Data Kata: hanya naik dari 0.006 ke 0.244 detik meski data naik 200x lipat.
- Data Angka: dari 0.005 ke 0.743 detik. Dengan kompleksitas **$O(n \log n)$,** algoritma ini sangat andalan untuk skala besar.

e. Quick Sort

Quick Sort adalah algoritma dengan kinerja terbaik di hampir semua skenario.

- Data Kata: dari 0.000 ke 0.205 detik (sangat cepat).
- Data Angka: dari 0.000 ke 0.378 detik. Kompleksitas **$O(n \log n)$** menjadikannya sangat efisien dan scalable.

f. Shell Sort
   - Data Kata: dari 0.007 ke 2.874 detik.
   - Data Angka: dari 0.004 ke 0.568 detik. Walaupun tidak secepat Merge dan Quick Sort, performanya tetap baik untuk data sedang hingga besar.

- Penggunaan Memori
  - Data Kata: Memori meningkat dari 159.664 byte (10.000 data) ke 20.080.196 byte (2 juta data).
  - Data Angka: Memori meningkat dari 40.000 byte ke 8.000.000 byte.

  Semua algoritma menggunakan memori dalam jumlah yang relatif sama pada ukuran data tertentu. Tipe data Kata menggunakan lebih banyak memori dibanding Angka karena struktur data yang lebih kompleks. Oleh karena itu, waktu eksekusi menjadi metrik utama untuk membandingkan efisiensi algoritma dalam pengujian ini.

- Kesimpulan
  1. Quick Sort dan Merge Sort adalah algoritma paling efisien dan stabil untuk semua ukuran dan jenis data.
  2. Shell Sort dapat menjadi alternatif yang baik jika tidak menggunakan rekursi, meskipun sedikit lebih lambat.
  3. Bubble Sort, Selection Sort, dan Insertion Sort sangat tidak disarankan untuk data besar karena waktu eksekusi yang sangat tinggi.
  4. Penggunaan memori meningkat proporsional terhadap jumlah data, namun tidak membedakan performa antar algoritma secara signifikan.