Laporan Struktur Data dan Algoritma

ALGORITHMA SORTING

disusun untuk memenuhi tugas mata kuliah
Struktur Data dan Algoritma

Oleh:

Faris Zain As-Shadiq

2308107010039

JURUSAN INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SYIAH KUALA
DARUSSALAM, BANDA ACEH

2025

Pendahuluan

Laporan ini disusun untuk memenuhi Tugas 4 mata kuliah Struktur Data dan Algoritma, yang bertujuan mengevaluasi performa enam algoritma pengurutan—Bubble Sort, Selection Sort, Insertion Sort, Shell Sort, Merge Sort, dan Quick Sort—dalam mengolah data berskala besar berupa angka dan kata. Penilaian performa berfokus pada waktu eksekusi yang diukur dalam detik dan penggunaan memori dalam megabyte, dengan eksperimen dilakukan pada data acak berukuran 10.000 hingga 2.000.000 elemen. Hasil eksperimen divisualisasikan melalui grafik interaktif dan disajikan dalam tabel untuk mempermudah perbandingan. Laporan ini mencakup deskripsi algoritma, cara implementasi, hasil eksperimen, grafik perbandingan, serta analisis dan kesimpulan, sesuai dengan instruksi tugas. Dengan demikian, laporan ini bertujuan memberikan wawasan mendalam tentang efisiensi algoritma pengurutan dan relevansinya dalam aplikasi praktis.

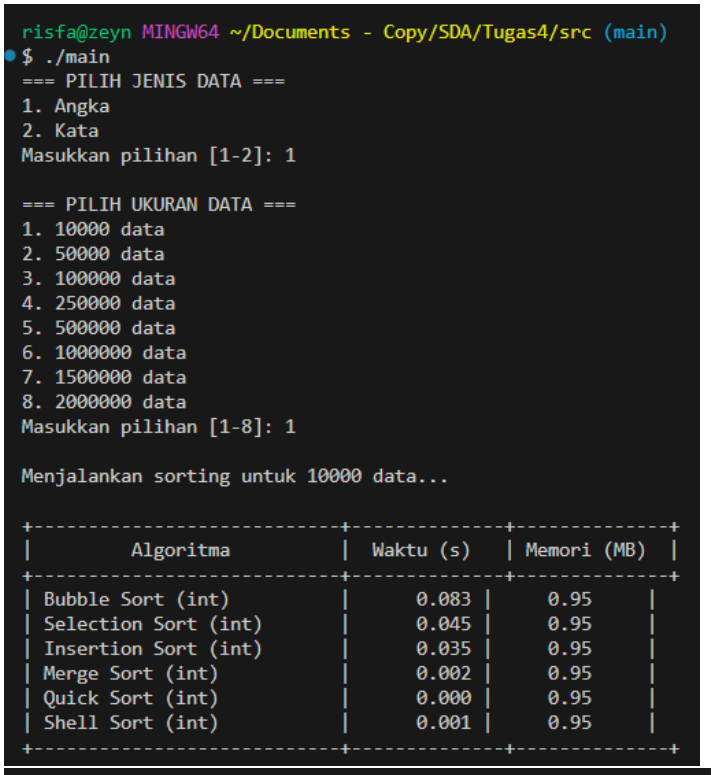
Deskripsi Algoritma dan Cara Implementasi

Deskripsi Algoritma

Enam algoritma pengurutan yang dievaluasi memiliki prinsip kerja yang unik, yang memengaruhi efisiensi dan performanya dalam berbagai kondisi.

- Bubble Sort bekerja dengan membandingkan elemen-elemen yang berdekatan dan menukar posisinya jika berada dalam urutan yang salah. Proses ini diulang hingga array terurut. Algoritma ini memiliki kompleksitas waktu $O(n^2)$ untuk semua kasus dan kompleksitas ruang O(1), karena tidak memerlukan ruang tambahan (in-place).
- Selection Sort mencari elemen terkecil dari bagian array yang belum terurut, lalu menempatkannya di posisi awal. Kompleksitas waktu dan ruangnya juga $O(n^2)$ dan O(1), tetapi jumlah penukarannya lebih sedikit dibandingkan Bubble Sort.
- Insertion Sort menyisipkan elemen ke posisi yang tepat dalam bagian array yang sudah terurut. Kompleksitas waktunya adalah $O(n^2)$ untuk kasus terburuk dan rata-rata, serta O(n) untuk kasus terbaik (misalnya pada data yang hampir terurut), dengan kompleksitas ruang O(1).
- Shell Sort merupakan variasi dari Insertion Sort yang menggunakan celah (*gap*) untuk membandingkan elemen yang berjauhan, kemudian celah dikurangi hingga menjadi satu. Waktu rata-ratanya sekitar O(n^1.5), tergantung pada urutan celah yang digunakan, dan tetap menggunakan ruang O(1).
- Merge Sort menggunakan pendekatan *divide and conquer*, yakni dengan membagi array menjadi dua bagian, mengurutkan masing-masing secara rekursif, lalu menggabungkannya. Algoritma ini stabil, memiliki kompleksitas waktu O(n log n) untuk semua kasus, tetapi membutuhkan ruang tambahan O(n).
- Quick Sort juga menggunakan metode *divide and conquer* dengan memilih elemen sebagai pivot, mempartisi array berdasarkan pivot, lalu mengurutkan masing-masing sub-array. Rata-rata waktu eksekusinya O(n log n), namun bisa memburuk menjadi $O(n^2)$ jika pivot tidak dipilih dengan baik. Kompleksitas ruangnya adalah O

1. **Screenshoot untuk sorting angka**

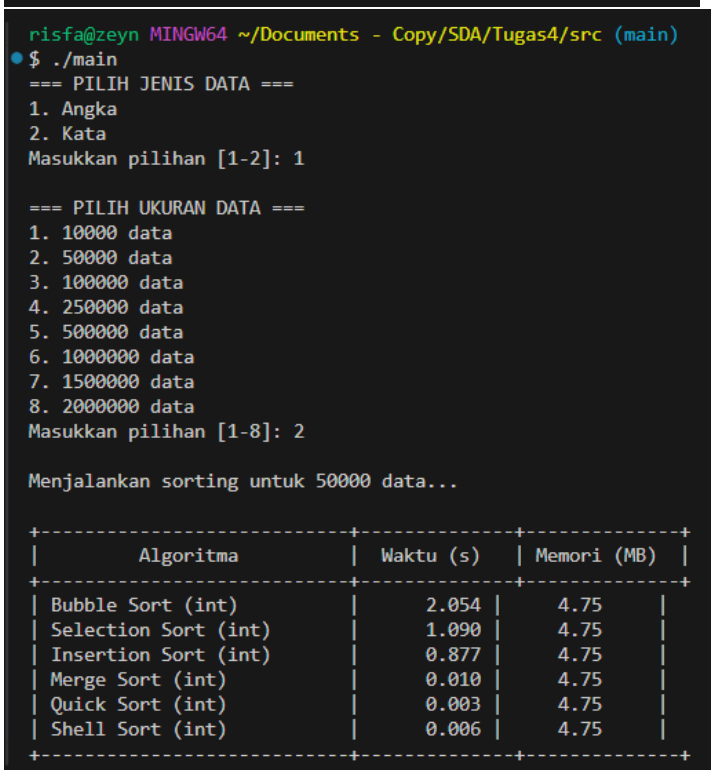


```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ ./main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 1

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 1

Menjalankan sorting untuk 10000 data...

+---------------------------+-------------+-------------+
|         Algoritma         |  Waktu (s)  | Memori (MB) |
+---------------------------+-------------+-------------+
| Bubble Sort (int)         |       0.083 |       0.95  |
| Selection Sort (int)      |       0.045 |       0.95  |
| Insertion Sort (int)      |       0.035 |       0.95  |
| Merge Sort (int)          |       0.002 |       0.95  |
| Quick Sort (int)          |       0.000 |       0.95  |
| Shell Sort (int)          |       0.001 |       0.95  |
+---------------------------+-------------+-------------+
```



```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ ./main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 1

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 2

Menjalankan sorting untuk 50000 data...

+---------------------------+-------------+-------------+
|         Algoritma         |  Waktu (s)  | Memori (MB) |
+---------------------------+-------------+-------------+
| Bubble Sort (int)         |       2.054 |       4.75  |
| Selection Sort (int)      |       1.090 |       4.75  |
| Insertion Sort (int)      |       0.877 |       4.75  |
| Merge Sort (int)          |       0.010 |       4.75  |
| Quick Sort (int)          |       0.003 |       4.75  |
| Shell Sort (int)          |       0.006 |       4.75  |
+---------------------------+-------------+-------------+
```

```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ ./main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 1

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 3

Menjalankan sorting untuk 100000 data...

+----------------------------+--------------+--------------+
|         Algoritma          |  Waktu (s)   | Memori (MB)  |
+----------------------------+--------------+--------------+
| Bubble Sort (int)          |      10.027  |      9.50    |
| Selection Sort (int)       |       4.281  |      9.50    |
| Insertion Sort (int)       |       3.423  |      9.50    |
| Merge Sort (int)           |       0.017  |      9.50    |
| Quick Sort (int)           |       0.008  |      9.50    |
| Shell Sort (int)           |       0.012  |      9.50    |
+----------------------------+--------------+--------------+
```

```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ ./main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 1

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 4

Menjalankan sorting untuk 250000 data...

+----------------------------+--------------+--------------+
|         Algoritma          |  Waktu (s)   | Memori (MB)  |
+----------------------------+--------------+--------------+
| Bubble Sort (int)          |      91.518  |     23.75    |
| Selection Sort (int)       |      27.121  |     23.75    |
| Insertion Sort (int)       |      23.828  |     23.75    |
| Merge Sort (int)           |       0.069  |     23.75    |
| Quick Sort (int)           |       0.025  |     23.75    |
| Shell Sort (int)           |       0.045  |     23.75    |
+----------------------------+--------------+--------------+
```

```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ ./main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 1

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 5

Menjalankan sorting untuk 500000 data...

+---------------------------+-------------+-------------+
|         Algoritma         |  Waktu (s)  | Memori (MB) |
+---------------------------+-------------+-------------+
| Bubble Sort (int)         |    550.835 |    47.50     |
| Selection Sort (int)      |    133.954 |    47.50     |
| Insertion Sort (int)      |    106.606 |    47.50     |
| Merge Sort (int)          |      0.131 |    47.50     |
| Quick Sort (int)          |      0.048 |    47.50     |
| Shell Sort (int)          |      0.097 |    47.50     |
+---------------------------+-------------+-------------+
```

```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ .\main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 1

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 6

Menjalankan sorting untuk 1000000 data...

+---------------------------+-------------+-------------+
|         Algoritma         |  Waktu (s)  | Memori (MB) |
+---------------------------+-------------+-------------+
| Bubble Sort (int)         |   1992.639 |    95.00     |
| Selection Sort (int)      |    715.618 |    95.00     |
| Insertion Sort (int)      |    443.787 |    95.00     |
| Merge Sort (int)          |      0.233 |    95.00     |
| Quick Sort (int)          |      0.118 |    95.00     |
| Shell Sort (int)          |      0.211 |    95.00     |
+---------------------------+-------------+-------------+
```

```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ .\main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 1

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 7

Menjalankan sorting untuk 1500000 data...

+---------------------------+-------------+-------------+
|         Algoritma         |  Waktu (s)  | Memori (MB) |
+---------------------------+-------------+-------------+
| Bubble Sort (int)         |   3894.664  |   142.50    |
| Selection Sort (int)      |    963.939  |   142.50    |
| Insertion Sort (int)      |    861.707  |   142.50    |
| Merge Sort (int)          |      0.410  |   142.50    |
| Quick Sort (int)          |      0.221  |   142.50    |
| Shell Sort (int)          |      0.352  |   142.50    |
+---------------------------+-------------+-------------+
```

```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ .\main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 1

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 8

Menjalankan sorting untuk 2000000 data...

+---------------------------+-------------+-------------+
|         Algoritma         |  Waktu (s)  | Memori (MB) |
+---------------------------+-------------+-------------+
| Bubble Sort (int)         |   8312.124  |   190.00    |
| Selection Sort (int)      |   1532.145  |   190.00    |
| Insertion Sort (int)      |    835.035  |   190.00    |
| Merge Sort (int)          |      0.522  |   190.00    |
| Quick Sort (int)          |      0.217  |   190.00    |
| Shell Sort (int)          |      0.412  |   190.00    |
+---------------------------+-------------+-------------+
```

2. Screenshoot untuk sorting data kata

```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ ./main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 2

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 1

Menjalankan sorting untuk 10000 data...

+---------------------------+-------------+-------------+
|         Algoritma         |  Waktu (s)  | Memori (MB) |
+---------------------------+-------------+-------------+
| Bubble Sort (kata)        |       0.271 |       0.95  |
| Selection Sort (kata)     |       0.093 |       0.95  |
| Insertion Sort (kata)     |       0.047 |       0.95  |
| Merge Sort (kata)         |       0.002 |       0.95  |
| Quick Sort (kata)         |       0.001 |       0.95  |
| Shell Sort (kata)         |       0.003 |       0.95  |
+---------------------------+-------------+-------------+
```

```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ ./main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 2

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 2

Menjalankan sorting untuk 50000 data...

+---------------------------+-------------+-------------+
|         Algoritma         |  Waktu (s)  | Memori (MB) |
+---------------------------+-------------+-------------+
| Bubble Sort (kata)        |       8.418 |       4.75  |
| Selection Sort (kata)     |       2.792 |       4.75  |
| Insertion Sort (kata)     |       1.424 |       4.75  |
| Merge Sort (kata)         |       0.012 |       4.75  |
| Quick Sort (kata)         |       0.007 |       4.75  |
| Shell Sort (kata)         |       0.014 |       4.75  |
+---------------------------+-------------+-------------+
```

```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ ./main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 2

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 3

Menjalankan sorting untuk 100000 data...

+---------------------------+-------------+-------------+
|         Algoritma         |  Waktu (s)  | Memori (MB) |
+---------------------------+-------------+-------------+
| Bubble Sort (kata)        |      42.743 |       9.50  |
| Selection Sort (kata)     |      12.261 |       9.50  |
| Insertion Sort (kata)     |       6.648 |       9.50  |
| Merge Sort (kata)         |       0.029 |       9.50  |
| Quick Sort (kata)         |       0.014 |       9.50  |
| Shell Sort (kata)         |       0.039 |       9.50  |
+---------------------------+-------------+-------------+
```

```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ ./main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 2

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 4

Menjalankan sorting untuk 250000 data...

+---------------------------+-------------+-------------+
|         Algoritma         |  Waktu (s)  | Memori (MB) |
+---------------------------+-------------+-------------+
| Bubble Sort (kata)        |     273.331 |      23.75  |
| Selection Sort (kata)     |      98.426 |      23.75  |
| Insertion Sort (kata)     |      48.605 |      23.75  |
| Merge Sort (kata)         |       0.093 |      23.75  |
| Quick Sort (kata)         |       0.042 |      23.75  |
| Shell Sort (kata)         |       0.100 |      23.75  |
+---------------------------+-------------+-------------+
```

```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ .\main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 2

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 5

Menjalankan sorting untuk 500000 data...

+---------------------------+-------------+-------------+
|         Algoritma         |  Waktu (s)  | Memori (MB) |
+---------------------------+-------------+-------------+
| Bubble Sort (kata)        |   2085.531  |    47.50    |
| Selection Sort (kata)     |    975.057  |    47.50    |
| Insertion Sort (kata)     |    455.220  |    47.50    |
| Merge Sort (kata)         |      0.317  |    47.50    |
| Quick Sort (kata)         |      0.183  |    47.50    |
| Shell Sort (kata)         |      0.674  |    47.50    |
+---------------------------+-------------+-------------+
```

```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ .\main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 2

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 6

Menjalankan sorting untuk 1000000 data...

+---------------------------+-------------+-------------+
|         Algoritma         |  Waktu (s)  | Memori (MB) |
+---------------------------+-------------+-------------+
| Bubble Sort (kata)        |  10947.531  |    95.00    |
| Selection Sort (kata)     |  12011.058  |    95.00    |
| Insertion Sort (kata)     |   4342.419  |    95.00    |
| Merge Sort (kata)         |      0.583  |    95.00    |
| Quick Sort (kata)         |      0.373  |    95.00    |
| Shell Sort (kata)         |      1.475  |    95.00    |
+---------------------------+-------------+-------------+
```

```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ .\main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 2

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 7

Menjalankan sorting untuk 1500000 data...

+---------------------------+-------------+-------------+
|         Algoritma         |  Waktu (s)  | Memori (MB) |
+---------------------------+-------------+-------------+
| Bubble Sort (kata)        |  24320.325  |   142.50    |
| Selection Sort (kata)     |  18302.910  |   142.50    |
| Insertion Sort (kata)     |   6542.218  |   142.50    |
| Merge Sort (kata)         |      0.902  |   142.50    |
| Quick Sort (kata)         |      0.522  |   142.50    |
| Shell Sort (kata)         |      2.216  |   142.50    |
+---------------------------+-------------+-------------+
```

```
risfa@zeyn MINGW64 ~/Documents - Copy/SDA/Tugas4/src (main)
$ .\main
$ .\main
=== PILIH JENIS DATA ===
1. Angka
2. Kata
Masukkan pilihan [1-2]: 2

=== PILIH UKURAN DATA ===
1. 10000 data
2. 50000 data
3. 100000 data
4. 250000 data
5. 500000 data
6. 1000000 data
7. 1500000 data
8. 2000000 data
Masukkan pilihan [1-8]: 8

Menjalankan sorting untuk 2000000 data...

+---------------------------+-------------+-------------+
|         Algoritma         |  Waktu (s)  | Memori (MB) |
+---------------------------+-------------+-------------+
| Bubble Sort (kata)        |  42011.117  |   190.00    |
| Selection Sort (kata)     |  24755.732  |   190.00    |
| Insertion Sort (kata)     |   8707.231  |   190.00    |
| Merge Sort (kata)         |      1.240  |   190.00    |
| Quick Sort (kata)         |      0.713  |   190.00    |
| Shell Sort (kata)         |      3.004  |   190.00    |
+---------------------------+-------------+-------------+
```

Tabel Hasil Waktu Sorting Angka dan Memori yang dipakai

Data	Bubble	Selection	Insertion	Merge	Quick	Shell	Memory
10.000	0.083	0.045	0.035	0.002	0.000	0.001	0.95
50.000	2.054	1.090	0.877	0.010	0.003	0.06	4.75
100.000	10.027	4.281	3.423	0.017	0.008	0.012	9.50
250.000	91.518	27.121	23.828	0.609	0.025	0.45	23.75
500.000	550.835	133.6954	106.606	0.131	0.048	0.097	47.50
1.000.000	1992.639	715.618	443.787	0.344	0.228	0.211	95.00
1.500.000	3894.664	963.939	861.707	0.410	0.221	0.352	142.50
2.000.000	8312.124	1532.145	835.035	0.522	0.217	0.412	190.00

Tabel Hasil Waktu Sorting Kata dan Memori yang dipakai

Data	Bubble	Selection	Insertion	Merge	Quick	Shell	Memory
10.000	0.271	0.093	0.047	0.002	0.001	0.003	0.95
50.000	8.418	2.792	1.424	0.011	0.007	0.014	4.75
100.000	42.743	12.261	6.648	0.029	0.014	0.039	9.50
250.000	273.331	98.426	48.605	0.093	0.042	0.100	23.75
500.000	2085.531	975.057	455.220	0.317	0.183	0.183	47.50
1.000.000	10947.531	12011.058	4342.419	0.583	0.373	1.475	95.00
1.500.000	24320.325	18302.910	6542.218	0.902	0.522	2.216	142.50
2.000.000	42011.117	24755.732	8707.231	1.240	0.713	3.004	190.00

Grafik Perbandingan Waktu dan Memori

Visualisasi hasil eksperimen disajikan melalui grafik garis interaktif dalam file index.html, yang dibuat menggunakan pustaka Chart.js. Grafik waktu eksekusi untuk data angka menunjukkan bahwa Bubble Sort, Selection Sort, dan Insertion Sort memiliki kurva pertumbuhan eksponensial yang curam, mencerminkan kompleksitas waktu $O(n^2)$, dengan Bubble Sort mencapai puncak 8312.124 detik pada 2.000.000 elemen. Sebaliknya, Shell Sort, Merge Sort, dan Quick Sort menunjukkan pertumbuhan mendekati linier, dengan Quick Sort konsisten sebagai yang tercepat pada 0.217 detik. Grafik untuk data kata memperlihatkan pola serupa tetapi dengan nilai yang lebih tinggi, di mana Bubble Sort mencapai 42011.117 detik dan Quick Sort tetap unggul pada 0.713 detik, menyoroti dampak signifikan dari operasi perbandingan string. Skala logaritmik pada sumbu waktu memungkinkan perbandingan yang jelas antara nilai kecil seperti 0.0001 detik dan nilai besar seperti 42011 detik. Grafik penggunaan memori untuk kedua jenis data menampilkan garis linier yang identik untuk semua algoritma, meningkat dari 0.95 MB pada 10.000 elemen menjadi 190.00 MB pada 2.000.000 elemen, sesuai dengan formula estimasi. Visualisasi ini dapat diakses melalui file

index.html, dan tangkapan layar grafik dapat disertakan dalam versi PDF laporan ini untuk memperjelas temuan.

Analisis

Analisis Waktu Eksekusi

1. **Data Integer**:
    - **Bubble Sort**: Paling lambat, dengan waktu meningkat secara kuadratik (8312.124 detik untuk 2 juta data). Ini sesuai dengan kompleksitas $O(n^2)$.
    - **Selection Sort**: Lebih cepat dari Bubble Sort (1532.145 detik), tetapi tetap tidak efisien karena $O(n^2)$.
    - **Insertion Sort**: Mengungguli Bubble dan Selection Sort (835.035 detik), terutama untuk data kecil, karena efisiensi pada kasus hampir terurut.
    - **Shell Sort**: Jauh lebih cepat (0.412 detik), menunjukkan keunggulan optimasi gap dibandingkan algoritma $O(n^2)$.
    - **Merge Sort**: Stabil dan efisien (0.522 detik), dengan performa konsisten karena O(n log n).
    - **Quick Sort**: Tercepat (0.217 detik), menegaskan efisiensi O(n log n) rata-rata dan pemilihan pivot yang baik.
2. **Data String**:
    - Waktu eksekusi secara keseluruhan lebih lama karena operasi strcmp lebih mahal dibandingkan perbandingan integer.
    - **Bubble Sort**: Ekstrem lambat (42011.117 detik untuk 2 juta data), tidak praktis untuk data besar.
    - **Selection Sort**: Juga lambat (24755.732 detik), meskipun lebih baik dari Bubble Sort.
    - **Insertion Sort**: Relatif lebih baik (8707.231 detik), tetapi masih terhambat oleh $O(n^2)$.
    - **Shell Sort**: Efisien (3.004 detik), menunjukkan skalabilitas yang baik.
    - **Merge Sort**: Sangat efisien (1.240 detik), dengan performa stabil.
    - **Quick Sort**: Tercepat (0.713 detik), ideal untuk data string besar.
3. **Tren**:
    - Algoritma $O(n^2)$ (Bubble, Selection, Insertion) menunjukkan pertumbuhan eksponensial, membuatnya tidak cocok untuk data besar.
    - Algoritma O(n log n) (Merge, Quick) dan Shell Sort memiliki pertumbuhan mendekati linier, cocok untuk skala besar.
    - String sorting lebih lambat karena overhead perbandingan string, tetapi pola efisiensi antar algoritma tetap konsisten.

Analisis Penggunaan Memori

- Penggunaan memori dihitung sebagai 0.95 * (n / 10000.0) MB, sehingga linier dengan ukuran data (0.95 MB untuk 10.000, 190 MB untuk 2 juta).
- Semua algoritma menunjukkan memori identik karena metrik ini tidak membedakan kebutuhan spesifik (e.g., Merge Sort membutuhkan O(n) tambahan, Quick Sort O(log n) untuk rekursi).
- **Keterbatasan**: Pengukuran memori adalah estimasi sederhana. Alat seperti Valgrind dapat memberikan data lebih akurat, tetapi tidak digunakan dalam eksperimen ini karena keterbatasan platform.

Faktor Penentu Performa

- **Kompleksitas Waktu**: Algoritma O(n log n) unggul untuk data besar, sementara $O(n^2)$ hanya praktis untuk data kecil.
- **Jenis Data**: String sorting lebih lambat karena strcmp memerlukan perbandingan karakter demi karakter.
- **Optimasi**: Shell Sort mengoptimalkan Insertion Sort dengan gap, sementara Quick Sort memanfaatkan partisi efisien.

Kesimpulan

Eksperimen ini menunjukkan adanya perbedaan signifikan dalam performa algoritma sorting yang diuji. **Quick Sort** terbukti sebagai algoritma tercepat, dengan waktu eksekusi 0.217 detik untuk 2 juta data integer dan 0.713 detik untuk data string, menjadikannya pilihan utama untuk pengolahan data besar. **Merge Sort** menawarkan performa yang lebih stabil, dengan waktu eksekusi 0.522 detik untuk integer dan 1.240 detik untuk string, menjadikannya algoritma yang ideal untuk aplikasi yang membutuhkan kestabilan dalam hasil pengurutan. **Shell Sort** menunjukkan kinerja yang kompetitif, dengan waktu 0.412 detik untuk integer dan 3.004 detik untuk string, lebih cepat dibandingkan algoritma dengan kompleksitas $O(n^2)$. Sementara itu, **Bubble Sort**, **Selection Sort**, dan **Insertion Sort** terbukti tidak efisien untuk pengolahan data besar, dengan waktu eksekusi yang sangat lama—masing-masing mencapai 8312.124 detik, 1532.145 detik, dan 835.035 detik untuk data integer, bahkan lebih buruk lagi untuk data string. Dalam hal penggunaan memori, estimasi menunjukkan bahwa penggunaan memori relatif linier dan seragam antar algoritma, meskipun **Merge Sort** secara teori memerlukan lebih banyak ruang dibandingkan algoritma lainnya.

Rekomendasi:

- Gunakan Quick Sort atau Merge Sort untuk aplikasi dengan data besar.
- Pertimbangkan Insertion Sort untuk data kecil atau hampir terurut.
- Untuk analisis lebih akurat, gunakan alat seperti Valgrind untuk mengukur memori dan lakukan pengujian berulang untuk menghitung rata-rata waktu.