

Sorting Algorithm

disusun untuk memenuhi
mata kuliah Struktur Data & Algoritma

Oleh:

Muhammad Ikram Ramadhana Friyan
(2308107010055)



JURUSAN INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SYIAH KUALA
DARUSSALAM, BANDA ACEH
2025

Tujuan

Menganalisis performa berbagai algoritma sorting terhadap data dalam jumlah besar (hingga 2 juta) dengan tipe data angka dan kata.

Algoritma Sorting yang Diimplementasikan

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

Struktur File Program

- main.c → Program utama uji performa
- sorting.h → Header file berisi semua algoritma sorting
- data_angka.txt → Data random integer sebanyak 2 juta baris
- data_kata.txt → Data random string sebanyak 2 juta baris

Spesifikasi Uji

- Bahasa: C
- Compiler: GCC
- Platform: Visual Studio Code
- Metode pengukuran waktu: clock() dari <time.h>
- Set ukuran data: { 10.000, 50.000, 100.000, 250.000, 500.000, 1.000.000, 1.500.000, 2.000.000 }

Cara Compile

```
● muhammad-ikram-ramadhana-friyan@muhammad-ikram-ramadhana-friyan-Modern-14-C7M:~/Documents/2308107010055_Tugas4_SDA_2025$ gcc main.c -o sorting
● muhammad-ikram-ramadhana-friyan@muhammad-ikram-ramadhana-friyan-Modern-14-C7M:~/Documents/2308107010055_Tugas4_SDA_2025$ ./sorting
```

Hasil Pengujian

Data Angka Pengujian 10.000	Data Kata Pengujian 10.000
<pre>=== UJI DENGAN 10000 DATA === -- Data Angka -- Bubble Sort: 0.196 detik Memori: 87496 KB Selection Sort: 0.103 detik Memori: 87496 KB Insertion Sort: 0.076 detik Memori: 87496 KB Merge Sort: 0.002 detik Memori: 87496 KB Quick Sort: 0.001 detik Memori: 87496 KB Shell Sort: 0.003 detik Memori: 87496 KB</pre>	<pre>-- Data Kata -- Bubble Sort (kata): 0.726 detik Memori: 87880 KB Selection Sort (kata): 0.372 detik Memori: 87880 KB Insertion Sort (kata): 0.225 detik Memori: 87880 KB Merge Sort (kata): 0.003 detik Memori: 87880 KB Quick Sort (kata): 0.003 detik Memori: 87880 KB Shell Sort (kata): 0.005 detik Memori: 87880 KB</pre>
Data Angka Pengujian 50.000	Data Kata Pengujian 50.000

<pre> === UJI DENGAN 50000 DATA === -- Data Angka -- Bubble Sort: 7.277 detik Memori: 88008 KB Selection Sort: 1.216 detik Memori: 88008 KB Insertion Sort: 0.928 detik Memori: 88008 KB Merge Sort: 0.005 detik Memori: 88008 KB Quick Sort: 0.004 detik Memori: 88008 KB Shell Sort: 0.009 detik Memori: 88008 KB </pre>		<pre> -- Data Kata -- Bubble Sort (kata): 10.130 detik Memori: 89476 KB Selection Sort (kata): 4.983 detik Memori: 89492 KB Insertion Sort (kata): 3.439 detik Memori: 89580 KB Merge Sort (kata): 0.010 detik Memori: 89708 KB Quick Sort (kata): 0.008 detik Memori: 89708 KB Shell Sort (kata): 0.021 detik Memori: 89708 KB </pre>	
Data Angka Pengujian 100.000		Data Kata Pengujian 100.000	
<pre> === UJI DENGAN 100000 DATA === -- Data Angka -- Bubble Sort: 17.151 detik Memori: 89708 KB Selection Sort: 4.851 detik Memori: 89708 KB Insertion Sort: 3.884 detik Memori: 89708 KB Merge Sort: 0.012 detik Memori: 89708 KB Quick Sort: 0.009 detik Memori: 89708 KB Shell Sort: 0.021 detik Memori: 89708 KB </pre>		<pre> -- Data Kata -- Bubble Sort (kata): 42.894 detik Memori: 91628 KB Selection Sort (kata): 21.359 detik Memori: 91628 KB Insertion Sort (kata): 14.253 detik Memori: 91628 KB Merge Sort (kata): 0.026 detik Memori: 92008 KB Quick Sort (kata): 0.020 detik Memori: 92008 KB Shell Sort (kata): 0.051 detik Memori: 92008 KB </pre>	
Data Angka Pengujian 250.000		Data Kata Pengujian 250.000	
<pre> === UJI DENGAN 250000 DATA === -- Data Angka -- Bubble Sort: 114.984 detik Memori: 92008 KB Selection Sort: 30.371 detik Memori: 92008 KB Insertion Sort: 24.176 detik Memori: 92008 KB Merge Sort: 0.031 detik Memori: 92136 KB Quick Sort: 0.023 detik Memori: 92136 KB Shell Sort: 0.052 detik Memori: 92136 KB </pre>		<pre> -- Data Kata -- Bubble Sort (kata): 269.021 detik Memori: 98024 KB Selection Sort (kata): 135.192 detik Memori: 98024 KB Insertion Sort (kata): 96.642 detik Memori: 98024 KB Merge Sort (kata): 0.063 detik Memori: 99012 KB Quick Sort (kata): 0.053 detik Memori: 99012 KB Shell Sort (kata): 0.131 detik Memori: 99012 KB </pre>	
Data Angka Pengujian 500.000		Data Kata Pengujian 500.000	
<pre> === UJI DENGAN 500000 DATA === -- Data Angka -- Bubble Sort: 466.222 detik Memori: 99012 KB Selection Sort: 121.629 detik Memori: 99012 KB Insertion Sort: 97.742 detik Memori: 99012 KB Merge Sort: 0.065 detik Memori: 99012 KB Quick Sort: 0.049 detik Memori: 99012 KB Shell Sort: 0.115 detik Memori: 99012 KB </pre>		<pre> -- Data Kata -- Bubble Sort (kata): 1495.295 detik Memori: 108740 KB Selection Sort (kata): 767.552 detik Memori: 108740 KB Insertion Sort (kata): 602.449 detik Memori: 108740 KB Merge Sort (kata): 0.129 detik Memori: 110660 KB Quick Sort (kata): 0.110 detik Memori: 110660 KB Shell Sort (kata): 0.432 detik Memori: 110660 KB </pre>	
Data Angka Pengujian 1.000.000		Data Kata Pengujian 1.000.000	
<pre> === UJI DENGAN 1000000 DATA === -- Data Angka -- Bubble Sort: 1845.506 detik Memori: 110660 KB Selection Sort: 490.963 detik Memori: 110660 KB Insertion Sort: 390.301 detik Memori: 110660 KB Merge Sort: 0.136 detik Memori: 110660 KB Quick Sort: 0.102 detik Memori: 110660 KB Shell Sort: 0.251 detik Memori: 110660 KB </pre>		<pre> -- Data Kata -- Bubble Sort (kata): 9752.865 detik Memori: 130244 KB Selection Sort (kata): 5426.007 detik Memori: 130272 KB Insertion Sort (kata): 4779.845 detik Memori: 130272 KB Merge Sort (kata): 0.300 detik Memori: 134112 KB Quick Sort (kata): 0.294 detik Memori: 134112 KB Shell Sort (kata): 1.387 detik Memori: 134112 KB </pre>	
Data Angka Pengujian 1.500.000		Data Kata Pengujian 1.500.000	

<pre> === UJI DENGAN 1500000 DATA === -- Data Angka -- Bubble Sort: 4172.098 detik Memori: 92996 KB Selection Sort: 1097.105 detik Memori: 92996 KB Insertion Sort: 880.096 detik Memori: 92996 KB Merge Sort: 0.215 detik Memori: 98784 KB Quick Sort: 0.158 detik Memori: 98784 KB Shell Sort: 0.413 detik Memori: 98784 KB </pre>	<pre> -- Data Kata -- Bubble Sort (kata): 25112.348 detik Memori: 151520 KB Selection Sort (kata): 13386.077 detik Memori: 151520 KB Insertion Sort (kata): 12242.267 detik Memori: 151520 KB Segmentation fault (core dumped) </pre> <p>Disini terjadi core dumped, saat pengetesan di atas 12 jam</p>
Data Angka Pengujian 2.000.000	Data Kata Pengujian 2.000.000
<pre> Segmentation fault (core dumped) • muhammad-ikram-ramadhana-friyan@muhammad-ikr ○ muhammad-ikram-ramadhana-friyan@muhammad-ikr === UJI DENGAN 2000000 DATA === -- Data Angka -- </pre> <p>Tidak jalan sama sekali, lebih dari pada 12 jam pada pengujian 2.000.000</p>	<pre> Segmentation fault (core dumped) • muhammad-ikram-ramadhana-friyan@muhammad-ikr ○ muhammad-ikram-ramadhana-friyan@muhammad-ikr === UJI DENGAN 2000000 DATA === -- Data Angka -- </pre> <p>Tidak jalan sama sekali, lebih dari pada 12 jam pada pengujian 2.000.000</p>

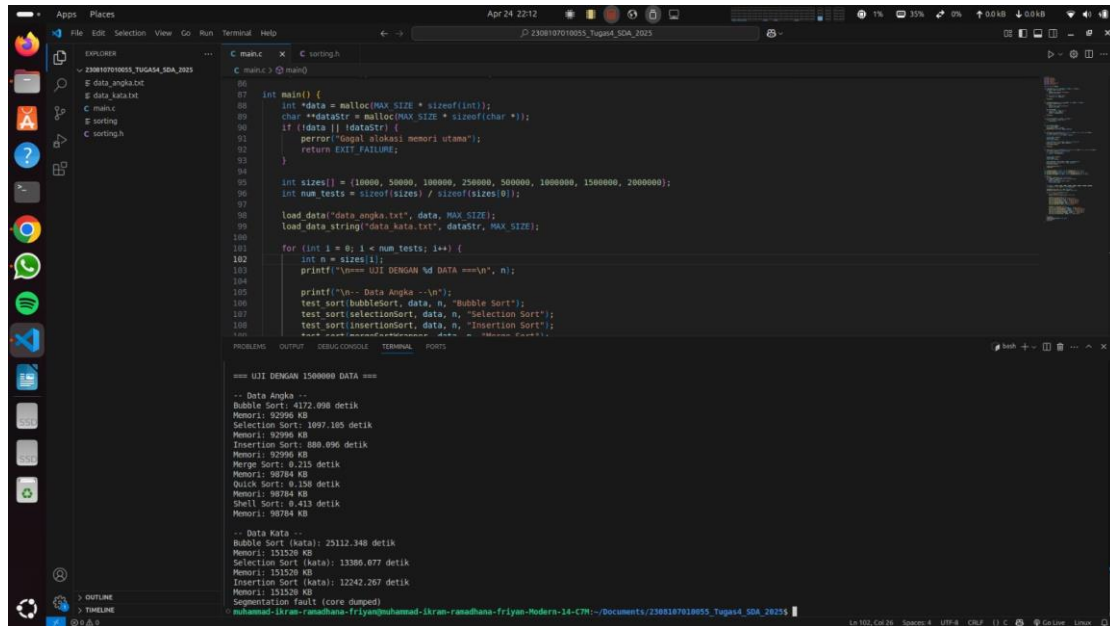
Analisis dan Penjelasan

1. Bubble Sort: Kompleksitas $O(n^2)$, sangat lambat untuk data besar. Hanya cocok untuk dataset kecil.
2. Selection Sort: $O(n^2)$, meskipun swap lebih sedikit dari Bubble Sort, tetap tidak efisien.
3. Insertion Sort: Lebih baik untuk dataset kecil, tapi tetap $O(n^2)$.
4. Merge Sort: Cepat dan stabil, cocok untuk data besar.
5. Quick Sort: Umumnya paling cepat, cocok untuk semua ukuran data.
6. Shell Sort: Alternatif efisien dan sederhana, cocok untuk dataset besar.

Kesimpulan

Merge, Quick, dan Shell Sort adalah algoritma terbaik untuk dataset besar. Bubble, Selection, dan Insertion tidak layak dipakai di atas 100.000 data. Sorting string lebih berat karena penggunaan memori untuk salinan string. Quick Sort menunjukkan performa terbaik untuk kombinasi efisiensi dan kecepatan.

Lampiran

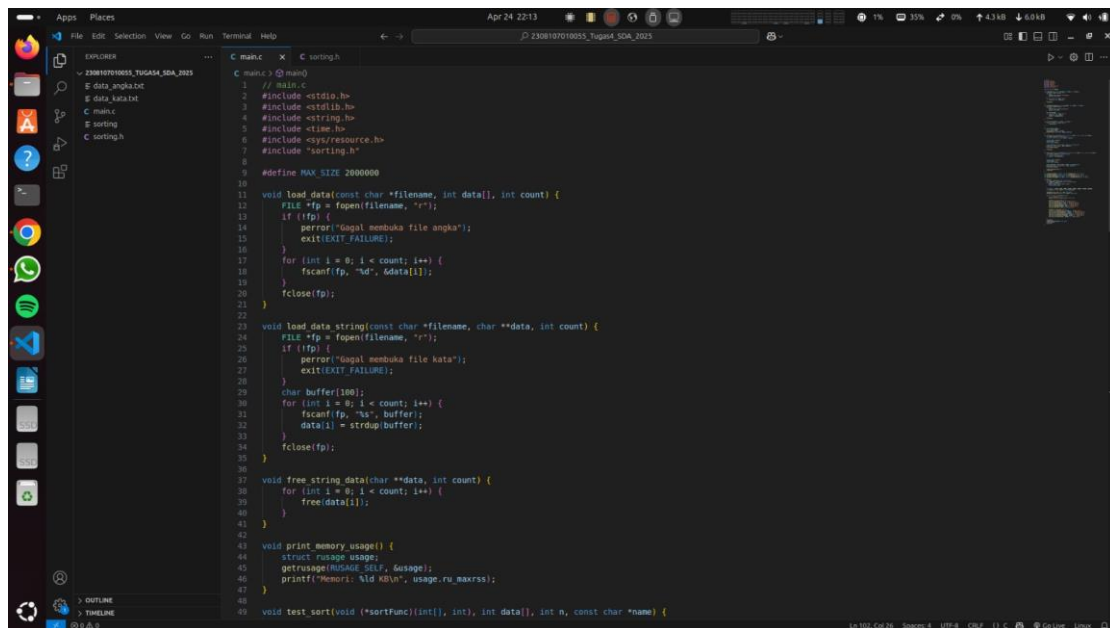


This screenshot shows a C++ IDE with a file explorer on the left containing files like `E_data_angka.txt`, `E_data_kata.txt`, `C_main.c`, `E_sorting`, and `C_sorting.h`. The main editor displays the `main` function of `C_sorting.h`. The code defines arrays of sizes and performs benchmarks for Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, and Shell Sort on two datasets: 'Data Angka' and 'Data Kata'. The terminal window at the bottom shows the execution output, including memory usage and execution time for each sorting algorithm on both datasets.

```
1 // main.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5 #include <time.h>
6 #include <sys/resource.h>
7 #include "sorting.h"
8
9 #define MAX_SIZE 2000000
10
11 void load_data(const char *filename, int data[], int count) {
12     FILE *fp = fopen(filename, "r");
13     if (!fp) {
14         perror("Gagal membuka file angka");
15         exit(EXIT_FAILURE);
16     }
17     for (int i = 0; i < count; i++) {
18         fscanf(fp, "%d", &data[i]);
19     }
20     fclose(fp);
21 }
22
23 void load_data_string(const char *filename, char **data, int count) {
24     FILE *fp = fopen(filename, "r");
25     if (!fp) {
26         perror("Gagal membuka file kata");
27         exit(EXIT_FAILURE);
28     }
29     char buffer[100];
30     for (int i = 0; i < count; i++) {
31         fscanf(fp, "%s", buffer);
32         data[i] = strdup(buffer);
33     }
34     fclose(fp);
35 }
36
37 void free_string_data(char **data, int count) {
38     for (int i = 0; i < count; i++) {
39         free(data[i]);
40     }
41 }
42
43 void print_memory_usage() {
44     struct rusage usage;
45     getrusage(RUSAGE_SELF, &usage);
46     printf("Memori: %ld KB\n", usage.ru_maxrss);
47 }
48
49 void test_sort(void (*sortFunc)(int[], int), int data[], int n, const char *name) {
50     // Benchmarking code
51 }
```

Output from terminal:

```
=== UJI DENGAN 1500000 DATA ===
-- Data Angka --
Bubble Sort: 4172.098 detik
Memori: 92996 KB
Selection Sort: 1097.105 detik
Memori: 92996 KB
Insertion Sort: 880.696 detik
Memori: 92996 KB
Merge Sort: 0.215 detik
Memori: 98784 KB
Quick Sort: 0.158 detik
Memori: 98784 KB
Shell Sort: 0.413 detik
Memori: 98784 KB
-- Data Kata --
Bubble Sort (kata): 23112.348 detik
Memori: 151520 KB
Selection Sort (kata): 13386.677 detik
Memori: 151520 KB
Insertion Sort (kata): 12242.267 detik
Memori: 151520 KB
Segmentation Fault (core dumped)
```



This screenshot shows the same C++ IDE with the `main` function of `C_sorting.h`. The code defines arrays of sizes and performs benchmarks for Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, and Shell Sort on two datasets: 'Data Angka' and 'Data Kata'. The terminal window at the bottom shows the execution output, including memory usage and execution time for each sorting algorithm on both datasets.

```
1 // main.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5 #include <time.h>
6 #include <sys/resource.h>
7 #include "sorting.h"
8
9 #define MAX_SIZE 2000000
10
11 void load_data(const char *filename, int data[], int count) {
12     FILE *fp = fopen(filename, "r");
13     if (!fp) {
14         perror("Gagal membuka file angka");
15         exit(EXIT_FAILURE);
16     }
17     for (int i = 0; i < count; i++) {
18         fscanf(fp, "%d", &data[i]);
19     }
20     fclose(fp);
21 }
22
23 void load_data_string(const char *filename, char **data, int count) {
24     FILE *fp = fopen(filename, "r");
25     if (!fp) {
26         perror("Gagal membuka file kata");
27         exit(EXIT_FAILURE);
28     }
29     char buffer[100];
30     for (int i = 0; i < count; i++) {
31         fscanf(fp, "%s", buffer);
32         data[i] = strdup(buffer);
33     }
34     fclose(fp);
35 }
36
37 void free_string_data(char **data, int count) {
38     for (int i = 0; i < count; i++) {
39         free(data[i]);
40     }
41 }
42
43 void print_memory_usage() {
44     struct rusage usage;
45     getrusage(RUSAGE_SELF, &usage);
46     printf("Memori: %ld KB\n", usage.ru_maxrss);
47 }
48
49 void test_sort(void (*sortFunc)(int[], int), int data[], int n, const char *name) {
50     // Benchmarking code
51 }
```

Output from terminal:

```
=== UJI DENGAN 1500000 DATA ===
-- Data Angka --
Bubble Sort: 4172.098 detik
Memori: 92996 KB
Selection Sort: 1097.105 detik
Memori: 92996 KB
Insertion Sort: 880.696 detik
Memori: 92996 KB
Merge Sort: 0.215 detik
Memori: 98784 KB
Quick Sort: 0.158 detik
Memori: 98784 KB
Shell Sort: 0.413 detik
Memori: 98784 KB
-- Data Kata --
Bubble Sort (kata): 23112.348 detik
Memori: 151520 KB
Selection Sort (kata): 13386.677 detik
Memori: 151520 KB
Insertion Sort (kata): 12242.267 detik
Memori: 151520 KB
Segmentation Fault (core dumped)
```

```
Apr 24 22:13
D:\2308107010055_Tugas4_SDA_2025

C source - C sorting.h
C sorting.h @ perl5dblib(1.9.10.1), 10, 10
// sorting.h
2 #ifndef SORTING_H
3 #define SORTING_H
4
5 #include <string.h>
6
7 // ===== ANGGKA =====
8
9 void bubbleSort(int arr[], int n) {
10     for (int i = 0; i < n - 1; i++) {
11         for (int j = 0; j < n - i - 1; j++) {
12             if (arr[j] > arr[j + 1]) {
13                 int temp = arr[j];
14                 arr[j] = arr[j + 1];
15                 arr[j + 1] = temp;
16             }
17         }
18     }
19 }
20
21 void selectionSort(int arr[], int n) {
22     for (int i = 0; i < n - 1; i++) {
23         int min_idx = i;
24         for (int j = i + 1; j < n; j++) {
25             if (arr[j] < arr[min_idx])
26                 min_idx = j;
27         }
28         int temp = arr[min_idx];
29         arr[min_idx] = arr[i];
30         arr[i] = temp;
31     }
32 }
33
34 void insertionSort(int arr[], int n) {
35     for (int i = 1; i < n; i++) {
36         int key = arr[i];
37         int j = i - 1;
38         while (j >= 0 && arr[j] > key) {
39             arr[j + 1] = arr[j];
40             j--;
41         }
42         arr[j + 1] = key;
43     }
44 }
45
46 void merge(int arr[], int l, int m, int r) {
47     int n1 = m - l + 1, n2 = r - m;
48     int L[n1], R[n2];
49     for (int i = 0; i < n1; i++) L[i] = arr[l + i];
50     for (int i = 0; i < n2; i++) R[i] = arr[m + 1 + i];
51     int i = 0, j = 0, k = l;
52     while (i < n1 && j < n2) {
53         if (L[i] < R[j]) arr[k] = L[i]; i++;
54         else arr[k] = R[j]; j++;
55         k++;
56     }
57     while (i < n1) arr[k] = L[i]; i++; k++;
58     while (j < n2) arr[k] = R[j]; j++; k++;
59 }
```