# TUGAS 4

disusun untuk memenuhi tugas mata kuliah
Struktur Data dan Algoritma

Oleh:

**Teuku Hafiz Izham**
**2308107010056**


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

# Pendahuluan

Tugas ini bertujuan untuk mengevaluasi performa berbagai algoritma pengurutan (Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, dan Shell Sort) berdasarkan waktu eksekusi dan penggunaan memori pada data skala besar (angka dan kata).

# Deskripsi Algoritma dan Implementasi

Implementasi algoritma ini tersedia dalam file sorting_algorithms.h, yang mencakup versi untuk data integer (int) dan string (char*). File ini berisi fungsi-fungsi pengurutan yang dirancang secara modular dan mudah dipahami, sehingga dapat digunakan kembali dalam berbagai eksperimen.

1. **Bubble Sort**
   Algoritma ini mengulang proses perbandingan dan pertukaran elemen berdekatan dalam array jika urutannya tidak sesuai. Proses berlanjut hingga tidak ada lagi pertukaran yang diperlukan. Untuk data string, perbandingan dilakukan dengan fungsi strcmp.

2. **Selection Sort**
   Selection Sort mencari elemen dengan nilai terkecil dari bagian array yang belum terurut, kemudian menukarnya dengan elemen di posisi awal bagian tersebut. Proses ini diulang hingga seluruh array terurut. Untuk string, pengurutan didasarkan pada urutan leksikografis.

3. **Insertion Sort**
   Insertion Sort menempatkan setiap elemen ke posisi yang sesuai dalam segmen array yang sudah terurut. Elemen-elemen yang lebih besar digeser untuk menyediakan tempat bagi elemen baru. Pada versi string, fungsi strcmp digunakan untuk menentukan urutan.

4. **Merge Sort**
   Mengadopsi strategi divide and conquer, Merge Sort membagi array menjadi dua bagian, mengurutkannya secara rekursif, lalu menggabungkan kembali hasilnya. Proses penggabungan membutuhkan memori tambahan. Untuk string, array pointer disalin dan digabung menggunakan perbandingan strcmp.

5. **Quick Sort**
   Quick Sort memilih sebuah elemen sebagai pivot, kemudian mempartisi array menjadi dua kelompok: elemen yang lebih kecil dan yang lebih besar dari pivot. Proses ini dilakukan secara rekursif. Partisi melibatkan perbandingan nilai atau string, dengan pertukaran elemen untuk menempatkannya di sisi yang sesuai relatif terhadap pivot.

6. **Shell Sort**
   Shell Sort adalah penyempurnaan dari Insertion Sort. Alih-alih membandingkan elemen berdekatan, algoritma ini membandingkan elemen dengan jarak tertentu (gap), yang secara bertahap dikurangi hingga array terurut.

**Implementasi**:

- Program ditulis dalam bahasa C dengan fungsi sorting terpisah di file header sorting_algorithms.h.
- File utama main.c menghasilkan data, membaca data, dan mengukur waktu eksekusi.
- Data dihasilkan menggunakan fungsi yang diberikan untuk 2 juta angka dan kata.

# Hasil Eksperimen

# Tabel Hasil

1. Data 10.000

```
--- Testing size: 10000 ---

+------------+------------+----------+----------------+------------------+
| Algorithm  | Data Type  | Size     | Time (seconds) | Est Mem (byte)   |
+------------+------------+----------+----------------+------------------+
| Bubble     | number     | 10000    | 0.365000       | 40000            |
| Selection  | number     | 10000    | 0.233000       | 40000            |
| Insertion  | number     | 10000    | 0.108000       | 40000            |
| Merge      | number     | 10000    | 0.005000       | 40000            |
| Quick      | number     | 10000    | 0.002000       | 40000            |
| Shell      | number     | 10000    | 0.004000       | 40000            |
+------------+------------+----------+----------------+------------------+

+------------+------------+----------+----------------+------------------+
| Algorithm  | Data Type  | Size     | Time (seconds) | Est Mem (byte)   |
+------------+------------+----------+----------------+------------------+
| Bubble     | word       | 10000    | 1.086000       | 199587           |
| Selection  | word       | 10000    | 0.435000       | 199587           |
| Insertion  | word       | 10000    | 0.181000       | 199587           |
| Merge      | word       | 10000    | 0.006000       | 199587           |
| Quick      | word       | 10000    | 0.004000       | 199587           |
| Shell      | word       | 10000    | 0.006000       | 199587           |
+------------+------------+----------+----------------+------------------+
```

2. Data 50.000

```
--- Testing size: 50000 ---

+------------+------------+----------+----------------+------------------+
| Algorithm  | Data Type  | Size     | Time (seconds) | Est Mem (byte)   |
+------------+------------+----------+----------------+------------------+
| Bubble     | number     | 50000    | 13.103000      | 200000           |
| Selection  | number     | 50000    | 5.797000       | 200000           |
| Insertion  | number     | 50000    | 2.714000       | 200000           |
| Merge      | number     | 50000    | 0.026000       | 200000           |
| Quick      | number     | 50000    | 0.011000       | 200000           |
| Shell      | number     | 50000    | 0.022000       | 200000           |
+------------+------------+----------+----------------+------------------+

+------------+------------+----------+----------------+------------------+
| Algorithm  | Data Type  | Size     | Time (seconds) | Est Mem (byte)   |
+------------+------------+----------+----------------+------------------+
| Bubble     | word       | 50000    | 33.854000      | 998519           |
| Selection  | word       | 50000    | 13.716000      | 998519           |
| Insertion  | word       | 50000    | 6.664000       | 998519           |
| Merge      | word       | 50000    | 0.035000       | 998519           |
| Quick      | word       | 50000    | 0.021000       | 998519           |
| Shell      | word       | 50000    | 0.052000       | 998519           |
+------------+------------+----------+----------------+------------------+
```

**3.** Data 100.000

```
--- Testing size: 100000 ---

+------------+------------+----------+----------------+------------------+
| Algorithm  | Data Type  | Size     | Time (seconds) | Est Mem (byte)   |
+------------+------------+----------+----------------+------------------+
| Bubble     | number     | 100000   | 55.086000      | 400000           |
| Selection  | number     | 100000   | 23.128000      | 400000           |
| Insertion  | number     | 100000   | 10.880000      | 400000           |
| Merge      | number     | 100000   | 0.050000       | 400000           |
| Quick      | number     | 100000   | 0.023000       | 400000           |
| Shell      | number     | 100000   | 0.047000       | 400000           |
+------------+------------+----------+----------------+------------------+

+------------+------------+----------+----------------+------------------+
| Algorithm  | Data Type  | Size     | Time (seconds) | Est Mem (byte)   |
+------------+------------+----------+----------------+------------------+
| Bubble     | word       | 100000   | 150.334000     | 1998878          |
| Selection  | word       | 100000   | 63.506000      | 1998878          |
| Insertion  | word       | 100000   | 30.917000      | 1998878          |
| Merge      | word       | 100000   | 0.070000       | 1998878          |
| Quick      | word       | 100000   | 0.050000       | 1998878          |
| Shell      | word       | 100000   | 0.126000       | 1998878          |
+------------+------------+----------+----------------+------------------+
```

**4.** Data 250.000

```
--- Testing size: 250000 ---

+------------+------------+----------+----------------+------------------+
| Algorithm  | Data Type  | Size     | Time (seconds) | Est Mem (byte)   |
+------------+------------+----------+----------------+------------------+
| Bubble     | number     | 250000   | 340.667000     | 1000000          |
| Selection  | number     | 250000   | 144.412000     | 1000000          |
| Insertion  | number     | 250000   | 69.120000      | 1000000          |
| Merge      | number     | 250000   | 0.137000       | 1000000          |
| Quick      | number     | 250000   | 0.060000       | 1000000          |
| Shell      | number     | 250000   | 0.129000       | 1000000          |
+------------+------------+----------+----------------+------------------+

+------------+------------+----------+----------------+------------------+
| Algorithm  | Data Type  | Size     | Time (seconds) | Est Mem (byte)   |
+------------+------------+----------+----------------+------------------+
| Bubble     | word       | 250000   | 1347.143000    | 4999435          |
| Selection  | word       | 250000   | 714.485000     | 4999435          |
| Insertion  | word       | 250000   | 422.952000     | 4999435          |
| Merge      | word       | 250000   | 0.199000       | 4999435          |
| Quick      | word       | 250000   | 0.158000       | 4999435          |
| Shell      | word       | 250000   | 0.481000       | 4999435          |
+------------+------------+----------+----------------+------------------+
```

5. Data 500.000

```
--- Testing size: 500000 ---

+------------+------------+----------+----------------+------------------+
| Algorithm  | Data Type  | Size     | Time (seconds) | Est Mem (byte)   |
+------------+------------+----------+----------------+------------------+
| Bubble     | number     | 500000   | 1428.937000    | 2000000          |
| Selection  | number     | 500000   | 661.986000     | 2000000          |
| Insertion  | number     | 500000   | 283.656000     | 2000000          |
| Merge      | number     | 500000   | 0.272000       | 2000000          |
| Quick      | number     | 500000   | 0.130000       | 2000000          |
| Shell      | number     | 500000   | 0.278000       | 2000000          |
+------------+------------+----------+----------------+------------------+

+------------+------------+----------+----------------+------------------+
| Algorithm  | Data Type  | Size     | Time (seconds) | Est Mem (byte)   |
+------------+------------+----------+----------------+------------------+
| Bubble     | word       | 500000   | 6372.021000    | 9999541          |
| Selection  | word       | 500000   | 3815.990000    | 9999541          |
| Insertion  | word       | 500000   | 2765.957000    | 9999541          |
| Merge      | word       | 500000   | 0.446000       | 9999541          |
| Quick      | word       | 500000   | 0.389000       | 9999541          |
| Shell      | word       | 500000   | 1.313000       | 9999541          |
+------------+------------+----------+----------------+------------------+
```

6. Data 1.000.000

```
+------------+------------+----------+----------------+------------------+
| Algorithm  | Data Type  | Size     | Time (seconds) | Est Mem (byte)   |
+------------+------------+----------+----------------+------------------+
| Bubble     | number     | 1000000  | 5674.471000    | 4000000          |
| Selection  | number     | 1000000  | 2426.014000    | 4000000          |
| Insertion  | number     | 1000000  | 1178.717000    | 4000000          |
| Merge      | number     | 1000000  | 0.603000       | 4000000          |
| Quick      | number     | 1000000  | 0.318000       | 4000000          |
| Shell      | number     | 1000000  | 0.661000       | 4000000          |
+------------+------------+----------+----------------+------------------+
```

## Data Perbandingan

### Ukuran data: 10.000

Saat data berukuran 10.000, perbedaan kecepatan antar algoritma mulai kelihatan. Untuk data angka, Bubble Sort butuh 0,365 detik, Selection Sort 0,233 detik, dan Insertion Sort 0,108 detik. Insertion Sort paling cepat di antara ketiganya, tapi masih kalah jauh dari Merge Sort (0,005 detik), Quick Sort (0,002 detik), dan Shell Sort (0,004 detik) yang super cepat. Untuk data teks, Bubble Sort jadi jauh lebih lambat (1,086 detik), Selection Sort 0,435 detik, dan Insertion Sort 0,181 detik, karena membandingkan teks lebih susah. Merge Sort (0,006 detik), Quick Sort (0,004 detik), dan Shell Sort (0,006 detik) tetap cepat. Memori untuk angka sama untuk semua algoritma (40.000 byte), tapi untuk teks lebih besar (199.587 byte) karena teks butuh ruang lebih banyak.

**Ukuran Data: 50.000**

Ketika ukuran data naik jadi 50.000, algoritma yang lambat mulai kewalahan. Untuk data angka, Bubble Sort butuh 13,103 detik, Selection Sort 5,797 detik, dan Insertion Sort 2,714 detik. Ketiganya sudah terasa lama, sementara Merge Sort (0,026 detik), Quick Sort (0,011 detik), dan Shell Sort (0,022 detik) masih sangat cepat. Untuk data teks, Bubble Sort makin parah (33,854 detik), Selection Sort 13,716 detik, dan Insertion Sort 6,664 detik. Merge Sort (0,035 detik), Quick Sort (0,021 detik), dan Shell Sort (0,052 detik) tetap unggul. Memori untuk angka 200.000 byte untuk semua algoritma, tapi untuk teks naik jadi 998.519 byte karena teks lebih besar.

**Ukuran Data: 100.000**

Pada ukuran 100.000, algoritma lambat makin tertinggal. Untuk data angka, Bubble Sort butuh 55,086 detik, Selection Sort 23,128 detik, dan Insertion Sort 10,880 detik, jadi sangat tidak praktis. Sebaliknya, Merge Sort (0,050 detik), Quick Sort (0,023 detik), dan Shell Sort (0,047 detik) masih cepat banget. Untuk teks, Bubble Sort super lama (150,334 detik), Selection Sort 63,506 detik, dan Insertion Sort 30,917 detik. Merge Sort (0,070 detik), Quick Sort (0,050 detik), dan Shell Sort (0,126 detik) tetap jago. Memori untuk angka 400.000 byte untuk semua algoritma, tapi untuk teks jadi 1.998.878 byte, menunjukkan teks butuh memori besar.

**Ukuran Data: 250.000**

Dengan data 250.000, perbedaan makin jelas. Untuk data angka, Bubble Sort butuh 340,667 detik, Selection Sort 144,412 detik, dan Insertion Sort 69,120 detik, jadi benar-benar lambat. Merge Sort (0,137 detik), Quick Sort (0,060 detik), dan Shell Sort (0,129 detik) tetap cepat, dengan Quick Sort paling unggul. Untuk teks, Bubble Sort memakan 1347,143 detik, Selection Sort 714,485 detik, dan Insertion Sort 422,952 detik, hampir tidak bisa dipakai. Merge Sort (0,199 detik), Quick Sort (0,158 detik), dan Shell Sort (0,481 detik) masih oke. Memori untuk angka 1.000.000 byte, tapi untuk teks naik jadi 4.999.435 byte, menunjukkan teks sangat boros memori.

**Ukuran Data: 500.000**

Saat data jadi 500.000, algoritma lambat benar-benar ketinggalan. Untuk angka, Bubble Sort butuh 1428,937 detik, Selection Sort 661,986 detik, dan Insertion Sort 283,656 detik, jadi tidak cocok untuk data besar. Merge Sort (0,272 detik), Quick Sort (0,130 detik), dan Shell Sort (0,278 detik) tetap cepat, dengan Quick Sort paling kencang. Untuk teks, Bubble Sort

sangat lambat (6372,021 detik), Selection Sort 3815,990 detik, dan Insertion Sort 2765,957 detik. Merge Sort (0,446 detik), Quick Sort (0,389 detik), dan Shell Sort (1,313 detik) masih bagus, tapi Shell Sort mulai agak lambat untuk teks. Memori untuk angka 2.000.000 byte, dan untuk teks jadi 9.999.541 byte, sangat besar.

**Ukuran Data: 1.000.000**

Untuk data 1.000.000 (hanya ada untuk angka), algoritma lambat benar-benar tidak bisa diandalkan. Bubble Sort butuh 5674,471 detik, Selection Sort 2426,014 detik, dan Insertion Sort 1178,717 detik, sangat lama. Merge Sort (0,603 detik), Quick Sort (0,318 detik), dan Shell Sort (0,661 detik) tetap cepat, dengan Quick Sort paling unggul. Memori untuk semua algoritma 4.000.000 byte, sama untuk data angka. Karena data teks tidak ada untuk ukuran ini, kita lihat tren bahwa Merge Sort dan Quick Sort pasti tetap cepat untuk teks, tapi butuh memori jauh lebih besar.

## Analisis

Dari data pengujian yang dilakukan terhadap berbagai algoritma sorting (Bubble, Selection, Insertion, Merge, Quick, dan Shell) dengan dua tipe data (number dan word) serta berbagai ukuran dataset (10.000 hingga 1.000.000 elemen), dapat ditarik beberapa kesimpulan penting:

1. **Performa Waktu Eksekusi**:
    - Algoritma **Bubble, Selection, dan Insertion** menunjukkan kompleksitas waktu $O(n^2)$, dengan waktu eksekusi yang meningkat signifikan seiring pertumbuhan data. Misalnya, untuk data number ukuran 1.000.000, Bubble Sort membutuhkan 5674 detik (~94 menit), sementara Selection dan Insertion masing-masing 2426 detik dan 1178 detik.
    - Algoritma **Merge, Quick, dan Shell** dengan kompleksitas O(n log n) jauh lebih efisien. Quick Sort konsisten tercepat (0.318 detik untuk 1 juta data number), diikuti Merge Sort (0.603 detik) dan Shell Sort (0.661 detik). Perbedaan ini terutama terlihat pada dataset besar.
2. **Pengaruh Tipe Data**:

- Operasi pengurutan **word** (string) secara signifikan lebih lambat daripada **number**. Contoh: Bubble Sort untuk 500.000 word membutuhkan 6372 detik (~106 menit), sementara untuk number "hanya" 1428 detik. Hal ini disebabkan operasi perbandingan string yang lebih kompleks dibanding angka.
- Algoritma efisien seperti Merge dan Quick tetap unggul untuk data word, dengan waktu di bawah 0.5 detik bahkan untuk 500.000 elemen.

3. **Penggunaan Memori**:
   - Algoritma **Merge Sort** cenderung menggunakan memori lebih besar (terlihat pada kolom "Est Mem") karena membutuhkan ruang tambahan untuk operasi divide-and-conquer.
   - **Quick Sort** dan **Shell Sort** lebih hemat memori, terutama untuk data number. Namun, perbedaan penggunaan memori antar algoritma tidak sebesar perbedaan waktu eksekusi.
4. **Kesesuaian Algoritma**:
   - Untuk dataset kecil (<10.000), semua algoritma masih feasible, meskipun algoritma sederhana seperti Insertion Sort bisa cukup cepat.
   - Untuk dataset besar (>100.000), algoritma O(n log n) seperti Quick Sort wajib dipilih untuk menghindari waktu eksekusi yang tidak praktis.

**Kesimpulan**

- Merge Sort dan Quick Sort adalah pilihan terbaik untuk data besar.
- Bubble Sort dan Selection Sort kurang efisien dan hanya cocok untuk data kecil.
- Shell Sort menawarkan keseimbangan antara efisiensi dan kesederhanaan.
- Pengurutan kata memerlukan waktu lebih lama dibandingkan angka karena kompleksitas operasi string.