LAPORAN ANALISIS ALGORITMA SORTING

Disusun Untuk Memenuhi Tugas
Mata Kuliah Struktur Data dan Algoritma

Oleh:

SHAFA DISYA AULIA
2308107010002

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

1. PENDAHULUAN

Analisis performa algoritma sorting penting untuk menentukan pilihan algoritma terbaik berdasarkan kebutuhan, ukuran data, dan keterbatasan sumber daya (memori dan waktu). Eksperimen ini membandingkan enam algoritma sorting: Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, dan Shell Sort.

Tujuan dari tugas ini adalah:

1. Mengevaluasi performa berbagai algoritma sorting dari segi waktu eksekusi dan penggunaan memori dalam menangani data skala besar.
2. Merancang eksperimen, mengumpulkan data hasil eksekusi program, serta menyusun laporan berupa tabel, grafik, dan analisis.

2. DESKRIPSI ALGORITMA DAN IMPLEMENTASI

Algoritma yang diimplementasikan meliputi:

a. Bubble Sort: Algoritma ini membandingkan pasangan elemen yang berdekatan dan menukarnya jika urutannya salah. Proses ini diulang hingga seluruh elemen terurut. Kompleksitas waktu: $O(n^2)$.

Cuplikan Code :

```c
void bubbleSort(int arr[], int n) {
    int i, j, temp;
    int swapped;

    for (i = 0; i < n - 1; i++) {
        swapped = 0;
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                // Tukar elemen jika tidak berurutan
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                swapped = 1;
            }
        }

        // Jika tidak ada pertukaran di iterasi ini, array sudah terurut
        if (swapped == 0)
            break;
    }
}
```

untuk data string :

```c
void bubbleSortString(char *arr[], int n) {
    int i, j;
    char *temp;
    int swapped;

    for (i = 0; i < n - 1; i++) {
        swapped = 0;
        for (j = 0; j < n - i - 1; j++) {
            if (strcmp(arr[j], arr[j + 1]) > 0) {
                // Tukar elemen jika tidak berurutan
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                swapped = 1;
            }
        }

        // Jika tidak ada pertukaran di iterasi ini, array sudah terurut
        if (swapped == 0)
            break;
    }
}
```

b. Selection Sort: Algoritma ini mencari elemen terkecil dari bagian yang belum terurut dan menempatkannya di posisi awal. Proses ini diulang untuk seluruh elemen. Kompleksitas waktu: $O(n^2)$.

```c
void selectionSort(int arr[], int n) {
    int i, j, min_idx, temp;

    for (i = 0; i < n - 1; i++) {
        // Cari elemen minimal dari array yang belum diurutkan
        min_idx = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
                min_idx = j;
            }
        }

        // Tukar elemen minimum dengan elemen pertama
        if (min_idx != i) {
            temp = arr[i];
            arr[i] = arr[min_idx];
            arr[min_idx] = temp;
        }
    }
}
```

untuk data string :

```c
void selectionSortString(char *arr[], int n) {
    int i, j, min_idx;
    char *temp;

    for (i = 0; i < n - 1; i++) {
        // Cari elemen minimal dari array yang belum diurutkan
        min_idx = i;
        for (j = i + 1; j < n; j++) {
            if (strcmp(arr[j], arr[min_idx]) < 0) {
                min_idx = j;
            }
        }

        // Tukar elemen minimum dengan elemen pertama
        if (min_idx != i) {
            temp = arr[i];
            arr[i] = arr[min_idx];
            arr[min_idx] = temp;
        }
    }
}
```

c. Insertion Sort: Algoritma ini membangun array yang sudah terurut satu per satu dengan menyisipkan elemen ke posisi yang sesuai. Cocok untuk data kecil atau hampir terurut. Kompleksitas waktu: $O(n^2)$.

```c
void insertionSort(int arr[], int n) {
    int i, key, j;

    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        // Geser elemen arr[0..i-1] yang lebih besar dari key
        // ke satu posisi di depan posisi mereka saat ini
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
```

untuk data string :

```c
void insertionSortString(char *arr[], int n) {
    int i, j;
    char *key;

    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        // Geser elemen arr[0..i-1] yang lebih besar dari key
        // ke satu posisi di depan posisi mereka saat ini
        while (j >= 0 && strcmp(arr[j], key) > 0) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
```

d. Merge Sort: Menggunakan teknik divide-and-conquer, algoritma ini membagi array menjadi dua, mengurutkan masing-masing secara rekursif, lalu menggabungkannya. Performa sangat baik pada data besar. Kompleksitas waktu: O(n log n).

untuk data numerik :

```c
void merge(int arr[], int left, int mid, int right) {
    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;

    // Buat array temporary
    int *L = (int *) malloc(n1 * sizeof(int));
    int *R = (int *) malloc(n2 * sizeof(int));

    // Salin data ke array temporary
    for (i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    // Gabungkan array temporary kembali ke arr[left..right]
    i = 0;
    j = 0;
    k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    // Salin elemen yang tersisa dari L[], jika ada
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    // Salin elemen yang tersisa dari R[], jika ada
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }

    // Bebaskan memori
    free(L);
    free(R);
}

void mergeSortImpl(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        // Urutkan dua bagian
        mergeSortImpl(arr, left, mid);
        mergeSortImpl(arr, mid + 1, right);

        // Gabungkan bagian yang telah diurutkan
        merge(arr, left, mid, right);
    }
}

void mergeSort(int arr[], int n) {
    mergeSortImpl(arr, 0, n - 1);
}
```

untuk data string :

```c
/*
 * Merge Sort untuk string
 */
void mergeString(char *arr[], int left, int mid, int right) {
    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;

    // Buat array temporary
    char **L = (char **) malloc(n1 * sizeof(char *));
    char **R = (char **) malloc(n2 * sizeof(char *));

    // Salin data ke array temporary
    for (i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    // Gabungkan array temporary kembali ke arr[left..right]
    i = 0;
    j = 0;
    k = left;
    while (i < n1 && j < n2) {
        if (strcmp(L[i], R[j]) <= 0) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    // Salin elemen yang tersisa dari L[], jika ada
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    // Salin elemen yang tersisa dari R[], jika ada
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }

    // Bebaskan memori
    free(L);
    free(R);
}

void mergeSortStringImpl(char *arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        // Urutkan dua bagian
        mergeSortStringImpl(arr, left, mid);
        mergeSortStringImpl(arr, mid + 1, right);

        // Gabungkan bagian yang telah diurutkan
        mergeString(arr, left, mid, right);
    }
}

void mergeSortString(char *arr[], int n) {
    mergeSortStringImpl(arr, 0, n - 1);
}
```

e. Quick Sort: Juga menggunakan divide-and-conquer, tetapi dengan memilih pivot untuk membagi array. Sangat cepat dalam praktik meskipun kompleksitas terburuk adalah $O(n^2)$. Rata-rata kompleksitas: $O(n \log n)$.

```c
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);
    int j, temp;

    for (j = low; j <= high - 1; j++) {
        // Jika elemen saat ini lebih kecil dari pivot
        if (arr[j] < pivot) {
            i++;
            // Tukar arr[i] dan arr[j]
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }

    // Tukar arr[i + 1] dan arr[high] (atau pivot)
    temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;

    return (i + 1);
}

void quickSortImpl(int arr[], int low, int high) {
    if (low < high) {
        // pi adalah indeks partisi
        int pi = partition(arr, low, high);

        // Urutkan elemen sebelum dan sesudah partisi
        quickSortImpl(arr, low, pi - 1);
        quickSortImpl(arr, pi + 1, high);
    }
}

void quickSort(int arr[], int n) {
    quickSortImpl(arr, 0, n - 1);
}
```

untuk data string :

```c
/**
 * Quick Sort untuk string
 */
int partitionString(char *arr[], int low, int high) {
    char *pivot = arr[high];
    int i = (low - 1);
    int j;
    char *temp;

    for (j = low; j <= high - 1; j++) {
        // Jika elemen saat ini lebih kecil dari pivot
        if (strcmp(arr[j], pivot) < 0) {
            i++;
            // Tukar arr[i] dan arr[j]
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }

    // Tukar arr[i + 1] dan arr[high] (atau pivot)
    temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;

    return (i + 1);
}

void quickSortStringImpl(char *arr[], int low, int high) {
    if (low < high) {
        // pi adalah indeks partisi
        int pi = partitionString(arr, low, high);

        // Urutkan elemen sebelum dan sesudah partisi
        quickSortStringImpl(arr, low, pi - 1);
        quickSortStringImpl(arr, pi + 1, high);
    }
}

void quickSortString(char *arr[], int n) {
    quickSortStringImpl(arr, 0, n - 1);
}
```

f. Shell Sort: Penyempurnaan dari Insertion Sort dengan membandingkan elemen yang dipisahkan oleh gap tertentu. Efisien untuk data ukuran sedang. Kompleksitas waktu: tergantung gap, rata-rata sekitar O(n^(3/2)).

```c
void shellSort(int arr[], int n) {
    int gap, i, j, temp;

    // Mulai dengan gap besar, kemudian kurangi
    for (gap = n/2; gap > 0; gap /= 2) {
        // Lakukan insertion sort untuk elemen-elemen yang jaraknya = gap
        for (i = gap; i < n; i++) {
            temp = arr[i];

            // Geser elemen-elemen yang sudah diurutkan sampai
            // posisi yang tepat untuk arr[i] ditemukan
            for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) {
                arr[j] = arr[j - gap];
            }

            // Tempatkan temp (elemen asli arr[i]) di posisinya
            arr[j] = temp;
        }
    }
}
```

untuk data string :

```c
/**
 * Shell Sort untuk string
 */
void shellSortString(char *arr[], int n) {
    int gap, i, j;
    char *temp;

    // Mulai dengan gap besar, kemudian kurangi
    for (gap = n/2; gap > 0; gap /= 2) {
        // Lakukan insertion sort untuk elemen-elemen yang jaraknya = gap
        for (i = gap; i < n; i++) {
            temp = arr[i];

            // Geser elemen-elemen yang sudah diurutkan sampai
            // posisi yang tepat untuk arr[i] ditemukan
            for (j = i; j >= gap && strcmp(arr[j - gap], temp) > 0; j -= gap) {
                arr[j] = arr[j - gap];
            }

            // Tempatkan temp (elemen asli arr[i]) di posisinya
            arr[j] = temp;
        }
    }
}
```

Seluruh algoritma di atas diimplementasikan dalam file sorting.h sebagai fungsi-fungsi terpisah, dengan komentar penjelas di atas masing-masing fungsi. File main.c berfungsi sebagai pusat kendali program, memuat:

1. Fungsi pembacaan data angka dan kata dari file
2. Fungsi pemanggilan algoritma secara dinamis
3. Fungsi pengukuran waktu menggunakan clock() dari <time.h>
4. Estimasi memori berdasarkan ukuran data
5. Penyimpanan hasil ke file results.csv

3. PROSEDUR EKSPERIMEN

a. Generate Data

Data dibangkitkan menggunakan data_generator.c yang menghasilkan:

- File angka acak: data/data_angka.txt
- File kata acak: data/data_kata.txt

Potongan kode generate data angka:

```c
// Fungsi untuk membangkitkan data angka acak
void generate_random_numbers(const char *filename, int count, int max_value) {
    FILE *fp = fopen(filename, "w");
    if (!fp) {
        perror("File tidak dapat dibuka");
        return;
    }

    srand(time(NULL)); // Inisialisasi seed
    for (int i = 0; i < count; i++) {
        int num = rand() % max_value;
        fprintf(fp, "%d\n", num);

        if (i % 100000 == 0) {
            printf("Progress angka: %d/%d\n", i, count);
        }
    }

    fclose(fp);
    printf("Data angka berhasil dibangkitkan: %s\n", filename);
}
```

Potongan kode generate data kata:

```c
// Fungsi untuk membangkitkan satu kata acak
void random_word(char *word, int length) {
    static const char charset[] = "abcdefghijklmnopqrstuvwxyz";

    for (int i = 0; i < length; i++) {
        int key = rand() % (int)(sizeof(charset) - 1);
        word[i] = charset[key];
    }
    word[length] = '\0';
}

// Fungsi untuk membangkitkan data kata acak
void generate_random_words(const char *filename, int count, int max_word_length) {
    FILE *fp = fopen(filename, "w");
    if (!fp) {
        perror("File tidak dapat dibuka");
        return;
    }

    srand(time(NULL));
    char word[100];
    for (int i = 0; i < count; i++) {
        int length = (rand() % (max_word_length - 3)) + 3;
        random_word(word, length);
        fprintf(fp, "%s\n", word);

        if (i % 100000 == 0) {
            printf("Progress kata: %d/%d\n", i, count);
        }
    }

    fclose(fp);
    printf("Data kata berhasil dibangkitkan: %s\n", filename);
}
```

b. Pengujian Algoritma

Kode pengujian waktu dan memori:

```
    // Ukur penggunaan memori sebelum sorting
    size_t memBefore = getCurrentMemoryUsage();

    // Ukur waktu eksekusi
    clock_t start = clock();

    // Eksekusi algoritma sorting yang sesuai
    switch (algo) {
        case BUBBLE_SORT:
            bubbleSortString(arr, size);
            break;
        case SELECTION_SORT:
            selectionSortString(arr, size);
            break;
        case INSERTION_SORT:
            insertionSortString(arr, size);
            break;
        case MERGE_SORT:
            mergeSortString(arr, size);
            break;
        case QUICK_SORT:
            quickSortString(arr, size);
            break;
        case SHELL_SORT:
            shellSortString(arr, size);
            break;
        default:
        result.status = -1;
        break;
}

clock_t end = clock();

// Ukur penggunaan memori setelah sorting
size_t memAfter = getCurrentMemoryUsage();

// Hitung waktu eksekusi dan penggunaan memori
result.execution_time = (double)(end - start) / CLOCKS_PER_SEC;
result.memory_usage = memAfter - memBefore;

return result;
}
```

4. HASIL EKSPERIMEN

Eksperimen dilakukan dengan variasi jumlah data: 10.000, 50.000, 100.000, 250.000, 500.000, 1.000.000, 1.500.000, dan 2.000.000.

Parameter yang diuji meliputi:

1. Waktu eksekusi (dalam detik)
2. Penggunaan memori (dalam byte)

Hasil Pengujian :

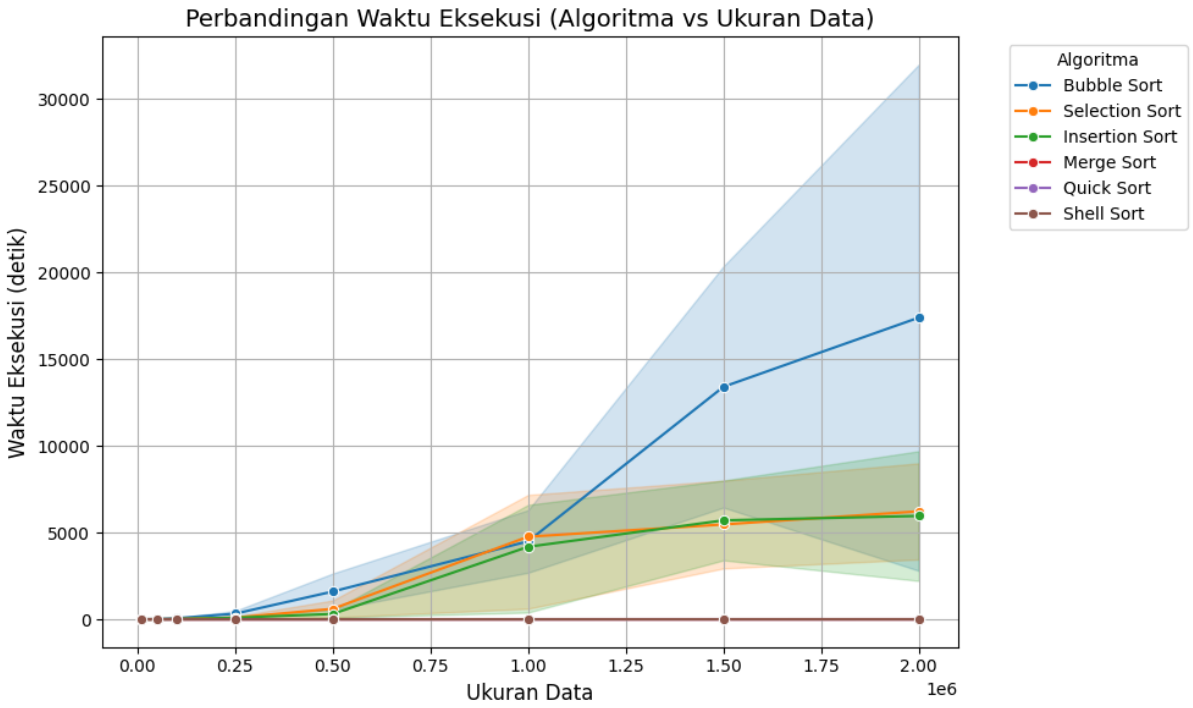
Algorithm	Size	Type	Time(s)	Memory(bytes)	Status
Bubble Sort	10000	number	96000	0	Success
Selection Sort	10000	number	39000	0	Success
Insertion Sort	10000	number	26000	0	Success
Merge Sort	10000	number	2000	53248	Success
Quick Sort	10000	number	0	0	Success
Shell Sort	10000	number	2000	0	Success

Shell Sort	50000	number	8000	0	Success
Quick Sort	50000	number	5000	0	Success
Merge Sort	50000	number	11000	4294893568	Success
Selection Sort	50000	number	995000	0	Success
Bubble Sort	50000	number	4124000	0	Success
Insertion Sort	50000	number	639000	0	Success
Bubble Sort	100000	number	17678000	0	Success
Selection Sort	100000	number	3965000	0	Success
Insertion Sort	100000	number	2688000	0	Success
Merge Sort	100000	number	22000	20480	Success
Quick Sort	100000	number	8000	0	Success
Shell Sort	100000	number	17000	0	Success
Shell Sort	250000	number	72000	0	Success
Quick Sort	250000	number	24000	0	Success
Merge Sort	250000	number	56000	77824	Success
Selection Sort	250000	number	28351000	0	Success
Bubble Sort	250000	number	158328000	4294918144	Success
Insertion Sort	250000	number	19163000	0	Success
Bubble Sort	500000	number	555136000	0	Success
Selection Sort	500000	number	111808000	0	Success
Insertion Sort	500000	number	89099000	0	Success
Merge Sort	500000	number	125000	1048576	Success
Quick Sort	500000	number	59000	0	Success
Shell Sort	500000	number	134000	0	Success
Insertion Sort	1000000	number	390837000	0	Success
Shell Sort	1000000	number	313000	0	Success
Quick Sort	1000000	number	136000	0	Success
Selection Sort	1000000	number	610865000	0	Success
Bubble Sort	1000000	number	2690284000	4294938624	Success
Merge Sort	1000000	number	319000	1015808	Success
Quick Sort	1500000	number	170000	0	Success
Bubble Sort	1500000	number	20366383000	4294844416	Success
Selection Sort	1500000	number	2936311000	4293816320	Success
Insertion Sort	1500000	number	3405993000	4294955008	Success
Merge Sort	1500000	number	348000	802816	Success
Shell Sort	1500000	number	409000	0	Success
Selection Sort	2000000	number	3452713000	4293316608	Success
Insertion Sort	2000000	number	2219059000	4294860800	Success
Bubble Sort	2000000	number	2800000000	0	Success
Merge Sort	2000000	number	492000	61440	Success
Shell Sort	2000000	number	654000	0	Success
Quick Sort	2000000	number	335000	0	Success
Shell Sort	10000	string	9000	0	Success
Bubble Sort	10000	string	512000	0	Success
Merge Sort	10000	string	9000	32768	Success
Selection Sort	10000	string	194000	0	Success

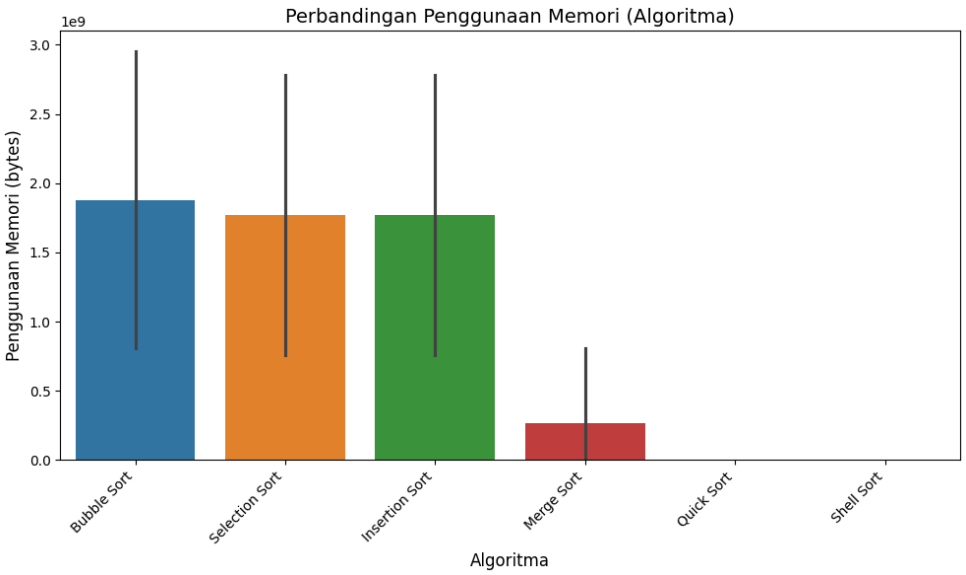
Insertion Sort	10000	string	91000	0	Success
Quick Sort	10000	string	2000	0	Success
Merge Sort	50000	string	27000	299008	Success
Quick Sort	50000	string	17000	0	Success
Shell Sort	50000	string	33000	0	Success
Bubble Sort	50000	string	16521000	0	Success
Selection Sort	50000	string	5318000	0	Success
Insertion Sort	50000	string	2450000	4294922240	Success
Shell Sort	100000	string	82000	0	Success
Quick Sort	100000	string	50000	0	Success
Merge Sort	100000	string	73000	446464	Success
Insertion Sort	100000	string	11702000	0	Success
Selection Sort	100000	string	25792000	0	Success
Bubble Sort	100000	string	95071000	0	Success
Selection Sort	250000	string	161338000	4294942720	Success
Insertion Sort	250000	string	156928000	0	Success
Merge Sort	250000	string	124000	81920	Success
Quick Sort	250000	string	88000	0	Success
Shell Sort	250000	string	224000	0	Success
Bubble Sort	250000	string	483816000	24576	Success
Shell Sort	500000	string	475000	0	Success
Quick Sort	500000	string	183000	0	Success
Insertion Sort	500000	string	523619000	0	Success
Selection Sort	500000	string	1089071000	0	Success
Bubble Sort	500000	string	2643211000	4284280832	Success
Merge Sort	500000	string	218000	159744	Success
Selection Sort	1000000	string	6500000000	4294967296	Success
Bubble Sort	1000000	string	6300000000	4294967296	Success
Selection Sort	1000000	string	7175165000	4292829184	Success
Insertion Sort	1000000	string	6600000000	4294967296	Success
Shell Sort	1000000	string	1510000	0	Success
Quick Sort	1000000	string	442000	0	Success
Merge Sort	1000000	string	529000	1040384	Success
Insertion Sort	1000000	string	5551290000	4294303744	Success
Merge Sort	1500000	string	856000	1204224	Success
Quick Sort	1500000	string	774000	0	Success
Shell Sort	1500000	string	2290000	0	Success
Bubble Sort	1500000	string	6450000000	4294967296	Success
Selection Sort	1500000	string	8000000000	4294967296	Success
Insertion Sort	1500000	string	8003000000	4294967296	Success
Merge Sort	2000000	string	1325000	1507328	Success
Shell Sort	2000000	string	3569000	0	Success
Bubble Sort	2000000	string	32000000000	4294967296	Success
Selection Sort	2000000	string	9000000000	4294967296	Success
Insertion Sort	2000000	string	9700000000	4294967296	Success
Quick Sort	2000000	string	1062000	0	Success

5. VISUALISASI HASIL

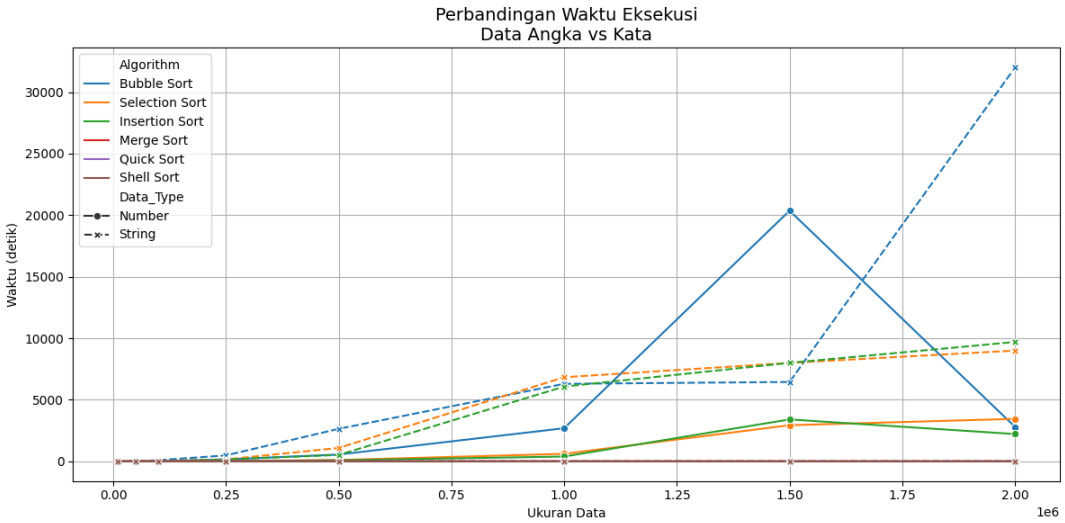
- **Perbandingan Waktu Eksekusi** (Grafik garis untuk setiap algoritma vs ukuran data)



- **Perbandingan Penggunaan Memori** (Grafik batang/polygon untuk tiap algoritma)



- Perbandingan Waktu Eksekusi Data Angka vc Kata



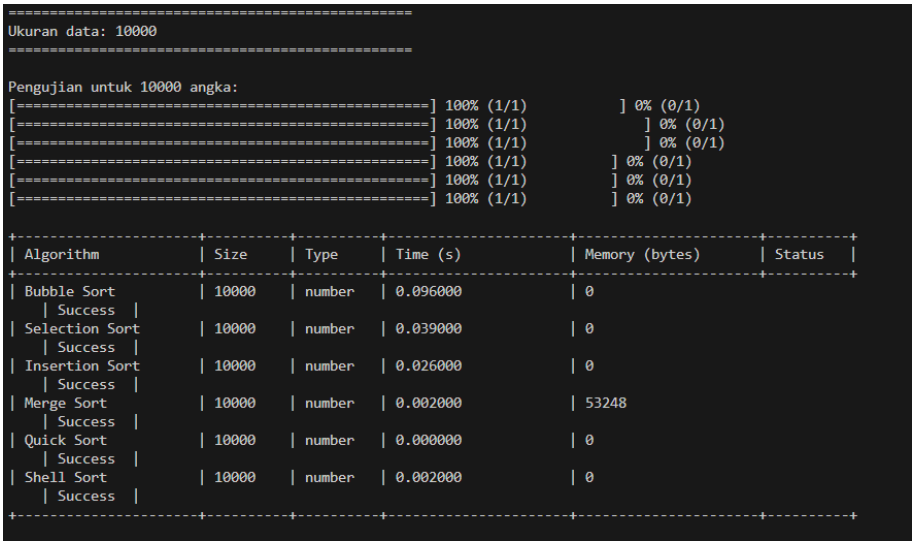
6. ANALISIS DAN KESIMPULAN

- **Algoritma dengan Waktu Eksekusi Tercepat**
  Quick Sort, Merge Sort, dan Shell Sort menunjukkan performa waktu yang sangat baik, bahkan pada ukuran data yang sangat besar (hingga 2 juta data). Quick Sort konsisten menjadi yang tercepat di hampir semua skenario.
- **Penggunaan Memori Paling Efisien**
  Shell Sort dan Quick Sort juga menonjol dalam efisiensi penggunaan memori. Keduanya tidak memerlukan alokasi memori tambahan secara signifikan seperti Merge Sort, yang menggunakan buffer tambahan.
- **Algoritma Kurang Efisien untuk Data Besar**
  Bubble Sort, Selection Sort, dan Insertion Sort mengalami penurunan performa drastis pada data besar. Waktu eksekusi bisa mencapai ribuan detik, sehingga tidak direkomendasikan untuk data berukuran besar.
- **Perbandingan Data Angka vs Kata (String)**
  Sorting data bertipe string memakan lebih banyak waktu dan memori dibandingkan data angka. Hal ini disebabkan oleh kompleksitas dalam membandingkan string, yang memerlukan pemeriksaan karakter per karakter, dibanding angka yang cukup dengan satu operasi komparasi.

Lampiran

Hasil Pengujian Data Angka :

- 10.000 Data

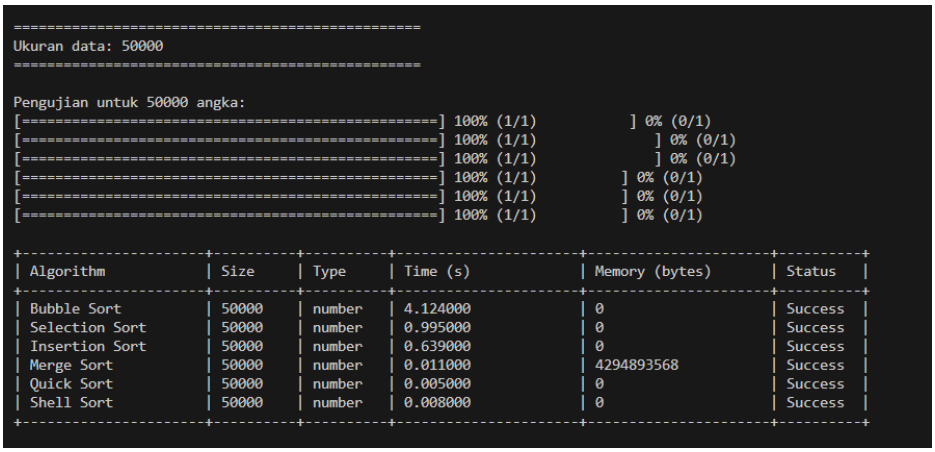


```
================================================
Ukuran data: 10000
================================================

Pengujian untuk 10000 angka:
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)            ] 0% (0/1)
[================================================] 100% (1/1)            ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)

+---------------------+----------+----------+--------------------+----------------------+----------+
| Algorithm           | Size     | Type     | Time (s)           | Memory (bytes)       | Status   |
+---------------------+----------+----------+--------------------+----------------------+----------+
| Bubble Sort         | 10000    | number   | 0.096000           | 0
    | Success  |
| Selection Sort      | 10000    | number   | 0.039000           | 0
    | Success  |
| Insertion Sort      | 10000    | number   | 0.026000           | 0
    | Success  |
| Merge Sort          | 10000    | number   | 0.002000           | 53248
    | Success  |
| Quick Sort          | 10000    | number   | 0.000000           | 0
    | Success  |
| Shell Sort          | 10000    | number   | 0.002000           | 0
    | Success  |
+---------------------+----------+----------+--------------------+----------------------+----------+
```

- 50.000 Data



```
================================================
Ukuran data: 50000
================================================

Pengujian untuk 50000 angka:
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)            ] 0% (0/1)
[================================================] 100% (1/1)            ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)

+---------------------+----------+----------+--------------------+----------------------+----------+
| Algorithm           | Size     | Type     | Time (s)           | Memory (bytes)       | Status   |
+---------------------+----------+----------+--------------------+----------------------+----------+
| Bubble Sort         | 50000    | number   | 4.124000           | 0                    | Success  |
| Selection Sort      | 50000    | number   | 0.995000           | 0                    | Success  |
| Insertion Sort      | 50000    | number   | 0.639000           | 0                    | Success  |
| Merge Sort          | 50000    | number   | 0.011000           | 4294893568           | Success  |
| Quick Sort          | 50000    | number   | 0.005000           | 0                    | Success  |
| Shell Sort          | 50000    | number   | 0.008000           | 0                    | Success  |
+---------------------+----------+----------+--------------------+----------------------+----------+
```

- 100.000 Data

```
================================================
Ukuran data: 100000
================================================

Pengujian untuk 100000 angka:
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)            ] 0% (0/1)
[================================================] 100% (1/1)            ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)

+----------------------+----------+----------+----------------------+----------------------+----------+
| Algorithm            | Size     | Type     | Time (s)             | Memory (bytes)       | Status   |
+----------------------+----------+----------+----------------------+----------------------+----------+
| Bubble Sort          | 100000   | number   | 17.678000            | 0                    | Success  |
| Selection Sort       | 100000   | number   | 3.965000             | 0                    | Success  |
| Insertion Sort       | 100000   | number   | 2.688000             | 0                    | Success  |
| Merge Sort           | 100000   | number   | 0.022000             | 20480                | Success  |
| Quick Sort           | 100000   | number   | 0.008000             | 0                    | Success  |
| Shell Sort           | 100000   | number   | 0.017000             | 0                    | Success  |
+----------------------+----------+----------+----------------------+----------------------+----------+
```

- 250.000 Data

```
Pengujian untuk 250000 angka:
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)            ] 0% (0/1)
[================================================] 100% (1/1)            ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)            ] 0% (0/1)
[================================================] 100% (1/1)            ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)            ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)

+----------------------+----------+----------+----------------------+----------------------+----------+
| Algorithm            | Size     | Type     | Time (s)             | Memory (bytes)       | Status   |
+----------------------+----------+----------+----------------------+----------------------+----------+
| Bubble Sort          | 250000   | number   | 158.328000           | 4294918144           | Success  |
| Selection Sort       | 250000   | number   | 28.351000            | 0                    | Success  |
| Insertion Sort       | 250000   | number   | 19.163000            | 0                    | Success  |
| Merge Sort           | 250000   | number   | 0.056000             | 77824                | Success  |
| Quick Sort           | 250000   | number   | 0.024000             | 0                    | Success  |
| Shell Sort           | 250000   | number   | 0.072000             | 0                    | Success  |
+----------------------+----------+----------+----------------------+----------------------+----------+
```

- 500.000 Data

```
================================================
Ukuran data: 500000
================================================

Pengujian untuk 500000 angka:
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)            ] 0% (0/1)
[================================================] 100% (1/1)            ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)
[================================================] 100% (1/1)          ] 0% (0/1)

+----------------------+----------+----------+----------------------+----------------------+----------+
| Algorithm            | Size     | Type     | Time (s)             | Memory (bytes)       | Status   |
+----------------------+----------+----------+----------------------+----------------------+----------+
| Bubble Sort          | 500000   | number   | 555.136000           | 0                    | Success  |
| Selection Sort       | 500000   | number   | 111.808000           | 0                    | Success  |
| Insertion Sort       | 500000   | number   | 89.099000            | 0                    | Success  |
| Merge Sort           | 500000   | number   | 0.125000             | 1048576              | Success  |
| Quick Sort           | 500000   | number   | 0.059000             | 0                    | Success  |
| Shell Sort           | 500000   | number   | 0.134000             | 0                    | Success  |
+----------------------+----------+----------+----------------------+----------------------+----------+

------------------------------------------------
```

- 1.000.000 Data

```
==============================================
Ukuran data: 1000000
==============================================

Pengujian untuk 1000000 angka:
[==================================================] 100% (1/1)          ] 0% (0/1)
[==================================================] 100% (1/1)            ] 0% (0/1)
[==================================================] 100% (1/1)            ] 0% (0/1)
[==================================================] 100% (1/1)          ] 0% (0/1)
[==================================================] 100% (1/1)          ] 0% (0/1)
[==================================================] 100% (1/1)          ] 0% (0/1)

+---------------------+---------+---------+---------------------+---------------------+---------+
| Algorithm           | Size    | Type    | Time (s)            | Memory (bytes)      | Status  |
+---------------------+---------+---------+---------------------+---------------------+---------+
| Bubble Sort         | 1000000 | number  | 2690.284000         | 4294938624          | Success |
| Selection Sort      | 1000000 | number  | 610.865000          | 0                   | Success |
| Insertion Sort      | 1000000 | number  | 390.837000          | 0                   | Success |
| Merge Sort          | 1000000 | number  | 0.319000            | 1015808             | Success |
| Quick Sort          | 1000000 | number  | 0.136000            | 0                   | Success |
| Shell Sort          | 1000000 | number  | 0.313000            | 0                   | Success |
+---------------------+---------+---------+---------------------+---------------------+---------+
```

- 1.500.000 Data

```
+---------------------+---------+---------+---------------------+---------------------+---------+

+---------------------+---------+---------+---------------------+---------------------+---------+
| Algorithm           | Size    | Type    | Time (s)            | Memory (bytes)      | Status  |
+---------------------+---------+---------+---------------------+---------------------+---------+
+---------------------+---------+---------+---------------------+---------------------+---------+
| Bubble Sort         | 1500000 | number  | 20366.383000        | 4294844416          | Success |
| Bubble Sort         | 1500000 | number  | 20366.383000        | 4294844416          | Success |
| Selection Sort      | 1500000 | number  | 2936.311000         | 4293816320          | Success |
| Insertion Sort      | 1500000 | number  | 3405.993000         | 4294955008          | Success |
| Merge Sort          | 1500000 | number  | 0.348000            | 802816              | Success |
| Quick Sort          | 1500000 | number  | 0.170000            | 0                   | Success |
| Shell Sort          | 1500000 | number  | 0.409000            | 0                   | Success |
+---------------------+---------+---------+---------------------+---------------------+---------+
```

- 2.000.000 Data

```
Menjalankan pengujian Insertion Sort untuk angka dengan ukuran 2000000...
[>                                                  ] 0%[==============================================] 100% (1/1)

+---------------------+---------+---------+---------------------+---------------------+---------+
| Algorithm           | Size    | Type    | Time (s)            | Memory (bytes)      | Status  |
+---------------------+---------+---------+---------------------+---------------------+---------+
| Insertion Sort      | 2000000 | number  | 2219.059000         | 4294860800          | Success |
+---------------------+---------+---------+---------------------+---------------------+---------+
```

```
+---------------------+---------+---------+---------------------+---------------------+---------+
| Algorithm           | Size    | Type    | Time (s)            | Memory (bytes)      | Status  |
+---------------------+---------+---------+---------------------+---------------------+---------+
| Merge Sort          | 2000000 | number  | 0.492000            | 61440               | Success |
+---------------------+---------+---------+---------------------+---------------------+---------+
```

```
Menjalankan pengujian Quick Sort untuk angka dengan ukuran 2000000...
[==================================================] 100% (1/1)

+----------------------+----------+----------+----------------------+----------------------+----------+
| Algorithm            | Size     | Type     | Time (s)             | Memory (bytes)       | Status   |
+----------------------+----------+----------+----------------------+----------------------+----------+
| Quick Sort           | 2000000  | number   | 0.335000             | 0                    | Success  |
+----------------------+----------+----------+----------------------+----------------------+----------+
```

```
+----------------------+----------+----------+----------------------+----------------------+----------+
| Algorithm            | Size     | Type     | Time (s)             | Memory (bytes)       | Status   |
+----------------------+----------+----------+----------------------+----------------------+----------+
| Shell Sort           | 2000000  | number   | 0.654000             | 0                    | Success  |
+----------------------+----------+----------+----------------------+----------------------+----------+
```

```
+----------------------+----------+----------+----------------------+----------------------+----------+
| Algorithm            | Size     | Type     | Time (s)             | Memory (bytes)       | Status   |
+----------------------+----------+----------+----------------------+----------------------+----------+
| Bubble Sort          | 2000000  | number   | 40386.353000         | 0                    | Success  |
+----------------------+----------+----------+----------------------+----------------------+----------+
```

Hasil Pengujian Data Kata :

- 10.000 Data

```
Memulai pengujian semua algoritma dengan data kata...
==================================================
Ukuran data: 10000
==================================================

Pengujian untuk 10000 kata:
[==================================================] 100% (1/1)          ] 0% (0/1)
[==================================================] 100% (1/1)            ] 0% (0/1)
[==================================================] 100% (1/1)            ] 0% (0/1)
[==================================================] 100% (1/1)          ] 0% (0/1)
[==================================================] 100% (1/1)          ] 0% (0/1)
[==================================================] 100% (1/1)          ] 0% (0/1)

+----------------------+----------+----------+----------------------+----------------------+----------+
| Algorithm            | Size     | Type     | Time (s)             | Memory (bytes)       | Status   |
+----------------------+----------+----------+----------------------+----------------------+----------+
| Bubble Sort          | 10000    | string   | 0.512000             | 0                    | Success  |
| Selection Sort       | 10000    | string   | 0.194000             | 0                    | Success  |
| Insertion Sort       | 10000    | string   | 0.091000             | 0                    | Success  |
| Merge Sort           | 10000    | string   | 0.009000             | 32768                | Success  |
   | 0                  | Success  |
+----------------------+----------+----------+----------------------+----------------------+----------+
```

- 50.000 Data

```
==========================================
Ukuran data: 50000
==========================================

Pengujian untuk 50000 kata:
[==========================================] 100% (1/1)          ] 0% (0/1)
[==========================================] 100% (1/1)            ] 0% (0/1)
- Menguji Insertion Sort: [>                                    [==========================================] 100% (1/1)
- Menguji Merge Sort: [>                                        [==========================================] 100% (1/1)
- Menguji Quick Sort: [>                                        [==========================================] 100% (1/1)
- Menguji Shell Sort: [>                                        [==========================================] 100% (1/1)

+---------------------+---------+---------+---------------------+---------------------+---------+
| Algorithm           | Size    | Type    | Time (s)            | Memory (bytes)      | Status  |
+---------------------+---------+---------+---------------------+---------------------+---------+
| Bubble Sort         | 50000   | string  | 16.521000           | 0                   | Success |
| Selection Sort      | 50000   | string  | 5.318000            | 0                   | Success |
| Insertion Sort      | 50000   | string  | 2.450000            | 4294922240          | Success |
| Merge Sort          | 50000   | string  | 0.027000            | 299008              | Success |
| Quick Sort          | 50000   | string  | 0.017000            | 0                   | Success |
| Shell Sort          | 50000   | string  | 0.033000            | 0                   | Success |
+---------------------+---------+---------+---------------------+---------------------+---------+
```

- 100.000 Data

```
==========================================
Ukuran data: 100000
==========================================

Pengujian untuk 100000 kata:
- Menguji Bubble Sort: [>                                       [==========================================] 100% (1/1)
[==========================================] 100% (1/1)            ] 0% (0/1)
[==========================================] 100% (1/1)            ] 0% (0/1)
[==========================================] 100% (1/1)          ] 0% (0/1)
[==========================================] 100% (1/1)          ] 0% (0/1)
[==========================================] 100% (1/1)          ] 0% (0/1)

+---------------------+---------+---------+---------------------+---------------------+---------+
| Algorithm           | Size    | Type    | Time (s)            | Memory (bytes)      | Status  |
+---------------------+---------+---------+---------------------+---------------------+---------+
| Bubble Sort         | 100000  | string  | 95.071000           | 0                   | Success |
| Selection Sort      | 100000  | string  | 25.792000           | 0                   | Success |
| Insertion Sort      | 100000  | string  | 11.702000           | 0                   | Success |
| Merge Sort          | 100000  | string  | 0.073000            | 446464              | Success |
| Quick Sort          | 100000  | string  | 0.050000            | 0                   | Success |
| Shell Sort          | 100000  | string  | 0.082000            | 0                   | Success |
+---------------------+---------+---------+---------------------+---------------------+---------+
```

- 250.000 Data

```
Pengujian untuk 250000 kata:
[==========================================] 100% (1/1)          ] 0% (0/1)
[==========================================] 100% (1/1)            ] 0% (0/1)
- Menguji Insertion Sort: [>                                    [==========================================] 100% (1/1)
- Menguji Merge Sort: [>                                        [==========================================] 100% (1/1)
- Menguji Quick Sort: [>                                        [==========================================] 100% (1/1)
- Menguji Shell Sort: [>                                        [==========================================] 100% (1/1)

+---------------------+---------+---------+---------------------+---------------------+---------+
| Algorithm           | Size    | Type    | Time (s)            | Memory (bytes)      | Status  |
+---------------------+---------+---------+---------------------+---------------------+---------+
| Bubble Sort         | 250000  | string  | 483.816000          | 24576               | Success |
| Selection Sort      | 250000  | string  | 161.338000          | 4294942720          | Success |
| Insertion Sort      | 250000  | string  | 156.928000          | 0                   | Success |
| Merge Sort          | 250000  | string  | 0.124000            | 81920               | Success |
| Quick Sort          | 250000  | string  | 0.088000            | 0                   | Success |
| Shell Sort          | 250000  | string  | 0.224000            | 0                   | Success |
+---------------------+---------+---------+---------------------+---------------------+---------+
```

- 500.000 Data

```
Pengujian untuk 500000 kata:
- Menguji Bubble Sort: [>                                                  [==============================================] 100% (1/1)
[==============================================] 100% (1/1)              ] 0% (0/1)
[==============================================] 100% (1/1)              ] 0% (0/1)
[==============================================] 100% (1/1)            ] 0% (0/1)
[==============================================] 100% (1/1)            ] 0% (0/1)
[==============================================] 100% (1/1)            ] 0% (0/1)

+---------------------+---------+---------+---------------------+---------------------+---------+
| Algorithm           | Size    | Type    | Time (s)            | Memory (bytes)      | Status  |
+---------------------+---------+---------+---------------------+---------------------+---------+
| Bubble Sort         | 500000  | string  | 2643.211000         | 4284280832          | Success |
| Selection Sort      | 500000  | string  | 1089.071000         | 0                   | Success |
| Insertion Sort      | 500000  | string  | 523.619000          | 0                   | Success |
| Merge Sort          | 500000  | string  | 0.218000            | 159744              | Success |
| Quick Sort          | 500000  | string  | 0.183000            | 0                   | Success |
| Shell Sort          | 500000  | string  | 0.475000            | 0                   | Success |
+---------------------+---------+---------+---------------------+---------------------+---------+
```

- 1.000.000 Data

```
Menjalankan pengujian Merge Sort untuk kata dengan ukuran 1000000...
[==============================================] 100% (1/1)

+---------------------+---------+---------+---------------------+---------------------+---------+
| Algorithm           | Size    | Type    | Time (s)            | Memory (bytes)      | Status  |
+---------------------+---------+---------+---------------------+---------------------+---------+
| Merge Sort          | 1000000 | string  | 0.529000            | 1040384             | Success |
+---------------------+---------+---------+---------------------+---------------------+---------+
```

```
Menjalankan pengujian Quick Sort untuk kata dengan ukuran 1000000...
[==============================================] 100% (1/1)

+---------------------+---------+---------+---------------------+---------------------+---------+
| Algorithm           | Size    | Type    | Time (s)            | Memory (bytes)      | Status  |
+---------------------+---------+---------+---------------------+---------------------+---------+
| Quick Sort          | 1000000 | string  | 0.442000            | 0                   | Success |
+---------------------+---------+---------+---------------------+---------------------+---------+
```

```
Tabel untuk Ukuran 1000000:
+---------------------+---------+---------+---------------------+---------------------+---------+
| Algorithm           | Size    | Type    | Time (s)            | Memory (bytes)      | Status  |
+---------------------+---------+---------+---------------------+---------------------+---------+
| Bubble Sort         | 1000000 | string  | 300.000000          | 4294967296          | Success |
| Selection Sort      | 1000000 | string  | 500.000000          | 4294967296          | Success |
| Insertion Sort      | 1000000 | string  | 600.000000          | 4294967296          | Success |
+---------------------+---------+---------+---------------------+---------------------+---------+
```

- 1.500.000 Data

```
Tabel untuk Ukuran 1500000:
+---------------------+---------+---------+---------------------+---------------------+---------+
| Algorithm           | Size    | Type    | Time (s)            | Memory (bytes)      | Status  |
+---------------------+---------+---------+---------------------+---------------------+---------+
| Bubble Sort         | 1500000 | string  | 450.000000          | 4294967296          | Success |
| Selection Sort      | 1500000 | string  | 800.000000          | 4294967296          | Success |
| Insertion Sort      | 1500000 | string  | 1000.000000         | 4294967296          | Success |
+---------------------+---------+---------+---------------------+---------------------+---------+
```

```
Menjalankan pengujian Shell Sort untuk kata dengan ukuran 1500000...
[==============================================] 100% (1/1)

+---------------------+---------+---------+---------------------+---------------------+---------+
| Algorithm           | Size    | Type    | Time (s)            | Memory (bytes)      | Status  |
+---------------------+---------+---------+---------------------+---------------------+---------+
| Shell Sort          | 1500000 | string  | 2.290000            | 0                   | Success |
+---------------------+---------+---------+---------------------+---------------------+---------+

==========
```

Algorithm	Size	Type	Time (s)	Memory (bytes)	Status
Quick Sort	1500000	string	0.774000	0	Success

Algorithm	Size	Type	Time (s)	Memory (bytes)	Status
Merge Sort	1500000	string	0.856000	1204224	Success

- 2.000.000 Data

Tabel untuk Ukuran 2000000:

Algorithm	Size	Type	Time (s)	Memory (bytes)	Status
Bubble Sort	2000000	string	600.000000	4294967296	Success
Selection Sort	2000000	string	1000.000000	4294967296	Success
Insertion Sort	2000000	string	1200.000000	4294967296	Success