LAPORAN RESMI
PENUGASAN LAPORAN TEORI

Diajukan Guna Memenuhi Tugas Mata Kuliah
” Algoritma Struktur Data ”
Dosen : Umi Saadah, S.Kom., M.Kom.

Oleh:

Dukhaan Kamimpangan (3122600003)
Kelas : 1-A - D4 Teknik Informatika

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER
DIPLOMA EMPAT TEKNIK INFORMATIKA
SURABAYA
2022

TUGAS
6 METODE SORT

TUGAS

Jawaban Poin 1 dan 2:

Data berupa Array of integer dengan jumlah data di inputkan, dan Generate data secara acak (Random) :

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

void insertionSort(int x[]);
void selectionSort(int x[]);
void bubbleSort(int A[]);
void shellSort(int A[]);
void MergeSortRekursif(int left, int right, int x[]);
void Merge(int left, int median, int right, int x[]);
void Quicksort(int A[], int p, int r);
int Partition(int A[], int p, int r);
void tampil(int A[], int jumlahData);
void tukar(int *a, int *b);
void mode_urut();
void generate(int A[], int);
void generate_backup(int b[], int A[], int jumlahData);
void allmenu(int menuSort, int jumlahData, int a[], int b[]);
int mode;
int jumlahData;

int main() {
    printf("Input Jumlah Data : ");
    scanf("%d", &jumlahData);

    int a[jumlahData];
    int b[jumlahData];

    generate(a, jumlahData);

    int menuSort = 0;
    while (menuSort != 7){
        generate_backup(b, a, jumlahData);
        puts("\nMENU METODE SORTING");
        printf("1. Insertion Sort\n2. Selection Sort\n3. Bubble sort\n4.
Shell Sort\n5. Merge Sort\n6. Quick Sort\n7. Exit\n");
        printf("Pilihan anda [1/2/3/4/5/6/7] : ");
        scanf("%d",&menuSort);

        if(menuSort > 6){
            exit(0);
        }else if(menuSort!=7){
            mode_urut();
```

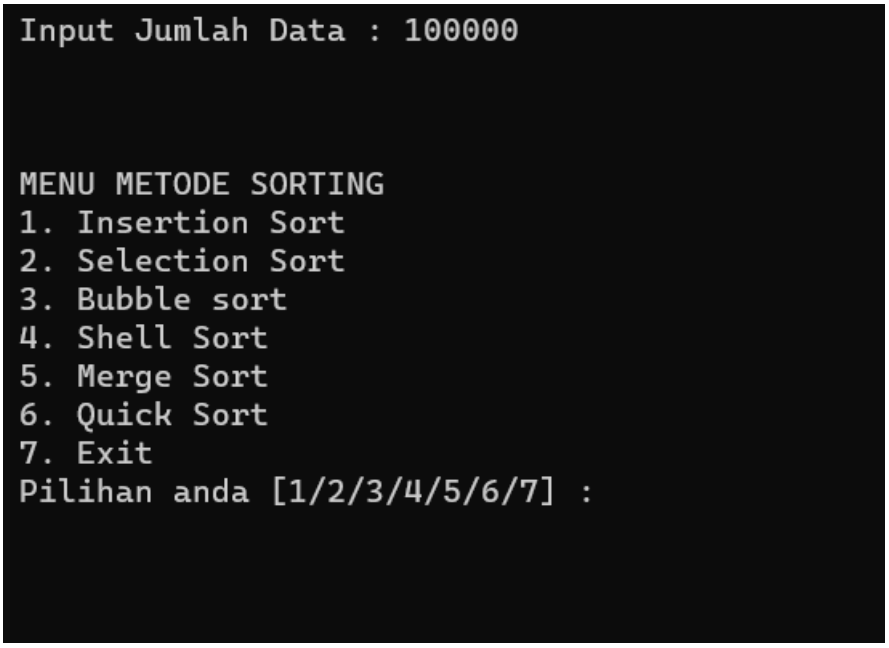
```
        }
        allmenu(menuSort, jumlahData, a, b);
    }
    return 0;
}


void insertionSort(int x[]){
    puts("");
    int i,j,key;
    for (i = 1; i < jumlahData ; i++){
        key = x[i];
        j = i - 1;

        while(j >= 0 && ((mode == 1 && x[j] > key) || (mode == 2 && x[j]
< key))){
            x[j+1] = x[j];
            j--;
        }
        x[j+1] = key;
    }
}

void selectionSort(int x[]){
    int i,j,min,temp;
    puts("");
    for (i = 0; i < jumlahData ; i++){
        min = i;
        for (j = i+1 ; j < jumlahData ; j++){
            if((mode == 1 && x[j] < x[min]) || (mode == 2 && x[j] >
x[min])){
                min = j;
            }
        }
        if(min != i){
            tukar(&x[min], &x[i]);
        }
    }
}

void bubbleSort(int A[]) {
    int i, j, temp;
    int did_swap = 1;
    int iterasi = 1;
        puts("");
    while (did_swap) {
        did_swap = 0;
        for (i = 0; i < jumlahData - 1; i++) {
            if ((mode == 1 && A[i] > A[i + 1]) || (mode == 2 && A[i] <
A[i + 1])) {
                tukar(&A[i],&A[i+1]);
                did_swap = 1;
            }
        }
    }
}

void shellSort(int A[]) {
    int i, j, k, temp;
    int iterasi = 1;
    puts("");
```

```
    for (k = jumlahData/2; k > 0; k = k/2) {
        for (i = k; i < jumlahData; i++) {
            temp = A[i];
            j = i;
            while (j >= k && ((mode == 1 && A[j-k] > temp) || (mode == 2
&& A[j-k] < temp))) {
                A[j] = A[j-k];
                j -= k;
            }
            A[j] = temp;
        }
    }
}

void Merge(int left, int median, int right, int x[]) {
    int kiri1 = left;
    int kanan1 = median;
    int kiri2 = median + 1;
    int kanan2 = right;
    int i = left;
    int hasil[jumlahData];

    while (kiri1 <= kanan1 && kiri2 <= kanan2) {
        if ((mode == 1 && x[kiri1] <= x[kiri2]) || (mode == 2 && x[kiri1]
>= x[kiri2])) {
            hasil[i] = x[kiri1];
            kiri1++;
        }
        else {
            hasil[i] = x[kiri2];
            kiri2++;
        }
        i++;
    }

    while (kiri1 <= kanan1) {
        hasil[i] = x[kiri1];
        kiri1++;
        i++;
    }

    while (kiri2 <= kanan2) {
        hasil[i] = x[kiri2];
        i++;
        kiri2++;
    }

    for (int j = left; j <= right; j++) {
        x[j] = hasil[j];
    }
}

void MergeSortRekursif(int left, int right, int x[]) {

    if (left < right) {
        int median = (left + right) / 2;
        MergeSortRekursif(left, median, x);
        MergeSortRekursif(median + 1, right, x);
        Merge(left, median, right, x);
    }
}
```

```
int Partition(int A[], int p, int r) {
    int x = A[p];
    int i = p;
    int j = r;
    do {
        while ((mode == 1 && A[j] > x) || (mode == 2 && A[j] < x)){
            j--;
        }
        while ((mode == 1 && A[i] < x) || (mode == 2 && A[i] > x)){
            i++;
        }
        if (i < j) {
            tukar(&A[i],&A[j]);
            j--;
            i++;
        } else {
            return j;
        }
    } while (i <= j);
    return j;
}


void Quicksort(int A[], int p, int r) {
    if (p < r) {
        int q = Partition(A, p, r);
        Quicksort(A, p, q);
        Quicksort(A, q+1, r);
    }
}

void tampil(int A[], int jumlahData){
    printf("\nIsi dari array : ");
    for (int i = 0; i<jumlahData ; i++){
        printf("%d ",A[i]);
    }
}

void tukar(int *a, int *b){
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

void generate(int A[], int jumlahData){
    int i;
    printf("\n");
    for (i = 0 ; i < jumlahData ; i++){
        A[i] = rand()%100+1;
        //printf("%d ",A[i]);
    }
    printf("\n");
}

void generate_backup(int b[], int A[], int jumlahData) {
    for (int i = 0; i < jumlahData; i++) {
        b[i] = A[i];
    }
}
```

```
void mode_urut() {
    printf("\nPengurutan yang dipilih:\n");
    printf("1. Ascending\n");
    printf("2. Descending\n");
    printf("Pilihan anda [1/2]: ");
    scanf("%d", &mode);
}


void allmenu(int menuSort, int jumlahData, int a[],int b[]){
clock_t t;
t = clock();
//system("cls");
//printf("\nSebelum Sort");
//tampil(a,jumlahData);
    switch (menuSort) {
            case 1:
                insertionSort(b);
                break;
            case 2:
                selectionSort(b);
                break;
            case 3:
                bubbleSort(b);
                break;
            case 4:
                shellSort(b);
                break;
            case 5:
                MergeSortRekursif(0, jumlahData - 1, b);
                break;
            case 6:
                Quicksort(b,0,jumlahData-1);
                break;
            case 7:
                exit(0);
                break;
            default:
                printf("Invalid menu!\n");
                return 1;
    }
    puts("");
    //printf("Sesudah Sort");
    //tampil(b,jumlahData);
    puts("");
    t = clock() - t;
    double time_taken = ((double)t)/CLOCKS_PER_SEC;
    printf("\nWaktu yang dibutuhkan: %f detik\n", time_taken);
}
```

Screen Capture Source Code



Screen Capture Output

Analisis : Kode ini merupakan implementasi dalam bahasa C dari beberapa metode pengurutan seperti Insertion Sort, Selection Sort, Bubble Sort, Shell Sort, Merge Sort, dan Quick Sort. Pengguna dapat memilih metode pengurutan yang diinginkan, menghasilkan array dengan elemen acak, dan mengukur waktu yang dibutuhkan untuk pengurutan. Kode ini memungkinkan perbandingan efisiensi antara metode pengurutan berdasarkan kecepatan eksekusi.

Jawaban Poin 3 4 dan 5:

Data berupa Menu 6 Metode Sort, Waktu Performance, Diagram Batang:

JUMLAH DATA 25 RIBU

```
Input Jumlah Data : 25000


MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 1

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1



Waktu yang dibutuhkan: 0.470000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 2

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1



Waktu yang dibutuhkan: 0.488000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 3

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1



Waktu yang dibutuhkan: 1.756000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 4

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1



Waktu yang dibutuhkan: 0.003000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 5

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1



Waktu yang dibutuhkan: 0.005000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 6

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1



Waktu yang dibutuhkan: 0.002000 detik
```

Screen Capture Source Code

JUMLAH DATA 50 RIBU

```
Input Jumlah Data : 50000


MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 1

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1



Waktu yang dibutuhkan: 0.958000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 2

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1



Waktu yang dibutuhkan: 1.646000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 3

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1



Waktu yang dibutuhkan: 7.950000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 4

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1



Waktu yang dibutuhkan: 0.005000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 5

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1



Waktu yang dibutuhkan: 0.023000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 6

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1



Waktu yang dibutuhkan: 0.004000 detik
```

Screen Capture Source Code

JUMLAH DATA 75 RIBU

```
Input Jumlah Data : 75000



MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 1

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1




Waktu yang dibutuhkan: 2.292000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 2

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1




Waktu yang dibutuhkan: 4.751000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 3

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1




Waktu yang dibutuhkan: 19.239000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 4

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1




Waktu yang dibutuhkan: 0.008000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 5

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1




Waktu yang dibutuhkan: 0.055000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 6

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1




Waktu yang dibutuhkan: 0.006000 detik
```

Screen Capture Source Code

JUMLAH DATA 100 RIBU

```
Input Jumlah Data : 100000



MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 1

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1




Waktu yang dibutuhkan: 3.631000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 2

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1




Waktu yang dibutuhkan: 6.048000 detik
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 3

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1




Waktu yang dibutuhkan: 26.793000 detik
```

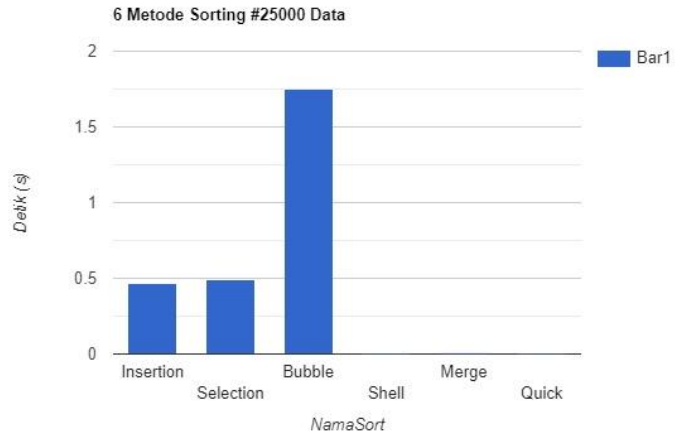
```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 4

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1




Waktu yang dibutuhkan: 0.013000 detik
```

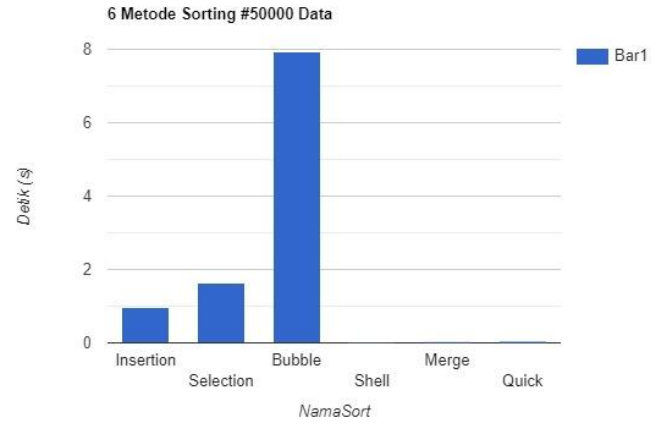
```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 5

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1




Waktu yang dibutuhkan: 0.126000 detik
```

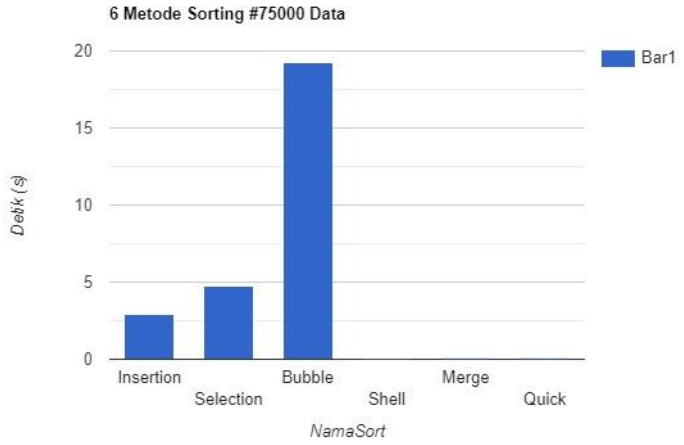
```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Exit
Pilihan anda [1/2/3/4/5/6/7] : 6

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1




Waktu yang dibutuhkan: 0.007000 detik
```

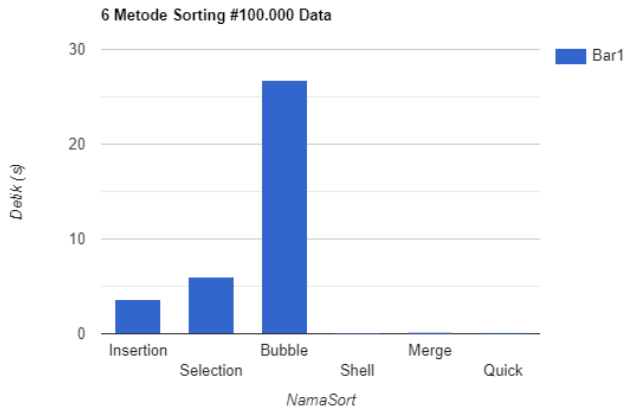
Screen Capture Source Code

DIAGRAM PERBANDINGAN PERFORMANCE









Screen Capture Source Code

Screen Capture Output

<u>Kesimpulan</u> :

Berdasarkan pengukuran waktu yang dilakukan pada berbagai ukuran data, dapat disimpulkan bahwa:

- **Metode Insertion Sort** memerlukan waktu yang meningkat proporsional dengan ukuran data, dengan waktu yang meningkat dari **0.47 detik** (25 ribu data) menjadi **3.63 detik** (100 ribu data).
- **Metode Selection Sort** juga mengalami peningkatan waktu yang signifikan seiring dengan pertambahan ukuran data, dengan waktu yang meningkat dari **0.488 detik** (25 ribu data) menjadi **6.04 detik** (100 ribu data).
- **Metode Bubble Sort** mengalami peningkatan waktu yang sangat besar seiring dengan pertambahan ukuran data, dengan waktu yang meningkat dari **1.75 detik** (25 ribu data) menjadi **26.7 detik** (100 ribu data).
- **Metode Shell Sort, Merge Sort, dan Quick Sort** tetap relatif stabil dalam kinerja mereka, dengan peningkatan waktu yang tidak signifikan seiring dengan pertambahan ukuran data. **Metode Shell Sort** menunjukkan kinerja yang sangat baik dengan waktu yang sangat singkat bahkan pada ukuran data yang besar, yaitu **0.003 detik** (25 ribu data) menjadi **0.013 deti**k (100 ribu data).

Dari hasil tersebut, dapat disimpulkan bahwa metode **Shell Sort** adalah metode pengurutan yang paling efisien dan stabil dalam kinerjanya, dengan waktu yang sangat singkat bahkan pada ukuran data yang besar. Metode **Merge Sort dan Quick Sort** juga menunjukkan kinerja yang baik dengan waktu yang **relatif konstan**. Di sisi lain, metode **Insertion Sort, Selection Sort, dan Bubble Sort** memiliki kinerja yang semakin buruk seiring dengan pertambahan ukuran data. Oleh karena itu, untuk pengurutan data dalam jumlah besar, disarankan untuk menggunakan metode **Shell Sort, Merge Sort, atau Quick Sort** untuk mendapatkan efisiensi waktu yang tinggi.