ALGORITMA STRUKTUR DATA

“PERFORMA SORTING – TEORI M12”

DOSEN :

Umi Sa’adah S.Kom., M.Kom. (197404162000032003)

PENYUSUN :

Mohammad Ihsan Septiano Firdaus (3125600041)

PROGRAM STUDI SARJANA TERAPAN

TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

TUGAS SORTING :

1. Implementasikan 6 metode sorting yang sudah dibahas
a. data berupa *array of int*
b. jumlah data min 25000 (lakukan 4 percobaan dg jml data yang berbeda)
c. generate data secara random
2. Lampirkan listing program dalam bentuk menu untuk 6 metode tsb
3. Capture output utk setiap metode, utk setiap jumlah data yang berbeda tampilkan performance berupa waktu komputasinya
4. Buatlah perbandingan performa masing-masing metode tampilkan dalam bentuk diagram batang.
5. Buat Analisis & Kesimpulan

Program

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

void insertionSort(int a[], int b, int n);
void selectionSort(int a[], int b, int n);
void bubbleSort(int a[], int b, int n);
void shellSort(int a[], int b, int n);
void mSortRek(int a[], int b , int l, int r);
void mSort(int a[], int med, int l, int r, int b);
void quickSort(int a[], int b, int l, int r);
void tukar(int *a, int *b);
void copy(int a[], int b[]);
void generate(int a[]);
void tampilkan(int a[]);
void compare(int a[]);
int n, backup[100000], a[1000000];

int main() {
    int sort, alur;
    srand(time(NULL));
    generate(backup);

    do {
        printf("\nMenu Sorting\n1. Insertion\n2. Selection\n3.
Bubble\n4. Shell\n5. Merge\n6. Quick\n7. Banding Performa\n8. Regenerate
Data\n9. Keluar\nPilihan Anda : ");
        scanf("%d", &sort);

        if (sort == 9) {
```

```
            break;
        }
        if (sort < 1 || sort > 8) {
            printf("Masukkan Opsi yang benar!\n");
            continue;
        }
        if (sort > 0 && sort < 7)
        { printf("\nMode urut\n1. Ascending\n2. Descending\nPilihan
Anda : ");
        scanf("%d", &alur);}

        copy(a, backup);
        // printf("Unsorted -> ");
        // tampilkan(a);
        clock_t start = clock();
        switch(sort) {
            case 1:
                insertionSort(a, alur, n);
                break;
            case 2:
                selectionSort(a, alur, n);
                break;
            case 3:
                bubbleSort(a, alur, n);
                break;
            case 4:
                shellSort(a, alur, n);
                break;
            case 5:
                mSortRek(a, alur,0, n-1 );
                break;
            case 6:
                quickSort(a, alur, 0, n-1);
                break;
            case 7:
                compare(a);
                continue;
            case 8:
                generate(backup);
                continue;
        }
        clock_t end = clock();

        double waktu = (double)(end - start) / CLOCKS_PER_SEC;
        printf("Durasi = %g\n", waktu);
```

```
        printf("-------------------------------------\n");
        // printf("Sorted -> ");
        // tampilkan(a);
        Sleep(3000);
    } while (sort != 9);

    return 0;
}

void tampilkan(int a[]) {
    for (int i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}

void generate(int a[]) {
    printf("Berapa jumlah data (maks 100000) ? ");
    scanf("%d", &n);
    printf("jml = %d\n", n);
    for(int i=0; i<n; i++)
        a[i] = rand()/1000;
}

void copy(int a[], int b[]) {
    for (int i = 0; i < n; i++)
        a[i] = b[i];
}

void insertionSort(int a[], int b, int n) {
    for (int i = 1; i < n; i++) {
        int key = a[i];
        int j=i-1;
        if(b == 1)
            while ( j >= 0 && a[j] > key ) {
                a[j+1] = a[j];
                j--; }
        if(b == 2)
            while ( j >= 0 && a[j] < key ) {
                a[j+1] = a[j];
                j--; }
        a[j+1]=key; }
}

void selectionSort(int a[], int b, int n) {
```

```
    for (int i = 0; i < n; i++) {
        int j = i+1, min = i;
        while (j < n) {
            if(b == 1)
                if (a[j] < a[min]) min= j;
            if(b == 2)
                if (a[j] > a[min]) min= j;
            j++ ; }
        tukar (&a[i], &a[min]);
    }
}

void bubbleSort(int a[], int b, int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            int kondisi = 0;
            if (b == 1) kondisi = (a[j] > a[j + 1]);
            if (b == 2) kondisi = (a[j] < a[j + 1]);
            if (kondisi) {
                tukar(&a[j], &a[j + 1]);
            }
        }
    }
}

void shellSort(int a[], int b, int n) {
    for (int gap = n / 2; gap > 0; gap /= 2) {
        for (int i = gap; i < n; i++) {
            for (int j = i - gap; j >= 0; j -= gap) {
                int kondisi = 0;
                if (b == 1) kondisi = (a[j] > a[j + gap]);
                if (b == 2) kondisi = (a[j] < a[j + gap]);
                if (kondisi) {
                    tukar(&a[j], &a[j + gap]);
                } else {
                    break;
                }
            }
        }
    }
}

void mSortRek(int a[], int b , int l, int r) {
    if (l<r) {
        int med = (l+r) / 2;
```

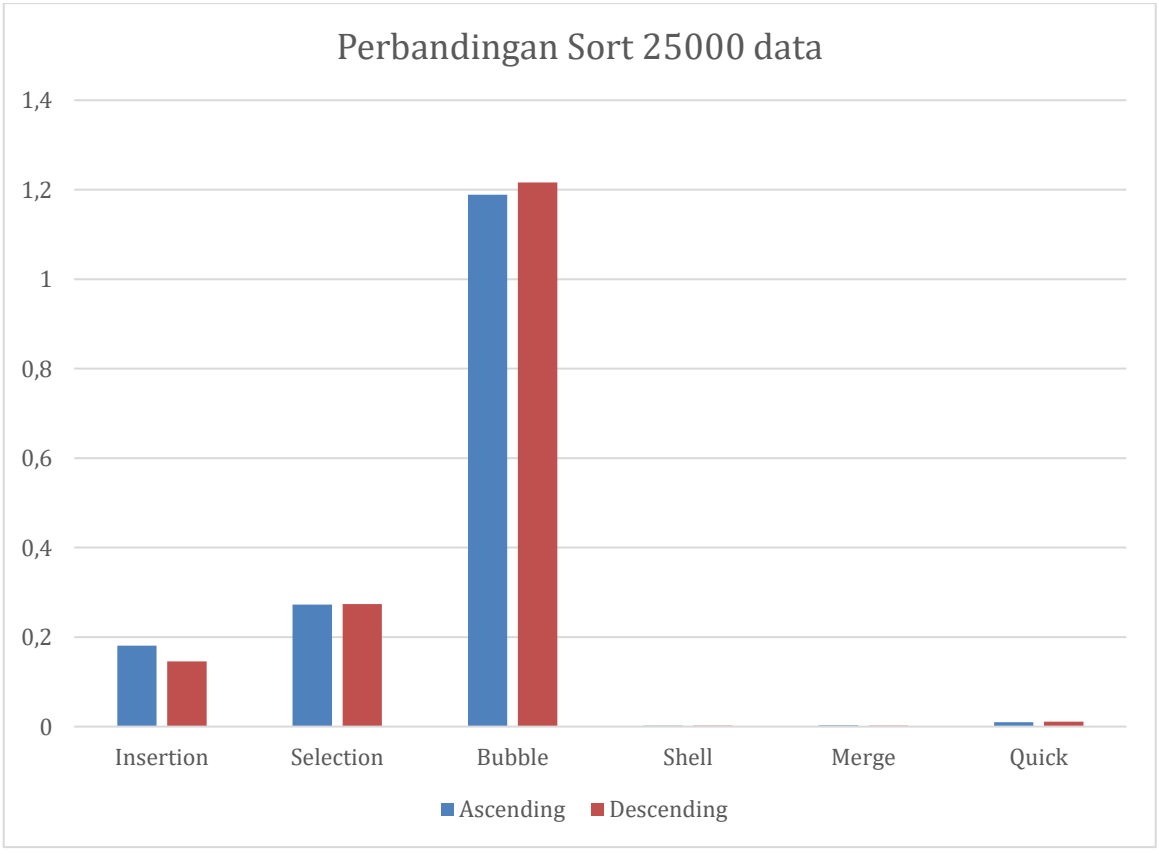
```
        mSortRek (a, b, l, med);
        mSortRek (a, b, med+1, r);
        mSort (a, med, l, r, b);
    }
}

void mSort(int a[], int med, int l, int r, int b) {
    int n1 = med - l + 1;
    int n2 = r - med;
    int f[n1], g[n2];
    for(int i = 0; i < n1; i++) f[i]  = a[l + i];
    for(int i = 0; i < n2; i++) g[i]  = a[med + 1 + i];
    int i = 0, j = 0, m = l;
    while (i < n1 && j < n2) {
int kondisi = 0;
        if (b == 1) kondisi = (f[i] <= g[j]);
        if (b == 2) kondisi = (f[i] >= g[j]);

        if (kondisi) a[m++] = f[i++];
        else a[m++] = g[j++];
    }
    while (i < n1) a[m++] = f[i++];
    while (j < n2) a[m++] = g[j++];
}

void quickSort(int a[], int b, int l, int r) {
    if (l < r) {
        int pivot = a[l];
        int i = l;
        for (int j = l + 1; j <= r; j++) {
            int kondisi = 0;
            if (b == 1) kondisi = (a[j] < pivot);
            if (b == 2) kondisi = (a[j] > pivot);

            if (kondisi) {
                i++;
                tukar(&a[i], &a[j]);
            }
        }

        tukar(&a[l], &a[i]);
        int pi = i;
        quickSort(a, b, l, pi - 1);
        quickSort(a, b, pi + 1, r);
    }}
```

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

void compare(int a[]) {
    int temp[n];
    clock_t start, end;
    double waktu;

    printf("\n--- HASIL PERBANDINGAN WAKTU EKSEKUSI ---\n");

    // 1. Insertion Sort
    printf("\nInsertion Sort\n");
    copy(temp, a); start = clock(); insertionSort(temp, 1, n); end =
clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Ascending  = %g detik\n", waktu);

    copy(temp, a); start = clock(); insertionSort(temp, 2, n); end =
clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Descending = %g detik\n", waktu);

    // 2. Selection Sort
    printf("\nSelection Sort\n");
    copy(temp, a); start = clock(); selectionSort(temp, 1, n); end =
clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Ascending  = %g detik\n", waktu);

    copy(temp, a); start = clock(); selectionSort(temp, 2, n); end =
clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Descending = %g detik\n", waktu);

    // 3. Bubble Sort
    printf("\nBubble Sort\n");
    copy(temp, a); start = clock(); bubbleSort(temp, 1, n); end =
clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Ascending  = %g detik\n", waktu);

    copy(temp, a); start = clock(); bubbleSort(temp, 2, n); end =
clock();
```

```
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Descending = %g detik\n", waktu);

    // 4. Shell Sort
    printf("\nShell Sort\n");
    copy(temp, a); start = clock(); shellSort(temp, 1, n); end =
clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Ascending  = %g detik\n", waktu);

    copy(temp, a); start = clock(); shellSort(temp, 2, n); end =
clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Descending = %g detik\n", waktu);

    // 5. Merge Sort
    printf("\nMerge Sort\n");
    copy(temp, a); start = clock(); mSortRek(temp, 1, 0, n - 1); end =
clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Ascending  = %g detik\n", waktu);

    copy(temp, a); start = clock(); mSortRek(temp, 2, 0, n - 1); end =
clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Descending = %g detik\n", waktu);

    // 6. Quick Sort
    printf("\nQuick Sort\n");
    copy(temp, a); start = clock(); quickSort(temp, 1, 0, n - 1); end =
clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Ascending  = %g detik\n", waktu);

    copy(temp, a); start = clock(); quickSort(temp, 2, 0, n - 1); end =
clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Descending = %g detik\n", waktu);

    printf("\n----------------------------------------\n");
    Sleep(5000);
}
```

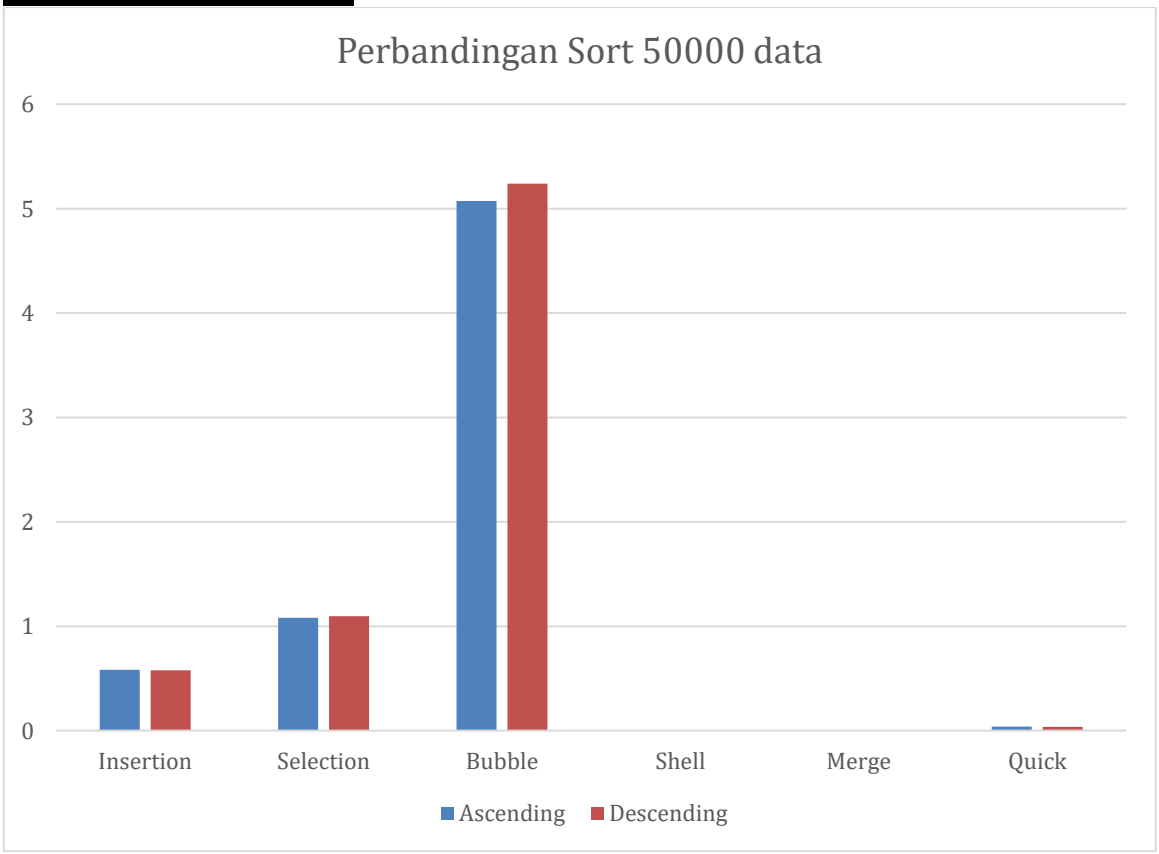
25000 Data

```
Berapa jumlah data (maks 100000) ? 25000
jml = 25000

Menu Sorting
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Banding Performa
8. Regenerate Data
9. Keluar
Pilihan Anda : 7

--- HASIL PERBANDINGAN WAKTU EKSEKUSI ---

Insertion Sort
Ascending  = 0.181 detik
Descending = 0.146 detik

Selection Sort
Ascending  = 0.273 detik
Descending = 0.274 detik

Bubble Sort
Ascending  = 1.189 detik
Descending = 1.216 detik

Shell Sort
Ascending  = 0.002 detik
Descending = 0.002 detik

Merge Sort
Ascending  = 0.003 detik
Descending = 0.002 detik

Quick Sort
Ascending  = 0.01 detik
Descending = 0.011 detik
```



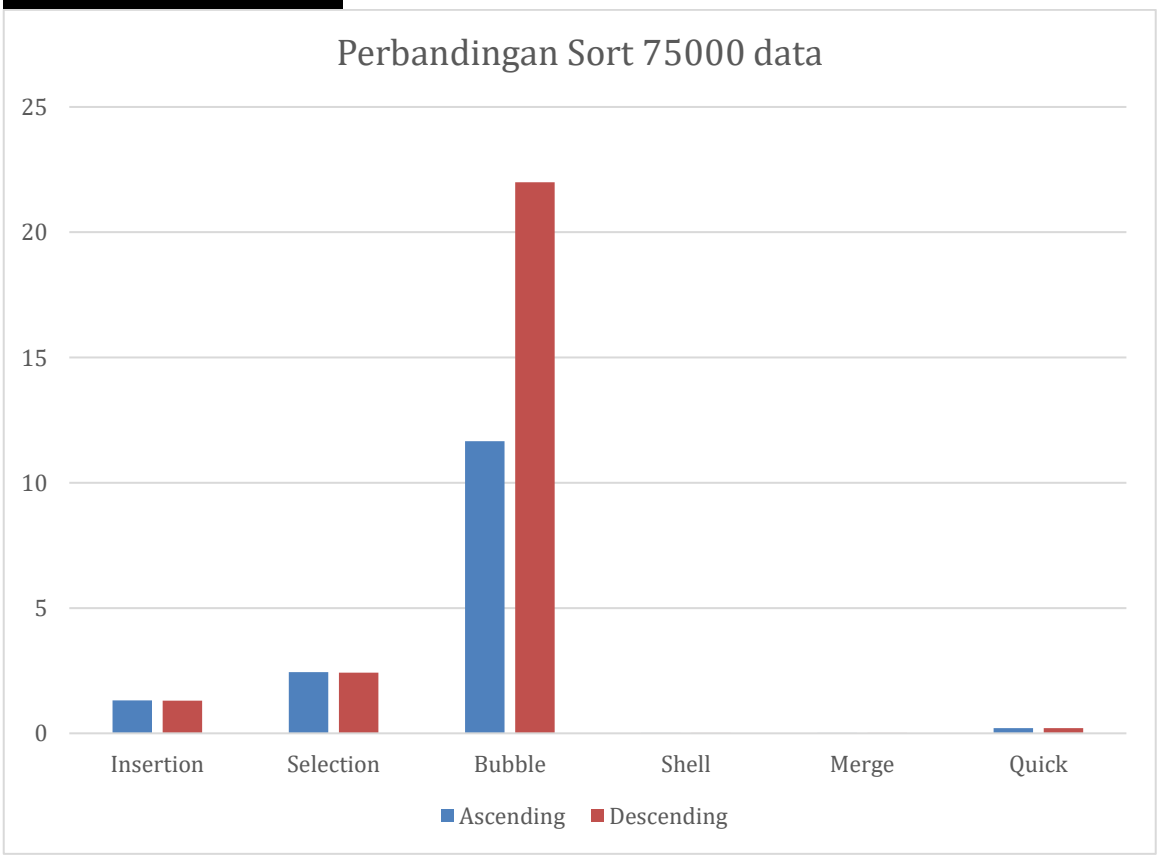
50000 Data

```
Berapa jumlah data (maks 100000) ? 50000
jml = 50000

Menu Sorting
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Banding Performa
8. Regenerate Data
9. Keluar
Pilihan Anda : 7

--- HASIL PERBANDINGAN WAKTU EKSEKUSI ---

Insertion Sort
Ascending  = 0.582 detik
Descending = 0.578 detik

Selection Sort
Ascending  = 1.081 detik
Descending = 1.097 detik

Bubble Sort
Ascending  = 5.074 detik
Descending = 5.239 detik

Shell Sort
Ascending  = 0.004 detik
Descending = 0.005 detik

Merge Sort
Ascending  = 0.005 detik
Descending = 0.006 detik

Quick Sort
Ascending  = 0.038 detik
Descending = 0.036 detik
```



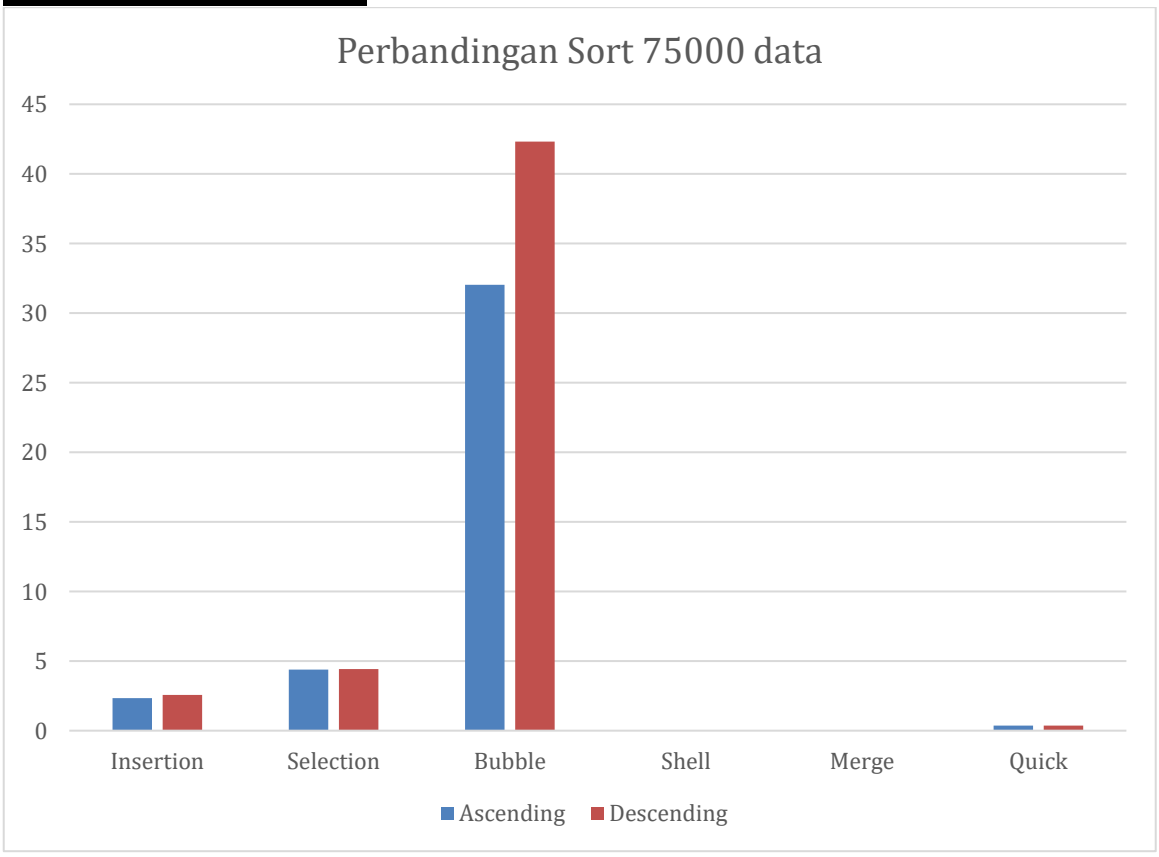
75000 Data

```
Berapa jumlah data (maks 100000) ? 75000
jml = 75000

Menu Sorting
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Banding Performa
8. Regenerate Data
9. Keluar
Pilihan Anda : 7

--- HASIL PERBANDINGAN WAKTU EKSEKUSI ---

Insertion Sort
Ascending  = 1.315 detik
Descending = 1.306 detik

Selection Sort
Ascending  = 2.443 detik
Descending = 2.417 detik

Bubble Sort
Ascending  = 11.656 detik
Descending = 21.996 detik

Shell Sort
Ascending  = 0.014 detik
Descending = 0.02 detik

Merge Sort
Ascending  = 0.017 detik
Descending = 0.018 detik

Quick Sort
Ascending  = 0.21 detik
Descending = 0.205 detik

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



100000 Data

```
Berapa jumlah data (maks 100000) ? 100000
jml = 100000

Menu Sorting
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Banding Performa
8. Regenerate Data
9. Keluar
Pilihan Anda : 7

--- HASIL PERBANDINGAN WAKTU EKSEKUSI ---

Insertion Sort
Ascending  = 2.344 detik
Descending = 2.565 detik

Selection Sort
Ascending  = 4.385 detik
Descending = 4.43 detik

Bubble Sort
Ascending  = 32.037 detik
Descending = 42.311 detik

Shell Sort
Ascending  = 0.022 detik
Descending = 0.027 detik

Merge Sort
Ascending  = 0.021 detik
Descending = 0.025 detik

Quick Sort
Ascending  = 0.364 detik
Descending = 0.367 detik

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



Kesimpulan

Pada data perbandingan performa sorting yanf telah didapatkan diatas, terlihat bahwa bubble sort konsisten meroket paling tinggi dibandingkan dengan sort yang lain. Terlihat juga bahwa shell sort dan merge sort memiliki data waktu komputasi yang saking kecilnya sampai tidak terlihat di chart.

Dapat disimpulkan bahwasanya shell sort dan merge sort memiliki waktu komputasional terkecil disusul oleh quick sort. Insertion sort dan Selection sort hanya bagus apabila jumlah datanya tidak terlalu masif. Bubble sort memiliki performa terburuk diantara sort yang lain