LAPORAN RESMI

SORTING

Umi Sa’adah S.Kom., M.Kom.

Disusun Oleh :

Nama	: Daniar Oktavian Dwiputra
Kelas	: 1 / D4 Teknik Informatika A
NRP	: 3122600030

1. Listing Program

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<time.h>
#define MAX 100000

void selection(int [], int);
void insertion(int [], int);
void bubble(int [], int);
void shell(int [], int);
void mergeSortRekursif(int [], int , int);
void merge(int [], int , int, int);
void quickSort(int [], int, int);
int partition(int [], int, int);
void tampil(int [], int);
void tukar(int *, int *);
void generate(int [], int);
void waktu(clock_t, clock_t, int);
void menu();

int main()
{
   menu();
   return 0;
}

void menu()
{
   int n, jwb, arr[MAX], backup[MAX];
   clock_t start, end;

   printf("Berapa jumlah data (maks 100000) ? ");
   scanf("%d", &n);

   generate(arr, n);

   do
   {
      memcpy(backup, arr, sizeof(int) * n);
      printf("\nMENU METODE SORTING\n");
      printf("1. Insertion\n");
printf("2. Selection\n");
printf("3. Bubble\n");
printf("4. Shell\n");
printf("5. Merge\n");
printf("6. Quick\n");
printf("7. Keluar\n");
printf("Pilihan Anda : ");
scanf("%d", &jwb);

switch (jwb)
{
case 1:
   insertion(backup, n);
   break;
case 2:
   selection(backup, n);
   break;
case 3:
   bubble(backup, n);
   break;
case 4:
   shell(backup, n);
   break;
case 5:
   start = clock();
   mergeSortRekursif(backup, 0, n);
   end = clock();
   waktu(start, end, n);
   break;
case 6:
   srand(time(NULL));
   start = clock();
   quickSort(backup, 0, n-1);
   end = clock();
   waktu(start, end, n);
   break;
case 7:
   printf("\nPROGRAM DIHENTIKAN\n");
   exit(0);
   break;
default:
   printf("\nPilihan Anda Invalid\n");
```

```
         break;
      }
   } while (jwb != 7);
}


void insertion(int x[], int n)
{
   int i, j, key;

   clock_t start, end;

   start = clock();
   i = 1;
   while(i < n)
   {
      key = x[i];
      j = i - 1;
      while (j >= 0 && (x[j] > key))
      {
         x[j+1] = x[j];
         j--;
      }
      x[j+1] = key;
      i++;
   }
   end = clock();
   waktu(start, end, n);
}

void selection(int x[], int n)
{
   int i, j, min;
   clock_t start, end;

   start = clock();
   while(i < n)
   {
      min = i;
      j = i + 1;
      while(j < n)
      {
            if(x[j] < x[min])
            {
                min = j;
            }
            j++;
        }
        tukar(&x[i], &x[min]);
        i++;
    }
    end = clock();
    waktu(start, end, n);
}

void bubble(int x[], int n)
{
    int i, j, did_swap;
    clock_t start, end;
    did_swap = 1;

    start = clock();
    for(i = 0; i < n-1; i++)
    {
        if (did_swap)
        {
            did_swap = 0;
            for(j = 0; j < (n - i - 1); j++)
            {
                if (x[j] > x[j+1])
                {
                    tukar(&x[j], &x[j+1]);
                    did_swap = 1;
                }
            }

        }
    }
    end = clock();
    waktu(start, end, n);
}

void shell(int x[], int n)
{
    int jarak = n / 2;
    int i, did_swap;
    clock_t start, end;
    start = clock();
    while (jarak > 0)
    {
        did_swap = 1;
        while (did_swap == 1)
        {
            did_swap = 0;
            for (i = 0; i < n - jarak; i++)
            {
                if (x[i] > x[i + jarak])
                {
                    tukar(&x[i], &x[i + jarak]);
                    did_swap = 1;
                }
            }
        }
        jarak = jarak / 2;
    }
    end = clock();
    waktu(start, end, n);
}

void mergeSortRekursif(int data[], int l, int r)
{
    int med;
    if(l < r)
    {
        med = (l+r) / 2;
        mergeSortRekursif(data, l, med);
        mergeSortRekursif(data, med+1, r);
        merge(data, l, med, r);
    }
}

void merge(int data[], int l , int m, int r)
{
    int i, j, ki1, ki2, ka1, ka2;
    int hasil [MAX];
```

```
   ki1 = l;
   ka1 = m;
   ki2 = m+1;
   ka2 = r;
   i = l;

   while (ki1 <= ka1 && ki2 <= ka2)
   {
      if(data[ki1] <= data[ki2])
      {
         hasil[i] = data[ki1];
         ki1++;
      }
      else
      {
         hasil[i] = data[ki2];
         ki2++;
      }
      i++;
   }
   while (ki1 <= ka1)
   {
      hasil[i] = data[ki1];
      ki1++;
      i++;
   }
   while (ki2 <= ka2)
   {
      hasil[i] = data[ki2];
      ki2++;
      i++;
   }

   j = l;
   while (j <= r)
   {
      data[j] = hasil[j];
      j++;
   }
}

void quickSort(int A[], int p, int r)
```

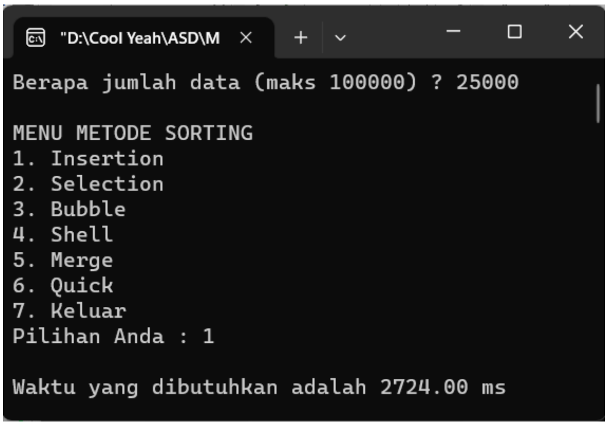
```
{
   int q;

   if (p < r)
   {
      q = partition(A, p, r);
      quickSort(A, p, q-1);
      quickSort(A, q+1, r);
   }

}

int partition(int A[], int p , int r)
{
   int i, j, x;
   x = A[r];
   i = p - 1;

   for (j = p; j < r; j++)
   {
      if (A[j] <= x)
      {
         i++;
         tukar(&A[i], &A[j]);
      }
   }
   tukar(&A[i+1], &A[r]);
   return (i+1);
}

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

   temp = *a;
   *a = *b;
   *b = temp;
}

void tampil(int A[], int n)
{
   int i;
```

```
    for (i = 0; i < n; i++)
    {
        printf("%d ", A[i]);
    }
    printf("\n");
}

void generate(int x[], int n)
{
    int i;
    srand(time(NULL));
    for(i = 0; i<n; i++)
    {
        x[i] = rand()/1000;
    }
}

void waktu(clock_t start, clock_t end, int n)
{
    double cpu_time_used;
    cpu_time_used = ((double)end - start) / CLOCKS_PER_SEC;

    printf("\nWaktu yang dibutuhkan adalah %.2f ms\n", cpu_time_used * 1000);
}
```

2. Implementasi 6 Metode Sorting

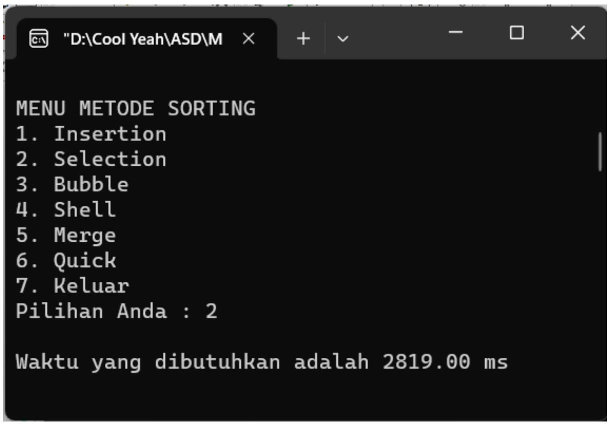
a) 25000 data

   1) Insertion



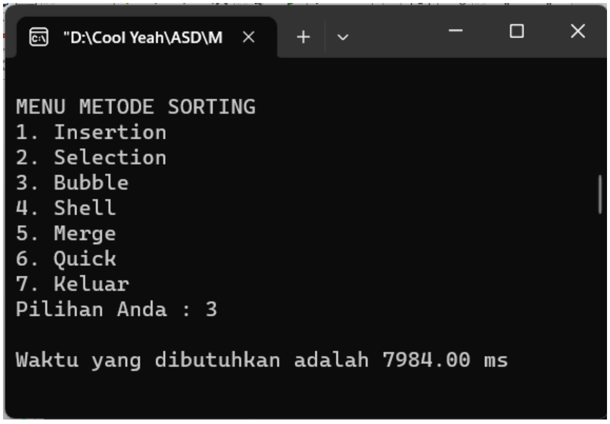
```
Berapa jumlah data (maks 100000) ? 25000

MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 1

Waktu yang dibutuhkan adalah 2724.00 ms
```

2) Selection



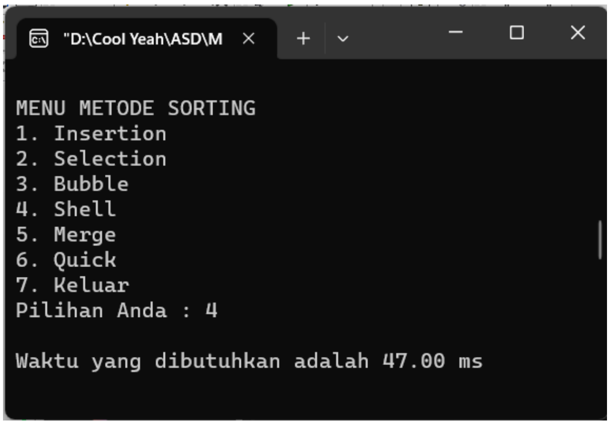
```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 2

Waktu yang dibutuhkan adalah 2819.00 ms
```

3) Bubble



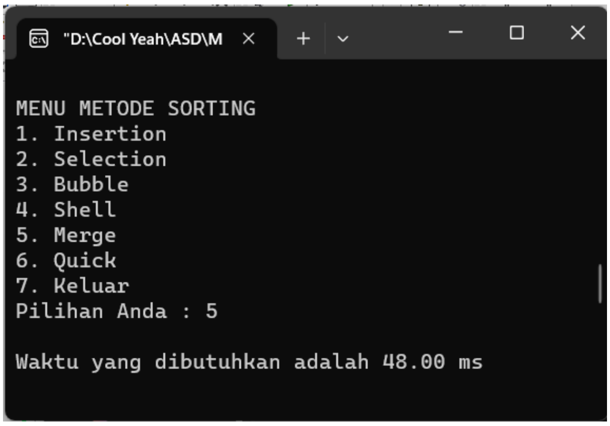
```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 3

Waktu yang dibutuhkan adalah 7984.00 ms
```

4) Shell



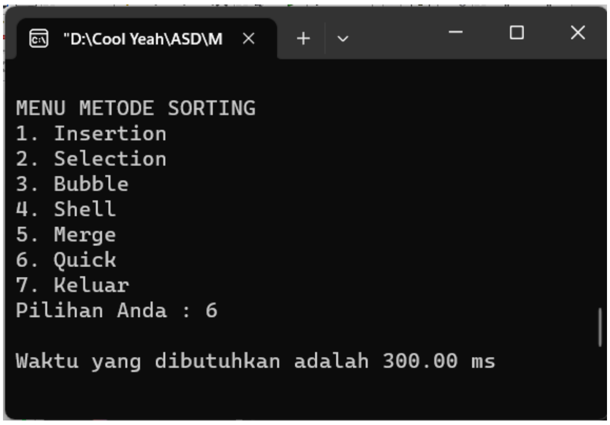
```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 4

Waktu yang dibutuhkan adalah 47.00 ms
```

5) Merge



```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 5

Waktu yang dibutuhkan adalah 48.00 ms
```

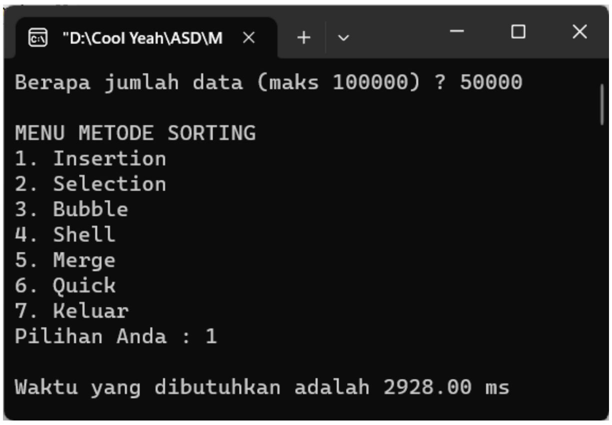
6) Quick



```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 6

Waktu yang dibutuhkan adalah 300.00 ms
```

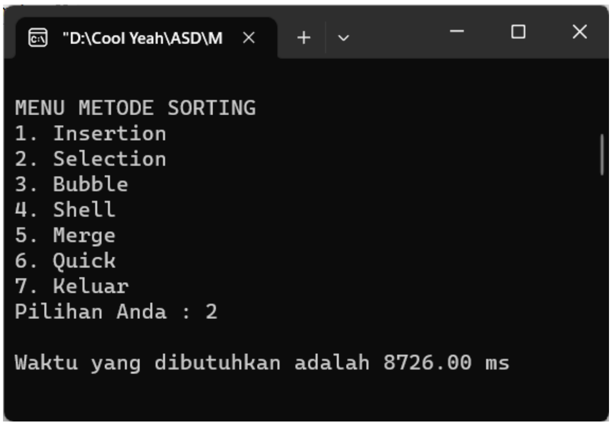
b) 50000 data

1) Insertion



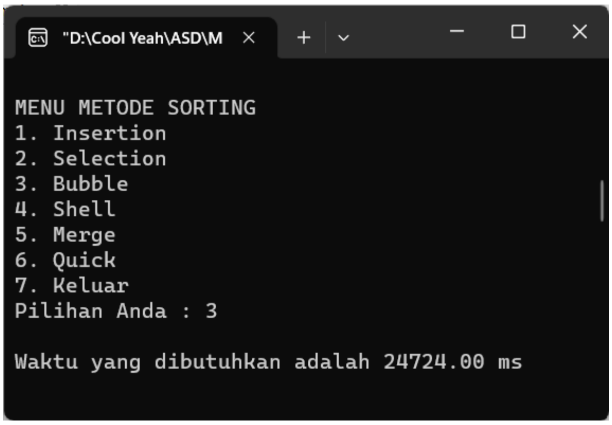
```
Berapa jumlah data (maks 100000) ? 50000

MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 1

Waktu yang dibutuhkan adalah 2928.00 ms
```

2) Selection



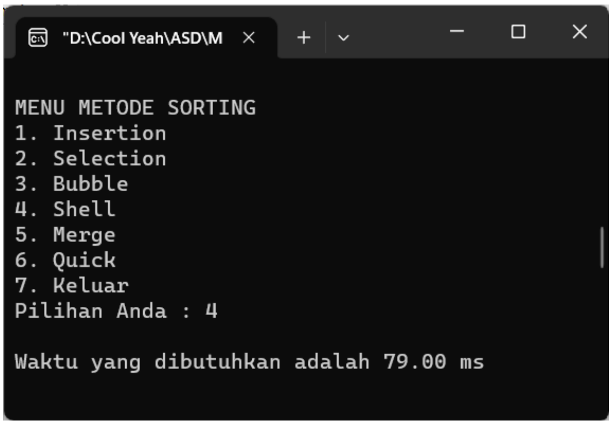
```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 2

Waktu yang dibutuhkan adalah 8726.00 ms
```

3) Bubble



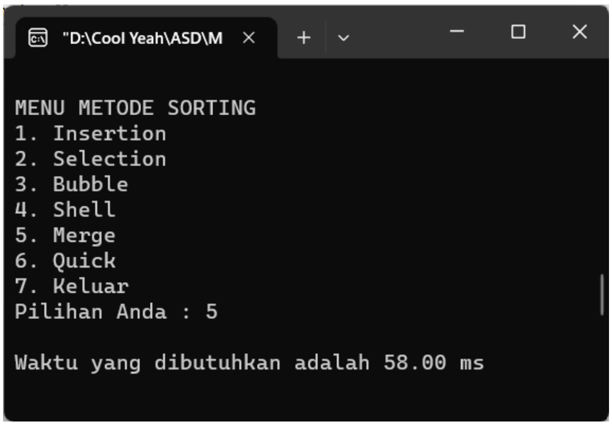
```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 3

Waktu yang dibutuhkan adalah 24724.00 ms
```

4) Shell



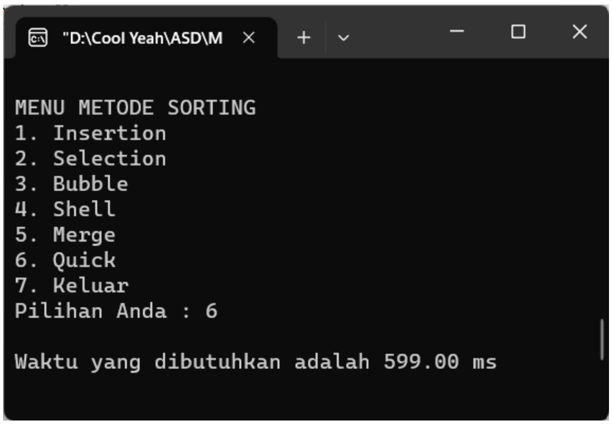
```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 4

Waktu yang dibutuhkan adalah 79.00 ms
```

5) Merge



```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 5

Waktu yang dibutuhkan adalah 58.00 ms
```

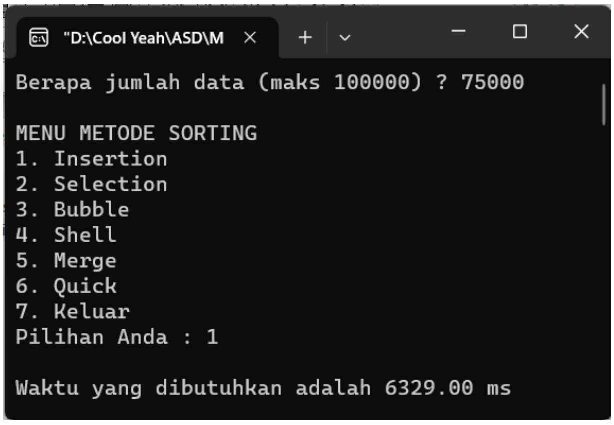
6) Quick



```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 6

Waktu yang dibutuhkan adalah 599.00 ms
```

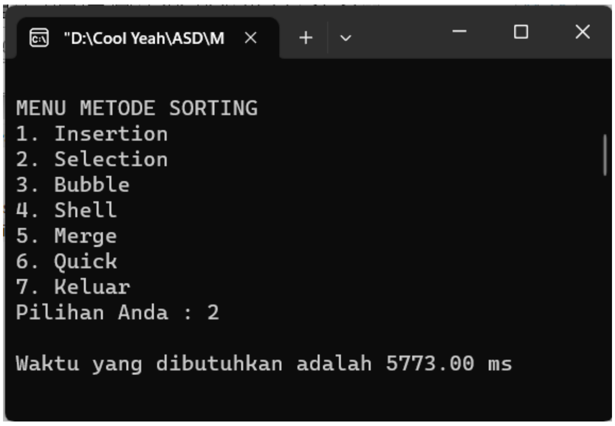
c) 75000 data

1) Insertion



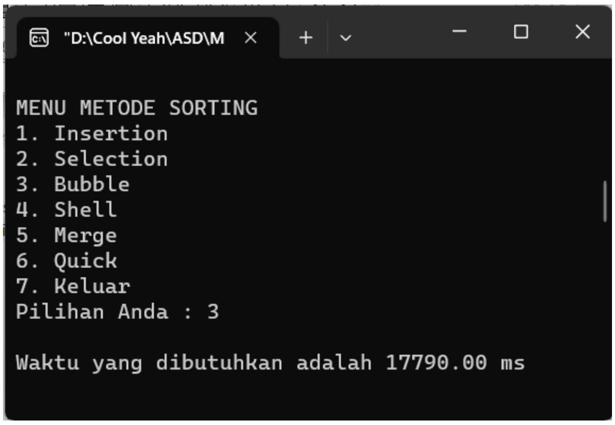
```
Berapa jumlah data (maks 100000) ? 75000

MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 1

Waktu yang dibutuhkan adalah 6329.00 ms
```

2) Selection



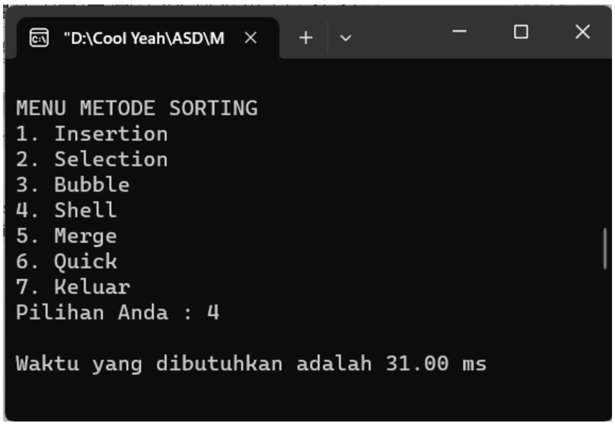
```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 2

Waktu yang dibutuhkan adalah 5773.00 ms
```

3) Bubble



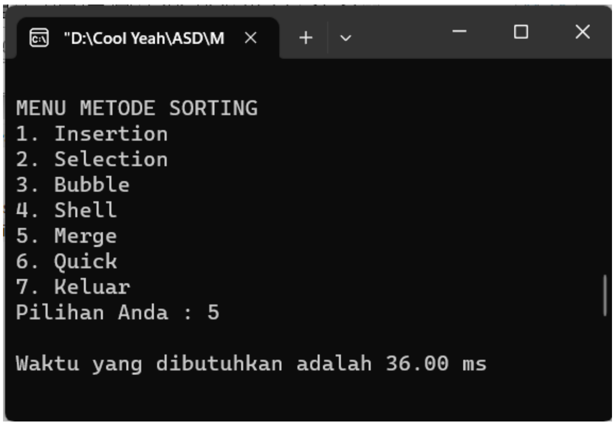
```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 3

Waktu yang dibutuhkan adalah 17790.00 ms
```

4) Shell



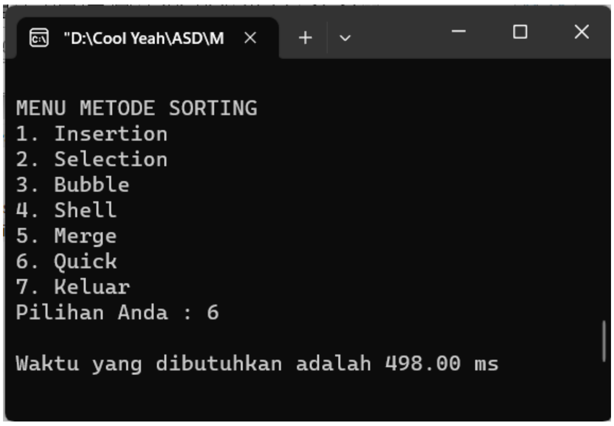
```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 4

Waktu yang dibutuhkan adalah 31.00 ms
```

5) Merge



```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 5

Waktu yang dibutuhkan adalah 36.00 ms
```

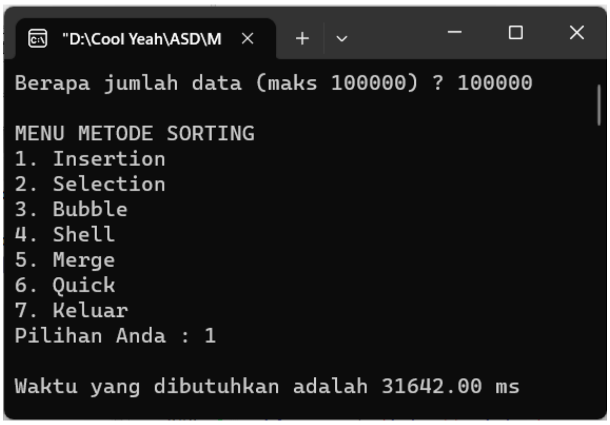
6) Quick



```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 6

Waktu yang dibutuhkan adalah 498.00 ms
```

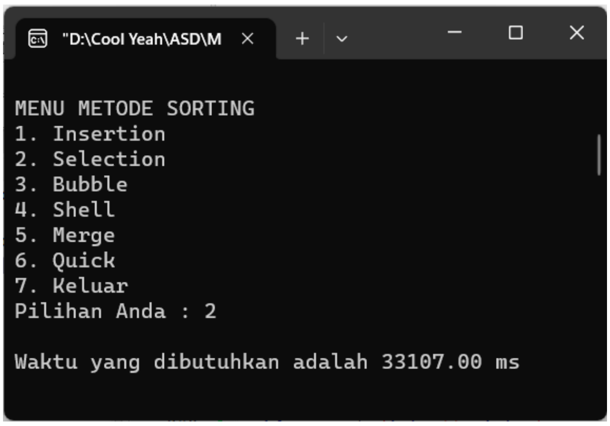
d) 100000 data

1) Insertion



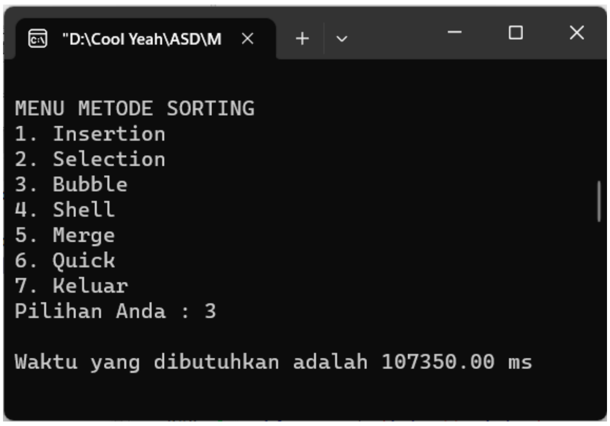
```
Berapa jumlah data (maks 100000) ? 100000

MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 1

Waktu yang dibutuhkan adalah 31642.00 ms
```

2) Selection



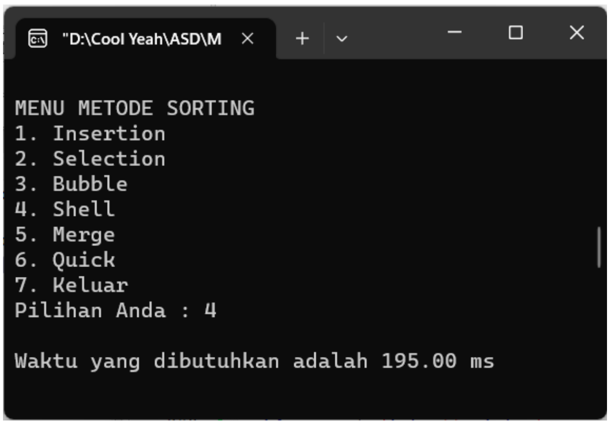
```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 2

Waktu yang dibutuhkan adalah 33107.00 ms
```

3) Bubble



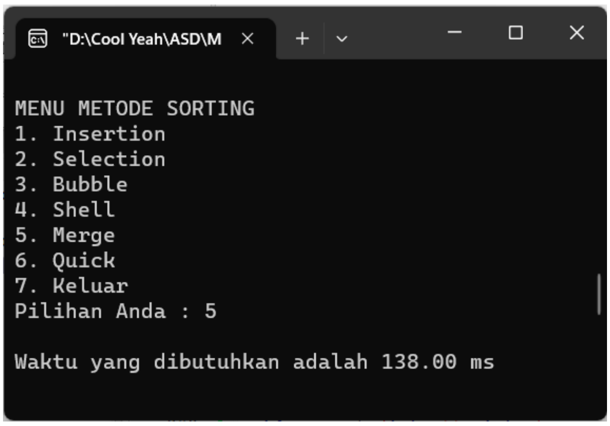
```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 3

Waktu yang dibutuhkan adalah 107350.00 ms
```

4) Shell



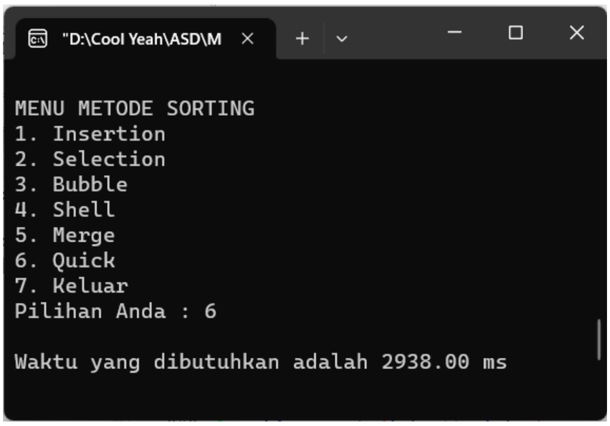
```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 4

Waktu yang dibutuhkan adalah 195.00 ms
```

5) Merge



```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 5

Waktu yang dibutuhkan adalah 138.00 ms
```

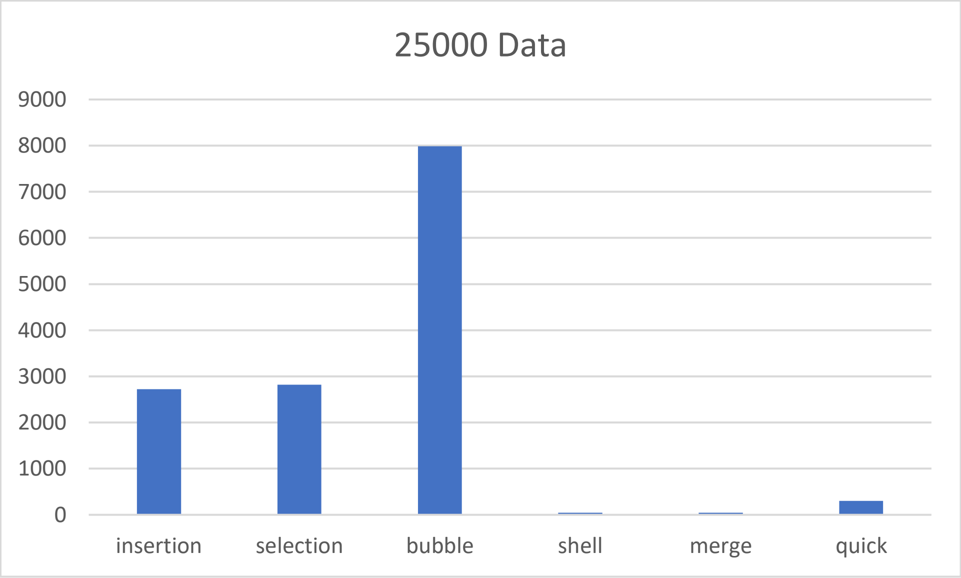
6) Quick



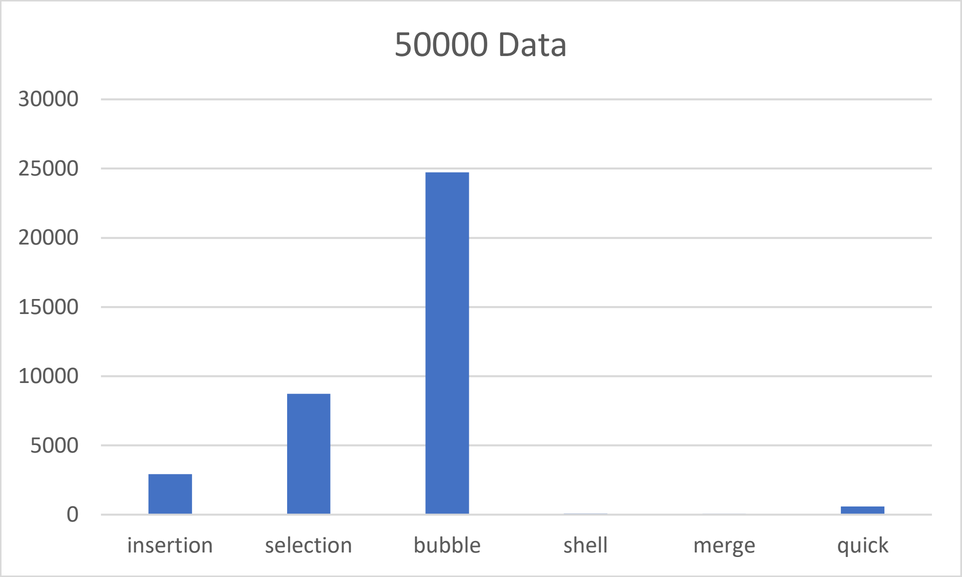
```
MENU METODE SORTING
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Keluar
Pilihan Anda : 6

Waktu yang dibutuhkan adalah 2938.00 ms
```

3. Perbandingan performa masing-masing metode

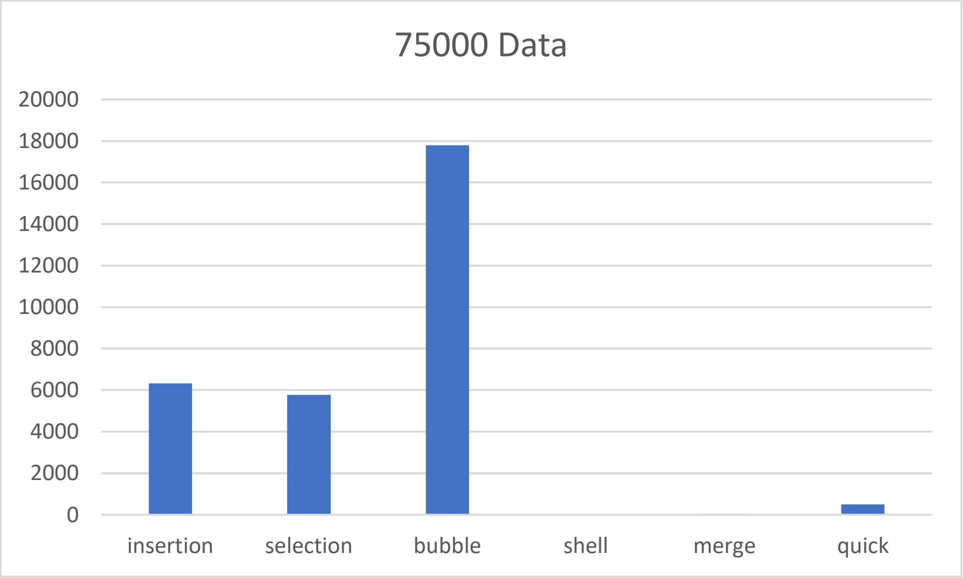
I. 25000 Data



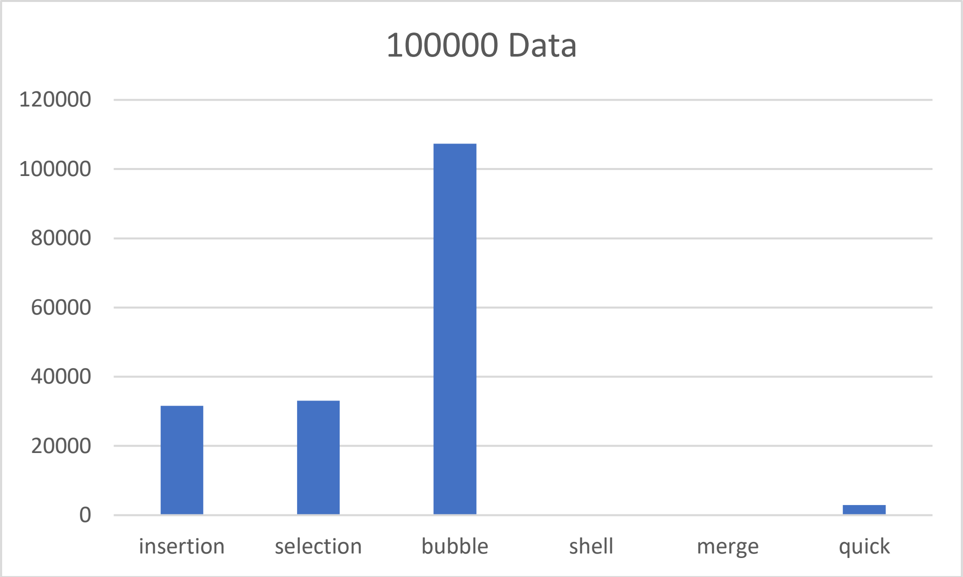
II. 50000 Data



III. 75000 Data



IV. 100000 Data



4. Analisa dan Kesimpulan
   Dari hasil percobaan yang telah dilakukan dan dapat dilihat dari data di atas, metode sorting yang tercepat adalah shell dan yang paling lambat adalah Bubble sort.