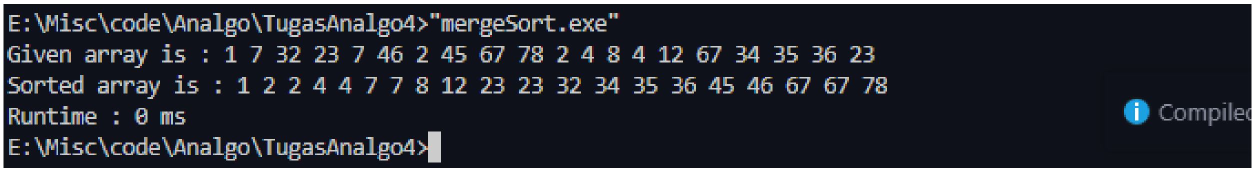
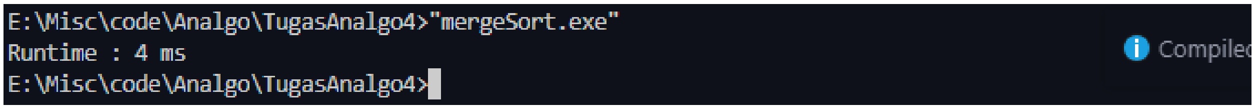
Case 1



```
E:\Misc\code\Analgo\TugasAnalgo4>"mergeSort.exe"
Given array is : 1 7 32 23 7 46 2 45 67 78 2 4 8 4 12 67 34 35 36 23
Sorted array is : 1 2 2 4 4 7 7 8 12 23 23 32 34 35 36 45 46 67 67 78
Runtime : 0 ms
E:\Misc\code\Analgo\TugasAnalgo4>
```

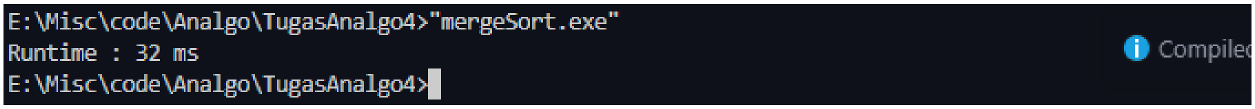
Dengan n = 20, jumlah data terlalu sedikit sehingga running timenya terlalu kecil.

n = 10000



```
E:\Misc\code\Analgo\TugasAnalgo4>"mergeSort.exe"
Runtime : 4 ms
E:\Misc\code\Analgo\TugasAnalgo4>
```

n = 100000



```
E:\Misc\code\Analgo\TugasAnalgo4>"mergeSort.exe"
Runtime : 32 ms
E:\Misc\code\Analgo\TugasAnalgo4>
```

Kasus 2:

```
for i ← n downto 2 do {pass sebanyak n-1 kali}
        imaks ← 1
        for j ← 2 to i do
           if $x_j > x_{imaks}$ then
              imaks ← j
           endif
        endfor
        {pertukarkan $x_{imaks}$ dengan $x_i$}
        temp ← $x_i$
        $x_i$ ← $x_{imaks}$
        $x_{imaks}$ ← temp
endfor
```

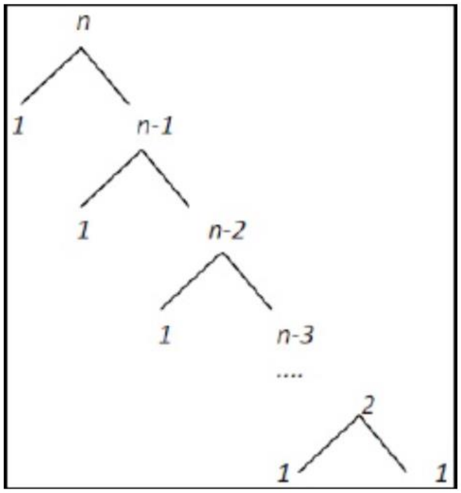
Subproblem = 1

Masalah setiap subproblem = n-1

Waktu proses pembagian = n

Waktu proses penggabungan = n

$$T(n) = \begin{cases} \Theta(1) \\ T(n-1) + \Theta(n) \end{cases}$$



$$= c((n{}^\wedge 2 - 3n + 2)/2) \ + \ cn$$
$$= c((n{}^\wedge 2)/2) - (3n/2) + 1 \ + \ cn$$
$$= O(n{}^\wedge 2)$$

$$\begin{aligned} T(n) &= cn \ + \ cn - c \ + cn - 2c \ + \ \ldots.. + 2c \ + cn \\ &= c\left(\frac{(n-1)(n-2)}{2}\right) + \ cn \\ &= c\left(\frac{n^2 - 3n + 2}{2}\right) + \ cn \\ &= c\left(\frac{n^2}{2}\right) - \left(\frac{3n}{2}\right) + 1 \ + \ cn \\ &= O(n{}^\wedge 2) \end{aligned}$$

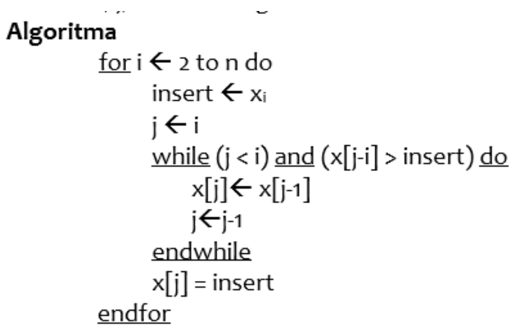
$$\begin{aligned} T(n) &= cn \ + \ cn - c \ + cn - 2c \ + \ \ldots.. + 2c \ + cn \\ &= c\left(\frac{(n-1)(n-2)}{2}\right) + \ cn \\ &= c\left(\frac{n^2 - 3n + 2}{2}\right) + \ cn \\ &= c\left(\frac{n^2}{2}\right) - \left(\frac{3n}{2}\right) + 1 \ + \ cn \\ &= \Omega \ (n{}^\wedge 2) \end{aligned}$$

$$\begin{aligned} T(n) &= cn^2 \\ &= \Theta(n^2) \end{aligned}$$

Source Code:

```c
// C program for implementation of selection sort
#include <stdio.h>

void swap(int *xp, int *yp)
{
      int temp = *xp;
      *xp = *yp;
      *yp = temp;
}

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

      // One by one move boundary of unsorted subarray
      for (i = 0; i < n-1; i++)
      {
            // Find the minimum element in unsorted array
            min_idx = i;
            for (j = i+1; j < n; j++)
            if (arr[j] < arr[min_idx])
                  min_idx = j;

            // Swap the found minimum element with the first element
            swap(&arr[min_idx], &arr[i]);
      }
}

/* Function to print an array */
void printArray(int arr[], int size)
{
      int i;
      for (i=0; i < size; i++)
            printf("%d ", arr[i]);
      printf("\n");
}

// Driver program to test above functions
int main()
{
      int arr[] = {64, 25, 12, 22, 11};
      int n = sizeof(arr)/sizeof(arr[0]);
      selectionSort(arr, n);
      printf("Sorted array: \n");
      printArray(arr, n);
      return 0;
}
```

Kasus 3:



Algoritma

```
    for i ← 2 to n do
        insert ← x_i
        j ← i
        while (j < i) and (x[j-i] > insert) do
            x[j]← x[j-1]
            j←j-1
        endwhile
        x[j] = insert
    endfor
```

Subproblem = 1

Masalah setiap subproblem = n-1

Waktu proses penggabungan = n

Waktu proses pembagian = n

$$T(n) = \begin{cases} \Theta(1) \\ T(n-1) + \Theta(n) \end{cases}$$

$$\begin{aligned} T(n) &= cn + cn - c + cn - 2c + \ldots.. + 2c + cn \leq 2cn^2 + cn^2 \\ &= c\left(\frac{(n-1)(n-2)}{2}\right) + cn \leq 2cn^2 + cn^2 \\ &= c\left(\frac{n^2 - 3n + 2}{2}\right) + cn \leq 2cn^2 + cn^2 \\ &= c\left(\frac{n^2}{2}\right) - c\left(\frac{3n}{2}\right) + c + cn \leq 2cn^2 + cn^2 \\ &= O(n\hat{}2) \end{aligned}$$

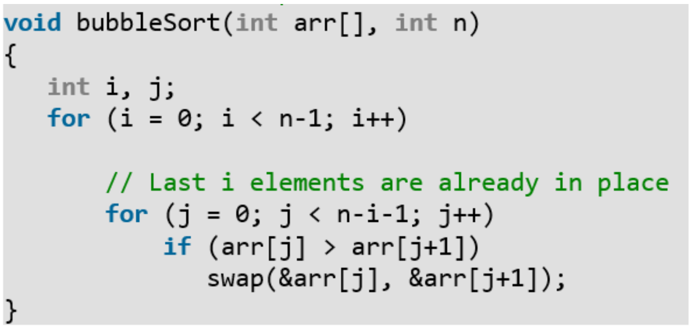
$$\begin{aligned} T(n) &= cn \leq cn \\ &= \Omega\,(n) \end{aligned}$$

$$\begin{aligned} T(n) &= \frac{cn + cn^2}{n} \\ &= \Theta(n) \end{aligned}$$

Source Code:

```
// C program for insertion sort
#include <math.h>
#include <stdio.h>

/* Function to sort an array using insertion sort*/
void insertionSort(int arr[], int n)
{
      int i, key, j;
      for (i = 1; i < n; i++) {
            key = arr[i];
            j = i - 1;

            /* Move elements of arr[0..i-1], that are
            greater than key, to one position ahead
            of their current position */
            while (j >= 0 && arr[j] > key) {
                  arr[j + 1] = arr[j];
                  j = j - 1;
            }
            arr[j + 1] = key;
      }
}

// A utility function to print an array of size n
void printArray(int arr[], int n)
{
      int i;
      for (i = 0; i < n; i++)
            printf("%d ", arr[i]);
      printf("\n");
}

/* Driver program to test insertion sort */
int main()
{
      int arr[] = { 12, 11, 13, 5, 6 };
      int n = sizeof(arr) / sizeof(arr[0]);

      insertionSort(arr, n);
      printArray(arr, n);

      return 0;
}
```

Kasus 4:



```
void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)

       // Last i elements are already in place
       for (j = 0; j < n-i-1; j++)
           if (arr[j] > arr[j+1])
              swap(&arr[j], &arr[j+1]);
}
```

Subproblem = 1

Masalah setiap subproblem = n-1

Waktu proses pembagian = n

Waktu proses penggabungan = n

$$T(n) = \begin{cases} \Theta(1) \\ T(n-1) + \Theta(n) \end{cases}$$

$$\begin{aligned} T(n) &= cn + cn - c + cn - 2c + \ldots..+ 2c + c \le 2cn^2 + cn^2 \\ &= c\left(\frac{(n-1)(n-2)}{2}\right) + c \le 2cn^2 + cn^2 \\ &= c\left(\frac{n^2 - 3n + 2}{2}\right) + c \le 2cn^2 + cn^2 \\ &= c\left(\frac{n^2}{2}\right) - c\left(\frac{3n}{2}\right) + 2c \le 2cn^2 + cn^2 \\ &= O(n^2) \end{aligned}$$

$$\begin{aligned} T(n) &= cn + cn - c + cn - 2c + \ldots..+ 2c + c \le 2cn^2 + cn^2 \\ &= c\left(\frac{(n-1)(n-2)}{2}\right) + c \le 2cn^2 + cn^2 \\ &= c\left(\frac{n^2 - 3n + 2}{2}\right) + c \le 2cn^2 + cn^2 \\ &= c\left(\frac{n^2}{2}\right) - c\left(\frac{3n}{2}\right) + 2c \le 2cn^2 + cn^2 \\ &= \Omega\ (n^2) \end{aligned}$$

$$\begin{aligned} T(n) &= cn^2 + cn^2 \\ &= \Theta(n^2) \end{aligned}$$

Source Code:

```
// Optimized implementation of Bubble sort
#include <stdio.h>

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

// An optimized version of Bubble Sort
void bubbleSort(int arr[], int n)
{
   int i, j;
   bool swapped;
   for (i = 0; i < n-1; i++)
   {
     swapped = false;
     for (j = 0; j < n-i-1; j++)
     {
        if (arr[j] > arr[j+1])
        {
           swap(&arr[j], &arr[j+1]);
           swapped = true;
        }
     }

     // IF no two elements were swapped by inner loop, then break
     if (swapped == false)
        break;
   }
}

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("n");
}

// Driver program to test above functions
int main()
{
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);
    bubbleSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```