LAPORAN PRAKTIKUM 4

ANALISIS ALGORITMA

Disusun oleh :

Putri Nabila

140810180007

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

2020

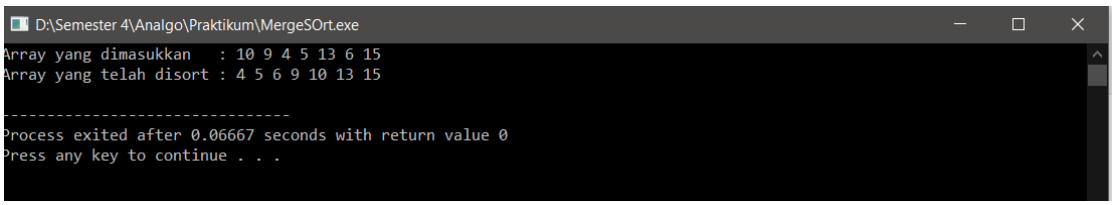
I. Studi Kasus 1: Merge Sort

Source code:

```
/*
Nama : Putri Nabila
NPM : 140810180007
Deskripsi : MergeSort
*/


#include <iostream>
using namespace std;
#define N 5;


void merge(int arr[],int l, int m, int r){

    int n1 = m - l + 1;
    int n2 = r-m;

    int L[n1], R[n2];

    // Copy array nya
    for(int i=0; i < n1; i++)
        L[i] = arr[l+i];
    for(int i=0; i < n2; i++)
        R[i] = arr[m+1+i];

    // gabungin aja  2 array secara berurutan
    int i = 0;
    int j = 0;
    int k = l;
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1)
    {
```

```
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[], int l, int r){
    if( l < r){
        int m = l+(r-l)/2;

        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);

        merge(arr, l, m, r);
    }
}

void cetakArray(int arr[], int n){
    for(int i=0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}

int main(){
    int arr[] = { 10, 9, 4, 5, 13, 6, 15};
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << "Array yang dimasukkan\t: "; cetakArray(arr,n);

    mergeSort(arr, 0, n-1);
    cout << "Array yang telah disort\t: ";cetakArray(arr,n);

    return 0;
}
```

Hasil Program:



Kompleksitas Algoritma:

Kompleksitas waktu algoritma merge sort adalah O(n lg n). Cari tahu kecepatan komputer Anda dalam memproses program. Hitung berapa running time yang dibutuhkan apabila input untuk merge sort-nya adalah 20?

$T(20 \log_{10} 20) = 26$

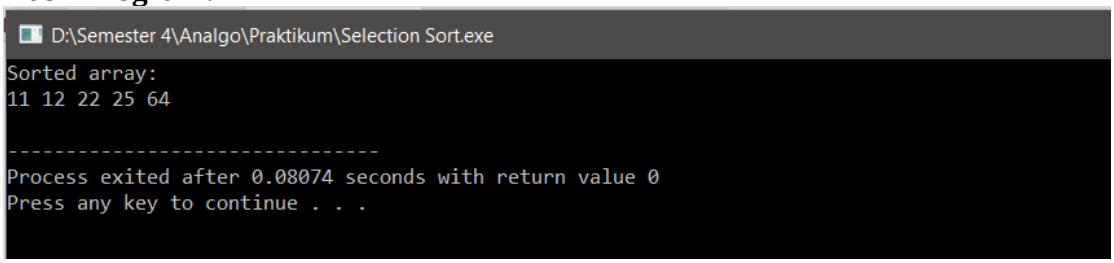
II. Studi Kasus 2: Selection Sort

Source code:

```
/*
Nama : Putri Nabila
NPM : 140810180007
Deskripsi : Selection Sort
*/


#include <stdio.h>

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

void selectionSort(int arr[], int n)
{
   int i, j, min_idx;
   for (i = 0; i < n - 1; i++)
   {
      min_idx = i;
      for (j = i + 1; j < n; j++)
         if (arr[j] < arr[min_idx])
            min_idx = j;
      swap(&arr[min_idx], &arr[i]);
   }
}

void printArray(int arr[], int size)
{
   int i;
   for (i = 0; i < size; i++)
      printf("%d ", arr[i]);
```

```
    printf("\n");
}

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

Hasil Program:



Kompleksitas Algoritma:

Menentukan T(n) :

$$T(n) = (n-1) + (n-2) + \cdots + 1 = \sum_{k=1}^{n-1} n - k = \frac{n(n-1)}{2}$$

Oleh karena itu :

$$T(n) = O(n^2)$$

$$T(n) = \Omega(n^2)$$

$$Karena\ O(n^2) = \Omega(n^2), maka\ \theta(n^2)$$

III. Studi Kasus 3: Insertion Sort

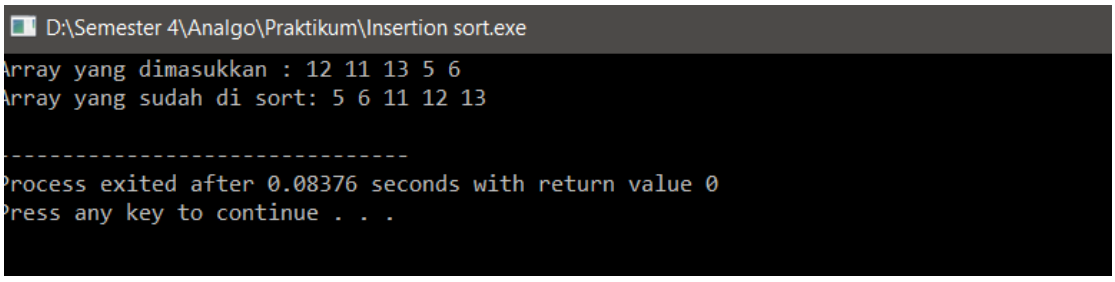
Source code:

```
/*

Nama : Putri Nabila

NPM : 140810180007

Deskripsi : Insertion Sort

*/
```

```
#include <iostream>
#include <math.h>
#include <stdio.h>
void insertionSort(int arr[], int n)
{
   int i, key, j;
   for (i = 1; i < n; i++) {
      key = arr[i];
      j = i - 1;

      while (j >= 0 && arr[j] > key) {
         arr[j + 1] = arr[j];
         j = j - 1;
      }
      arr[j + 1] = key;
   }
}

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

int main()
{
   int arr[] = { 12, 11, 13, 5, 6 };
```

```
   int n = sizeof(arr) / sizeof(arr[0]);

   printf("Array yang dimasukkan : ");

   printArray(arr, n);

   insertionSort(arr, n);

   printf("Array yang sudah di sort: ");

   printArray(arr, n);


   return 0; }
```

Hasil Program:



Kompleksitas Algoritma:

Menentukan T(n) :

$$T(n) = 2(1) + 2(2) + 2(3) + 2(4) \ldots + 2(n-1) = 2\big(1 + 2 + 3 + 4 + \cdots (n-1)\big)$$

$$= \frac{2\big((n-1)n\big)}{2} = (n-1)n = n^2 - n = O(n^2)$$

T(n) = $\Omega(5)$

IV. Studi Kasus 4: Bubble Sort

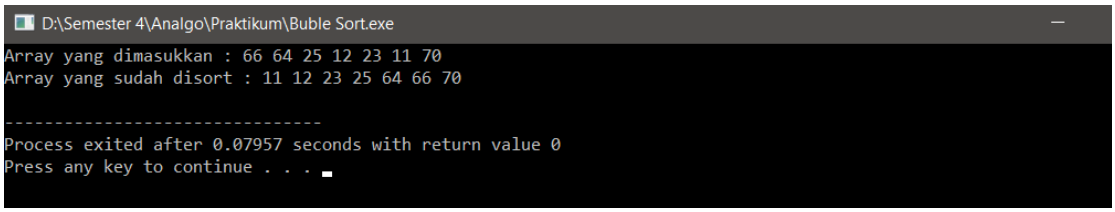
Source code:

```
/*

Nama : Putri Nabila

NPM : 140810180007

Deskripsi : Buble Sort

*/

#include <iostream>
```

```
#include <stdio.h>


void swap(int *xp, int *yp)
{
   int temp = *xp;
   *xp = *yp;
   *yp = temp;
}
void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n - 1; i++)
      for (j = 0; j < n - i - 1; j++)
         if (arr[j] > arr[j + 1])
            swap(&arr[j], &arr[j + 1]);
}


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


int main()
{
   int arr[] = {66, 64, 25, 12, 23, 11, 70};
   int n = sizeof(arr) / sizeof(arr[0]);
```

```
    printf("Array yang dimasukkan : ");
    printArray(arr, n);
    bubbleSort(arr, n);
    printf("Array yang sudah disort : ");
    printArray(arr, n);
    return 0;
}
```

Hasil Program:



Kompleksitas Algoritma:

Menentukan T(n) :

$T(n) = n^2 + n = O(n^2)$

$T(n) = n - 1 = \Omega(n)$