A logo of a university

Description automatically generated

NAMA : RAKASYAEL HIZKIA KOLONDAM

NIM : 2023105490

MATA KULIAH : TI0091 – PRAKTIKUM DATA TERSTRUKTUR

PRAKTIKUM KE - : 12

|  |  |  |
| --- | --- | --- |
| **No.** | **Kode** | **Penjelasan** |
| 1 | public static void ascendingBubbleSort(int[] array)  {  for (int i = array.length - 1; i > 0; i--)  {  for (int j = 0; j < i; j++)  {  if (array[j] > array[j + 1])  {  int temp = array[j];  array[j] = array[j + 1];  array[j + 1] = temp;  }  }  }  } | Metode ini melakukan sorting dengan algoritma Bubble Sorting dengan cara membandingkan satu persatu setiap node dalam array dengan node setelahnya, jika lebih besar maka akan ditukar posisinya, dan akan mulai dari awal lagi sampai array tersusun. |
| 2 | public static void ascendingSelectionSort(int[] array)  {  for (int i = 0; i < array.length; i++)  {  int minIndex = i;  for (int j = i + 1; j < array.length; j++)  {  if (array[j] < array[minIndex])  {  minIndex = j;  }  }  if (i != minIndex)  {  int temp = array[i];  array[i] = array[minIndex];  array[minIndex] = temp;  }  }  } | Metode ini melakukan sorting dengan algoritma Selection Sort. Metode ini akan menandai node pertama sebagai node terkecil. Lalu akan di cek dengan node selanjutnya, jika lebih kecil maka node yang baru akan ditandai sebagai node terkecil dan begitu seterusnya. Lalu akan dilakukan penukaran ketika I bukan minimum index, maka akan ditukar dengan node yang merupakan node terkecil. Lalu iterasi dilakukan lagi dari awal. |
| 3 | public static void ascendingInsertionSort(int[] array)  {  for (int i = 1; i < array.length; i++)  {  int temp = array[i];  int j = i - 1;  while (j > -1 && temp < array[j])  {  array[j + 1] = array[j];  array[j] = temp;  j--;  }  }  } | Melakukan sorting dengan cara Insertion. Metode akan menandai temp sebagai node pertama dalam array dan j sebagai indeks sebelum i. Lalu setelah itu dalam while loop di cek jika j lebih dari -1 dan temp lebih dari node pada indeks j, maka akan ditukar. |
| 4 | public static void **descendingBubbleSort**(int[] array) {  *// Modifikasi bagian ini*  *for* (int i **=** array.length **-** 1; i **>** 0; i**--**) {  *for* (int j **=** 0; j **<** i; j**++**) {  *if* (array[j] **<** array[j **+** 1]) {                    int temp **=** array[j];                    array[j] **=** array[j **+** 1];                    array[j **+** 1] **=** temp;                }            }        }      } | Sama seperti ascending bubble sort, descending melakukan pengecekan jika node selanjutnya lebih besar dari pada node yang ditunjuk. |
| 5 | public static void **descendingSelectionSort**(int[] array) {  *// Modifikasi bagian ini*  *for* (int i **=** 0; i **<** array.length; i**++**) {            int maxIndex **=** i;  *for* (int j **=** i **+** 1; j **<** array.length; j**++**) {  *if* (array[j] **>** array[maxIndex]) {                    maxIndex **=** j;                }            }  *if* (i **!=** maxIndex) {                int temp **=** array[i];                array[i] **=** array[maxIndex];                array[maxIndex] **=** temp;            }        }      } | Sama seperti ascending tapi mengecek node ketika node yang ditunjuk lebih besar dari pada indeks yang maksimum (sekarang) maka akan ditunjuk sebagai max indeks yang baru lalu akan ditukarkan. |
| 6 | public static void **descendingInsertionSort**(int[] array) {  *// Modifikasi bagian ini*  *for* (int i **=** 1; i **<** array.length; i**++**) {            int temp **=** array[i];            int j **=** i **-** 1;  *while* (j **>** **-**1 **&&** temp **>** array[j]) {                array[j **+** 1] **=** array[j];                array[j] **=** temp;                j**--**;            }        }      } | Sama seperti algoritma yang ascending, tetapi melakukan pengecekan dalam while loop ketika temp lebih besar dari pada node pada indeks j atau I - 1 |