**LAPORAN PRAKTIKUM IV**

**ALGORITMA STRUKTUR DATA**

**“Quick Sort dan Branching”**



Nama : Muhammad Azka Raki

NIM : 2311016110005

**KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN**

**UNIVERSITAS LAMBUNG MANGKURAT**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**PROGRAM STUDI S1 ILMU KOMPUTER**

**BANJARBARU**

**2024**

1. **Tinjauan Pustaka**

*Quick Sort* merupakan algoritma pengurutan data yang bergantung pada elemen pivot dengan dengan cara kerja metodenya mereduksi tahap demi tahap sehingga menjadi 2 bagian yang lebih kecil. *Divide and Conquer* adalah sebutan dari cara kerja metode tersebut. Elemen angka atau n yang lebih besar akan dipartisi menjadi dua *sub*-elemen yang mana salah satunya berisi elemen yang lebih kecil dari pivot. Kemudian akan secara rekursif untuk mengurutkan elemen angka.

Algoritma *Quick Sort* merupakan algoritma yang sangat cepat jika dibandingkan dengan algoritma pengurutan data yang lain karena algoritma ini melakukan pengurutan data dengan membagi masalah tersebut sehingga menghasilkan pengurutan data yang lebih cepat.

*Tree* adalah struktur data hirarkis yang terdiri dari simpul-simpul. Setiap node memiliki node anak, dan salah satu node adalah node akar. Dalam sebuah pohon terdapat node yang bisa memiliki lebih dari satu node anak. *Root* adalah node yang tidak memiliki induk. *Leaf* adalah adalah node tanpa anak.

*Binary Tree* adalah tipe *tree* yang terdiri dari simpul-simpul, yang masing-masing memiliki paling banyak dua node anak.

Dalam mengunjungi semua elemen dalam sebuah *binary tree*, terdapat 3 cara untuk melakukannya. Proses membaca atau mengunjungi *binary tree* disebut sebagai traversal. Adapun 3 cara traversal pada *binary tree* antara lain:

1. Inorder Traversal

Inorder traversal bekerja dengan cara mengunjungi *subtree* kiri, selanjutnya ke *root*, kemudian ke *subtree* kanan.

1. Preorder Traversal

Preorder traversal mengunjungi *root* sebelum mengunjungi *subtrees*

1. Postorder Traversal

Postorder traversal mengunjungi *root* setelah mengunjung *subtrees*

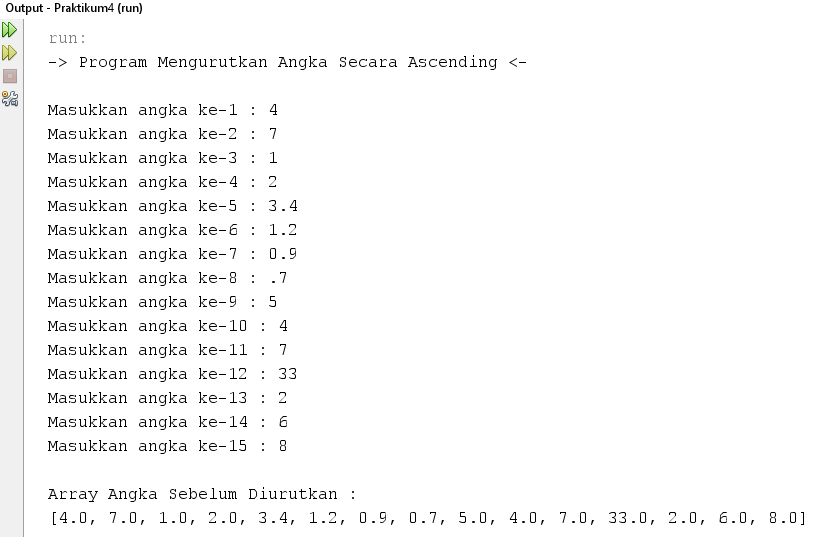
1. **Source Code**
2. Main.java

|  |
| --- |
| package praktikum4;  public class Main {  public static void main(String[] args) {  ArrayAngka arrayAngka = new ArrayAngka();  System.out.println("-> Program Mengurutkan Angka Secara Ascending <-");  System.out.println("");  arrayAngka.isiArrayAngka();  System.out.println("");  arrayAngka.printArray("Array Angka Sebelum Diurutkan");  System.out.println("");  arrayAngka.quickSort(0, arrayAngka.length - 1);  System.out.println("");  arrayAngka.printArray("Array Angka Setelah Diurutkan");  }  } |

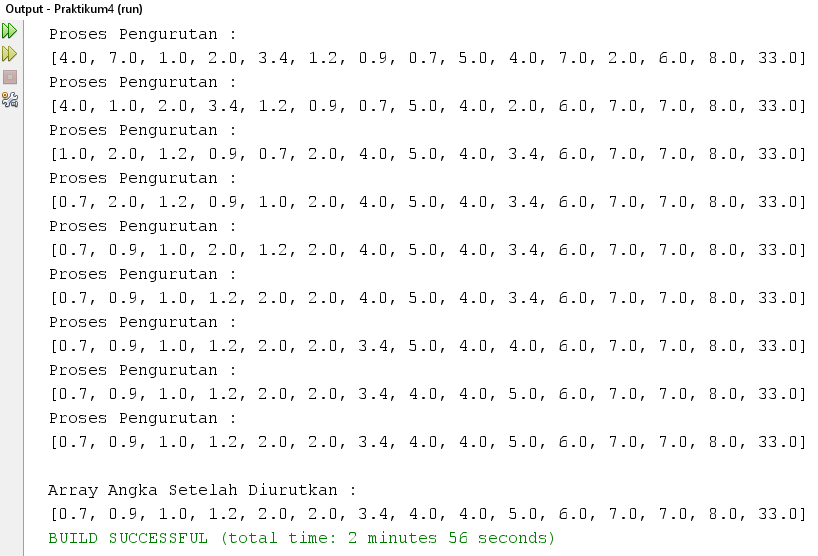
1. ArrayAngka.java

|  |
| --- |
| package praktikum4;  import java.util.Arrays;  import java.util.Scanner;  public class ArrayAngka {  public int length = 15;  private double[] arrayAngka = new double[length];  private void setAngkaDiIndex(int index, double angka) {  arrayAngka[index] = angka;  }  void isiArrayAngka() {  Scanner scan = new Scanner(System.in);  for (int i = 0; i < arrayAngka.length; i++) {  System.out.print("Masukkan angka ke-" + (i + 1) + " : ");  double angka = scan.nextDouble();  setAngkaDiIndex(i, angka);  }  }  void printArray(String message) {  System.out.println(message + " :\n" + Arrays.toString(arrayAngka));  }  void quickSort(int low, int high) {  if (low < high) {  int pivotIndex = partition(low, high);  printArray("Proses Pengurutan");  quickSort(low, pivotIndex - 1);  quickSort(pivotIndex + 1, high);  }  }  private int partition(int low, int high) {  double pivot = arrayAngka[high];  int i = low - 1;  for (int j = low; j < high; j++) {  if (arrayAngka[j] <= pivot) {  i++;  double temp = arrayAngka[i];  arrayAngka[i] = arrayAngka[j];  arrayAngka[j] = temp;  }  }  double temp = arrayAngka[i + 1];  arrayAngka[i + 1] = arrayAngka[high];  arrayAngka[high] = temp;  return i + 1;  }  } |

1. **Output**



Gambar 3.1 hasil output (input angka secara berurutan = 4, 7, 1, 2, 3.4, 1.2, 0.9, .7, 5, 4, 7, 33, 2, 6, 8)



Gambar 3.2 hasil output (lanjutan gambar 3.1)

**Sumber :**

Heryanto, Y., & Wira Harjanti, T. (2023). Analisis Perbandingan Ruang dan Waktu pada Algoritma Sorting Menggunakan Bahasa Pemrograman Python. Jurnal Penerapan Sistem Informasi (Komputer & Manajemen)

Horstmann, C. S. (2017). Big Java: Late Objects. John Wiley & Sons.

Poetra, D. R. (2022). Performa Algoritma Bubble Sort dan Quick Sort pada Framework Flutter dan Dart SDK(Studi Kasus Aplikasi E-Commerce). JATISI (Jurnal Teknik Informatika Dan Sistem Informasi). <https://doi.org/10.35957/jatisi.v9i2.1886>