**LAPORAN PRAKTIKUM STRUKTUR DATA**

Modul ke : 6

Judul Praktikum : SORTING II

Hari dan Tanggal Pelaksanaan : Selasa, 26 April 2022

Tempat Pelaksanaan : Lab Desain

Dosen Pengampu Praktikum : Khoirul Umam, S.Pd, M.Kom

Nama Mahasiswa Pelaksana : Andri Wijaksono

NIM Pelaksana : 362155401206

1. **Tugas Pendahuluan**

Tuliskan hasil pengerjaan Tugas Pendahuluan pada bagian ini.

1. **Kegiatan Praktikum**

Cantumkan apa saja yang dilakukan pada latihan-latihan praktikum, *source code* yang dipakai, *screen shot* hasil eksekusi kode, dan jawaban dari pertanyaan-pertanyaan yang muncul pada ti­ap kegiatan latihan.

Gambarkan flowchart untuk masing-masing algoritma shell sort, quick sort, dan merge sort untuk pengurutan data secara ascending!

**Latihan 1: Ascending bubble sort**

1. Buat file dengan nama **BubbleSort**.**java** kemudian tuliskan kode berikut

import java.lang.Comparable;

public class ShellSort {

public static <T extends Comparable<? super T>> void sort(T[] arr) {

int i, jarak;

boolean didSwap;

T temp;

jarak = arr.length;

while (jarak > 1) {

jarak = jarak / 2;

didSwap = true;

while (didSwap) {

didSwap = false;

i = 0;

while (i < arr.length - jarak) {

if (arr[i].compareTo(arr[i + jarak]) > 0) {

temp = arr[i];

arr[i] = arr[i + jarak];

arr[i + jarak] = temp;

didSwap = true;

}

i++;

}

}

// log tahapan sorting

System.out.print("Jarak " + jarak + " : ");

tampil(arr);

}

}

public static <T> void tampil(T data[]) {

for (T objek : data) {

System.out.print(objek + " ");

}

System.out.println();

}

public static void main(String[] args) {

    Integer data[] = { 3, 10, 4, 6, 8, 9, 7, 2, 1, 5 };

    System.out.print("Data awal: ");

    tampil(data);

    long awal = System.currentTimeMillis();

    sort(data);

    long selisihWaktu = System.currentTimeMillis() - awal;

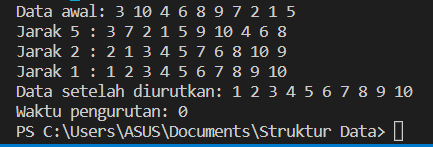
    System.out.print("Data setelah diurutkan: ");

    tampil(data);

    System.out.println("Waktu pengurutan: " + selisihWaktu);

    }

    }

****

3. Jarak berapa saja yang digunakan dalam kasus di atas?

Jawab: Jarak 5, jarak 2, jarak 1

4. Berapa total perbandingan (compare) yang terjadi pada kasus di atas?

Jawab: Perbandingan 3

**Latihan 2: Ascending quick sort**

1. Buat file bernama QuickSort.java kemudian tuliskan kode berikut:

import java.lang.Comparable;

public class QuickSort {

public static <T extends Comparable<? super T>> int partition(T[] arr, int

p, int r) {

int i, j;

T pivot, temp;

pivot = arr[p]; // pivot pada index 0

System.out.println("Index " + p + "-" + r + " | Pivot = " + pivot);

i = p;

j = r;

while (i <= j) {

while (pivot.compareTo(arr[j]) < 0)

j--;

while (pivot.compareTo(arr[i]) > 0)

i++;

if (i < j) {

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

System.out.print("> Swap indeks " + i + " & " + j + ": ");

tampil(arr);

j--;

i++;

} else {

return j;

}

}

return j;

}

public static <T extends Comparable<? super T>> void sort(T[] arr, int p,

int r) {

int q;

if (p < r) {

q = partition(arr, p, r);

System.out.println("=> Partisi: " + q);

sort(arr, p, q);

sort(arr, q + 1, r);

}

}

public static <T> void tampil(T data[]) {

for (T objek : data) {

System.out.print(objek + " ");

}

System.out.println();

}

public static void main(String[] args) {

Integer data[] = { 3, 10, 4, 6, 8, 9, 7, 2, 1, 5 };

System.out.print("Data awal: ");

tampil(data);

long awal = System.currentTimeMillis();

sort(data, 0, data.length - 1);

long selisihWaktu = System.currentTimeMillis() - awal;

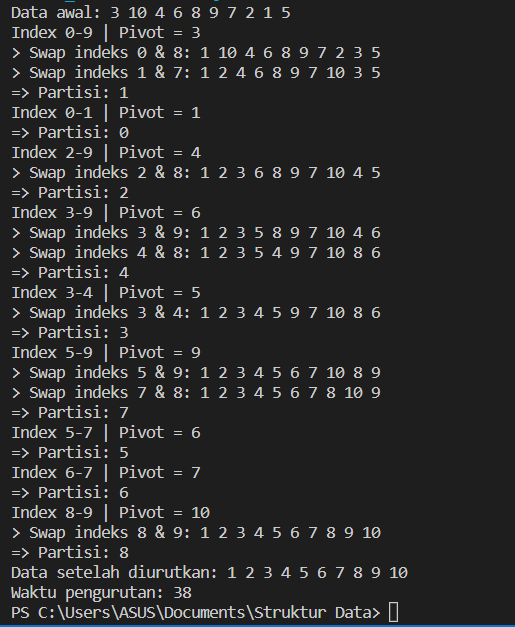
System.out.print("Data setelah diurutkan: ");

tampil(data);

System.out.println("Waktu pengurutan: " + selisihWaktu);

}

}



3. Berapa kali proses pencarian partisi terjadi?

Jawab: 8 kali Partisipasi

4. Berapa kali proses swapping terjadi?

Jawab: 9 kali partisipasi

**Latihan 3: Ascending merge sort**

1. Buat file baru bernama MergeSort.java kemudian tuliskan kode berikut:

import java.lang.Comparable;

public class MergeSort {

public static <T extends Comparable<? super T>> void merge(T[] arr, int

left, int median, int right) {

Object[] temp = new Object[arr.length];

int kiri1, kanan1, kiri2, kanan2, i, j;

kiri1 = left;

kanan1 = median;

kiri2 = median + 1;

kanan2 = right;

i = left;

while ((kiri1 <= kanan1) && (kiri2 <= kanan2)) {

if (arr[kiri1].compareTo(arr[kiri2]) <= 0) {

temp[i] = arr[kiri1];

kiri1++;

} else {

temp[i] = arr[kiri2];

kiri2++;

}

i++;

}

while (kiri1 <= kanan1) {

temp[i] = arr[kiri1];

kiri1++;

i++;

}

while (kiri2 <= kanan2) {

temp[i] = arr[kiri2];

kiri2++;

i++;

}

j = left;

while (j <= right) {

arr[j] = (T) temp[j];

j++;

}

}

public static <T extends Comparable<? super T>> void sort(T[] arr, int l,

int r) {

int med;

if (l < r) {

med = (l + r) / 2;

sort(arr, l, med);

sort(arr, med + 1, r);

merge(arr, l, med, r);

// log tahapan sorting

System.out.print("l = " + l + ", r = " + r + ", med = " + med + " : ");

tampil(arr);

}

}

public static <T> void tampil(T data[]) {

for (T objek : data) {

System.out.print(objek + " ");

}

System.out.println();

}

public static void main(String[] args) {

Integer data[] = { 3, 10, 4, 6, 8, 9, 7, 2, 1, 5 };

System.out.print("Data awal: ");

tampil(data);

long awal = System.currentTimeMillis();

sort(data, 0, data.length - 1);

long selisihWaktu = System.currentTimeMillis() - awal;

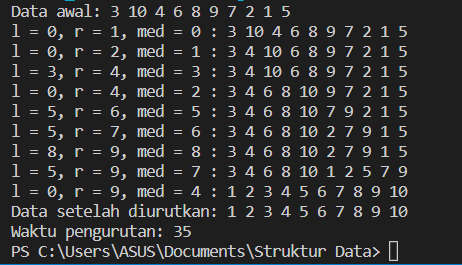
System.out.print("Data setelah diurutkan: ");

tampil(data);

System.out.println("Waktu pengurutan: " + selisihWaktu);

}

}

****

3. Berapa kali proses merging terjadi pada kasus di atas?

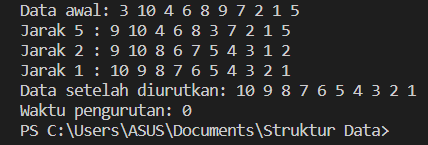
Jawab: 9 kali melakukan proses marging

1. **Tugas Praktikum**

Tuliskan dan jabarkan hasil pengerjaan Tugas Praktikum yang tertera di dalam modul lengkap dengan *source code* yang digunakan.

1. Tambahkan method descendingSort() ke dalam class ShellSort, QuickSort, maupun MergeSort agar pengurutan menggunakan metode-metode tersebut dapat menghasilkan urutan secara menurun (descending).

* **Shellsort**
* import java.lang.Comparable;
* public class ShellSort {
* public static <T extends Comparable<? super T>> void sort(T[] arr) {
* int i, jarak;
* boolean didSwap;
* T temp;
* jarak = arr.length;
* while (jarak > 1) {
* jarak = jarak / 2;
* didSwap = true;
* while (didSwap) {
* didSwap = false;
* i = 0;
* while (i < arr.length - jarak) {
* if (arr[i].compareTo(arr[i + jarak]) < 0) {
* temp = arr[i];
* arr[i] = arr[i + jarak];
* arr[i + jarak] = temp;
* didSwap = true;
* }
* i++;
* }
* }
* System.out.print("Jarak " + jarak + " : ");
* tampil(arr);
* }
* }
* public static <T> void tampil(T data[]) {
* for (T objek : data) {
* System.out.print(objek + " ");
* }
* System.out.println();
* }
* public static void main(String[] args) {
* Integer data[] = { 3, 10, 4, 6, 8, 9, 7, 2, 1, 5 };
* System.out.print("Data awal: ");
* tampil(data);
* long awal = System.currentTimeMillis();
* sort(data);
* long selisihWaktu = System.currentTimeMillis() - awal;
* System.out.print("Data setelah diurutkan: ");
* tampil(data);
* System.out.println("Waktu pengurutan: " + selisihWaktu);
* }
* }



* QuickSort
* import java.lang.Comparable;
* public class QuickSort {
* public static <T extends Comparable<? super T>> int partition(T[] arr, int p, int r) {
* int i, j;
* T pivot, temp;
* pivot = arr[p];
* System.out.println("Index " + p + "-" + r + " | Pivot = " + pivot);
* i = p;
* j = r;
* while (i <= j) {
* while (pivot.compareTo(arr[j]) > 0)
* j--;
* while (pivot.compareTo(arr[i]) < 0)
* i++;
* if (i < j) {
* temp = arr[i];
* arr[i] = arr[j];
* arr[j] = temp;
* System.out.print("> Swap indeks " + i + " & " + j + ": ");
* tampil(arr);
* j--;
* i++;
* } else {
* return j;
* }
* }
* return j;
* }
* public static <T extends Comparable<? super T>> void sort(T[] arr, int p, int r) {
* int q;
* if (p < r) {
* q = partition(arr, p, r);
* System.out.println("=> Partisi: " + q);
* sort(arr, p, q);
* sort(arr, q + 1, r);
* }
* }
* public static <T> void tampil(T data[]) {
* for (T objek : data) {
* System.out.print(objek + " ");
* }
* System.out.println();
* }
* public static void main(String[] args) {
* Integer data[] = { 3, 10, 4, 6, 8, 9, 7, 2, 1, 5 };
* System.out.print("Data awal: ");
* tampil(data);
* long awal = System.currentTimeMillis();
* sort(data, 0, data.length - 1);
* long selisihWaktu = System.currentTimeMillis() - awal;
* System.out.print("Data setelah diurutkan: ");
* tampil(data);
* System.out.println("Waktu pengurutan: " + selisihWaktu);
* }
* }

