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

A. Tugas Pendahuluan

Tuliskan hasil pengerjaan Tugas Pendahuluan pada bagian ini.

B. 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);
    }
    }
```

```
Data awal: 3 10 4 6 8 9 7 2 1 5
Jarak 5 : 3 7 2 1 5 9 10 4 6 8
Jarak 2 : 2 1 3 4 5 7 6 8 10 9
Jarak 1 : 1 2 3 4 5 6 7 8 9 10
Data setelah diurutkan: 1 2 3 4 5 6 7 8 9 10
Waktu pengurutan: 0
PS C:\Users\ASUS\Documents\Struktur Data>
```

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);
}
}
```

```
Data awal: 3 10 4 6 8 9 7 2 1 5
Index 0-9 | Pivot = 3
> Swap indeks 0 & 8: 1 10 4 6 8 9 7 2 3 5
> Swap indeks 1 & 7: 1 2 4 6 8 9 7 10 3 5
=> Partisi: 1
Index 0-1 | Pivot = 1
=> Partisi: 0
Index 2-9 | Pivot = 4
> Swap indeks 2 & 8: 1 2 3 6 8 9 7 10 4 5
=> Partisi: 2
Index 3-9 | Pivot = 6
> Swap indeks 3 & 9: 1 2 3 5 8 9 7 10 4 6
> Swap indeks 4 & 8: 1 2 3 5 4 9 7 10 8 6
=> Partisi: 4
Index 3-4 | Pivot = 5
> Swap indeks 3 & 4: 1 2 3 4 5 9 7 10 8 6
=> Partisi: 3
Index 5-9 | Pivot = 9
> Swap indeks 5 & 9: 1 2 3 4 5 6 7 10 8 9
> Swap indeks 7 & 8: 1 2 3 4 5 6 7 8 10 9
=> Partisi: 7
Index 5-7 | Pivot = 6
=> Partisi: 5
Index 6-7 | Pivot = 7
=> Partisi: 6
Index 8-9 | Pivot = 10
> Swap indeks 8 & 9: 1 2 3 4 5 6 7 8 9 10
=> Partisi: 8
Data setelah diurutkan: 1 2 3 4 5 6 7 8 9 10
Waktu pengurutan: 38
PS C:\Users\ASUS\Documents\Struktur Data>
```

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);
}
}
```

```
Data awal: 3 10 4 6 8 9 7 2 1 5
l = 0, r = 1, med = 0 : 3 10 4 6 8 9 7 2 1 5
l = 0, r = 2, med = 1 : 3 4 10 6 8 9 7 2 1 5
l = 3, r = 4, med = 3 : 3 4 10 6 8 9 7 2 1 5
l = 0, r = 4, med = 2 : 3 4 6 8 10 9 7 2 1 5
l = 5, r = 6, med = 5 : 3 4 6 8 10 7 9 2 1 5
l = 5, r = 7, med = 6 : 3 4 6 8 10 2 7 9 1 5
l = 8, r = 9, med = 8 : 3 4 6 8 10 2 7 9 1 5
l = 5, r = 9, med = 7 : 3 4 6 8 10 1 2 5 7 9
l = 0, r = 9, med = 4 : 1 2 3 4 5 6 7 8 9 10
Data setelah diurutkan: 1 2 3 4 5 6 7 8 9 10
Waktu pengurutan: 35
PS C:\Users\ASUS\Documents\Struktur Data>
```

3. Berapa kali proses merging terjadi pada kasus di atas?
Jawab: 9 kali melakukan proses marging

C. 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);
        }
    }
```

```
Data awal: 3 10 4 6 8 9 7 2 1 5
Jarak 5 : 9 10 4 6 8 3 7 2 1 5
Jarak 2 : 9 10 8 6 7 5 4 3 1 2
Jarak 1 : 10 9 8 7 6 5 4 3 2 1
Data setelah diurutkan: 10 9 8 7 6 5 4 3 2 1
Waktu pengurutan: 0
PS C:\Users\ASUS\Documents\Struktur Data>
```

- 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);
    }
}
```

```
Data awal: 3 10 4 6 8 9 7 2 1 5
Index 0-9 | Pivot = 3
> Swap indeks 0 & 9: 5 10 4 6 8 9 7 2 1 3
=> Partisi: 6
Index 0-6 | Pivot = 5
> Swap indeks 0 & 6: 7 10 4 6 8 9 5 2 1 3
> Swap indeks 2 & 5: 7 10 9 6 8 4 5 2 1 3
=> Partisi: 4
Index 0-4 | Pivot = 7
> Swap indeks 0 & 4: 8 10 9 6 7 4 5 2 1 3
=> Partisi: 2
Index 0-2 | Pivot = 8
> Swap indeks 0 & 2: 9 10 8 6 7 4 5 2 1 3
=> Partisi: 1
Index 0-1 | Pivot = 9
> Swap indeks 0 & 1: 10 9 8 6 7 4 5 2 1 3
=> Partisi: 0
Index 3-4 | Pivot = 6
> Swap indeks 3 & 4: 10 9 8 7 6 4 5 2 1 3
=> Partisi: 3
Index 5-6 | Pivot = 4
> Swap indeks 5 & 6: 10 9 8 7 6 5 4 2 1 3
=> Partisi: 5
Index 7-9 | Pivot = 2
> Swap indeks 7 & 9: 10 9 8 7 6 5 4 3 1 2
=> Partisi: 7
Index 8-9 | Pivot = 1
> Swap indeks 8 & 9: 10 9 8 7 6 5 4 3 2 1
=> Partisi: 8
Data setelah diurutkan: 10 9 8 7 6 5 4 3 2 1
Waktu pengurutan: 42
PS C:\Users\ASUS\Documents\Struktur Data>
```