

PRAKTIKUM 11-12

ALGORITMA PENGURUTAN (BUBBLE DAN SHELL)

A. TUJUAN PEMBELAJARAN

1. Memahami mengenai algoritma pengurutan *bubble sort* dan *shell sort*.
2. Mampu mengimplementasikan algoritma pengurutan *bubble sort* dan *shell sort* secara *ascending* dan *descending*.

B. DASAR TEORI

1. Algoritma *Bubble Sort*

Bubble sort (metode gelembung) adalah metode/algoritma pengurutan dengan dengan cara melakukan penukaran data dengan tepat disebelahnya secara terus menerus sampai bisa dipastikan dalam satu iterasi tertentu tidak ada lagi perubahan. Jika tidak ada perubahan berarti data sudah terurut. Disebut pengurutan gelembung karena masing-masing kunci akan dengan lambat menggelembung ke posisinya yang tepat.

Metode pengurutan gelembung (*Bubble Sort*) diinspirasi oleh gelembung sabun yang berada dipermukaan air. Karena berat jenis gelembung sabun lebih ringan daripada berat jenis air, maka gelembung sabun selalu terapung ke atas permukaan. Prinsip di atas dipakai pada pengurutan gelembung.

Algoritma *bubble sort* adalah salah satu algoritma pengurutan yang paling simple, baik dalam hal pengertian maupun penerapannya. Ide dari algoritma ini adalah mengulang proses perbandingan antara tiap-tiap elemen array dan menukarnya apabila urutannya salah. Perbandingan elemen-elemen ini akan terus diulang hingga tidak perlu dilakukan penukaran lagi. Algoritma ini termasuk dalam golongan algoritma *comparison sort*, karena menggunakan perbandingan dalam operasi antar elemennya. Berikut ini adalah gambaran dari algoritma *bubble sort*. Misalkan kita mempunyai sebuah array dengan. Elemen-elemen “4 2 5 3 9”. Proses yang akan terjadi apabila digunakan algoritma *bubble sort* adalah sebagai berikut.

Pass pertama

(4 2 5 3 9) menjadi (2 4 5 3 9)

(2 4 5 3 9) menjadi (2 4 5 3 9)

(2 4 5 3 9) menjadi (2 4 3 5 9)

(2 4 3 5 9) menjadi (2 4 3 5 9)

Pass kedua

(2 4 3 5 9) menjadi (2 4 3 5 9)

(2 4 3 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

Pass ketiga

(2 3 4 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

Dapat dilihat pada proses di atas, sebenarnya pada pass kedua, langkah kedua, array telah terurut. Namun algoritma tetap dilanjutkan hingga pass kedua berakhir. Pass ketiga dilakukan karena definisi terurut dalam algoritma *bubble sort* adalah tidak ada satupun penukaran pada suatu pass, sehingga pass ketiga dibutuhkan untuk memverifikasi keurutan array tersebut.

2. Algoritma *Shell Sort*

Metode *shell sort* dikembangkan oleh Donald L. Shell pada tahun 1959. Dalam metode ini jarak antara dua elemen yang dibandingkan dan ditukarkan tertentu. Secara singkat metode ini dijelaskan sebagai berikut. Pada langkah pertama, kita ambil elemen pertama dan kita bandingkan dan kita bandingkan dengan elemen pada jarak tertentu dari elemen pertama tersebut. Kemudian elemen kedua kita bandingkan dengan elemen lain dengan jarak yang sama seperti jarak yang sama seperti diatas. Demikian seterusnya sampai seluruh elemen dibandingkan. Pada langkah kedua proses diulang dengan langkah

yang lebih kecil, pada langkah ketiga jarak tersebut diperkecil lagi seluruh proses dihentikan jika jarak sudah sama dengan satu.

Contoh dari proses pengurutan dengan menggunakan metode *Shell Sort* dijelaskan pada tabel 1.

Tabel 1. Proses pengurutan menggunakan metode Shell Sort

Jarak	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
Awal	22	10	15	3	2	8
Jarak = 3	<u>22</u>	10	15	<u>3</u>	2	8
	3	<u>10</u>	15	22	<u>2</u>	8
	3	2	<u>15</u>	22	<u>10</u>	<u>8</u>
	3	2	8	22	10	15
Jarak = 1	<u>3</u>	<u>2</u>	8	22	10	15
	2	<u>3</u>	<u>8</u>	22	10	15
	2	3	<u>8</u>	<u>22</u>	10	15
	2	3	8	<u>22</u>	<u>10</u>	15
	2	3	8	10	<u>22</u>	<u>15</u>
	2	3	8	10	15	22
Akhir	2	3	8	10	15	22

C. TUGAS PENDAHULUAN

Jawablah pertanyaan berikut ini :

1. Tuliskan algoritma pengurutan *bubble sort* secara *ascending*.
2. Tuliskan algoritma pengurutan *shell sort* secara *ascending*.

D. PERCOBAAN

Percobaan 1 : *Bubble sort* secara *ascending*.

```
public class Bubble {
    public static <T extends Comparable<? super T>> void
        bubbleSort(T[]arr) {
        int i, j;
        T temp;
        for(i=0 ; i<arr.length-1; i++){
            for(j=0 ; j<arr.length-i-1; j++){
                if(arr[j].compareTo(arr[j+1])>0){
                    temp = arr[j+1];
                    arr[j+1] = arr[j];
                    arr[j]= temp;
                }
            }
        }
    }
}
```

```

    }

    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[] = new Integer[10];
        for(int a=0 ; a<data.length ; a++){
            data[a]= (int) (Math.random()*13+1);
        }
        long awal = System.currentTimeMillis();
        bubbleSort(data);
        long sisaWaktu = System.currentTimeMillis() - awal;
        tampil(data);
        System.out.println("Waktu " + sisaWaktu);
    }
}

```

Percobaan 2 : Bubble sort secara ascending dengan flag.

```

public class BubbleFlag {
    public static <T extends Comparable<? super T>> void
        bubbleSortFlag(T[]arr) {
        int i=0, j;
        Boolean did_swap=true;
        T temp;
        while(i<arr.length-1 && did_swap){
            for(j=0 ; j<arr.length-i-1; j++){
                did_swap=false;
                if(arr[j].compareTo(arr[j+1])>0) {
                    temp = arr[j+1];
                    arr[j+1] = arr[j];
                    arr[j]= temp;
                    did_swap=true;
                }
            }
            i++;
        }
    }

    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[] = new Integer[10];
for(int a=0 ; a<data.length ; a++){
    data[a]= (int) (Math.random()*13+1);
}
long awal = System.currentTimeMillis();
bubbleSortFlag(data);
long sisaWaktu = System.currentTimeMillis() - awal;
tampil(data);
System.out.println("Waktu " + sisaWaktu);
}
}

```

Percobaan 3 : Shell sort secara ascending.

```

public class Shell {
    public static <T extends Comparable<? super T>> void
        shellSort(T[]arr){
        int i, jarak;
        Boolean did_swap=true;
        T temp;
        jarak=arr.length;
        while(jarak>1){
            jarak=jarak/2;
            did_swap=true;
            while(did_swap){
                did_swap=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;
                        did_swap=true;
                    }
                    i++;
                }
            }
        }
    }

    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[] = new Integer[10];
        for(int a=0 ; a<data.length ; a++){
            data[a]= (int) (Math.random()*13+1);

```

```

    }

    long awal = System.currentTimeMillis();
    shellSort(data);
    long sisaWaktu = System.currentTimeMillis() - awal;
    tampil(data);
    System.out.println("Waktu " + sisaWaktu);
}
}

```

E. LATIHAN

1. Dari percobaan 1 tambahkan method untuk melakukan pengurutan *bubble sort* secara *descending*.
2. Dari percobaan 2 tambahkan method untuk melakukan pengurutan *bubble sort* dengan *flag* secara *descending*.
3. Dari percobaan 3 tambahkan method untuk melakukan pengurutan *shell sort* secara *descending*.
4. Buatlah class Mahasiswa dengan variable `nrp` dan `nama` yang memiliki tipe `String`! Class Mahasiswa mengimplementasikan interface `Comparable`, selanjutnya implementasikan fungsi abstract `compareTo()`, untuk membandingkan dua objek mahasiswa berdasarkan `nrp`.

```

public class Mahasiswa implements Comparable <Mahasiswa> {
    private String nrp ;
    private String nama ;
    @Override
    public int compareTo(Mahasiswa o) {...}

    @Override
    public String toString() {...}
}

```

Lakukan pengujian fungsi `bubbleSort()` lagi, sebelumnya tambahkan pada fungsi `main()` seperti di bawah ini !

```

public class Latihan {
    public static <T extends Comparable<? super T>> void
        bubbleSort(T[] arr){
        .....
    }
    public static <T> void tampil(T data[]) {

```

```
.....
}
public static void main(String[] args) {
    Mahasiswa arr8[] = {new Mahasiswa("02", "Budi"),
                        new Mahasiswa("01", "Andi"),
                        new Mahasiswa("04", "Udin"),
                        new Mahasiswa("03", "Candra")};

    long awal = System.currentTimeMillis();
    bubbleSort (arr8);
    tampil(arr8);
    long sisaWaktu = System.currentTimeMillis() - awal;
    System.out.println("Waktu " + sisaWaktu);
}
}
```

5. Pada soal 4 lakukan untuk algoritma *shell sort*.

F. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.