LAPORAN

Struktur Data Linear
Praktikum 5 : AdvancedSort

NAMA : Johanes Yogtan WR

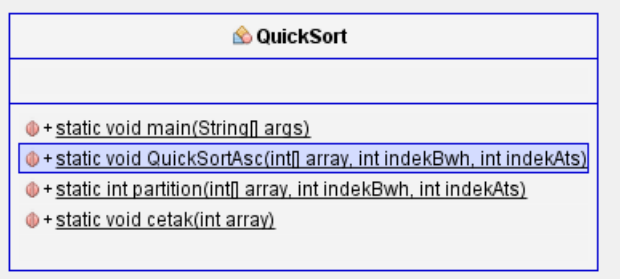
NIM : 215314105

Program Studi INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA

1. Advancedsort

Quicksort

- **Diagram UML**



- **Input (screenshot)**

Ascending :

```
package Modul3_Pengurutan;
public class QuickSort {
    public static void main(String[] args) {
        int array[] = {5, 8, 26, 15, 11, 31};
        System.out.println("DATA SEBELUM DIURUTKAN");
        cetak(array);

        QuickSortAsc(array, 0, array.length - 1);
        System.out.println("\nDATA SESUDAH DIURUTKAN");
        cetak(array);
        System.out.println("");
    }
    public static void QuickSortAsc(int[] array, int indekBwh, int indekAts) {
        if (indekBwh < indekAts) {
            int p = partition(array, indekBwh, indekAts);
            QuickSortAsc(array, indekBwh, p);
            QuickSortAsc(array, p + 1, indekAts);
        }
    }

    public static int partition(int[] array, int indekBwh, int indekAts) {
        double pivot = array[indekBwh];
        int i = indekBwh - 1;
        int j = indekAts + 1;
        for (; ; ) {
            do {
                i++;
            }while (array[i] < pivot) ;
            do {
                j--;
            } while (array[j] > pivot);
            if (i < j) {
                int t = array[i];
                array[i] = array[j];
                array[j] = t;
            } else {
                return j;
            }
        }
    }
    public static void cetak(int array[]) {
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
    }
}
```

Descending :

```
package Modul3_Pengurutan;
public class QuickSortDescending {
    public static void main(String[] args) {
        int array[] = {5, 8, 26, 15, 11, 31};
        System.out.println("DATA SEBELUM DIURUTKAN");
        cetak(array);

        QuickSortDsc(array, 0, array.length - 1);
        System.out.println("\nDATA SESUDAH DIURUTKAN");
        cetak(array);
        System.out.println("");
    }
   public static void QuickSortDsc(int[] larik, int low, int high) {
        if (low < high) {
            int p = partition(larik, low, high);
            QuickSortDsc(larik, low, p);
            QuickSortDsc(larik, p + 1, high);
        }
    }
    public static int partition(int[] larik, int low, int high) {
        double pivot = larik[low];
        int i = low - 1;
        int j = high + 1;
        for (; ; ) {
            do {
                i++;
            }while (larik[i] > pivot) ;
            do {
                j--;
            } while (larik[j] < pivot);
            if (i < j) {
                int t = larik[i];
                larik[i] = larik[j];
                larik[j] = t;
            } else {
                return j;
            }
        }
    }

    public static void cetak(int array[]) {
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
    }
}
```

- Output (screenshot)
  Ascending :

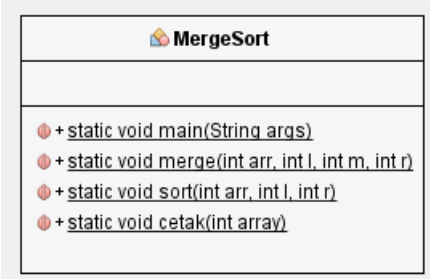
```
run:
DATA SEBELUM DIURUTKAN
5 8 26 15 11 31
DATA SESUDAH DIURUTKAN
5 8 11 15 26 31
BUILD SUCCESSFUL (total time: 0 seconds)
```

Descending :

```
run:
DATA SEBELUM DIURUTKAN
5 8 26 15 11 31
DATA SESUDAH DIURUTKAN
31 26 8 15 11 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

Mergesort

- **Diagram UML**



- **Input (screenshot)**

Ascending :

```
package Modul3_Pengurutan;
public class MergeSort {
    public static void main(String args[]) {
        int array[] = {5, 8, 26, 15, 11, 31};
        System.out.println("DATA SEBELUM DIURUTKAN");
        cetak(array);

        MergeSort ob = new MergeSort();
        ob.sort(array, 0, array.length-1);
        System.out.println("\nDATA SESUDAH DIURUTKAN");
        cetak(array);
        System.out.println("");
    }
    //Ascending
   public static void merge(int arr[], int l, int m, int r) {
        int n1 = m - l + 1;
        int n2 = r - m;

        int L[] = new int[n1];
        int R[] = new int[n2];

        for (int i = 0; i < n1; ++i) {
            L[i] = arr[l + i];
        }
        for (int j = 0; j < n2; ++j) {
            R[j] = arr[m + 1 + j];
        }

        int i = 0, j = 0;

        int k = l;
        while (i < n1 && j < n2) {
            if (L[i] <= R[j]) {
                arr[k] = L[i];
                i++;
            } else {
                arr[k] = R[j];
                j++;
            }
            k++;
        }
```

```
        while (i < n1) {
            arr[k] = L[i];
            i++;
            k++;
        }

        while (j < n2) {
            arr[k] = R[j];
            j++;
            k++;
        }
    }

    public static void sort(int arr[], int l, int r) {
        if (l < r) {
            int m = (l + r) / 2;

            sort(arr, l, m);
            sort(arr, m + 1, r);

            merge(arr, l, m, r);
        }
    }
    public static void cetak(int array[]) {
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
    }
}
```

Descending :

```
package Modul3_Pengurutan;
public class MergeSort {
    public static void main(String args[]) {
        int array[] = {5, 8, 26, 15, 11, 31};
        System.out.println("DATA SEBELUM DIURUTKAN");
        cetak(array);

        MergeSort ob = new MergeSort();
        ob.sort(array, 0, array.length-1);
        System.out.println("\nDATA SESUDAH DIURUTKAN");
        cetak(array);
        System.out.println("");
    }
    //Ascending
    public static void merge(int arr[], int l, int m, int r) {
        int n1 = m - l + 1;
        int n2 = r - m;

        int L[] = new int[n1];
        int R[] = new int[n2];

        for (int i = 0; i < n1; ++i) {
            L[i] = arr[l + i];
        }
        for (int j = 0; j < n2; ++j) {
            R[j] = arr[m + 1 + j];
        }

        int i = 0, j = 0;

        int k = l;
        while (i < n1 && j < n2) {
            if (L[i] >= R[j]) {
                arr[k] = L[i];
                i++;
            } else {
                arr[k] = R[j];
                j++;
            }
            k++;
        }
```

```
        while (i < n1) {
            arr[k] = L[i];
            i++;
            k++;
        }

        while (j < n2) {
            arr[k] = R[j];
            j++;
            k++;
        }
    }

    public static void sort(int arr[], int l, int r) {
        if (l < r) {
            int m = (l + r) / 2;

            sort(arr, l, m);
            sort(arr, m + 1, r);

            merge(arr, l, m, r);
        }
    }
    public static void cetak(int array[]) {
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
    }
}
```

- Output (screenshot)

  Ascending :

```
run:
DATA SEBELUM DIURUTKAN
5 8 26 15 11 31
DATA SESUDAH DIURUTKAN
5 8 11 15 26 31
BUILD SUCCESSFUL (total time: 0 seconds)
```

  Descending :

```
run:
DATA SEBELUM DIURUTKAN
5 8 26 15 11 31
DATA SESUDAH DIURUTKAN
31 26 15 11 8 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

- **Ilustrasi**

Ilustrasi Bublesort

Iterasi 1

X[0]	Dengan	X[1]	(5 dengan 8)	Tak Berubah
X[1]	Dengan	X[2]	(8 dengan 26)	Tidak Berubah
X[2]	Dengan	X[3]	(26 dengan 15)	Ditukar
X[3]	Dengan	X[4]	(26 dengan 11)	Ditukar
X[4]	Dengan	X[5]	(26 dengan 31)	Tidak Berubah

Iterasi 2

X[0]	Dengan	X[1]	(5 dengan 8)	Tak Berubah
X[1]	Dengan	X[2]	(8 dengan 15)	Tidak Berubah
X[2]	Dengan	X[3]	(15 dengan 11)	Ditukar
X[3]	Dengan	X[4]	(15 dengan 26)	Tidak Berubah
X[4]	Dengan	X[5]	(26 dengan 31)	Tidak Berubah

Iterasi 3 **SAMPAI** Iterasi 6

X[0]	Dengan	X[1]	(5 dengan 8)	Tak Berubah
X[1]	Dengan	X[2]	(8 dengan 11)	Tidak Berubah
X[2]	Dengan	X[3]	(11 dengan 15)	Tidak Berubah
X[3]	Dengan	X[4]	(15 dengan 26)	Tidak Berubah
X[4]	Dengan	X[5]	(26 dengan 31)	Tidak Berubah

Penjelasan

Program akan melakukan perulangan bersangkar, perulangan pertama untuk menjalankan iterasi dan perulangan kedua untuk menjalankan indek data setiap iterasi. Selama perulangan kedua akan menjalankan fungsi if, apakah array [j – 1] / array 0 lebih besar dari array 1. Jika iya, indek 0 akan disimpan ke variabel sementara dan data indek 1 ditukar ke indek 0, sehingga data indek 0 ditukar ke indek 1. jika tidak, array 1 akan berjalan membandingkan dengan data lainnya.

Seperti iterasi 1 dan 2 diatas, indek 0 dan 1 di bandingkan, ternyata 3 lebih besar dari 2, sehingga data ditukar, dilanjutkan indek 1 dan 2, ternyata 3 tidak lebih besar dari 5, sehingga data tidak ditukar, dilanjutkan indek 2 dan 3, ternyata 5 tidak lebih besar dari 7, sehingga data tidak ditukar, dilanjutkan indek 2 tadi yang sudah dipindahkan untuk membandingkan, indek 3 dan 4, ternyata 7 tidak lebih besar dari 8, sehingga data tidak ditukar, dilanjutkan indek 4 dan 5, ternyata 8 tidak lebih besar dari 9, sehingga data tidak berubah. Begitu seterusnya hingga perulangan 1 selesai dan data tidak ditukar.

Ilustrasi Selectionsort

Iterasi 1

X[0]	Dengan	X[1]	(5 dengan 8)	Tidak Berubah
X[1]	Dengan	X[2]	(5 dengan 26)	Tidak Berubah
X[2]	Dengan	X[3]	(5 dengan 15)	Tidak Berubah
X[3]	Dengan	X[4]	(5 dengan 11)	Tidak Berubah
X[4]	Dengan	X[5]	(5 dengan 31)	Tidak Berubah

Iterasi 2

X[0]	Dengan	X[1]	(8 dengan 26)	Tidak Berubah
X[1]	Dengan	X[2]	(8 dengan 15)	Tidak Berubah
X[2]	Dengan	X[3]	(8 dengan 11)	Tidak Berubah
X[3]	Dengan	X[4]	(8 dengan 31)	Tidak Berubah

Iterasi 3

X[0]	Dengan	X[1]	(26 dengan 15)	Tidak Berubah
X[1]	Dengan	X[2]	(26 dengan 11)	Ditukar
X[2]	Dengan	X[3]	(11 dengan 31)	Tidak Berubah

Iterasi 4 – Data Terurut

Penjelasan

Program akan melakukan perulangan bersangkar, perulangan pertama untuk menjalankan iterasi dan menyimpan indek yang sedang dijalankan dan perulangan kedua untuk menjalankan indek data setiap iterasi. Selama perulangan kedua akan menjalankan fungsi if, apakah array [j] / array 0 lebih besar dari array 1. Jika iya, indek 0 akan disimpan ke variabel sementara , jika tidak, array akan berjalan membandingkan dengan data lainnya.

Seperti iterasi 0 dan 1 diatas, indek 1 dan 0 di bandingkan, ternyata 7 tidak lebih kecil dari 5, sehingga data tidak ditukar, dilanjutkan indek 2 dan 0, ternyata 5 tidak lebih kecil dari 3, sehingga data tidak ditukar, dilanjutkan indek 3 dan 0, ternyata 9 tidak lebih kecil dari 3, sehingga data tidak ditukar", dilanjutkan indek 4 dan 0, ternyata 3 tidak lebih kecil dari 2, sehingga data tidak ditukar, dilanjutkan indek 5 dan 0, ternyata 8 tidak lebih kecil dari 2, sehingga data tidak ditukar. Begitu seterusnya hingga perulangan 1 selesai dan data tidak ditukar.

Ilustrasi Insertionsort

Iterasi 1

X[0]	Dengan	X[1]	(5 dengan 8)	Tidak Berubah
X[1]	Dengan	X[2]	(8 dengan 26)	Tidak Berubah
X[2]	Dengan	X[3]	(26 dengan 15)	Ditukar
X[2]	Dengan	X[3]	(15 dengan 26)	Tidak Berubah
X[4]	Dengan	X[5]	(11 dengan 31)	Tidak Berubah

Iterasi 2

X[1]	Dengan	X[2]	(8 dengan 15)	Tidak Berubah
X[2]	Dengan	X[3]	(15 dengan 26)	Tidak Berubah
X[3]	Dengan	X[4]	(26 dengan 11)	Ditukar
X[3]	Dengan	X[4]	(11 dengan 26)	Tidak Berubah
X[4]	Dengan	X[5]	(26 dengan 31)	Tidak Berubah

Iterasi 3

X[2]	Dengan	X[3]	(15 dengan 11)	Ditukar
X[2]	Dengan	X[3]	(11 dengan 15)	Tidak Berubah
X[3]	Dengan	X[4]	(15 dengan 26)	Tidak Berubah
X[4]	Dengan	X[5]	(26 dengan 31)	Tidak Berubah

Iterasi 4

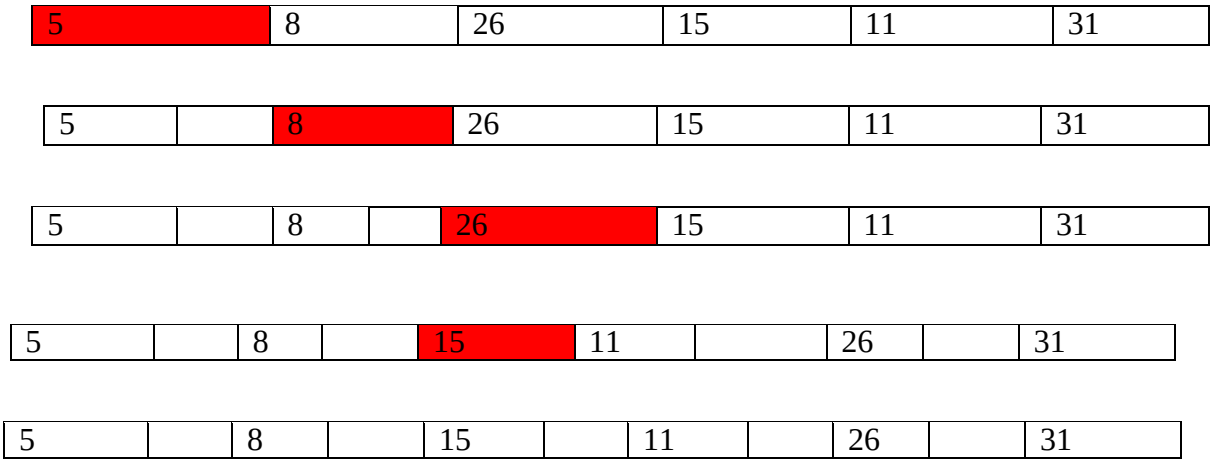
X[4]	Dengan	X[5]	(26 dengan 31)	Tidak Berubah

Penjelasan

Program akan melakukan perulangan , perulangan pertama untuk menjalankan iterasi dan menyimpan indek yang sedang dijalankan dan menyimpan indek dikurangi satu yang sedang dijalanin. Selama perulangan berlangsng akan menjalankan fungsi while, apakah I / array 0 lebih besar dari array -1 dan array[i]/array 0 lebih besar dari 1. Jika iya, maka data array 0 akan disimpan array 1 dan indeknya dikurangin, jika tidak maka data aknan disimpan seperti biasa.

Seperti iterasi 0 dan 1 diatas, indek 0 dan 1 di bandingkan, ternyata 5 tidak ebih besar dari 8, sehingga data tidak ditukar, dilanjutkan indek 1 dan 2, ternyata 8 tidak lebih kecil dari 26, sehingga data tidak ditukar, dilanjutkan indek 2 dan 3, ternyata 26 lebih kecil dari 3, sehingga data tidak ditukar", dilanjutkan indek 4 dan 0, ternyata 3 tidak lebih besar dari 15, sehingga ditukar, dilanjutkan indek 3 dan 2, ternyata 15 tidak lebih besar dari 26, sehingga data tidak ditukar. Begitu seterusnya hingga perulangan 1 selesai dan data tidak ditukar.

Ilustrasi Insertionsort



Penjelasan

Dikarenakan saya sedikit bingung menjelaskan jalannya program, maka saya menjelaskan jalan quiksortnya aja.

Pertama-tama membandingkan index 1-6 dengan pivot 5, ternyata tidak ditukar karena semua data dikanan pivot lebih besar dari pivot, dilanjutkan dengan membandingkan index 2-6 dengan pivot yaitu 8, ternyata posisi 7 ditukar ke belakang pivot, dilanjutkan membandingkan index 3-9 dengan pivot 26, ternyata posisi 15,11 ditukar menjadi dibelakang pivot, dilanjutkan membandingkan index 4 dengan pivot 15, ternyata posisi 15 dan 11 ditukar, terakhir posisi 26 dan 31 tidak ditukar.

- **Tabel Perbandingan Iterasi**

Perbandingan Bublesort

	[0]	[1]	[2]	[3]	[4]	[5]	Jumlah Perbandingan (jumlah pertukaran yang terjadi)
Awal	5	8	26	15	11	31	-
Iterasi 1	5	8	15	11	26	31	2
Iterasi 2	5	8	11	15	26	31	1
Iterasi 3	5	8	11	15	26	31	0
Iterasi 4	5	8	11	15	26	31	0
Iterasi 5	5	8	11	15	26	31	0
Iterasi 6	5	8	11	15	26	31	0

Perbandingan Selectionsort

	[0]	[1]	[2]	[3]	[4]	[5]	Jumlah Perbandingan (jumlah pertukaran yang terjadi)
Awal	5	8	26	15	11	31	-
Iterasi 1	5	8	26	15	11	31	0
Iterasi 2	5	8	26	15	11	31	0
Iterasi 3	5	8	11	15	26	31	1
Iterasi 4	5	8	11	15	26	31	0
Iterasi 5	5	8	11	15	26	31	0
Iterasi 6	5	8	11	15	26	31	0

Perbandingan InsertionSort

	[0]	[1]	[2]	[3]	[4]	[5]	Jumlah Perbandingan (jumlah pertukaran yang terjadi)
Awal	5	8	26	15	11	31	-
Iterasi 1	5	8	15	26	11	31	1
Iterasi 2	5	8	15	11	26	31	1
Iterasi 3	5	8	11	15	26	31	1
Iterasi 4	5	8	11	15	26	31	-

Perbandingan QuickSort

Pivot = Merah
Pertukaran Kuning dan Orange

	[0]	[1]	[2]	[3]	[4]	[5]	Jumlah Perbandingan (jumlah pertukaran yang terjadi)
Awal	5	8	26	15	11	31	-
Iterasi1	5	8	26	15	11	31	-
Iterasi2	5	8	26	15	15	31	-
Iterasi3	5	8	26	15	11	31	-
Iterasi4	5	8	15	11	26	31	2
Iterasi5	5	8	11	15	26	31	-

2. Tabel Perbandingan Waktu

Perbandingan Bublesort

N	Nano Time
1000	2029945910511200
10.000	2029960435115000
100.000	2029998527876600
1.000.000	Too long wait…
10.000.000	Too long wait…

Perbandingan Selectionsort

N	Nano Time
1000	2030067614516600
10.000	2030083652602600
100.000	2030101995251700
1.000.000	Too long wait…
10.000.000	Too long wait…

Perbandingan Insertionsort

N	Nano Time
1000	2030148311283600
10.000	2030165390262800
100.000	2030192856825500
1.000.000	Too long wait…
10.000.000	Too long wait…

Perbandingan QuickSort

N	Nano Time
1000	2030209820244500
10.000	2030242190829800
100.000	2030265086097100
1.000.000	2030282268953800
10.000.000	Too long wait…

Perbandingan Mergesort

N	Nano Time
1000	2030314746857300
10.000	2030329101728800
100.000	2030342930358100
1.000.000	2030359429424700
10.000.000	Too long wait…