# PRAKTIKUM 14
# ALGORITMA PENGURUTAN (MERGE SORT)

## A. TUJUAN PEMBELAJARAN

1. Memahami step by step algoritma pengurutan merge sort.
2. Mampu mengimplementasikan algoritma pengurutan merge sort dengan berbagai macam parameter berupa tipe data primitif atau tipe Generic.
3. Mampu mengimplementasikan algoritma pengurutan merge sort secara ascending dan descending.

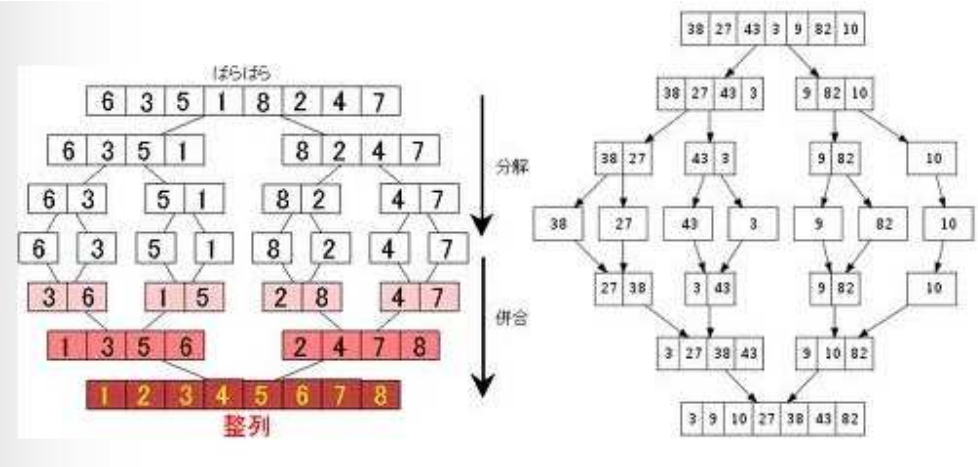
## B. DASAR TEORI

**Algoritma *Merge Sort***

Merge sort merupakan algoritma pengurutan dalam ilmu komputer yang dirancang untuk memenuhi kebutuhan pengurutan atas suatu rangkaian data yang tidak memungkinkan untuk ditampung dalam memori komputer karena jumlahnya yang terlalu besar. Algoritma ini ditemukan oleh John von Neumann pada tahun 1945.

Algoritma pengurutan data merge sort dilakukan dengan menggunakan cara divide and conquer yaitu dengan memecah kemudian menyelesaikan setiap bagian kemudian menggabungkannya kembali. Pertama data dipecah menjadi 2 bagian dimana bagian pertama merupakan setengah (jika data genap) atau setengah minus satu (jika data ganjil) dari seluruh data, kemudian dilakukan pemecahan kembali untuk masing-masing blok sampai hanya terdiri dari satu data tiap blok.

Setelah itu digabungkan kembali dengan membandingkan pada blok yang sama apakah data pertama lebih besar daripada data ke-tengah+1, jika ya maka data ke-tengah+1 dipindah sebagai data pertama, kemudian data ke-pertama sampai ke-tengah digeser menjadi data ke-dua sampai ke-tengah+1, demikian seterusnya sampai menjadi satu blok utuh seperti awalnya. Sehingga metode merge sort merupakan metode yang membutuhkan fungsi rekursi untuk penyelesaiannya.




Gambar 1. Contoh pengurutan menggunakan *MergeSort*

Contoh penerapan atas sebuah larik/array dengan data yang akan diurutkan {6,3,5,1,8,2,4,7} ditunjukkan pada gambar 1. Pertama kali larik tersebut dibagi menjadi dua bagian, {6,3,5,1} dan {8,2,4,7}. Larik {6,3,5,1} dibagi menjadi dua bagian yaitu, {6,3} dan {5,1}. Larik {6,3} dibagi menjadi dua bagian yaitu, {6} dan {3}. Selanjutnya karena {6} dan {3} sudah tidak bisa dibagi lagi maka di merge dan diurutkan menjadi {3,6}. Larik {5,1} dibagi menjadi dua bagian yaitu, {5} dan {1}. Selanjutnya {5} dan {1} dilakukan merge dan diurutkan menjadi {1,5}. Larik {3,6} dan {1,5} dimerge dan diurutkan menjadi {1,3,5,6}.

Larik {8,2,4,7} dibagi menjadi dua bagian yaitu, {8,2} dan {4,7}. Larik {8,2} dibagi menjadi dua bagian yaitu, {8} dan {2}. Selanjutnya {8} dan {2} di merge dan diurutkan menjadi {2,8}. Larik {4,7} dibagi menjadi dua bagian yaitu, {4} dan {7}. Selanjutnya {4} dan {7} dilakukan merge dan diurutkan menjadi {4,7}. Larik {2,8} dan {4,7} dimerge dan diurutkan menjadi {2,4,7,8}. Larik {1,3,5,6} dan larik {2,4,7,8} dimerge dan diurutkan menjadi {1,2,3,4,5,6,7,8}.

## C. TUGAS PENDAHULUAN

Jelaskan algoritma pengurutan *merge sort* secara *ascending* dengan data 5 6 3 1 2

## D. PERCOBAAN

**Percobaan 1 : *Merge sort* secara *ascending* dengan data int.**

```
public class MergeDemo {

    private int[] array;
    private int[] tempMergArr;
    private int length;

    public MergeDemo(int[] array) {
        this.array = array;
        this.length = array.length;
        this.tempMergArr = new int[length];
    }

    public void mergeSort(int lowerIndex, int higherIndex) {
        if (lowerIndex < higherIndex) {
            int middle = lowerIndex + (higherIndex - lowerIndex) / 2;
            // Below step sorts the left side of the array
            mergeSort(lowerIndex, middle);
            // Below step sorts the right side of the array
            mergeSort(middle + 1, higherIndex);
            // Now merge both sides
            mergeParts(lowerIndex, middle, higherIndex);
        }
    }

    private void mergeParts(int lowerIndex, int middle, int
higherIndex) {

        for (int i = lowerIndex; i <= higherIndex; i++) {
            tempMergArr[i] = array[i];
        }
        int i = lowerIndex;
        int j = middle + 1;
        int k = lowerIndex;
        while (i <= middle && j <= higherIndex) {
            if (tempMergArr[i] <= tempMergArr[j]) {
                array[k] = tempMergArr[i];
                i++;
            } else {
                array[k] = tempMergArr[j];
                j++;
            }
            k++;
        }
        while (i <= middle) {
            array[k] = tempMergArr[i];
            k++;
            i++;
        }
    }
```

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

public class MainMerge {
    public static void main(String[] args) {
        int[] inputArr = {45, 23, 11, 89, 77, 98, 4, 28, 65, 43};
        MergeDemo mms = new MergeDemo(inputArr);
        mms.tampil();
        mms.mergeSort(0,inputArr.length-1);
        mms.tampil();
    }
}
```

**Percobaan 2 : *Merge sort* secara *descending* dengan data int.**

```
public class MergeDemo {

    private int[] array;
    private int[] tempMergArr;
    private int length;

    public MergeDemo(int[] array) {
        this.array = array;
        this.length = array.length;
        this.tempMergArr = new int[length];
    }

    public void mergeSort(int lowerIndex, int higherIndex) {
        if (lowerIndex < higherIndex) {
            int middle = lowerIndex + (higherIndex - lowerIndex) / 2;
            // Below step sorts the left side of the array
            mergeSort(lowerIndex, middle);
            // Below step sorts the right side of the array
            mergeSort(middle + 1, higherIndex);
            // Now merge both sides
            mergeParts(lowerIndex, middle, higherIndex);
        }
    }

    private void mergeParts(int lowerIndex, int middle, int
higherIndex) {

        for (int i = lowerIndex; i <= higherIndex; i++) {
            tempMergArr[i] = array[i];
        }
        int i = lowerIndex;
        int j = middle + 1;
        int k = lowerIndex;
```

```
        while (i <= middle && j <= higherIndex) {
            if (tempMergArr[i] >= tempMergArr[j]) {
                array[k] = tempMergArr[i];
                i++;
            } else {
                array[k] = tempMergArr[j];
                j++;
            }
            k++;
        }
        while (i <= middle) {
            array[k] = tempMergArr[i];
            k++;
            i++;
        }
    }

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

public class MainMerge {
    public static void main(String[] args) {
        int[] inputArr = {45, 23, 11, 89, 77, 98, 4, 28, 65, 43};
        MergeDemo mms = new MergeDemo(inputArr);
        mms.tampil();
        mms.mergeSort(0,inputArr.length-1);
        mms.tampil();
    }
}
```

## E. LATIHAN

1. Buatlah class MergeDemo2, copy program pada percobaan 1, lakukan modifikasi pada program tersebut dengan menggunakan tipe Generic.

```
public class MergeDemo2<T extends Number> { }
```

Ujilah program tersebut dengan

```
public class MainMerge2 {
    public static void main(String[] args) {
        Double[] inputArr = {45.4, 23.1, 11.4, 89.1, 77.9, 98.7,
4.2, 28.6, 65.8, 43.9};
        MergeDemo2 mms = new MergeDemo2(inputArr);
        mms.tampil();
```

```
        mms.mergeSort(0,inputArr.length-1);
      mms.tampil();
    }
}
```

2. Buatlah class MergeDemo3, copy program pada percobaan 1, lakukan modifikasi pada program tersebut dengan menggunakan tipe Generic.

```
public class MergeDemo3<T extends Comparable> { }
```

Ujilah program tersebut dengan

```
public class MainMerge3 {
    public static void main(String[] args) {
        String inputArr []=
  {"jeruk","anggur","belimbing","jambu","kelengkeng"};
        MergeDemo3 mms = new MergeDemo3(inputArr);
        mms.tampil();
        mms.mergeSort(0,inputArr.length-1);
    }
}
```

3. 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() {…}
}
```

Ujilah program pada soal no 2 dengan

```
public class MainMerge4 {
    public static void main(String[] args) {
        Mahasiswa inputArr[] = {new Mahasiswa("02", "Budi"), new
  Mahasiswa("01", "Andi"), new Mahasiswa("04", "Udin"), new Mahasiswa("03",
  "Candra")};

        MergeDemo3 mms = new MergeDemo3(inputArr);
```

```
            mms.tampil();
            mms.mergeSort(0,inputArr.length-1);
        }
    }
```

## F. LAPORAN RESMI

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