Mochamad Farid Maulana

1941720012

TI-1D/18

LAPORAN TUGAS JOBSHEET 4

1. Fungsi Main

```
public class MainMinMax {                                                   // -

    public static void main(String[] args) {                               // -
        minMax[] ppArray = new minMax[5];                                   // O(1)
        ppArray[0] = new minMax();                                          // O(1)

        Scanner sc = new Scanner(System.in);                                // O(1)

        for (int i = 0; i < 5; i++) {                                       // O(n)
            ppArray[i] = new minMax();                                      // O(1)
            System.out.println("Nilai array indeks ke-" + i);               // O(1)
            System.out.print("Masukkan nilai: ");                          // O(1)
            ppArray[i].nilaiArray = sc.nextInt();                           // O(1)
        }                                                                   // -

        System.out.println("---------------------------------------");     // O(1)
        System.out.println("Nilai yang Anda masukkan : ");                  // O(1)
        for (int i = 0; i < 5; i++) {                                       // O(n)
            System.out.println("Indeks ke-" + i + " : "
                    + ppArray[i].nilaiArray);                               // O(1)
        }                                                                   // -

        int arr[] = new int[5];                                             // O(1)
        for (int i = 0; i < 5; i++) {                                       // O(n)
            arr[i] = ppArray[i].nilaiArray;                                 // O(1)
        }                                                                   // -

        minMax hsl = new minMax();                                          // O(1)
        hsl.MinMax(arr, 5);                                                 // O(1)

        System.out.println("---------------------------------------");     // O(1)
        System.out.println("Brute Force");                                  // O(1)
        System.out.println("Nilai Minimal : " + hsl.getMin());              // O(1)
        System.out.println("Nilai Maksimal : " + hsl.getMax());             // O(1)

        max_min(arr, 0, 4, hsl);                                            // O(1)

        System.out.println("---------------------------------------");     // O(1)
        System.out.println("Divide Conquer");                               // O(1)
        System.out.print("Nilai Minimal: " + hsl.min
                + "\nNilai Maksimal: " + hsl.max);                          // O(1)
        System.out.print("\n");                                             // O(1)
    }                                                                       // -
}                                                                           // -
```

Hasil perhitungan :

1 + 1 + 1 + (n * 1) + 1 + (n * 1) + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1

16 + (2 * O(n))
O(n)

Brute Force

```
public class minMax {                                                   // -

    public int nilaiArray, min, max;                                    // -

    minMax() {}                                                         // -

    public minMax(int min, int max) {                                   // -
        this.min = min;                                                 // O(1)
        this.max = max;                                                 // O(1)
    }                                                                   // -

    public void MinMax(int arr[], int n) {                              // -
        min = arr[0];                                                   // O(1)
        max = arr[0];                                                   // O(1)

        for (int i = 0; i < n; i++) {                                   // O(n)
            if (arr[i] < min) {                                         // -
                min = arr[i];                                           // O(n)
            } else if (arr[i] > max) {                                  // -
                max = arr[i];                                           // O(n)
            }                                                           // -
        }                                                               // -
    }                                                                   // -

    public int getMax() {                                               // -
        return max;                                                     // O(1)
    }                                                                   // -

    public int getMin() {                                               // -
        return min;                                                     // O(1)
    }                                                                   // -
```

Hasil perhitungan :

1 + 1 + n + n + n = 2 + 3 * O(n)

= O(n)

Divide and Conquer

```
    public static void max_min
        (int[] arr, int indeks_awal, int indeks_akhir, minMax hasil) {  // -
        int indeks_tengah;                                              // -
        minMax hasil1 = new minMax();                                   // O(1)
        minMax hasil2 = new minMax();                                   // O(1)

        if (indeks_awal == indeks_akhir) {                              // -
            hasil.min = hasil.max = arr[indeks_awal];                   // O(1)
        } else if (indeks_akhir - indeks_awal == 1) {                   // -
            if (arr[indeks_awal] > arr[indeks_akhir]) {                 // -
                hasil.min = arr[indeks_akhir];                          // O(1)
                hasil.max = arr[indeks_awal];                           // O(1)
            } else {                                                    // -
                hasil.min = arr[indeks_awal];                           // O(1)
                hasil.max = arr[indeks_akhir];                          // O(1)
            }                                                           // -
        } else {                                                        // -
            indeks_tengah = (indeks_awal + indeks_akhir) / 2;           // O(1)
            max_min(arr, indeks_awal, indeks_tengah, hasil1);           // O(n)
            max_min(arr, indeks_tengah + 1, indeks_akhir, hasil2);      // O(n)

            hasil.min = (hasil1.min < hasil2.min)
                    ? hasil1.min : hasil2.min;                          // O(1)
            hasil.max = (hasil1.max > hasil2.max)
                    ? hasil1.max : hasil2.max;                          // O(1)
        }                                                               // -
    }                                                                   // -
}                                                                       // -
```

Hasil Perhitungan

$1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + (2^n) + 1 + 1 = 8 + n^2 + 2$

$= 10 + n^2$

$= O(k^n)$

2. Brute Force

```
public void fiboBF(int n) {                                                    // -
    int a, b, hasil;                                                           // -
    a = 2;                                                                     // O(1)
    b = 3;                                                                     // O(1)
    hasil = 0;                                                                 // O(1)
    for (int i = 2; i < 5; i++) {                                              // O(n)
        hasil = a + b;                                                         // O(1)
        System.out.println(a + " " + b + " " + hasil);                         // O(1)
        a = b;                                                                 // O(1)
        b = hasil;                                                             // O(1)
    }                                                                          // -
}                                                                              // -
```

Hasil perhitungan :

1 + 1 + 1 + (O(n) * 1 * 1 * 1 * 1) = 3 + O(n) = O(n)

Divide and Conquer

```
public int fiboDC(int n) {                                                     // -
    if (n == 0) {                                                              // -
        return 0;                                                              // O(1)
    }                                                                          // -
    if (n == 1) {                                                              // -
        return 1;                                                              // O(1)
    }                                                                          // -
    return fiboDC(n - 1) + fiboDC(n - 2);                                      // O(n)
}                                                                              // -
```

Hasil perhitungan :

1 + 1 + O(n) = 2 + O(n) = O(n)

3. Fungsi main

```
public class MainFaktorial {                                                    // -

    public static void main(String[] args) {                                   // -
        Scanner farlan = new Scanner(System.in);                                // O(1)

        System.out.println("=======================================");          // O(1)
        System.out.print("Masukkan jumlah elemen yang "
                + "ingin dihitung : ");                                         // O(1)
        int elemen = farlan.nextInt();                                          // O(1)

        Faktorial fk[] = new Faktorial[elemen];                                 // O(1)
        for (int i = 0; i < elemen; i++) {                                      // O(n)
            fk[i] = new Faktorial();                                            // O(1)
            System.out.print("Masukkan nilai data ke-"
                    + (i + 1) + " : ");                                         // O(1)
            fk[i].nilai = farlan.nextInt();                                     // O(1)
        }                                                                       // -

        System.out.println("=======================================");          // O(1)
        System.out.println("Hasil Faktorial dengan Brute Force");               // O(1)
        for (int i = 0; i < elemen; i++) {                                      // O(n)
            System.out.println("Faktorial dari nilai "
                    + fk[i].nilai + " adalah : "
                    + fk[i].faktorialBF(fk[i].nilai));                          // O(1)
        }                                                                       // -

        System.out.println("=======================================");          // O(1)
        System.out.println("Hasil Faktorial dengan "
                + "Divide and Conquer");                                        // O(1)
        for (int i = 0; i < elemen; i++) {                                      // O(n)
            System.out.println("Faktorial dari nilai "
                    + fk[i].nilai + " adalah : "
                    + fk[i].faktorialDC(fk[i].nilai));                          // O(1)
        }                                                                       // -
    }                                                                           // -
}                                                                               // -
```

Hasil perhitungan :
1 + 1 + 1 + 1 + 1 + (O(n) * 1 * 1 * 1) + 1 + 1 + (O(n) * 1) + 1 + 1 + (O(n) * 1)
9 + (3 * O(n)) = O(n)

Brute Force

```
public class Faktorial {                        // -
    int nilai;                                  // -

    int faktorialBF(int n) {                    // -
        int fakto = 1;                          // O(1)
        int i = 1;                              // O(1)
        while(i <= n){                          // O(n)
            fakto = fakto * i;                  // O(1)
            i++;                                // O(1)
        }                                       // -
        return fakto;                           // O(1)
    }                                           // -
```

Hasil Perhitungan :

1 + 1 + (O(n) * 1 * 1) + 1 = 3 + O(n)

= O(n)

Divide and Conquer

```
    int faktorialDC(int n) {                          // -
        if (n == 1) {                                 // -
            return 1;                                 // O(1)
        } else {                                      // -
            int fakto = n * faktorialDC(n - 1);       // O(n)
            return fakto;                             // O(1)
        }                                             // -
    }                                                 // -
}                                                     // -
```

Hasil Perhitungan

1 + O(n) + 1 = 2 + O(n)

= O(n)