# Laporan Algoritma Dan Struktur Data

## Jobsheet 4


Ghazwan Ababil

244107020151

1E – Teknik Informatika

Program Studi D-IV Teknik Informatika

Jurusan Teknologi Informasi

Politeknik Negeri Malang

2025

## 1. Praktikum

### 1.1 Percobaan 1: Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

#### 1.1.1 Kode Program

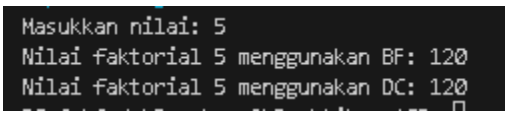
Membuat package Jobsheet4, lalu membuat class, dan atribut pada class Faktorial

```
package Jobsheet4;

public class Faktorial{
    int faktorialBF(int n){
        int fakto = 1;
        for (int i = 1; i <= n; i++)
            fakto = fakto * i;
        return fakto;
    }
    int faktorialDC(int n){
        if (n == 1) return 1;
        else {
            int fakto = n * faktorialDC(n-1);
            return fakto;
        }
    }
}
```

Membuat Class untuk mengeksekusi dengan nama MainFaktorial, menambahkan fungsi main pada file tersebut, menambahkan scanner untuk menerima nilai berdasarkan input pengguna, dan membuat objek dari class Faktorial dan menampilkan hasil pemanggilan dari method faktorialBC() dan faktorialDC()

```
package Jobsheet4;

import java.util.Scanner;

public class MainFaktorial{
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan nilai: ");
        int nilai = input.nextInt();

        Faktorial fk = new Faktorial();
        System.out.println("Nilai faktorial " + nilai + " menggunakan BF:
" + fk.faktorialBF(nilai));
        System.out.println("Nilai faktorial " + nilai + " menggunakan DC:
" + fk.faktorialDC(nilai));
    }
}
```

### 1.1.2 Hasil Running Kode Program




### 1.1.3 Pertanyaan

1) Pada method faktorialDC terdapat bagian if dan else, bagian if berperan sebagai base case untuk menghentikan perulangan rekursif dengan kondisi n ==1, lalu bagian else merupakan proses pemecahan permasalahan faktorial dengan memecah n menjadi nilai yang lebih kecil hingga mencapai base case.
2) Method faktorialBF() dapat diubah perulangannya selain menggunakan for. Berikut kode programnya:

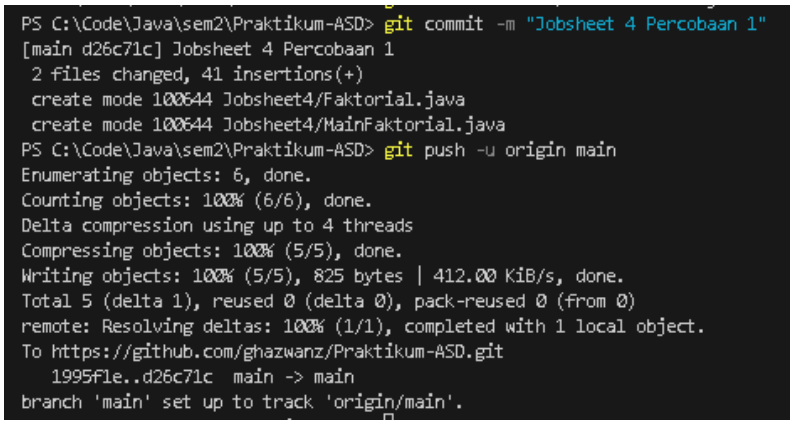
```
int faktorialBF(int n){
     int fakto = 1;
     int index = 1;
     while (index <= n) {
         fakto = fakto * index;
         index++;
     }
     return fakto;
}
```

3) Perbedaan proses perhitungan faktorial antara fakto *= i; dan int fakto = n * faktorialDC(n-1);
   - fakto *= i;

     proses perulangan yang terjadi dengan cara fakto *= i berjalan secara bertahap perkalian dilakukan dari yang terkecil.
   - int fakto = n * faktorialDC(n-1)

     proses perulangan yang dilakukan dengan cara int fakto = n * faktorialDC(n-1) berjalan dengan cara dari terbesar kemudian ke terkecil.
4) Method faktorialBF() bekerja secara bertahap melakukan proses perkalian faktorial dari terkecil hingga ke-n. Sedangkan method faktorialDC() bekerja dari bilangan terbesar (n) kemudian disederhanakan hingga bentuk paling kecil (1).

5) Commit dan push kode program




## 1.2 Percobaan 2: Menghitung Hasil Pangkat dengan Algoritma Brute Force dan Divide and Conquer

### 1.2.1 Kode Program

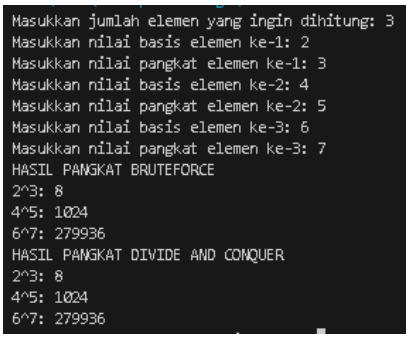
Membuat class baru bernama Pangkat, kemudian menambahkan konstruktor berparameter, atribut berupa nilai dan pangkat, dan method untuk menghitung pangkat secara brute force dan devide and conquer.

```
package Jobsheet4;

public class Pangkat11 {
    int nilai, pangkat;

    Pangkat11(int nilai, int pangkat){
        this.nilai = nilai;
        this.pangkat = pangkat;
    }

    int pangkatBF(int a, int n){
        int hasil = 1;
        for (int i = 0; i < n; i++) hasil = hasil * a;
        return hasil;
    }

    int pangkatDC(int a, int n){
        if (n == 1) return a;
        else {
            if (n % 2 == 1) return (pangkatDC(a, n/2) * pangkatDC(a, n/2)*a);
            else return (pangkatDC(a, n/2) * pangkatDC(a, n/2));
        }
    }
}
```

Kemudian membuat class baru bernama MainPangkat untuk mengeksekusi, menambahkan fungsi main pada class tersebut, menambahkan scanner untuk menginput jumlah elemen yang akan dihitung pangkatnya, serta instansiasi array of object dan melakukan pemanggilan method untuk menghitung pangkat secara brute force dan devide and conquer.

```
package Jobsheet4;

import java.util.Scanner;

public class MainPangkat {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan jumlah elemen yang ingin dihitung: ");
        int elemen = input.nextInt();

        Pangkat[] png = new Pangkat[elemen];
        for (int i = 0; i < elemen; i++) {
            System.out.print("Masukkan nilai basis elemen ke-" + (i + 1) + ": ");
            int basis = input.nextInt();
            System.out.print("Masukkan nilai pangkat elemen ke-" + (i + 1) + ": ");
            int pangkat = input.nextInt();
            png[i] = new Pangkat(basis, pangkat);
        }

        System.out.println("HASIL PANGKAT BRUTEFORCE");
        for (Pangkat p : png) {
            System.out.println(p.nilai + "^" + p.pangkat +": " +
p.pangkatBF(p.nilai, p.pangkat));
        }
        System.out.println("HASIL PANGKAT DIVIDE AND CONQUER");
        for (Pangkat p : png) {
            System.out.println(p.nilai + "^" + p.pangkat +": " +
p.pangkatDC(p.nilai, p.pangkat));
        }
    }
}
```

### 1.2.2 Hasil Run Program



### 1.2.3 Pertanyaan

1) Perbedaan dari method pangkatBF() dan pangkatDC() adalah
   - pangkatBF() melakukan proses perkalian nilai a hingga n kali secara iteratif
   - Sedangkan pangkatDC() proses yang dilakukan untuk menghitung pangkat yaitu, pangkat n dipecah (divide) menjadi lebih kecil dengan dibagi 2 hingga n menjadi bentuk paling kecil / 1, sehingga proses yang dilakukan tidak dilakukan secara iteratif sebanyak n kali, melainkan menghitung nilai pangkat dengan a ^ (n/2) dan digunakan kembali.
2) Ya, proses combine sudah termasuk ke dalam kode tersebut

```
if (n % 2 == 1) return (pangkatDC(a, n/2) * pangkatDC(a, n/2)*a);
else return (pangkatDC(a, n/2) * pangkatDC(a, n/2));
```

   Dimana pada return statement, Proses ini menggabungkan hasil penyelesaian dari submasalah yang dimana jika nilai n bernilai ganjil maka akan mengembalikan proses penghitungan pangkatDC(a,n/2) * pangkatDC(a,n/2) * a, sedangkan jika nilai n bernilai genap maka akan mengembalikan proses pangkatDC(a,n/2) * pangkatDC(a,n/2)
3) Method pangkatBF() tetap relevan meski memiliki parameter sendiri, akan tetapi nilai yang diinputkan pada parameter harus sama dengan nilai pada atribut nilai dan pangkat agar bisa tetap relevan. Jadi menurut saya menurut saya method tersebut kurang relevan untuk memiliki parameter, karena diperlukan kondisi tertentu agar method tersebut dikatakan cukup relevan.

   Method pangkatBF() dapat atau lebih baik dibuat tanpa parameter, berikut modifikasi method pangkatBF() tanpa menggunakan parameter

```
int pangkatBF(){
     int hasil = 1;
     for (int i = 0; i < pangkat; i++) hasil = hasil * nilai;
     return hasil;
}
```

4) pangkatBF() melakukan proses perhitungan pangkat nilai a dikalikan dengan nilai a hingga n kali secara iteratif.

Sedangkan pangkatDC() melakukan proses perhitungan pangkat n nilai a dengan metode divide and conquer, yang dimana membagi proses menghitung pangkat menjadi lebih kecil, dan menyatukannya kembali.

5) Push dan commit Kode Program

```
PS C:\Code\Java\sem2\Praktikum-ASD> git commit -m "Jobsheet 4 Percobaan 2 DC dan BF menghitung hasil pangkat"
[main 144865e] Jobsheet 4 Percobaan 2 DC dan BF menghitung hasil pangkat
 2 files changed, 61 insertions(+)
 create mode 100644 Jobsheet4/MainPangkat.java
 create mode 100644 Jobsheet4/Pangkat.java
PS C:\Code\Java\sem2\Praktikum-ASD> git push -u origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 1.08 KiB | 220.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ghazwanz/Praktikum-ASD.git
   d26c71c..144865e  main -> main
branch 'main' set up to track 'origin/main'.
```

## 1.3 Percobaan 3: Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

### 1.3.1 Kode Program

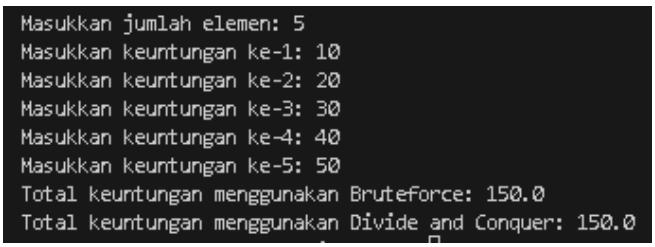
Membuat class baru dengan nama Sum11

```
package Jobsheet4;

public class Sum11 {
    double keuntungan[];

    Sum11(int el) {
        keuntungan = new double[el];
    }

    double totalBF() {
        double total = 0;
        for (int i = 0; i < keuntungan.length; i++) {
            total = total + keuntungan[i];
        }
        return total;
    }

    double totalDC(double arr[], int l, int r) {
        if (l == r) {
            return arr[l];
        }

        int mid = (l + r) / 2;
        double lsum = totalDC(arr, l, mid);
        double rsum = totalDC(arr, mid + 1, r);
        return lsum + rsum;
    }
}
```

Membuat class baru MainSum11 dan menambahkan fungsi main, melakukan instansiasi array of object kemudian menampilkan hasil perhitungannya.

```
package Jobsheet4;

import java.util.Scanner;

public class MainSum11 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan jumlah elemen: ");
        int elemen = input.nextInt();

        Sum11 sm = new Sum11(elemen);
        for (int i = 0; i < elemen; i++) {
            System.out.print("Masukkan keuntungan ke-" + (i + 1) + ": ");
            sm.keuntungan[i] = input.nextDouble();
        }

        System.out.println("Total keuntungan menggunakan Bruteforce: "+
sm.totalBF());
        System.out.println("Total keuntungan menggunakan Divide and Conquer: "+
sm.totalDC(sm.keuntungan, 0, elemen - 1));
    }
}
```

### 1.3.2 Hasil Run Program




### 1.3.3 Pertanyaan

1) Variabel mid diperlukan pada method TotalDC(), variabel mid digunakan untuk mengetahui indeks titik tengah (median) dari attribut array keuntungan yang akan digunakan sebagai pembatas dari perhitungan kiri (leftsum) dan sebagai awalan dari perhitungan kanan (rightsum)
2) Statement
   - double lsum = totalDC(arr,l,mid);

     berfungsi untuk menghitung penjumlahan array pada indeks 0 – mid (indeks tengah pada array) atau proses penjumlahan pada sisi kiri array secara rekursif.
   - double rsum = totalDC(arr,mid+1,r);

     sedangkan rsum berfungsi untuk menghitung proses penjumlahan array pada indeks tengah (mid+1) hingga batas indeks dari array atau bisa dibilang proses penjumlahan pada sisi kanan array secara rekursif.
3) Penjumlahan hasil lsum dan rsum dilakukan untuk menggabungkan solusi dari permasalahan yang telah dipecah. Hasil lsum merupakan solusi dari sub masalah pertama (perhitungan dari indeks 0 sampai median) dan hasil rsum merupakan solusi dari sub masalah kedua(perhitungan dari tengah hingga indeks terakhir). Untuk menemukan solusi akhir maka diperlukan untuk menjumlahkan lsum dan rsum.
4) Base case dari method totalDC yaitu jika posisi dari indeks l dan indeks r bernilai sama maka akan mengembalikan nilai array pada indeks l.
5) Method totalDC() merupakan method untuk mengetahui jumlah dari seluruh elemen pada array dengan metode divide and conquer. Pada method totalDC solusi untuk menghitung jumlah seluruh elemen pada array dilakukan dengan membagi

proses perhitungan menjadi dua bagian, yaitu sisi kiri dan sisi kanan hingga mencapai basis rekursi, yaitu hingga posisi indeks kiri dan kanan bernilai sama sehingga menunjuk elemen pada indeks yang sama. Setelah itu menemukan hasil dari sisi kiri dan kanan, maka return penjumlahan dari hasil sisi kiri dan sisi kanan untuk mendapatkan hasil akhirnya.

6) Push dan commit ke github