

LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA
JOBSHEET 5



Rani Miftahus Saádah

244107020057

TI_1E

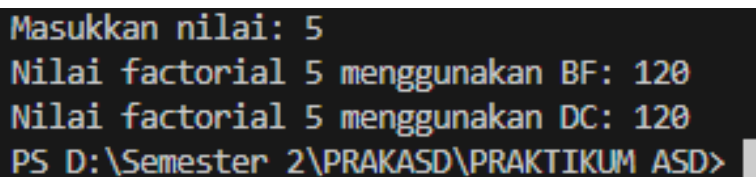
PROGRAM STUDI D_IV TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
PRAKTIKUM 2025

5.2.1 Faktorial

```
public class Faktorial25 {  
  
    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;  
        }  
    }  
}
```

```
import java.util.Scanner;  
  
public class MainFaktorial25 {  
    public static void main(String[] args) {  
        Scanner input = new Scanner(System.in);  
        Faktorial25 fk = new Faktorial25();  
  
        System.out.print("Masukkan nilai: ");  
        int nilai = input.nextInt();  
  
        System.out.println("Nilai factorial "+nilai+ " menggunakan BF:  
"+ fk.faktorialBF(nilai));  
        System.out.println("Nilai factorial "+nilai+ " menggunakan DC:  
"+ fk.faktorialDC(nilai));  
    }  
}
```

5.2.2 Verifikasi hasil



```
Masukkan nilai: 5  
Nilai factorial 5 menggunakan BF: 120  
Nilai factorial 5 menggunakan DC: 120  
PS D:\Semester 2\PRAKASD\PRAKTIKUM_ASD>
```

5.2.3 Pertanyaan

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!

If digunakan untuk menghentikan rekursi, sedangkan else digunakan untuk membagi masalah menjadi sub-masalah lebih kecil hingga mencapai base case.

2. . Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan for? Buktikan!

```
public class Faktorial25 {  
  
    int faktorialBF(int n){  
        int fakto = 1;  
        int i = 1;  
        while (i <= n) {  
            fakto = fakto * i;  
            i++;  
        }  
        return fakto;  
    }  
  
    int faktorialDC(int n){  
        if (n==1) {  
            return 1;  
        }else{  
            int fakto = n * faktorialDC(n-1);  
            return fakto;  
        }  
    }  
}
```

```
Masukkan nilai: 5  
Nilai factorial 5 menggunakan BF: 120  
Nilai factorial 5 menggunakan DC: 120  
PS D:\Semester 2\PRAKASD\PRAKTIKUM_ASD>
```

3. Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !

*fakto *= i; digunakan dalam metode iteratif (Brute Force) dengan loop, sedangkan int fakto = n * faktorialDC(n-1); digunakan dalam metode rekursif (Divide and Conquer) untuk memecah masalah menjadi sub-masalah lebih kecil.*

4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!

faktorialBF()	faktorialDC()
menggunakan loop untuk menghitung faktorial secara langsung dengan mengalikan angka dari 1 hingga n.	menggunakan pemanggilan rekursif, memecah perhitungan faktorial menjadi sub-masalah yang lebih kecil hingga mencapai base case.
lebih cepat karena hanya menggunakan perulangan tanpa ada overhead rekursi.	lebih lambat karena memerlukan banyak pemanggilan fungsi rekursif,

	yang meningkatkan overhead komputasi.
jika menginginkan efisiensi dan tidak ada batasan penggunaan loop.	jika ingin memanfaatkan konsep rekursi atau bekerja dengan masalah yang lebih kompleks dan terstruktur secara rekursif.

5.3.1 Pangkat

```

public class Pangkat25 {
    int nilai, pangkat;

    Pangkat25(int n, int p){
        nilai = n;
        pangkat = p;
    }
    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));
            }
        }
    }
}

```

```

import java.util.Scanner;

public class MainPangkat25 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan jumlah elemen: ");
        int elemen = input.nextInt();

        Pangkat25[] png = new Pangkat25[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 Pangkat25(basis, pangkat);
        }
        System.out.println("=== HASIL PANGKAT BRUTEFORCE ===");
        for (Pangkat25 p : png) {
            System.out.println(p.nilai + "^" + p.pangkat + ": " +
p.pangkatBF(p.nilai, p.pangkat));
        }
        System.out.println("=== HASL PANGKAT DIVIDE AND CONQUER ===");
        for (Pangkat25 p : png) {
            System.out.println(p.nilai + "^" + p.pangkat + ": " +
p.pangkatDC(p.nilai, p.pangkat));
        }
    }
}

```

5.3.2 Verifikasi hasil

```

Masukkan jumlah elemen: 3
Masukkan nilai basis elemen ke-1: 2
Masukkan nilai pangkat elemen ke-1: 3
Masukkan nilai basis elemen ke-2: 4
Masukkan nilai pangkat elemen ke-2: 5
Masukkan nilai basis elemen ke-3: 6
Masukkan nilai pangkat elemen ke-3: 7
=== HASIL PANGKAT BRUTEFORCE ===
2^3: 8
4^5: 1024
6^7: 279936
=== HASL PANGKAT DIVIDE AND CONQUER ===2^3: 8
4^5: 1024
6^7: 279936
PS D:\Semester 2\PRAKASD\PRAKTIKUM_ASD>

```

5.3.3 Pertanyaan

1. . Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!

pada method pangkatBF() menggunakan loop untuk menghitung pangkat secara iterativ sedangkan untuk method pangkatDC() yang menerapkan divide and conquer menggunakan rekursi.

2. Apakah tahap combine sudah termasuk dalam kode tersebut? Tunjukkan!

sudah, tahap combine ditunjukkan oleh sintaks berikut:

```
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));
        }
    }
}
```

3. Pada method pangkatBF()terdapat parameter untuk melewati nilai yang akan dipangkatkan dan pangkat berapa, padahal di sisi lain di class Pangkat telah ada atribut nilai dan pangkat, apakah menurut Anda method tersebut tetap relevan untuk memiliki parameter? Apakah bisa jika method tersebut dibuat dengan tanpa parameter? Jika bisa, seperti apa method pangkatBF() yang tanpa parameter?

pangkatBF() masih relevan memiliki parameter, terutama jika ingin memungkinkan pemanggilan metode dengan nilai yang berbeda setiap kali. ini bisa menghitung pangkat tanpa harus bergantung pada atribut dalam class Pangkat.

Namun, jika atribut nilai dan pangkat sudah ada dalam class, parameter a dan b bisa dihilangkan dalam pangkatBF(), sehingga metode hanya menggunakan nilai yang sudah disimpan dalam objek.

```

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

```

4. Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!

pangkatBF()	pangkatDC()
Menggunakan perulangan untuk mengalikan a sebanyak b kali.	Membagi masalah menjadi sub-masalah lebih kecil menggunakan rekursi.
Tidak membagi masalah, langsung iterasi.	Membagi masalah menjadi dua bagian (b/2).
Lambat jika b besar karena membutuhkan b iterasi.	Menggunakan lebih banyak memori, karena menyimpan call stack rekursi.

5.4.1 Sum

```

public class Sum25 {
    double keuntungan[];

    Sum25(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;
    }
}

```

```
import java.util.Scanner;

public class MainSum25 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan jumlah elemen: ");
        int elemen = input.nextInt();

        Sum25 sm = new Sum25(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));
    }
}
```

5.4.2 Verifikasi hasil

```
Masukkan jumlah elemen: 5
Masukkan keuntungan ke-1: 10
Masukkan keuntungan ke-2: 20
Masukkan keuntungan ke-3: 30
Masukkan keuntungan ke-4: 40
Masukkan keuntungan ke-5: 50
Total keuntungan menggunakan Bruteforce: 150.0
Total keuntungan menggunakan Divide and Conquer: 150.0
PS D:\Semester 2\PRAKASD\PRAKTIKUM_ASD>
```

5.4.3 Pertanyaan

1. Kenapa dibutuhkan variable mid pada method TotalDC()?

Variabel mid pada metode totalDC() dibutuhkan untuk menentukan titik tengah dari array yang sedang di proses. Tujuannya adalah membagi array menjadi dua bagian, sehingga algoritma Divide and Conquer bisa bekerja dengan membagi masalah menjadi sub-masalah yang lebih kecil.

2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?

```
double lsum = totalDC(arr, l, mid);
double rsum = totalDC(arr, mid+1, r);
```

Digunakan untuk membagi array menjadi dua bagian dan menghitung totalnya secara rekursif. Dengan pendekatan ini, algoritma dapat bekerja lebih efisien dibandingkan metode Brute Force, terutama untuk array yang sangat besar.

3. . Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?

```
return lsum+rsum;
```

Diperlukan karena dalam metode Divide and Conquer, permasalahan dipecah menjadi dua bagian yang lebih kecil (sub-masalah), lalu hasil dari masing-masing bagian harus digabungkan kembali untuk mendapatkan hasil akhir.

4. Apakah base case dari totalDC()?

Base case dari totalDC() adalah ketika $l == r$, yaitu ketika hanya satu elemen tersisa dalam array, sehingga rekursi berhenti dan nilai elemen tersebut langsung dikembalikan.

5. Tarik Kesimpulan tentang cara kerja totalDC()

Fungsi totalDC() bekerja dengan cara membagi array menjadi dua bagian secara rekursif, menghitung jumlahnya secara terpisah, lalu menggabungkan kembali hasilnya untuk mendapatkan total keseluruhan elemen dalam array.