

**Laporan Hasil Praktikum
Algoritma dan Struktur Data
Jobsheet 5**



**Muhammad Firman Aditiasmara
244107020094
TI-1E
Program Studi Teknik Informatika
Jurusan Teknologi Informasi
Politeknik Negeri malang
2025**

1. Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Mahasiswa mampu membuat algoritma bruteforce dan divide-conquer
2. Mahasiswa mampu menerapkan penggunaan algoritma bruteforce dan divide-conquer

2. Praktikum

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

2.1.1 Langkah-langkah Percobaan

1. Membuat class Faktorial dan melengkapi dengan atribut dan method

```
package Jobsheet5;

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;
        }

    }
}
```

2. Membuat class baru MainFaktorial dengan penambahan scanner dan pemanggilan method

```

package Jobsheet5;

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));

    }
}

```

2.1.2 Verifikasi Hasil Percobaan

```

Masukkan nilai: 5
Nilai faktorial 5menggunakan BF: 120
Nilai faktorial 5menggunakan DC: 120

```

Hasil percobaan telah sesuai dengan ketentuan

2.1.3 Pertanyaan

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

Jawab : Bagian kode if berfungsi sebagai base case yaitu kondisi ketika rekursi berhenti. Sedangkan kode else berfungsi sebagai rekursif case, yang berarti akan terus memanggil dirinya sendiri hingga mencapai base case

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

Jawab : Pada method faktorialBF() mungkin diubah menggunakan perulangan while ataupun do-while loop

- a. Perulangan while:

```

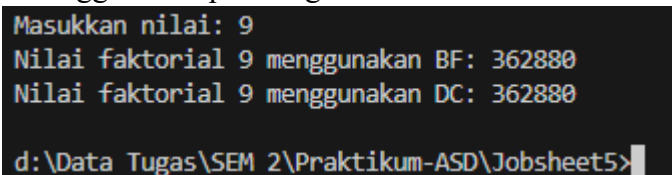
int faktorialBF(int n) {
    int fakto = 1;
    int i = 1;
    while (i <= n) {
        fakto = fakto * i;
        i++;
    }
    return fakto;
}

```

b. Perulangan do-while

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

Hasil menggunakan perulangan while dan do-while



```
Masukkan nilai: 9  
Nilai faktorial 9 menggunakan BF: 362880  
Nilai faktorial 9 menggunakan DC: 362880  
d:\Data Tugas\SEM 2\Praktikum-ASD\Jobsheet5>
```

3. Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !
Jawab : Kode fakto *=1 biasanya digunakan dalam metode **Brute Force**, karena nilai fakto akan diperbarui setiap iterasi.seperti dalam perulangan for, while. Jika fakto = n* faktorialDC(n-1) digunakan pada Divide Conquer karena nilai factorial dihitung dengan memanggil method faktorialDC secara rekursif hingga mencapai base case
4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()
Jawab : pada FaktorialBF menggunakan iterasi looping dengan for untuk menghitung factorial , dengan menngalikan angka 1 hingga n. untuk FaktorialDC menggunakan rekursif dengan memanggil dirinya sendiri hingga mencapai base case

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

2.2.1 Langkah-langkah Percobaan

1. Membuat class Pangkat dan membuat atribut untuk angka yang dipangkatkan dan pemangkatnya, serta menambahkan method PangkatDC()

```
package Jobsheet5;

public class Pangkat {

    int nilai, pangkat;

    Pangkat(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)*a);
            }
        }
    }
}
```

2. Membuat class MainPangkat dengan menambahkan scanner untuk input elemen, menambahkan instansiasi array dan menambahkan proses pengisian beberapa nilai yang akan dipangkatkan sekaligus dengan pemangkatnya.

```

package Jobsheet5;

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: ");
        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 BRUTERFORCE");
        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));
        }
        input.close();
    }
}

```

2.2.2 Verifikasi Hasil Percobaan

```

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 BRUTERFORCE
2^3: 8
4^5: 1024
6^7: 279936
HASIL PANGKAT DIVIDE AND CONQUER
2^3: 8
4^5: 16384
6^7: 279936

```

2.2.3 Pertanyaan

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

Pada method PangkatBF menggunakan perulangan for, dimulai dari hasil = 1, lalu mengalikan a sebanyak n kali

Sedangkan pada PangkatDC menggunakan rekursif, jika sudah sampai n=1 maka akan langsung mengembalikan nilai a, jika n lebih dari 1, maka n dibagi dua dan dipanggil secara rekursif

```
for (int i = 0; i < 3; i++) {  
    System.out.println("Data  
Mahasiswa ke-" + (i + 1));  
    arrayOfMahasiswa18[i].cetakInfo()  
;  
}
```

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

Jawab : Tahap combine sudah ada pada kode berikut

```
return (pangkatDC(a, n / 2) * pangkatDC(a, n /  
2) * a);
```

Penggabungan tersebut merupakan hasil pemecahan pada saat proses divide conquer

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?

Jawab : Tetap relevan jika digunakan untuk pemanggilan dari luar kelas, namun juga bisa tanpa parameter dan akan seperti berikut

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

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

Pada method pangkatBF() menggunakan bruteforce dengan cara melakukan perulangan sebanyak n, pada setiap iterasi, hasil dikalikan dengan n. Jika PangkatDC() menggunakan rekursi

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

2.3.1 Langkah-langkah Percobaan

1. Membuat class baru bernama sum, membuat method TotalBF() untuk menghitung total nilai array dengan iterative, dan membuat method TotalDC() untuk perhitungan nilai total array menggunakan algoritma Divide and Conquer

```
package Jobsheet5;

public class Sum {

    double keuntungan[];

    Sum(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;
    }

}
```

2. Membuat class baru MainSum untuk menuliskan berapa bulan keuntungan yang dihitung, dan membuat instansiasi objek untuk memanggil atribut atau fungsi pada class Sum


```

package Jobsheet5;

import java.util.Scanner;

public class MainSum {

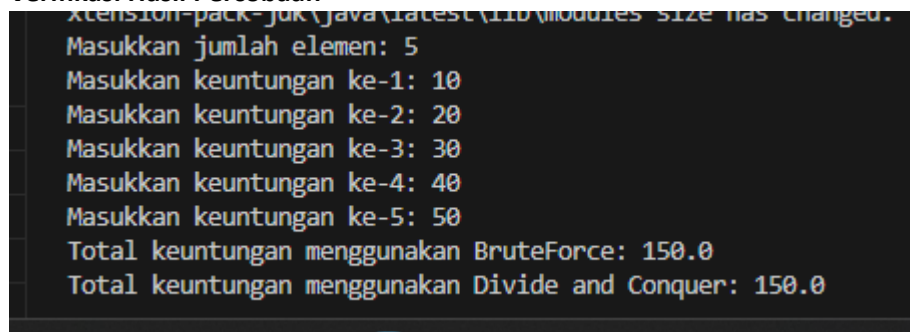
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan jumlah elemen: ");
        int elemen = input.nextInt();

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

}

```

2.3.2 Verifikasi Hasil Percobaan



```

xtension-pack-jdk\java\latest\lib\modules\size has changed.
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

```

2.3.3 Pertanyaan

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

Jawab :

Variable mid diperlukan untuk membagi array menjadi dua bagian yang lebih kecil sehingga algoritma bisa bekerja secara rekursif hingga menyelesaikan perhitungan jumlah elemen dalam array.

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

```

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

```

Jawab :

totalDC(arr, l, mid) berperan Memproses dan menghitung total elemen dari bagian kiri array, yaitu dari indeks l hingga mid.
Sedangkan totalDC(arr, mid, r) berfungsi untuk menghitung total elemen dari bagian kanan array, indeks mid + 1 hingga r

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

```
return lsum+rsum;
```

Jawab :

Pernyataan `return lsum + rsum;` diperlukan untuk menggabungkan hasil perhitungan dua bagian array yang telah dipecah

4. Apakah base case dari totalDC()?

Jawab :

Base case dari fungsi totalDC() adalah Ketika memenuhi if ($l == r$) maka akan dilakukan `return arr[l]`

5. Tarik Kesimpulan tentang cara kerja totalDC()

Jawab :

Fungsi totalDC() bekerja menggunakan metode Divide and Conquer. Dengan base case yaitu ($l == r$). array pada method ini dibagi dua dengan nilai Tengah berupa mid. Setiap bagian dibagi terus hingga mencapai base case. Lsum dan rsum digabungkan untuk menghasilkan total keseluruhan


```

import java.util.Scanner;

public class DosenDemol8 {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        Dosen18[] arrayOfDosen18 = new Dosen18[3];
        String kode, nama, dummy;
        boolean jenisKelamin;
        int usia;

        System.out.println("==== Input Data Dosen =====");
        for (int i = 0; i < 3; i++) {
            System.out.println("Masukkan Data Dosen ke-" + (i +
1));
            System.out.print("Kode                                : ");
            kode = input.nextLine();
            System.out.print("Nama                                : ");
            nama = input.nextLine();
            System.out.print("Jenis Kelamin (L/P)   : ");
            dummy = input.nextLine();
            jenisKelamin = dummy.equalsIgnoreCase("L") ? true :
false;
            System.out.print("Usia                                : ");
            dummy = input.nextLine();
            usia = Integer.parseInt(dummy);
            System.out.println("-----");

            arrayOfDosen18[i] = new Dosen18(kode, nama,
jenisKelamin, usia);
        }
        int data = 0;
        for (Dosen18 dosen : arrayOfDosen18) {
            System.out.println("Data Dosen ke-" + data++);
            System.out.println("Kode                : " + dosen.kode);
            System.out.println("Nama                : " + dosen.nama);
            System.out.println("Jenis Kelamin : " +
(dosen.jenisKelamin ? "Pria" : "Wanita"));
            System.out.println("Usia                : " + dosen.usia);
            System.out.println("-----");
            data++;
        }

        DataDosen18 dataDosen18 = new DataDosen18();
        dataDosen18.dataSemuaDosen(arrayOfDosen18);
        dataDosen18.jumlahDosenPerJenisKelamin(arrayOfDosen18);
        dataDosen18.rerataUsiaDosenPerJenisKelamin(arrayOfDosen18);
        dataDosen18.infoDosenPalingTua(arrayOfDosen18);
    }
}

```

