

# **LAPORAN HASIL PRAKTIKUM BASIS DATA**

## **Jobsheet 5**



**Adi Luhung**

**244107020088**

**Kelas 1E**

**Program Studi Teknik Informatika**

**Jurusan Teknologi Informasi**

**Politeknik Negeri Malang**

**2025**

## 1. Praktikum

### 1.1 Percobaan 1

1. Kode program untuk menghitung faktorial menggunakan Brute Force dan Divide Qonquer.

```
package Minggu5;
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;
        }
    }
}
```

```
package Minggu5;
import java.util.Scanner;
public class MainFaktorial {
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);
        System.out.print("Masukkan nilai: ");
        int nilai = sc.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. Hasil kode program.

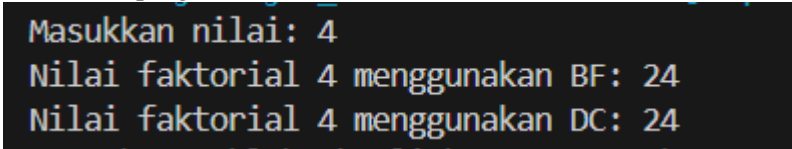
```
Masukkan nilai: 5
Nilai faktorial 5 menggunakan BF: 120
Nilai faktorial 5 menggunakan DC: 120
```

### 1.1.1 Pertanyaan

1. Pada kode if berfungsi sebagai pembatas perulangan yang disebut *base case*, sedangkan kode else berfungsi sebagai perulangan atau *recursive case*.
2. Memodifikasi kode program agar dapat menggunakan selain perulangan for.

```
package Minggu5;
public class Faktorial {
    int faktorialBF (int n) {
        int fakto = 1;
        int i = 1;
        do {
            fakto = fakto * i;
            i++;
        } while (i <= n);
        return fakto;
    }
    int faktorialDC (int n) {
        if (n == 1) {
            return 1;
        } else {
            int fakto = n * faktorialDC (n-1);
            return fakto;
        }
    }
}
```

Hasil kode program.



```
Masukkan nilai: 4
Nilai faktorial 4 menggunakan BF: 24
Nilai faktorial 4 menggunakan DC: 24
```

3. Perbedaan fakto \*= i dan fakto = n \* faktorialDC (n-1)
  - fakto \*= i  
menggunakan cara perulangan atau bisa disebut brute force
  - fakto = n \* faktorialDC (n-1)  
menggunakan rekursi dengan cara memanggil fungsi itu sendiri berkali-kali.
4. Perbedaan cara kerja fakto \*= i dan fakto = n \* faktorialDC (n-1)
  - fakto \*= i  
cara ini menggunakan perulangan dan menghitung faktorial secara langsung dalam urutan maju
  - fakto = n \* faktorialDC (n-1)  
menggunakan rekursi untuk memecah masalah menjadi submasalah kecil hingga mencapai basis kasus, lalu menggabungkan hasilnya kembali.

## 1.2 Percobaan 2

1. Membuat kode program untuk mengoperasikan perhitungan dengan pangkat menggunakan Brute Force dan Divide Qonquer.

```
package Minggu5;
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));
            }
        }
    }
}
```

```
package Minggu5;
import java.util.Scanner;
public class MainPangkat {
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);
        System.out.print("Masukkan jumlah elemen: ");
        int elemen = sc.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 = sc.nextInt();
            System.out.print("Masukkan nilai basis pangkat ke-" + (i+1)+ ": ");
            int pangkat = sc.nextInt();
            png [i] = new Pangkat(basis, pangkat);
        }

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

## 2. Hasil kode program

```
Masukkan jumlah elemen: 3
Masukkan nilai basis elemen ke-1: 2
Masukkan nilai basis pangkat ke-1: 3
Masukkan nilai basis elemen ke-2: 4
Masukkan nilai basis pangkat ke-2: 5
Masukkan nilai basis elemen ke-3: 6
Masukkan nilai basis pangkat ke-3: 7
HASIL PANGKAT BRUTE FORCE:
2^3: 8
4^5: 1024
6^7: 279936
HASIL PANGKAT DIVIDE QONQUER:
2^3: 8
4^5: 1024
6^7: 279936
```

### 1.2.1 Pertanyaan

1. Method pangkatBF(), dibuat untuk melakukan perhitungan pangkat dengan menggunakan Brute Force (loop), sedangkan method pangkatDC() menggunakan Divide Qonquer (rekursif).
2. Tahap combine sudah termasuk dalam kode tersebut, yaitu pada kode

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

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

3. Iya, bisa.

```
package BruteForceDivideQonquer.Minggu5;
import java.util.Scanner;
public class MainPangkat01 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Masukkan jumlah elemen: ");
        int elemen = sc.nextInt();
        Pangkat01[] png = new Pangkat01[elemen];

        for (int i = 0; i < elemen; i++) {
            System.out.print("Masukkan nilai basis elemen ke-" + (i + 1) + ": ");
            int basis = sc.nextInt();
            System.out.print("Masukkan nilai pangkat ke-" + (i + 1) + ": ");
            int pangkat = sc.nextInt();
            png[i] = new Pangkat01(basis, pangkat);
        }
        System.out.println("HASIL PANGKAT BRUTE FORCE:");
        for (Pangkat01 p : png) {
            System.out.println(p.nilai + "^" + p.pangkat + ": " + p.pangkatBF());
        }
        System.out.println("HASIL PANGKAT DIVIDE & CONQUER:");
        for (Pangkat01 p : png) {
            System.out.println(p.nilai + "^" + p.pangkat + ": " + p.pangkatDC());
        }
        sc.close();
    }
}
```

```

package BruteForceDivideQonquer.Minggu5;
public class Pangkat01 {
    int nilai, pangkat;

    Pangkat01(int n, int p) {
        nilai = n;
        pangkat = p;
    }
    int pangkatBF() {
        int hasil = 1;
        for (int i = 0; i < pangkat; i++) {
            hasil *= nilai;
        }
        return hasil;
    }
    int pangkatDC() {
        return hitungPangkatDC(pangkat);
    }
    private int hitungPangkatDC(int n) {
        if (n == 1) {
            return nilai;
        } else {
            int hasil = hitungPangkatDC(n / 2);
            if (n % 2 == 1) {
                return hasil * hasil * nilai;
            } else {
                return hasil * hasil;
            }
        }
    }
}

```

#### 4. Cara kerja method `pangkatBF()`, dan `pangkatDC()`.

- `pangkatBF()`  
Mengalikan nilai variabel `nilai` sebanyak nilai didalam variabel `pangkat` dengan menggunakan loop.
- `pangkatDC()`  
Memecah variabel `pangkat` menjadi setengah ( $n/2$ ) dan menggabungkan hasilnya secara rekursif

### 1.3 Percobaan 3

#### 1. Menghitung total keuntungan menggunakan Brute Force dan Divide Qonquer

```
package Minggu5;
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;
    }
}
```

```
package Minggu5;
import java.util.Scanner;
public class MainSum {
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);
        System.out.print("Masukkan jumlah elemen: ");
        int elemen = sc.nextInt();

        Sum sm = new Sum(elemen);
        for (int i = 0; i < elemen; i++) {
            System.out.print("Masukan keuntungan ke-" + (i+1)+ ": ");
            sm.keuntungan[i] = sc.nextDouble();
        }
        System.out.println("Total keuntungan menggunakan Brute Force: " +
sm.totalBF());
        System.out.println("Total keuntungan menggunakan Divide Qonquer: " +
sm.totalDC(sm.keuntungan, 0, elemen-1));
    }
}
```

## 2. Hasil kode program

```
Masukkan jumlah elemen: 5
Masukan keuntungan ke-1: 10
Masukan keuntungan ke-2: 20
Masukan keuntungan ke-3: 30
Masukan keuntungan ke-4: 40
Masukan keuntungan ke-5: 50
Total keuntungan menggunakan Brute Force: 150.0
Total keuntungan menggunakan Divide Qonquer: 150.0
```

### 1.3.2 Pertanyaan

1. Variabel mid digunakan dalam metode totalDC() sebagai pembagi untuk membagi array menjadi dua bagian yang lebih kecil dalam pendekatan Divide and Conquer.
2. totalDC(arr, l, mid) untuk menghitung total keuntungan dari bagian kiri array. Sedangkan totalDC(arr, mid + 1, r) untuk menghitung total keuntungan dari bagian kanan array.
3. lsum + rsum adalah tahap combine dari masalah yang telah dipecah diawal proses.
4. Base case dari program diatas adalah

```
if (l == r) {
    return arr[l];
}
```

5. Pada awal proses, array dibagi menjadi dua bagian menggunakan mid. Bagian kiri dihitung dengan totalDC(arr, l, mid), bagian kanan dihitung dengan totalDC(arr, mid + 1, r). Kemudian jika hanya ada satu elemen dalam array (l == r), langsung dikembalikan sebagai base case. Setelah nilai dari dua bagian kecil dihitung, hasilnya dijumlahkan.
6. Push kode program ke github

```
PS C:\Users\luhun\Kuliah1 Semester2\Praktikum_ASD> git add .
PS C:\Users\luhun\Kuliah1 Semester2\Praktikum_ASD> git commit -m "Modifikasi"
[main 525ced4] Modifikasi
14 files changed, 145 insertions(+), 9 deletions(-)
create mode 100644 BruteForceDivideQonquer/Minggu5/Faktorial01.java
create mode 100644 BruteForceDivideQonquer/Minggu5/MainFaktorial01.java
create mode 100644 BruteForceDivideQonquer/Minggu5/MainPangkat01.java
create mode 100644 BruteForceDivideQonquer/Minggu5/MainSum.java
create mode 100644 BruteForceDivideQonquer/Minggu5/Pangkat01.java
create mode 100644 BruteForceDivideQonquer/Minggu5/Sum01.java
create mode 100644 BruteForceDivideQonquer/test.md
PS C:\Users\luhun\Kuliah1 Semester2\Praktikum_ASD> git push origin main
Enumerating objects: 29, done.
Counting objects: 100% (29/29), done.
Delta compression using up to 8 threads
Compressing objects: 100% (19/19), done.
```