

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



Disusun Oleh :

Nama : Fadhil Taufiqurrachman  
NIM : 244107020090  
Kelas : Teknik Informatika 1E

**Program Studi Teknik Informatika**  
**Jurusan Teknologi Informasi**  
**Politeknik Negeri Malang 2025**

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

### 5.2.1 Kode Program

Kode program pada class Faktorial08 :

```
package Jobsheet5;

public class Faktorial08 {
    public int nilai;

    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;
        }
    }
}
```

Kode program pada class MainFaktorial08 :

```
package Jobsheet5;
import java.util.Scanner;

public class MainFaktorial08 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan Nilai : ");
        int nilai = input.nextInt();
        input.nextLine();

        Faktorial08 fk = new Faktorial08();
        System.out.println("Nilai Faktorial " + nilai + " Menggunakan Brute Force : " + fk.faktorialBF(nilai));
        System.out.println("Nilai Faktorial " + nilai + " Menggunakan Divide And Conquer : " + fk.faktorialDC(nilai));
    }
}
```

### 5.2.2 Verifikasi

```
Masukkan Nilai : 5
Nilai Faktorial 5 Menggunakan Brute Force : 120
Nilai Faktorial 5 Menggunakan Divide And Conquer : 120
```

### 5.2.3 Pertanyaan

1.

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

Kode program tersebut merupakan fungsi/method rekursif, yaitu fungsi yang dapat memanggil (Mengcall) dirinya sendiri. Hal ini berkaitan dengan konsep Divide And Conquer, yaitu memecah masalah menjadi beberapa bagian sederhana lalu menyelesaikannya satu persatu. IF merupakan base case, sedangkan ELSE merupakan recursion call. IF bertindak sebagai Stopping Condition jika kasus sudah terbagi menjadi kasus terkecil dan akan mengembalikan nilai 1. Sedangkan ELSE menyediakan pengulangan yang dibutuhkan untuk menyederhanakan permasalahan. Bagian inilah yang sesuai dengan konteks divide and conquer, fase "conquer" (menyelesaikan sub-permasalahan secara rekursif) setelah fase "divide" (memecah masalah. Kemudian, hasil dari sub-permasalahan (faktorial n-1) dikombinasikan dengan n (fase "combine" secara implisit melalui perkalian) untuk mendapatkan solusi dari masalah awal.

2. Bisa dan memungkinkan, berikut adalah contohnya yaitu menggunakan perulangan While, dan hasilnya akan tetap sama sesuai yang seharusnya.

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

3. `fakto *= i;`. Kode program tersebut secara bertahap menghitung hasil faktorial dengan mengalikan hasil sebelumnya dengan angka berikutnya secara berurutan, hal ini sesuai dengan konsep BruteForce itu sendiri, yakni memproses satu per satu.  
`int fakto = n * faktorialDC(n-1);`. Kode program tersebut merupakan bagian dari implementasi rekursif untuk menghitung faktorial. Seperti penjelasan pada no 1, kode ini sesuai dengan konsep Divide And Conquer
4. Method `faktorialBF()`, merupakan method untuk menghitung nilai factorial menggunakan metode Brute Force, yakni dengan cara mengkalikan hasil sebelumnya dengan n secara satu per satu.  
Method `faktorialDC()`, merupakan method untuk menghitung nilai factorial menggunakan metode Divide and conquer, yakni dengan cara rekursif, memecah masalah menjadi bagian sederhana kemudian memprosesnya dan digabungkan kembali untuk menghasilkan jawaban dari masalah awal.

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

### 5.3.1 Kode Program

Kode program pada class Pangkat08 :

```
package Jobsheet5;

public class Pangkat08 {
    public int nilai, pangkat;

    Pangkat08() {

    }

    Pangkat08(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));
            }
        }
    }
}
```

### Kode program pada class MainPangkat08 :

```
package Jobsheet5;
import java.util.Scanner;

public class MainPangkat08 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan Jumlah Elemen : ");
        int elemen = input.nextInt();

        Pangkat08[] png = new Pangkat08[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 Pangkat08(basis, pangkat);
        }

        System.out.println("Hasil Pangkat Brute Force : ");
        for (Pangkat08 p : png) {
            System.out.println(p.nilai + "^" + p.pangkat + " : " +
p.pangkatBF(p.nilai, p.pangkat));
        }

        System.out.println("Hasil Pangkat Divide And Conquer : ");
        for (Pangkat08 p : png) {
            System.out.println(p.nilai + "^" + p.pangkat + " : " +
p.pangkatDC(p.nilai, p.pangkat));
        }
    }
}
```

### 5.3.2 Verifikasi

```
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 Brute Force :
2^3 : 8
4^5 : 1024
6^7 : 279936
Hasil Pangkat Divide And Conquer :
2^3 : 8
4^5 : 1024
6^7 : 279936
```

### 5.3.3 Pertanyaan

1. Perbedaan kedua method tersebut terletak pada konsep penyelesaiannya.

Method `pangkatBF()`, merupakan method menggunakan metode Brute Force, yakni dengan cara mengkalikan hasil sebelumnya dengan basis yang diinputkan secara satu per satu sejumlah bilangan pangkat.

Method `pangkatDC()`, merupakan method menggunakan metode Divide and conquer, yakni dengan cara rekursif, memecah masalah menjadi bagian sederhana kemudian memprosesnya, Jika bilangan pangkat masih genap, maka akan menghitung `pangkatDC(a, n/2)` dan mengalikannya dengan mengcall dirinya sendiri. Jika bilangan pangkat sudah ganjil, maka akan menghitung `pangkatDC(a, n/2)`, mengalikannya dengan dirinya sendiri, dan terakhir mengalikannya lagi dengan `a` dengan tujuan untuk berhenti memecah.

2. Sudah termasuk, yaitu pada kode program berikut.

```
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. Memiliki parameter tetaplah relevan, hasilnya akan tetap sama. Parameter berfungsi sebagai nilai yang disimpan sementara pada method. Jadi variabelnya hanya akan berfungsi pada method tersebut. Namun jika kita tidak ingin menggunakan parameter tetap bisa. Berikut adalah kode programnya.

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

4. Method `pangkatBF()` menggunakan metode Brute Force, yakni dengan cara mengkalikan basis yang diinputkan dengan basis itu sendiri, kemudian hasilnya akan dikalikan kembali dengan nilai basis lagi, secara satu per satu sejumlah bilangan pangkat.

Method `pangkatDC()` menggunakan metode Divide and conquer, yakni dengan cara rekursif, memecah masalah menjadi bagian sederhana kemudian memprosesnya, Jika bilangan pangkat masih genap, maka akan menghitung `pangkatDC(a, n/2)` dan mengalikannya dengan mengcall dirinya sendiri. Jika bilangan pangkat sudah ganjil, maka akan menghitung `pangkatDC(a, n/2)`, mengalikannya dengan dirinya sendiri, dan terakhir mengalikannya lagi dengan `a` dengan tujuan untuk berhenti memecah. Namun jika pengguna menginputkan nilai pangkat adalah 1, maka akan langsung menjalankan perintah `if` di awal dengan langsung mereturn nilai basis itu sendiri.



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

### 5.4.1 Kode Program

Kode program pada class Sum08 :

```
package Jobsheet5;

public class Sum08 {
    double keuntungan[];

    Sum08(int e1) {
        keuntungan = new double[e1];
    }

    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;
    }
}
```

Kode program pada class MainSum08 :

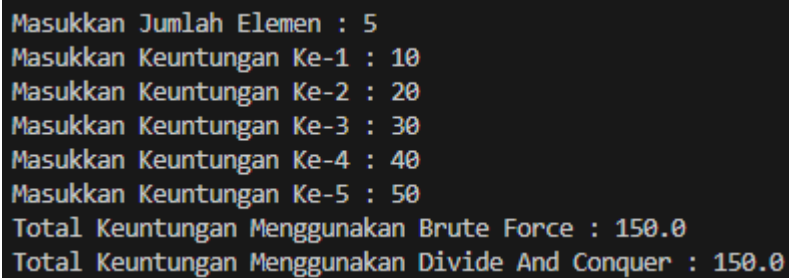
```
package Jobsheet5;
import java.util.Scanner;

public class MainSum08 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan Jumlah Elemen : ");
        int elemen = input.nextInt();

        Sum08 sm = new Sum08(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 Brute Force : "
+ sm.totalBF());
        System.out.println("Total Keuntungan Menggunakan Divide And
Conquer : " + sm.totalDC(sm.keuntungan, 0, elemen-1));
    }
}
```

#### 5.4.2 Verifikasi



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 Brute Force : 150.0  
Total Keuntungan Menggunakan Divide And Conquer : 150.0

#### 5.4.3 Pertanyaan

1. Karena variabel mid berguna untuk membagi array menjadi dua bagian yang lebih kecil, sehingga memungkinkan untuk melakukan pemecahan masalah secara rekursif. Hal ini akan menerapkan prinsip Divide and Conquer dengan benar.
2. `double lsum = totalDC(arr, l, mid);`, Memanggil rekursi untuk menghitung total keuntungan di bagian kiri array hingga mid.  
`double rsum = totalDC(arr, mid + 1, r);``totalDC(arr, mid + 1, r);`, Memanggil rekursi untuk menghitung total elemen di bagian kanan array, yaitu dari indeks mid + 1 hingga r.

3. Kode Program tersebut merupakan tahap combine, jadi hasil yang diperoleh dari penyelesaian pecahan sebelumnya, akan dijumlah untuk menghasilkan hasil akhir yang utuh dan benar.

4. Berikut ini adalah base case dari method `totalDC()`.

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

5. Cara kerjanya yaitu dengan cara membagi menjadi dua bagian menggunakan variabel `mid`. Masing-masing akan dihitung secara terpisah. Bagian kiri array hingga `mid` dihitung, kemudian bagian kanan array setelah `mid` akan dihitung. Setelah itu, hasil kedua tersebut akan dijumlahkan dan akan menghasilkan nilai yang seharusnya. Namun jika hanya terdapat satu elemen, maka akan langsung mereturn bilangan itu sendiri.