



---

**PERTEMUAN 5**  
**JOBSHEET 4 – BRUTE FORCE and DEVIDE CONQUER**

Untuk Memenuhi Salah Satu Tugas

Mata Kuliah Praktikum Algoritma Struktur Dasar  
Dosen: Ibu Rokhimatul Wakhidah, S.Pd., M.T.



Disusun oleh: Revani Nanda Putri (NIM: 2341760056)

**PROGRAM STUDI SISTEM INFORMASI BISNIS**  
**JURUSAN TEKNOLOGI INFORMASI**  
**POLITEKNIK NEGERI MALANG**  
**2024**

## JOBSHEET IV

### BRUTE FORCE DAN DIVIDE CONQUER

#### 4.1 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

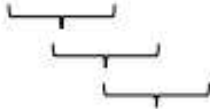
#### 4.2 Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

Perhatikan Diagram Class berikut ini :


Faktorial
nilai: int
faktorialBF(): int
faktorialDC(): int

Berdasarkan diagram class di atas, akan dibuat program class dalam Java. Untuk menghitung nilai faktorial suatu angka menggunakan 2 jenis algoritma, Brute Force dan Divide and Conquer. Jika digambarkan terdapat perbedaan proses perhitungan 2 jenis algoritma tersebut sebagai berikut :

Tahapan pencarian nilai faktorial dengan algoritma Brute Force :

$$20! = 20 \times 19 \times 18 \times 17 \times \dots \times 5 \times 4 \times 3 \times 2 \times 1$$


Tahapan pencarian nilai faktorial dengan algoritma Divide and Conquer :

$$20! = 20 \times 19 \times 18 \times 17 \times \dots \times 5 \times 4 \times 3 \times 2 \times 1$$


##### 4.2.1 Langkah-langkah Percobaan

1. Buat Project baru, dengan nama "BruteForceDivideConquer". Buat package dengan nama minggu5.
2. Buatlah class baru dengan nama **Faktorial**
3. Lengkapi class **Faktorial** dengan atribut dan method yang telah digambarkan di dalam diagram class di atas, sebagai berikut:
  - a) Tambahkan atribut nilai

```
public int nilai;
```

- b) Tambahkan method faktorialBF() nilai

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

- c) Tambahkan method faktorialDC() nilai

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

4. Coba jalankan (Run) class Faktorial dengan membuat class baru MainFaktorial.

- a) Di dalam fungsi main sediakan komunikasi dengan user untuk menginputkan jumlah angka yang akan dicari nilai faktorialnya

```
Scanner sc = new Scanner(System.in);
System.out.println("=====");
System.out.print("Masukkan jumlah elemen yang ingin dihitung : ");
int elemen = sc.nextInt();
```

- b) Buat Array of Objek pada fungsi main, kemudian inputkan beberapa nilai yang akan dihitung faktorialnya

```
Faktorial [] fk = new Faktorial[elemen];
for (int i = 0; i < elemen; i++) {
    fk[i] = new Faktorial();
    System.out.print("Masukkan nilai data ke-"+(i+1)+" : ");
    fk[i].nilai = sc.nextInt();
}
```

- c) Tampilkan hasil pemanggilan method faktorialDC() dan faktorialBF()

```
System.out.println("=====");
System.out.println("Hasil Faktorial dengan Brute Force");
for (int i = 0; i < elemen; i++) {
    System.out.println("Faktorial dari nilai "+fk[i].nilai+" adalah : "+fk[i].faktorialBF(fk[i].nilai));
}
System.out.println("=====");
System.out.println("Hasil Faktorial dengan Divide and Conquer");
for (int i = 0; i < elemen; i++) {
    System.out.println("Faktorial dari nilai "+fk[i].nilai+" adalah : "+fk[i].faktorialDC(fk[i].nilai));
}
System.out.println("=====");
```

d) Pastikan program sudah berjalan dengan baik!

#### 4.2.2 Verifikasi Hasil Percobaan

File Class Faktorial.java

```
1 package minggu5;
2
3 /**
4  * Faktorial
5  */
6 public class Faktorial {
7     public int nilai;
8
9     public int faktorialBF(int n) {
10         int fakto = 1;
11         for (int i = 1; i <= n; i++) {
12             fakto = fakto * i;
13         }
14         return fakto;
15     }
16
17     public int faktorialDC(int n) {
18         if (n == 1) {
19             return 1;
20         } else {
21             int fakto = n * faktorialDC(n - 1);
22             return fakto;
23         }
24     }
25 }
```

File MainFaktorial.java

```

1 package minggu5;
2
3 import java.util.Scanner;
4
5 public class MainFaktorial {
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         System.out.println("=====");
9         System.out.print("Masukkan jumlah elemen yang ingin dihitung: ");
10        int elemen = sc.nextInt();
11
12        Faktorial[] fk = new Faktorial[elemen];
13        for (int i = 0; i < elemen; i++) {
14            fk[i] = new Faktorial();
15            System.out.print("Masukkan nilai data ke-" + (i + 1) + ": ");
16            fk[i].nilai = sc.nextInt();
17        }
18
19        System.out.println("=====");
20        System.out.println("Hasil Faktorial dengan Brute Force");
21        for (int i = 0; i < elemen; i++) {
22            System.out.println("Faktorial dari nilai "+fk[i].nilai+" adalah : "+fk[i].faktorialBF(fk[i].nilai));
23        }
24
25        System.out.println("=====");
26        System.out.println("Hasil Faktorial dengan Divide and Conquer");
27        for (int i = 0; i < elemen; i++) {
28            System.out.println("Faktorial dari nilai "+fk[i].nilai+" adalah : "+fk[i].faktorialDC(fk[i].nilai));
29        }
30        System.out.println("=====");
31    }
32 }
33

```

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

```

run:
=====
Masukkan jumlah elemen yang ingin dihitung : 3
Masukkan nilai data ke-1 : 5
Masukkan nilai data ke-2 : 8
Masukkan nilai data ke-3 : 3
=====
Hasil Faktorial dengan Brute Force
Faktorial dari nilai 5 adalah : 120
Faktorial dari nilai 8 adalah : 40320
Faktorial dari nilai 3 adalah : 6
=====
Hasil Faktorial dengan Divide and Conquer
Faktorial dari nilai 5 adalah : 120
Faktorial dari nilai 8 adalah : 40320
Faktorial dari nilai 3 adalah : 6
=====
BUILD SUCCESSFUL (total time: 7 seconds)

```

```
Files\Java\jdk1.8.0_231\bin\java.exe' '-cp' 'C:\Users\LENOVO\AppData\Roaming\Code\User\workspaceStorage\11ee10a54d74d40911b624a05b5ef356\redhat.java\jdt_ws\Pertemuan5_5bc7abd7\bin' 'minggu5.MainFaktorial'
=====
Masukkan jumlah elemen yang ingin dihitung: 3
Masukkan nilai data ke-1: 5
Masukkan nilai data ke-2: 8
Masukkan nilai data ke-3: 3
=====
Hasil Faktorial dengan Brute Force
Faktorial dari nilai 5 adalah : 120
Faktorial dari nilai 8 adalah : 40320
Faktorial dari nilai 3 adalah : 6
=====
Hasil Faktorial dengan Divide and Conquer
Faktorial dari nilai 5 adalah : 120
Faktorial dari nilai 8 adalah : 40320
Faktorial dari nilai 3 adalah : 6
=====
PS D:\kampus\smt2\Algoritma Struktur Dasar\Praktek\Pertemuan5> []
```

### 4.2.3 Pertanyaan

1. Jelaskan mengenai base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial!
  - Algoritma divide conquer

ketika nilai n sama dengan 1. Faktorial 1 (1!) didefinisikan sebagai 1, sama seperti faktorial 0.



```
1 public int faktorialDC(int n) {
2     if (n == 1) {
3         return 1;
4     } else {
5         int fakto = n * faktorialDC(n - 1);
6         return fakto;
7     }
8 }
```

2. Pada implementasi Algoritma Divide and Conquer Faktorial apakah lengkap terdiri dari 3 tahapan divide, conquer, combine? Jelaskan masing-masing bagiannya pada kode program!

Jawab:

Tahapan divide and conquer tersebut terletak pada blok kode program berikut :

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

- Divide : Terjadi dalam faktorialDC(int n). Pada bagian ini, masalah besar (mencari faktorial dari n) dibagi menjadi submasalah yang lebih kecil dengan mengurangi nilai n satu per satu hingga mencapai kasus dasar ketika n sama dengan 1. Setiap panggilan rekursif pada faktorialDC(n - 1) mewakili proses pembagian ini.
- Conquer : Tahap ini ada pada blok else statement, karena ketika nilai n tidak sama dengan 1, hal tersebut merupakan tahap di mana submasalah diselesaikan secara rekursif.
- Combine : Dalam kode program tersebut, combine atau penggabungan solusi dari banyak sub-masalah terletak pada baris berikut:

```
int fakto = n * faktorialDC(n - 1);
return fakto;
```

Hasil dari pemanggilan rekursif faktorialDC(n - 1) (solusi submasalah yang lebih kecil) digunakan untuk mengalikan nilai n, sehingga menghasilkan solusi untuk masalah aslinya. Kemudian solusi submasalah digabungkan untuk membentuk solusi untuk masalah aslinya

3. Apakah memungkinkan perulangan pada method faktorialBF() dirubah selain menggunakan for?Buktikan!

Memungkinkan

```
int faktorialBF(int n) {
    if (n == 0 || n == 1) {
        return 1;
    } else {
        return n * faktorialBF(n - 1);
    }
}
```

faktorialBF() diimplementasikan dengan menggunakan rekursi. Jika nilai n adalah 0 atau 1, maka fungsi akan mengembalikan 1 sebagai nilai faktorialnya. Jika n lebih besar dari 1, maka fungsi akan

memanggil dirinya sendiri dengan parameter  $n - 1$  dan mengalikan hasilnya dengan  $n$ . Implementasi ini menghasilkan hasil yang sama dengan implementasi menggunakan perulangan `for`, namun dengan pendekatan yang berbeda

4. Tambahkan pengecekan waktu eksekusi kedua jenis method tersebut!

Untuk menambahkan pengecekan waktu eksekusi pada kedua jenis method, kita dapat menggunakan `System.nanoTime()`, sebelum dan sesudah pemanggilan method faktorisasi. Program tersebut berfungsi untuk mencatat waktu awal sebelum pemanggilan rekursif dan waktu akhir setelah rekursi selesai. Selisih kedua waktu tersebut memberikan waktu yang diperlukan untuk menjalankan method faktorisasi dalam satuan nanodetik.

```
BruteForceDivideConquer > minggu5 > Faktorial.java > Faktorial > faktorialDC(int)
1  package minggu5;
2
3  /**
4   * Faktorial
5   */
6  public class Faktorial {
7      public int nilai;
8
9      int faktorialBF(int n) {
10         long startTime = System.nanoTime();
11         if (n == 0 || n == 1) {
12             return 1;
13         } else {
14             int result = n * faktorialBF(n - 1);
15             long endTime = System.nanoTime();
16             System.out.println("waktu yang diperlukan untuk fBF(" + n + "): " + (endTime - startTime));
17             return result;
18         }
19     }
20
21     public int faktorialDC(int n) {
22         long startTime = System.nanoTime();
23         if (n == 1) {
24             return 1;
25         } else {
26             long endTime = System.nanoTime();
27             int fakto = n * faktorialDC(n - 1);
28             System.out.println("waktu yang diperlukan untuk fBF(" + n + "): " + (endTime - startTime));
29             return fakto;
30         }
31     }
32 }
```



5. Buktikan dengan inputan elemen yang di atas 20 angka, apakah ada perbedaan waktu eksekusi?

```
PS D:\kampus\smt2\Algoritma Struktur Dasar\Praktek\Pertemuan5> & 'C:\Program Files\Java\jdk1.8.0_231\bin\java.exe'
dhat_java\jdt_ws\Pertemuan5_5bc7abd7\bin' 'minggu5.MainFaktorial'
=====
Masukkan jumlah elemen yang ingin dihitung: 2
Masukkan nilai data ke-1: 3
Masukkan nilai data ke-2: 2
=====
Hasil Faktorial dengan Brute Force
waktu yang diperlukan untuk fBF(2): 700
waktu yang diperlukan untuk fBF(3): 563300
Faktorial dari nilai 3 adalah : 6
waktu yang diperlukan untuk fBF(2): 9300
Faktorial dari nilai 2 adalah : 2
=====
Hasil Faktorial dengan Divide and Conquer
waktu yang diperlukan untuk fBF(2): 200
waktu yang diperlukan untuk fBF(3): 8800
Faktorial dari nilai 3 adalah : 6
waktu yang diperlukan untuk fBF(2): 400
Faktorial dari nilai 2 adalah : 2
=====
PS D:\kampus\smt2\Algoritma Struktur Dasar\Praktek\Pertemuan5> |
```

#### 4.3 Menghitung Hasil Pangkat dengan Algoritma Brute Force dan Divide and Conquer

Pada praktikum ini kita akan membuat program class dalam Java. Untuk menghitung nilai pangkat suatu angka menggunakan 2 jenis algoritma, Brute Force dan Divide and Conquer.

#### 4.3.1 Langkah-langkah Percobaan

1. Di dalam paket `minggu5`, buatlah class baru dengan nama `Pangkat`. Dan di dalam class `Pangkat` tersebut, buat atribut angka yang akan dipangkatkan sekaligus dengan angka pemangkatnya

```
public int nilai, pangkat;
```

2. Pada class Pangkat tersebut, tambahkan method `PangkatBF()`

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

3. Pada class Pangkat juga tambahkan method `PangkatDC()`

```
public int pangkatDC(int a,int n){
    if (n==0) {
        return 1;
    }
    else
    {
        if(n%2==1)//bilangan ganjil
            return (pangkatDC(a,n/2)*pangkatDC(a,n/2)*a);
        else//bilangan genap
            return (pangkatDC(a,n/2)*pangkatDC(a,n/2));
    }
}
```

4. Perhatikan apakah sudah tidak ada kesalahan yang muncul dalam pembuatan class Pangkat
5. Selanjutnya buat class baru yang di dalamnya terdapat method main. Class tersebut dapat dinamakan MainPangkat. Tambahkan kode pada class main untuk menginputkan jumlah nilai yang akan dihitung pangkatnya.

```
Scanner sc = new Scanner(System.in);
System.out.println("=====");
System.out.print("Masukkan jumlah elemen yang ingin dihitung : ");
int elemen = sc.nextInt();
```

6. Nilai pada tahap 5 selanjutnya digunakan untuk instansiasi array of objek. Di dalam Kode berikut ditambahkan proses pengisian beberapa nilai yang akan dipangkatkan sekaligus dengan pemangkatnya.

```
Pangkat [] png = new Pangkat[elemen];

for (int i = 0; i < elemen; i++) {
    png[i] = new Pangkat();
    System.out.print("Masukkan nilai yang akan dipangkatkan ke-"+(i+1)+" : ");
    png[i].nilai = sc.nextInt();
    System.out.print("Masukkan nilai pemangkat ke-"+(i+1)+" : ");
    png[i].pangkat = sc.nextInt();
}
```

7. Kemudian, panggil hasil nya dengan mengeluarkan return value dari method PangkatBF() dan PangkatDC().

```
System.out.println("=====");
System.out.println("Hasil Pangkat dengan Brute Force");
for (int i = 0; i < elemen; i++) {
    System.out.println("Nilai "+png[i].nilai+" pangkat "+png[i].pangkat+" adalah : "+png[i].pangkatBF(png[i].nilai,png[i].pangkat));
}
System.out.println("=====");
System.out.println("Hasil Pangkat dengan Divide and Conquer");
for (int i = 0; i < elemen; i++) {
    System.out.println("Nilai "+png[i].nilai+" pangkat "+png[i].pangkat+" adalah : "+png[i].pangkatDC(png[i].nilai,png[i].pangkat));
}
System.out.println("=====");
```

```

1  package minggu5;
2
3  import java.util.Scanner;
4
5  public class MainPangkat {
6      public static void main(String[] args) {
7          Scanner sc = new Scanner(System.in);
8          System.out.println("-----");
9          System.out.print("Masukkan jumlah elemen: ");
10         int elemen = sc.nextInt();
11
12         Pangkat[] png = new Pangkat[elemen];
13         for (int i = 0; i < elemen; i++) {
14             png[i] = new Pangkat();
15             System.out.print("Masukkan nilai yang hendak dipangkatkan: ");
16             png[i].nilai = sc.nextInt();
17             System.out.print("Masukkan nilai pemangkat: ");
18             png[i].pangkat = sc.nextInt();
19         }
20
21         System.out.println("HASIL - BRUTE FORCE");
22         for (int i = 0; i < elemen; i++) {
23             System.out.println("Hasil dari "
24                 + png[i].nilai+ " pangkat "
25                 + png[i].pangkat+ " adalah "
26                 + png[i].pangkatBF(png[i].nilai, png[i].pangkat));
27         }
28
29         System.out.println("HASIL - DIVIDE AND CONQUER");
30         for (int i = 0; i < elemen; i++) {
31             System.out.println("Hasil dari "
32                 + png[i].nilai+ " pangkat "
33                 + png[i].pangkat+ " adalah "
34                 + png[i].pangkatDC(png[i].nilai, png[i].pangkat));
35         }
36
37         sc.close();
38     }
39 }
40

```

```

1  package minggu5;
2
3  public class Pangkat {
4
5      public int nilai, pangkat;
6
7      int pangkatBF(int a, int n) {
8          int hasil = 1;
9          for (int i = 0; i < n; i++) {
10             hasil *= a;
11         }
12         return hasil;
13     }
14
15     int pangkatDC(int a, int n) {
16         if(n==1) {
17             return a;
18         } else {
19             if(n%2==1) {
20                 return (pangkatDC(a, n/2) * pangkatDC(a, n/2) * a);
21             } else {
22                 return (pangkatDC(a, n/2) * pangkatDC(a, n/2));
23             }
24         }
25     }
26 }
27

```

#### 4.3.2 Verifikasi Hasil Percobaan

Pastikan output yang ditampilkan sudah benar seperti di bawah ini.

```

run:
=====
Masukkan jumlah elemen yang ingin dihitung : 2
Masukkan nilai yang akan dipangkatkan ke-1 : 6
Masukkan nilai pemangkat ke-1 : 2
Masukkan nilai yang akan dipangkatkan ke-2 : 4
Masukkan nilai pemangkat ke-2 : 3
=====
Hasil Pangkat dengan Brute Force
Nilai 6 pangkat 2 adalah : 36
Nilai 4 pangkat 3 adalah : 64
=====
Hasil Pangkat dengan Divide and Conquer
Nilai 6 pangkat 2 adalah : 36
Nilai 4 pangkat 3 adalah : 64
=====
BUILD SUCCESSFUL (total time: 10 seconds)

```

```
Pertemuan5_SDC/abd/\\bin minggus.MainPangkat
-----
Masukkan jumlah elemen: 2
Masukkan nilai yang hendak dipangkatkan: 6
Masukkan nilai pemangkat: 2
Masukkan nilai yang hendak dipangkatkan: 4
Masukkan nilai pemangkat: 3
HASIL - BRUTE FORCE
Hasil dari 6 pangkat 2 adalah 36
Hasil dari 4 pangkat 3 adalah 64
HASIL - DIVIDE AND CONQUER
Hasil dari 6 pangkat 2 adalah 36
Hasil dari 4 pangkat 3 adalah 64
PS D:\kampus\smt2\Algoritma Struktur Dasar\Praktek\Pertemuan5>
```

#### 4.3.3 Pertanyaan

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu `PangkatBF()` dan `PangkatDC()` !

Jawaban: `PangkatBF()` melakukan perulangan iteratif, sedangkan `PangkatDC()` menggunakan rekursif

2. Pada method `PangkatDC()` terdapat potongan program sebagai berikut:

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

Jelaskan arti potongan kode tersebut

Jawaban:

Kode tersebut merupakan bagian dari method `pangkatDC` jika bernilai, yang berarti jika  $n$  modulo (hasil sisa pembagian)  $2 == 1$ , bilangan tersebut merupakan bilangan ganjil dan akan mengembalikan nilai dari pemanggilan rekursif ke method `pangkatDC` dengan membagi  $n$  menjadi dua dan mengalikan hasilnya dengan dirinya sendiri serta dengan bilangan  $a$ . Lalu kondisi jika  $n$  adalah bilangan genap, maka algoritma akan melakukan pemanggilan rekursif ke method `pangkatDC` dengan membagi  $n$  menjadi dua dan mengalikan hasilnya dengan dirinya sendiri. Karena pada case ini, pangkat  $n$  bisa dianggap sebagai hasil perkalian dua pangkat  $n/2$ .

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

Ya, dalam algoritma Divide and Conquer, tahap *combine* terjadi pada saat hasil dari submasalah yang lebih kecil digabungkan untuk menghasilkan solusi dari masalah asli. Tahap *combine* terjadi pada kondisi jika  $n$  bilangan ganjil :

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

Pada kode tersebut, hasil dari kedua pemanggilan rekursif `pangkatDC(a, n/2)` digabungkan dengan mengalikan kembali hasilnya dengan  $a$ . Selanjutnya pada kondisi jika  $n$  adalah bilangan genap, berikut adalah kodenya :

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

4. Modifikasi kode program tersebut, anggap proses pengisian atribut dilakukan dengan konstruktor.

Pangkat.java

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

MainPangkat.java

```
Pangkat[] png = new Pangkat[elemen];
for (int i = 0; i < elemen; i++) {
    System.out.print(s:"Masukkan nilai yang hendak dipangkatkan: ");
    int nilai = sc.nextInt();
    System.out.print(s:"Masukkan nilai pemangkat: ");
    int pangkat = sc.nextInt();
    png[i] = new Pangkat(nilai, pangkat);
}
```

5. Tambahkan menu agar salah satu method yang terpilih saja yang akan dijalankan!

```

1 package minggu5;
2
3 import java.util.Scanner;
4
5 public class MainPangkat {
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         System.out.println("-----");
9         System.out.print("Masukkan jumlah elemen: ");
10        int elemen = sc.nextInt();
11
12        Pangkat[] png = new Pangkat[elemen];
13        for (int i = 0; i < elemen; i++) {
14            System.out.print("Masukkan nilai yang hendak dipangkatkan: ");
15            int nilai = sc.nextInt();
16            System.out.print("Masukkan nilai pemangkat: ");
17            int pangkat = sc.nextInt();
18            png[i] = new Pangkat(nilai, pangkat);
19        }
20
21        System.out.println("\nMethod yang tersedia");
22        System.out.println("1. Brute Force");
23        System.out.println("2. Divide and Conquer");
24        System.out.print("Masukkan Method Pilihan (1/2): ");
25
26        int metode = 0;
27        metode = sc.nextInt();
28
29        switch (metode) {
30            case 1:
31                System.out.println("HASIL - BRUTE FORCE");
32                for (int i = 0; i < elemen; i++) {
33                    System.out.println("Hasil dari "
34                        + png[i].nilai+ " pangkat "
35                        + png[i].pangkat+ " adalah "
36                        + png[i].pangkatBF(png[i].nilai, png[i].pangkat));
37                }
38                break;
39
40            case 2:
41                System.out.println("HASIL - DIVIDE AND CONQUER");
42                for (int i = 0; i < elemen; i++) {
43                    System.out.println("Hasil dari "
44                        + png[i].nilai+ " pangkat "
45                        + png[i].pangkat+ " adalah "
46                        + png[i].pangkatDC(png[i].nilai, png[i].pangkat));
47                }
48
49                sc.close();
50                break;
51
52            default:
53                break;
54        }
55    }
56 }

```





```

-----
Masukkan jumlah elemen: 2
Masukkan nilai yang hendak dipangkatkan: 6
Masukkan nilai pemangkat: 2
Masukkan nilai yang hendak dipangkatkan: 4
Masukkan nilai pemangkat: 3

Method yang tersedia
1. Brute Force
2. Divide and Conquer
Masukkan Method Pilihan (1/2): 1
HASIL - BRUTE FORCE
Hasil dari 6 pangkat 2 adalah 36
Hasil dari 4 pangkat 3 adalah 64
PS D:\kampus\smt2\Algoritma Struktur Dasar\Praktek\Pertemuan5>

```

```

-----
Masukkan jumlah elemen: 2
Masukkan nilai yang hendak dipangkatkan: 6
Masukkan nilai pemangkat: 2
Masukkan nilai yang hendak dipangkatkan: 4
Masukkan nilai pemangkat: 3

Method yang tersedia
1. Brute Force
2. Divide and Conquer
Masukkan Method Pilihan (1/2): 2
HASIL - DIVIDE AND CONQUER
Hasil dari 6 pangkat 2 adalah 36
Hasil dari 4 pangkat 3 adalah 64
PS D:\kampus\smt2\Algoritma Struktur Dasar\Praktek\Pertemuan5>

```



#### 4.4 Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

Di dalam percobaan ini, kita akan mempraktekkan bagaimana proses *divide*, *conquer*, dan *combine* diterapkan pada studi kasus penjumlahan keuntungan suatu perusahaan dalam beberapa bulan.

##### 4.4.1 Langkah-langkah Percobaan

1. Pada paket minggu5. Buat class baru yaitu class `Sum`. Di dalam class tersebut terdapat beberapa atribut jumlah elemen array, array, dan juga total. Tambahkan pula konstruktor pada class `Sum`.

```
public int elemen;
public double keuntungan[];
public double total;
```

```
Sum(int elemen){
    this.elemen = elemen;
    this.keuntungan=new double[elemen];
    this.total = 0;
}
```

2. Tambahkan method `TotalBF()` yang akan menghitung total nilai array dengan cara *iterative*.

```
double totalBF(double arr[]){
    for (int i = 0; i < elemen; i++) {
        total = total + arr[i];
    }
    return total;
}
```

3. Tambahkan pula method `TotalDC()` untuk implementasi perhitungan nilai total array menggunakan algoritma Divide and Conquer

```
double totalDC(double arr[], int l, int r){
    if(l==r)
        return arr[l];
    else if(l<r){
        int mid=(l+r)/2;
        double lsum=totalDC(arr,l,mid-1);
        double rsum=totalDC(arr,mid+1,r);
        return lsum+rsum+arr[mid];
    }

    return 0;
}
```

4. Buat class baru yaitu `MainSum`. Di dalam kelas ini terdapat method `main`. Pada method ini user dapat menuliskan berapa bulan keuntungan yang akan dihitung. Dalam kelas ini sekaligus dibuat instansiasi objek untuk memanggil atribut ataupun fungsi pada class `Sum`

```
Scanner sc = new Scanner(System.in);
System.out.println("=====");
System.out.println("Program Menghitung Keuntungan Total (Satuan Juta. Misal 5.9)");
System.out.print("Masukkan jumlah bulan : ");
int elm = sc.nextInt();
```

5. Karena yang akan dihitung adalah total nilai keuntungan, maka ditambahkan pula pada method main mana array yang akan dihitung. Array tersebut merupakan atribut yang terdapat di class Sum, maka dari itu dibutuhkan pembuatan objek Sum terlebih dahulu.

```
Sum sm = new Sum(elm);
System.out.println("=====");
for (int i = 0; i < sm.elemen; i++) {
    System.out.print("Masukkan untung bulan ke - " + (i+1) + " = ");
    sm.keuntungan[i] = sc.nextDouble();
}
```

6. Tampilkan hasil perhitungan melalui objek yang telah dibuat untuk kedua cara yang ada (Brute Force dan Divide and Conquer)

```
System.out.println("=====");
System.out.println("Algoritma Brute Force");
System.out.println("Total keuntungan perusahaan selama " + sm.elemen + " bulan adalah = "+sm.totalBF(sm.keuntungan));
System.out.println("=====");
System.out.println("Algoritma Divide Conquer");
System.out.println("Total keuntungan perusahaan selama " + sm.elemen + " bulan adalah = "+sm.totalDC(sm.keuntungan, 0, sm.elemen-1));
```

```

1 package minggu5;
2
3 import java.util.Scanner;
4
5 public class MainSum {
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         System.out.println("=====");
9         System.out.println("Program Menghitung Keuntungan Total (Satuan Juta, Misal 5.9)");
10        System.out.print("Masukkan jumlah bulan : ");
11        int elm = sc.nextInt();
12
13        Sum sm = new Sum(elm);
14        System.out.println("=====");
15        for (int i = 0; i < sm.elmen; i++) {
16            System.out.print("Masukkan untung bulan ke - " + (i+1) + " = ");
17            sm.keuntungan[i] = sc.nextDouble();
18        }
19
20        System.out.println("=====");
21        System.out.println("Algoritma Brute Force");
22        System.out.println("Total keuntungan perusahaan selama " + sm.elmen + " bulan adalah " + sm.totalBf(sm.keuntungan));
23        System.out.println("=====");
24        System.out.println("Algoritma Divide Conquer");
25        System.out.printf("Total keuntungan perusahaan selama %d bulan adalah %.2f", sm.elmen, sm.totalDC(sm.keuntungan, 0, sm.elmen-1));
26
27        sc.close();
28    }
29 }
30

```

```

1  package minggu5;
2
3  public class Sum {
4
5      int elemen;
6      double keuntungan[], total;
7
8      Sum(int elemen) {
9          this.elemen = elemen;
10         this.keuntungan = new double[elemen];
11         this.total = 0;
12     }
13
14     double totalBF(double arr[]) {
15         for(int i=0; i < elemen; i++) {
16             total = total + arr[i];
17         }
18         return total;
19     }
20
21     double totalDC(double arr[], int l, int r) {
22         if(l==r){
23             return arr[l];
24         }else if(l < r) {
25             int mid = (l+r)/2;
26             double lsum = totalDC(arr, l, mid-1);
27             double rsum = totalDC(arr, mid+1, r);
28             return lsum+rsum+arr[mid];
29         }
30         return 0;
31     }
32 }
33

```

#### 4.4.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

```
run:
=====
Program Menghitung Keuntungan Total (Satuan Juta. Misal 5.9)
Masukkan jumlah bulan : 5
=====
Masukkan untung bulan ke - 1 = 8.5
Masukkan untung bulan ke - 2 = 9.54
Masukkan untung bulan ke - 3 = 7.2
Masukkan untung bulan ke - 4 = 9.1
Masukkan untung bulan ke - 5 = 6
=====
Algoritma Brute Force
Total keuntungan perusahaan selama 5 bulan adalah = 40.339999999999996
=====
Algoritma Divide Conquer
Total keuntungan perusahaan selama 5 bulan adalah = 40.34
BUILD SUCCESSFUL (total time: 11 seconds)
```

```
=====  
Program Menghitung Keuntungan Total (Satuan Juta, Misal 5.9)  
Masukkan jumlah bulan : 5  
=====  
Masukkan untung bulan ke - 1 = 8.5  
Masukkan untung bulan ke - 2 = 9.54  
Masukkan untung bulan ke - 3 = 7.2  
Masukkan untung bulan ke - 4 = 9.1  
Masukkan untung bulan ke - 5 = 6  
=====  
Algoritma Brute Force  
Total keuntungan perusahaan selama 5 bulan adalah 40.339999999999996  
=====  
Algoritma Divide Conquer  
Total keuntungan perusahaan selama 5 bulan adalah 40.34  
PS D:\kampus\smt2\Algoritma Struktur Dasar\Praktek\Pertemuan5> |
```



#### 4.4.3 Pertanyaan

1. Berikan ilustrasi perbedaan perhitungan keuntungan dengan method `TotalBF()` ataupun `TotalDC()`

Jawaban: Ilustrasinya adalah sebagai berikut :

Misalkan kita memiliki data keuntungan harian selama seminggu (7 hari), dan kita ingin menghitung total keuntungan dari data tersebut. Contoh data keuntungan harian:

Hari 1: \$100

Hari 2: \$150

Hari 3: \$200

Hari 4: \$250

Hari 5: \$300

Hari 6: \$350

Hari 7: \$400

Dengan menggunakan metode `totalBF`, kita akan menjumlahkan semua keuntungan harian secara berurutan, seperti berikut : Total keuntungan =  $100 + 150 + 200 + 250 + 300 + 350 + 400 = \$1750$ .

Sementara itu, dengan menggunakan metode `totalDC`, kita akan membagi data menjadi submasalah yang lebih kecil. Kita akan mencari total keuntungan untuk setengah pertama hari (hari 1 hingga hari 4) dan setengah kedua hari (hari 5 hingga hari 7) terlebih dahulu, lalu menambahkan keduanya bersamaan dengan keuntungan pada hari keempat. Berikut adalah perhitungannya :

Total keuntungan untuk setengah pertama hari:  $100 + 150 + 200 + 250 = \$700$

Total keuntungan untuk setengah kedua hari:  $300 + 350 + 400 = \$1050$

Total keuntungan =  $700 + 1050 + 250 = \$2000$

Jadi, kita bisa melihat bahwa hasil perhitungan total keuntungan menggunakan metode `totalDC` akan berbeda dari hasil yang didapatkan menggunakan metode `totalBF`. Meskipun keduanya memberikan hasil yang benar, namun pendekatan yang digunakan untuk menghitungnya berbeda. Metode `totalDC` membagi masalah menjadi submasalah yang lebih kecil dan menggabungkan hasilnya, sementara metode `totalBF` langsung menjumlahkan semua data secara berurutan.

- Perhatikan output dari kedua jenis algoritma tersebut bisa jadi memiliki hasil berbeda di belakang koma. Bagaimana membatasi output di belakang koma agar menjadi standar untuk kedua jenis algoritma tersebut.

Untuk membatasi output di belakang koma agar menjadi standar untuk kedua jenis algoritma, kita dapat menggunakan DecimalFormat untuk memformat output sehingga memiliki jumlah angka di belakang koma yang sama. Berikut adalah kode modifikasinya :

```
1 package minggu5;
2
3 import java.text.DecimalFormat;
4
5 public class Sum {
6
7     int elemen;
8     double keuntungan[], total;
9
10    Sum(int elemen) {
11        this.elemen = elemen;
12        this.keuntungan = new double[elemen];
13        this.total = 0;
14    }
15
16    double totalBF(double arr[]) {
17        for (int i = 0; i < elemen; i++) {
18            total = total + arr[i];
19        }
20        DecimalFormat df = new DecimalFormat("#.##");
21        return Double.parseDouble(df.format(total));
22    }
23
24    double totalDC(double arr[], int l, int r) {
25        if (l == r) {
26            return arr[l];
27        } else if (l < r) {
28            int mid = (l + r) / 2;
29            double lsum = totalDC(arr, l, mid - 1);
30            double rsum = totalDC(arr, mid + 1, r);
31            double result = lsum + rsum + arr[mid];
32            DecimalFormat df = new DecimalFormat("#.##");
33            return Double.parseDouble(df.format(result));
34        }
35        return 0;
36    }
37 }
38
```

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\kampus\smt2\Algoritma Struktur Dasar\Praktek\Pertemuan5> d:; cd 'd:\kampus\smt2\Algoritma Struktur Dasar\Praktek\Pertemuan5'; & 'C:\Program Files\Java\jdk1.8.0_231\bin\java.exe' '-cp' 'C:\Users\LENOVO\AppData\Roaming\Code\User\workspaceStorage\11b624a05b5ef356f356f356f356f356\redhat.java\jdt_ws\Pertemuan5_5bc7abd7\bin' 'minggu5.MainSum'
=====
Program Menghitung Keuntungan Total (Satuan Juta, Misal 5.9)
Masukkan jumlah bulan : 5
=====
Masukkan untung bulan ke - 1 = 8.5
Masukkan untung bulan ke - 2 = 9.54
Masukkan untung bulan ke - 3 = 7.2
Masukkan untung bulan ke - 4 = 9.1
Masukkan untung bulan ke - 5 = 6
=====
Algoritma Brute Force
Total keuntungan perusahaan selama 5 bulan adalah 40.34
=====
Algoritma Divide Conquer
Total keuntungan perusahaan selama 5 bulan adalah 40.34
PS D:\kampus\smt2\Algoritma Struktur Dasar\Praktek\Pertemuan5>
```



Dengan menggunakan DecimalFormat, kita dapat memastikan bahwa output dari kedua metode memiliki jumlah angka di belakang koma yang sama

3. Mengapa terdapat formulasi *return value* berikut?Jelaskan!

```
return lsum+rsum+arr[mid];
```

Jawaban:

Kode tersebut merupakan bagian dari algoritma Divide and Conquer untuk menghitung total keuntungan dari suatu array.

- 1) lsum adalah total keuntungan dari submasalah bagian kiri.
- 2) rsum adalah total keuntungan dari submasalah bagian kanan.
- 3) arr[mid] adalah keuntungan dari hari tengah (atau elemen tengah) di antara submasalah bagian kiri dan kanan.

Dengan demikian, formulasi `return lsum + rsum + arr[mid];` tersebut memungkinkan algoritma Divide and Conquer untuk menggabungkan hasil dari submasalah yang lebih kecil menjadi solusi dari masalah asli, yaitu menghitung total keuntungan dari seluruh array.

4. Kenapa dibutuhkan variable `mid` pada method `TotalDC()` ?

Jawaban: Variabel `mid` dibutuhkan dalam metode `totalDC()` karena method tersebut menggunakan pendekatan Divide and Conquer. Ketika kita membagi array menjadi dua bagian, `mid` digunakan untuk menentukan indeks tengah dari array tersebut. Dengan mengetahui indeks tengah, kita dapat membagi array menjadi dua bagian yang sama besar (atau hampir sama besar jika jumlah elemen ganjil). ada tahap rekursif, submasalah tersebut kemudian dibagi kembali menjadi submasalah yang lebih kecil dengan cara yang sama, sampai mencapai submasalah yang dapat diselesaikan langsung (misalnya, submasalah dengan hanya satu elemen). Kemudian, hasil dari submasalah yang lebih kecil digabungkan kembali untuk mendapatkan solusi dari masalah asli.



5. Program perhitungan keuntungan suatu perusahaan ini hanya untuk satu perusahaan saja. Bagaimana cara menghitung sekaligus keuntungan beberapa bulan untuk beberapa perusahaan. (Setiap perusahaan bisa saja memiliki jumlah bulan berbeda-beda)? Buktikan dengan program!

Jawaban:

```

1  package minggu5;
2
3  import java.util.Scanner;
4
5  public class MainSum {
6
7      public static void main(String[] args) {
8
9          Scanner sc = new Scanner(System.in);
10
11          System.out.println("PROGRAM MENGHITUNG KEUNTUNGAN TOTAL");
12          System.out.println("=====");
13          System.out.print("Masukkan jumlah perusahaan : ");
14          int jml = sc.nextInt();
15
16          for (int i = 1; i <= jml; i++) {
17              System.out.println("=====");
18              System.out.println("Program Menghitung Keuntungan Total Perusahaan ke-" + i + " (Satuan Juta. Misal 5.9)");
19              System.out.print("Masukkan jumlah bulan : ");
20              int elm = sc.nextInt();
21              sc.nextLine();
22
23              Sum sm = new Sum(elm);
24              System.out.println("=====");
25              for (int j = 0; j < sm.elemen; j++) {
26                  System.out.print("Masukkan untung bulan ke - " + (j + 1) + " = ");
27                  sm.keuntungan[j] = sc.nextDouble();
28                  sc.nextLine();
29              }
30
31              System.out.println("\nTOTAL KEUNTUNGAN PERUSAHAAN KE-" + i);
32              System.out.println("=====");
33              System.out.println("Algoritma Brute Force");
34              System.out.println("Total keuntungan perubahan selama " + sm.elemen + " bulan adalah = "
35                  + sm.totalBF(sm.keuntungan));
36              System.out.println("=====");
37              System.out.println("Algoritma Divide Conquer");
38              System.out.printf("Total keuntungan perubahan selama %d bulan adalah = %.2f\n", sm.elemen,
39                  sm.totalDC(sm.keuntungan, 0, sm.elemen - 1));
40              System.out.println();
41          }
42
43          sc.close();
44      }
45  }

```



```
a Struktur Dasar\Praktek\Pertemuan5'; & 'C:\Program Files\Java\jdk1.8.0_231\bin\java.exe' '-cp'
'C:\Users\LENOVO\AppData\Roaming\Code\User\workspaceStorage\11ee10a54d74d40911b624a05b5ef356\r
edhat.java\jdt_ws\Pertemuan5_5bc7abd7\bin' 'minggu5.MainSum'
PROGRAM MENGHITUNG KEUNTUNGAN TOTAL
=====
Masukkan jumlah perusahaan : 2
=====
Program Menghitung Keuntungan Total Perusahaan ke-1 (Satuan Juta. Misal 5.9)
Masukkan jumlah bulan : 2
=====
Masukkan untung bulan ke - 1 = 9
Masukkan untung bulan ke - 2 = 7

TOTAL KEUNTUNGAN PERUSAHAAN KE-1
=====
Algoritma Brute Force
Total keuntungan perubahan selama 2 bulan adalah = 16.0
=====
Algoritma Divide Conquer
Total keuntungan perubahan selama 2 bulan adalah = 16.00

=====
Program Menghitung Keuntungan Total Perusahaan ke-2 (Satuan Juta. Misal 5.9)
Masukkan jumlah bulan : 4
=====
Masukkan untung bulan ke - 1 = 7
Masukkan untung bulan ke - 2 = 5
Masukkan untung bulan ke - 3 = 9
Masukkan untung bulan ke - 4 = 7

TOTAL KEUNTUNGAN PERUSAHAAN KE-2
=====
Algoritma Brute Force
Total keuntungan perubahan selama 4 bulan adalah = 28.0
=====
Algoritma Divide Conquer
Total keuntungan perubahan selama 4 bulan adalah = 28.00

PS D:\kampus\smt2\Algoritma Struktur Dasar\Praktek\Pertemuan5>
```

## 4.5 Latihan Praktikum

Buatlah kode program untuk menghitung nilai akar dari suatu bilangan dengan algoritma Brute Force dan Divide Conquer! *Jika bilangan tersebut bukan merupakan kuadrat sempurna, bulatkan angka ke bawah.*

```

1 package minggu5;
2 public class Akar {
3     int n;
4
5     public Akar(int n) {
6         this.n = n;
7     }
8
9     public int findSqrtBruteForce() {
10        int sqrt = 0;
11        for (int i = 1; i <= n; i++) {
12            if (i * i <= n) {
13                sqrt = i;
14            }
15        }
16        return sqrt;
17    }
18
19    public int findSqrtDivideConquer() {
20        if (n < 2) {
21            return n;
22        }
23        int low = 1;
24        int high = n;
25        int mid;
26        while (low <= high) {
27            mid = (low + high) / 2;
28            if (mid * mid <= n && (mid + 1) * (mid + 1) > n) {
29                return mid;
30            } else if (mid * mid > n) {
31                high = mid - 1;
32            } else {
33                low = mid + 1;
34            }
35        }
36        return -1;
37    }
38 }

```

```

1 package minggu5;
2 import java.util.*;
3 public class MainAkar {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         System.out.print("Masukkan akar: ");
7         int n = sc.nextInt();
8
9         // Create an instance of the Akar class (assuming it's defined below)
10        Akar sqrt = new Akar(n);
11        int sqrtBruteForce = sqrt.findSqrtBruteForce();
12        int sqrtDivideConquer = sqrt.findSqrtDivideConquer();
13        System.out.println("Akar dari " + n + " dengan Brute Force: " + sqrtBruteForce);
14        System.out.println("Akar dari " + n + " dengan Divide Conquer: " + sqrtDivideConquer);
15    }
16 }
17

```



```
Masukkan akar: 4  
Akar dari 4 dengan Brute Force: 2  
Akar dari 4 dengan Divide Conquer: 2  
PS D:\kampus\smt2\Algoritma Struktur Dasar\Praktek\Pertemuan5> |
```