LAPORAN PRAKTIKUM
MATA KULIAH ALGORITMA DAN STRUKTUR DATA

Dosen Pengampu : Triana Fatmawati, S.T, M.T

PERTEMUAN - 5 Brute Force & Divide Conquer

Nama : **M. Zidna Billah Faza**

NIM : **2341760030**

Prodi : **D-IV Sistem Informasi Bisnis**

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2023

A. PERCOBAAN 1

1) Buatlah class baru dengan nama Faktorial

2) 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;
    }

}
```

3) 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 input18 = new Scanner(System.in);

System.out.println(x:"============================================================");
System.out.print(s:"Masukkan jumlah elemen yang ingin dihitung : ");
int elemen = input18.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 = input18.nextInt();
}
```

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

```
System.out.println(x:"============================================================");
System.out.println(x:"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(x:"============================================================");
System.out.println(x:"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(x:"============================================================");
```

   d. Pastikan program sudah berjalan dengan baik!

```
============================================================
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
============================================================
```

Pertanyaan Percobaan 1

1) Jelaskan mengenai base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial!

Algoritma Divide-and-Conquer untuk mencari nilai faktor membagi masalah menjadi submasalah yang lebih kecil, menyelesaikan setiap submasalah secara terpisah, dan menggabungkan solusi untuk mendapatkan solusi akhir.

Sebagai bagian dari pencarian faktorial, algoritma pertama-tama memeriksa apakah nilai yang diinginkan adalah 1. Jika demikian, ia segera mengembalikan nilai 1. Jika tidak, algoritma membagi permasalahan menjadi permasalahan yang lebih kecil dengan mengalikan nilai yangg ditemukan dengan faktorial dari nilai sebelumnya. Proses ini diulangi hingga kasus dasar tercapai, dimana nilai yang diinginkan adalah 1.

Hasil dari setiap langkah rekursi kemudian digabungkan untuk mendapatkan nilai faktor akhir. Oleh karena itu, algoritma ini menggunakan pendekatan rekursif untuk menyelesaikan masalah, menggabungkan solusi hingga sampai pada solusi akhir.

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

Pada percobaan 1 hanya terdapat 2 tahapan yaitu Divide dan Conquer karena tahapan Combine tidak selalu eksplisit dalam implementasi faktorial menggunakan algoritma Divide and Conquer.

- Devide

Terjadi ketika pada fungsi faktorialDC(int n), kita membagi masalah menjadi masalah yang lebih kecil dengan mengurangi nilai faktorial yang dicari (n) satu per satu hingga mencapai kasus dasar (n==1).

- Conquer

Terjadi di dalam blok else pada fungsi faktorialDC(int n). Pada tahap ini, kita menyelesaikan setiap submasalah secara rekursif dengan mengalikan nilai faktorial yang sedang dicari (n) dengan nilai faktorial dari masalah yang lebih kecil (faktorialDC(n-1)). Setiap langkah rekursif ini akan terus berulang sampai mencapai kasus dasar.

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

Perulangan pada method faktorialBF() dapat diubah menjadi rekursif dengan menggunakan if else seperti berikut :

```
// SESUDAH PERTANYAAN

if (n == 0 || n == 1) {

    return 1;

} else {

    return n * faktorialBF(n - 1);

}
```

4) Tambahkan pegecekan waktu eksekusi kedua jenis method tersebut!

```
============================================================
Masukkan jumlah elemen yang ingin dihitung : 3
Masukkan nilai data ke-1 : 8
Masukkan nilai data ke-2 : 5
Masukkan nilai data ke-3 : 3
============================================================
Hasil Faktorial dengan Brute Force
Waktu eksekusi : 5867 m/s
Faktorial dari nilai 8 adalah : 40320
Waktu eksekusi : 3911 m/s
Faktorial dari nilai 5 adalah : 120
Waktu eksekusi : 3911 m/s
Faktorial dari nilai 3 adalah : 6
============================================================
Hasil Faktorial dengan Divide and Conquer
Waktu eksekusi : 5867 m/s
Faktorial dari nilai 8 adalah : 40320
Waktu eksekusi : 3911 m/s
Faktorial dari nilai 5 adalah : 120
Waktu eksekusi : 3422 m/s
Faktorial dari nilai 3 adalah : 6
============================================================
```

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

```
Masukkan nilai data ke-20 : 20
============================================================
Hasil Faktorial dengan Brute Force
Waktu eksekusi : 6356 ms
Faktorial dari nilai 1 adalah : 1
Waktu eksekusi : 2933 ms
Faktorial dari nilai 2 adalah : 2
Waktu eksekusi : 4889 ms
Faktorial dari nilai 3 adalah : 6
Waktu eksekusi : 3911 ms
Faktorial dari nilai 4 adalah : 24
Waktu eksekusi : 26889 ms
Faktorial dari nilai 5 adalah : 120
Waktu eksekusi : 3911 ms
Faktorial dari nilai 6 adalah : 720
Waktu eksekusi : 10561468 ms
Faktorial dari nilai 7 adalah : 5040
Waktu eksekusi : 3911 ms
Faktorial dari nilai 8 adalah : 40320
Waktu eksekusi : 4400 ms
Faktorial dari nilai 9 adalah : 362880
Waktu eksekusi : 4400 ms
Faktorial dari nilai 10 adalah : 3628800
Waktu eksekusi : 3422 ms
Faktorial dari nilai 11 adalah : 39916800
Waktu eksekusi : 4888 ms
Faktorial dari nilai 12 adalah : 479001600
Waktu eksekusi : 5866 ms
Faktorial dari nilai 13 adalah : 1932053504
Waktu eksekusi : 4400 ms
Faktorial dari nilai 14 adalah : 1278945280
Waktu eksekusi : 4889 ms
Faktorial dari nilai 15 adalah : 2004310016
Waktu eksekusi : 69912 ms
Faktorial dari nilai 16 adalah : 2004189184
Waktu eksekusi : 5378 ms
Faktorial dari nilai 17 adalah : -288522240
```

B. PERCOBAAN 2

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 class Pangkat {

    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 input18 = new Scanner(System.in);

System.out.println(x:"============================================================");
System.out.print(s:"Masukkan jumlah elemen yang ingin dihitung : ");
int elemen = input18.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 = input18.nextInt();
    System.out.print("Masukkan nilai pemangkat ke-" + (i+1) + " : ");
    png[i].pangkat = input18.nextInt();
}
```

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

```
System.out.println(x:"============================================================");
System.out.println(x:"Hasil Faktorial 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(x:"============================================================");
System.out.println(x:"Hasil Faktorial 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(x:"============================================================");
```

8) Verifikasi Hasil Percobaan

```
============================================================
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 Faktorial dengan Brute Force
Nilai 6 pangkat 2 adalah : 36
Nilai 4 pangkat 3 adalah : 64
============================================================
Hasil Faktorial dengan Divide and Conquer
Nilai 6 pangkat 2 adalah : 36
Nilai 4 pangkat 3 adalah : 64
============================================================
```

Pertanyaan Percobaan 2

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

- PangkatBF()
  a. Menggunakan iterasi sederhana dengan looping for untuk mengalikan bilangan a sebanyak n kali
  b. Algoritma ini efektif untuk nilai n yang relatif kecil, tetapi memiliki kompleksitas waktu O(n) karena membutuhkan waktu linear sesuai dengan nilai eksponen n

- PangkatDC()
  a. Membagi masalah menjadi submasalah yang lebih kecil dan menyelesaikannya secara rekursif.
  b. Algoritma ini memanfaatkan sifat matematis bahwa a^n dapat dipecah menjadi a^(n/2) * a^(n/2) untuk eksponen n genap dan (a^(n/2) * a^(n/2) * a) untuk eksponen n ganjil.
  c. Algoritma ini memiliki kompleksitas waktu yang lebih baik daripada Brute Force, yaitu O(log n), karena melakukan pembagian masalah secara rekursif menjadi submasalah yang lebih kecil

2) Pada method PangkatDC() terdapat potongan program sebagai berikut: Jelaskan arti potongan kode tersebut

```
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));

}
```

- if (n % 2 == 1) { // Bilangan Ganjil

Pernyataan ini memeriksa apakah eksponen n adalah bilangan ganjil. Jika sisa pembagian n dengan 2 sama dengan 1, itu berarti n adalah bilangan ganjil.

- return (pangkatDC(a, n/2) * pangkatDC(a, n/2) * a)

Jika eksponen n adalah bilangan ganjil, maka masalah dibagi menjadi dua bagian: a^(n/2) dan a^(n/2). Kedua submasalah ini diselesaikan secara rekursif menggunakan metode pangkatDC(). Kemudian, hasilnya dikalikan dengan a, karena eksponen n ganjil menambahkan satu faktor a tambahan.

- else { // Bilangan Genap

Ini adalah bagian dari struktur if-else yang menangani kasus ketika eksponen n adalah bilangan genap

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

Jika eksponen n adalah bilangan genap, maka masalah dibagi menjadi dua bagian yang sama: a^(n/2) dan a^(n/2). Kedua submasalah ini diselesaikan secara rekursif menggunakan metode pangkatDC(), dan hasilnya dikalikan bersama.

Dengan cara ini, pendekatan Divide and Conquer secara rekursif memecah masalah menjadi submasalah yang lebih kecil dan menyelesaikannya secara efisien.

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

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

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

```java
public class Pangkat {

    public int nilai, pangkat;

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

    public int getNilai() {
        return nilai;
    }

    public int getPangkat() {
        return pangkat;
    }

    public int pangkatBF(int a, int n) {

        int hasil = 1;

        for (int i = 0; i < n; i++) {
            hasil = hasil * a;
        }

        return hasil;
    }

    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));

            }
        }
    }
}
```

```java
import java.util.Scanner;

public class MainPangkat {
    public static void main(String[] args) {
        Scanner input18 = new Scanner(System.in);

        System.out.println("============================================================");
        System.out.print("Masukkan jumlah elemen yang ingin dihitung : ");
        int elemen = input18.nextInt();

        Pangkat[] png = new Pangkat[elemen];

        for (int i = 0; i < elemen; i++) {
            System.out.print("Masukkan nilai yang akan dipangkatkan ke-" + (i + 1) + " : ");
            int nilai = input18.nextInt();
            System.out.print("Masukkan nilai pemangkat ke-" + (i + 1) + " : ");
            int pangkat = input18.nextInt();
            png[i] = new Pangkat(nilai, pangkat);
        }

        System.out.println("============================================================");
        System.out.println("Hasil Faktorial dengan Brute Force");

        for (int i = 0; i < elemen; i++) {
            System.out.println("Nilai " + png[i].getNilai() + " pangkat " + png[i].getPangkat() + " adalah : "
                    + png[i].pangkatBF(png[i].getNilai(), png[i].getPangkat()));
        }

        System.out.println("============================================================");
        System.out.println("Hasil Faktorial dengan Divide and Conquer");

        for (int i = 0; i < elemen; i++) {
            System.out.println("Nilai " + png[i].getNilai() + " pangkat " + png[i].getPangkat() + " adalah : "
                    + png[i].pangkatDC(png[i].getNilai(), png[i].getPangkat()));
        }

        System.out.println("============================================================");
    }
}
```

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

```
============================================================
                          MENU
1. Hitung Pangkat dengan Brute Force
2. Hitung Pangkat dengan Divide and Conquer
3. Keluar
============================================================
Pilih menu : 1
============================================================
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 Faktorial dengan Brute Force
Nilai 6 pangkat 2 adalah : 36
Nilai 4 pangkat 3 adalah : 64
============================================================
```

```java
import java.util.Scanner;

public class MainPangkat {
    public static void main(String[] args) {
        Scanner input18 = new Scanner(System.in);
        int pilihan;

        do {
            System.out.println("============================================================");
            System.out.println("                              MENU");
            System.out.println("1. Hitung Pangkat dengan Brute Force");
            System.out.println("2. Hitung Pangkat dengan Divide and Conquer");
            System.out.println("3. Keluar");
            System.out.println("============================================================");
            System.out.print("Pilih menu : ");
            pilihan = input18.nextInt();

            switch (pilihan) {
                case 1:
                    PangkatBF();
                    break;
                case 2:
                    PangkatDC();
                    break;
                case 3:
                    System.out.println("Program selesai.");
                    break;
                default:
                    System.out.println("Pilihan tidak valid. Silakan pilih menu yang benar.");
            }
        } while (pilihan != 3);
    }

    public static void PangkatBF() {
        Scanner input18 = new Scanner(System.in);

        System.out.println("============================================================");
        System.out.print("Masukkan jumlah elemen yang ingin dihitung : ");
        int elemen = input18.nextInt();

        Pangkat[] png = new Pangkat[elemen];

        for (int i = 0; i < elemen; i++) {
            System.out.print("Masukkan nilai yang akan dipangkatkan ke-" + (i + 1) + " : ");
            int nilai = input18.nextInt();
            System.out.print("Masukkan nilai pemangkat ke-" + (i + 1) + " : ");
            int pangkat = input18.nextInt();
            png[i] = new Pangkat(nilai, pangkat);
        }

        System.out.println("============================================================");
        System.out.println("Hasil Faktorial dengan Brute Force");

        for (int i = 0; i < elemen; i++) {
            System.out.println("Nilai " + png[i].getNilai() + " pangkat " + png[i].getPangkat() + " adalah : "
                    + png[i].pangkatBF(png[i].getNilai(), png[i].getPangkat()));
        }
    }

    public static void PangkatDC() {
        Scanner input18 = new Scanner(System.in);

        System.out.println("============================================================");
        System.out.print("Masukkan jumlah elemen yang ingin dihitung : ");
        int elemen = input18.nextInt();

        Pangkat[] png = new Pangkat[elemen];

        for (int i = 0; i < elemen; i++) {
            System.out.print("Masukkan nilai yang akan dipangkatkan ke-" + (i + 1) + " : ");
            int nilai = input18.nextInt();
            System.out.print("Masukkan nilai pemangkat ke-" + (i + 1) + " : ");
            int pangkat = input18.nextInt();
            png[i] = new Pangkat(nilai, pangkat);
        }

        System.out.println("============================================================");
        System.out.println("Hasil Faktorial dengan Divide and Conquer");

        for (int i = 0; i < elemen; i++) {
            System.out.println("Nilai " + png[i].getNilai() + " pangkat " + png[i].getPangkat() + " adalah : "
                    + png[i].pangkatDC(png[i].getNilai(), png[i].getPangkat()));
        }
    }
}
```

C. PERCOBAAN 3

1) Pada paket minggu5. Buat class baru yaitu class Sum. DI salam 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 input18 = new Scanner(System.in);

System.out.println(x:"============================================================");
System.out.println(x:"Program Menghitung Keuntungan Total (Satuan Juta. Misal 5.9)");
System.out.print(s:"Masukkan jumlah bulan : ");
int elemen = input18.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(elemen);
System.out.println(x:"============================================================");
for (int i = 0; i < sm.elemen; i++) {

    System.out.print("Masukkan untung bulan ke-" + (i + 1) + " : ");
    sm.keuntungan[i] = input18.nextDouble();

}
```

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

```
System.out.println(x:"============================================================");
System.out.println(x:"Algoritma Brute Force");
System.out.println("Total keuntungan perusahaan selama " + sm.elemen + " bulan adalah : " + sm.totalBF(sm.keuntungan));

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

System.out.println(x:"------------------------------------------------------------");
```

7) Verifikasi hasil percobaan

```
============================================================
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
============================================================
```

Pertanyaan Percobaan 3

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

- totalBF()
  a. Pada metode ini akan menelusuri seluruh elemen dalam array keuntungan
  b. Setiap elemen ditambahkan ke total secara berurutan dengan kompleksitas waktu O(n), di mana n adalah jumlah elemen dalam array

- totalDC()
  a. Metode ini membagi masalah menjadi submasalah yang lebih kecil, menyelesaikan submasalah tersebut secara rekursif, dan menggabungkan hasilnya
  b. Ketika mencapai kasus dasar (basis), yaitu ketika kisaran (range) dari l dan r hanya satu elemen, maka nilai dari elemen tersebut dikembalikan
  c. Jika kisaran lebih dari satu elemen, kisaran dibagi menjadi dua bagian pada tengahnya (mid), dan dua submasalah dipecahkan secara rekursif untuk setiap bagian kiri dan kanan
  d. Kemudian, total dari kiri, kanan, dan elemen tengah dihitung dan digabungkan untuk memberikan total keseluruhan
  e. Proses ini memiliki kompleksitas waktu O(n log n) karena membagi masalah menjadi dua setengah pada setiap tingkat rekursi.

Jadi, TotalBF() sederhana dan langsung menambahkan keuntungan secara berurutan, sementara TotalDC() membagi masalah menjadi submasalah lebih kecil, menyelesaikannya secara rekursif, dan kemudian menggabungkan hasilnya untuk mendapatkan total keseluruhan

2) 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.

Dibulatkan menjadi 2 angka setelah koma, maka menghasilkan output seperti ini

```
============================================================
Algoritma Brute Force
Total keuntungan perusahaan selama 5 bulan adalah : 40.34
============================================================
Algoritma Divide Conquer
Total keuntungan perusahaan selama 5 bulan adalah : 40.34
============================================================
```

```
return Math.round(total * 100.0) / 100.0;
return Math.round((lsum + rsum + arr[mid]) * 100.0) / 100.0;
```

3) Mengapa terdapat formulasi return value berikut?Jelaskan!

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

- lsum

Ini adalah hasil dari pemanggilan rekursif totalDC() pada setengah bagian kiri array (dari indeks l hingga mid - 1). Metode ini mengembalikan total keuntungan dari setengah bagian kiri

- rsum

Ini adalah hasil dari pemanggilan rekursif totalDC() pada setengah bagian kanan array (dari indeks mid + 1 hingga r). Metode ini mengembalikan total keuntungan dari setengah bagian kanan

- arr[mid]

Ini adalah nilai keuntungan pada indeks mid, yang merupakan nilai keuntungan tengah dari array.

Jadi, return lsum + rsum + arr[mid]; menggabungkan total keuntungan dari setengah bagian kiri, setengah bagian kanan, dan nilai keuntungan tengah, yang merupakan total keseluruhan keuntungan dari seluruh array. Ini adalah langkah combine dalam algoritma Divide and Conquer, di mana hasil dari dua submasalah yang lebih kecil digabungkan bersama-sama untuk membentuk solusi untuk masalah yang lebih besar.

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

Variabel mid dibutuhkan dalam metode totalDC() karena itu merupakan indeks yang menandai titik tengah dari array yang sedang diproses. Dalam pendekatan Divide and Conquer, array dibagi menjadi dua bagian setiap kali metode rekursif dipanggil. Variabel mid digunakan untuk menandai pembagian tersebut

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!

```
============================================================
Masukkan jumlah perusahaan : 2
Masukkan jumlah bulan untuk perusahaan ke-1: 5
Masukkan jumlah bulan untuk perusahaan ke-2: 3
============================================================
Keuntungan perusahaan ke-1
Masukkan keuntungan perusahaan ke-1 pada bulan ke-1: 8,5
Masukkan keuntungan perusahaan ke-1 pada bulan ke-2: 9,54
Masukkan keuntungan perusahaan ke-1 pada bulan ke-3: 7,2
Masukkan keuntungan perusahaan ke-1 pada bulan ke-4: 9,1
Masukkan keuntungan perusahaan ke-1 pada bulan ke-5: 6
============================================================
Keuntungan perusahaan ke-2
Masukkan keuntungan perusahaan ke-2 pada bulan ke-1: 8,9
Masukkan keuntungan perusahaan ke-2 pada bulan ke-2: 9,01
Masukkan keuntungan perusahaan ke-2 pada bulan ke-3: 9,92
============================================================
Algoritma Brute Force
Total keuntungan perusahaan ke-1 adalah 40.34
Total keuntungan perusahaan ke-2 adalah 27.83
============================================================
Algoritma Divide Conquer
Total keuntungan perusahaan ke-1 adalah 40.34
Total keuntungan perusahaan ke-2 adalah 27.83
============================================================
```

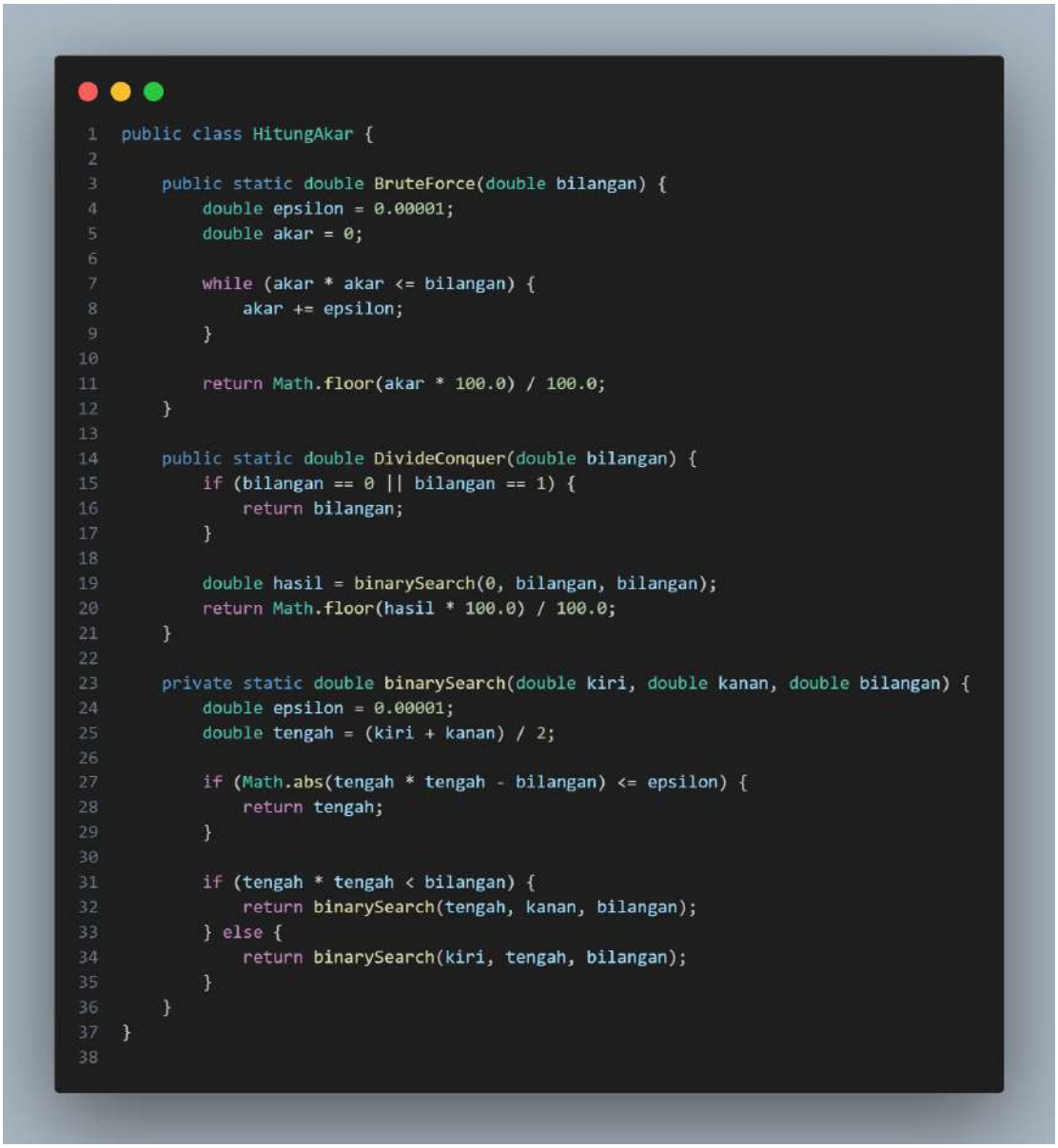
```java
public class Sum_1 {

    public double[][] keuntungan;

    public Sum_1(int[] jumlahBulan) {
        int jumlahPerusahaan = jumlahBulan.length;
        this.keuntungan = new double[jumlahPerusahaan][];
        for (int i = 0; i < jumlahPerusahaan; i++) {
            this.keuntungan[i] = new double[jumlahBulan[i]];
        }
    }

    public double totalBF(double[] arr) {
        double total = 0;
        for (double d : arr) {
            total += d;
        }
        return Math.round(total * 100.0) / 100.0;
    }

    public double totalDC(double[] arr, int l, int r) {
        double total = 0;
        for (double d : arr) {
            total += d;
        }
        return Math.round(total * 100.0) / 100.0;
    }

}
```

```java
import java.util.Scanner;

public class MainSum_1 {
    public static void main(String[] args) {
        Scanner input18 = new Scanner(System.in);

        System.out.println("============================================================");
        System.out.print("Masukkan jumlah perusahaan : ");
        int jumlahPerusahaan = input18.nextInt();

        int[] jumlahBulan = new int[jumlahPerusahaan];
        for (int i = 0; i < jumlahPerusahaan; i++) {
            System.out.print("Masukkan jumlah bulan untuk perusahaan ke-" + (i + 1) + ": ");
            jumlahBulan[i] = input18.nextInt();
        }

        Sum_1[] sums = new Sum_1[jumlahPerusahaan];
        for (int i = 0; i < jumlahPerusahaan; i++) {
            System.out.println("------------------------------------------------------------");
            System.out.println("Keuntungan perusahaan ke-" + (i + 1));
            sums[i] = new Sum_1(jumlahBulan);
            for (int j = 0; j < jumlahBulan[i]; j++) {
                System.out.print("Masukkan keuntungan perusahaan ke-" + (i + 1) + " pada bulan ke-" + (j + 1) + ": ");
                sums[i].keuntungan[i][j] = input18.nextDouble();
            }
        }

        System.out.println("============================================================");
        System.out.println("Algoritma Brute Force");

        for (int i = 0; i < jumlahPerusahaan; i++) {
            System.out.println("Total keuntungan perusahaan ke-" + (i + 1) + " adalah " + sums[i].totalBF(sums[i].keuntungan[i]));
        }

        System.out.println("============================================================");
        System.out.println("Algoritma Divide Conquer");

        for (int i = 0; i < jumlahPerusahaan; i++) {
            System.out.println("Total keuntungan perusahaan ke-" + (i + 1) +  " adalah " + sums[i].totalDC(sums[i].keuntungan[i], 0, jumlahBulan[i] - 1));
        }
        System.out.println("============================================================");
    }
}

```

D. TUGAS

1) 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.

- Program HitungAkar.java



```java
public class HitungAkar {

    public static double BruteForce(double bilangan) {
        double epsilon = 0.00001;
        double akar = 0;

        while (akar * akar <= bilangan) {
            akar += epsilon;
        }

        return Math.floor(akar * 100.0) / 100.0;
    }

    public static double DivideConquer(double bilangan) {
        if (bilangan == 0 || bilangan == 1) {
            return bilangan;
        }

        double hasil = binarySearch(0, bilangan, bilangan);
        return Math.floor(hasil * 100.0) / 100.0;
    }

    private static double binarySearch(double kiri, double kanan, double bilangan) {
        double epsilon = 0.00001;
        double tengah = (kiri + kanan) / 2;

        if (Math.abs(tengah * tengah - bilangan) <= epsilon) {
            return tengah;
        }

        if (tengah * tengah < bilangan) {
            return binarySearch(tengah, kanan, bilangan);
        } else {
            return binarySearch(kiri, tengah, bilangan);
        }
    }
}

```

- Program MainHitungAkar.java

```java
import java.util.Scanner;

public class MainHitungAkar {
    public static void main(String[] args) {
        Scanner input18 = new Scanner(System.in);

        System.out.println("================================");
        System.out.print("Masukkan bilangan          : ");
        double bilangan = input18.nextDouble();

        double akarBruteForce = HitungAkar.BruteForce(bilangan);
        double akarDivideConquer = HitungAkar.DivideConquer(bilangan);

        System.out.println("================================");
        System.out.println("Akar dengan Brute Force     : " + akarBruteForce);
        System.out.println("Akar dengan Divide Conquer  : " + akarDivideConquer);
        System.out.println("================================");

    }
}

```

- Output

```
==================================
Masukkan bilangan          : 4
==================================
Akar dengan Brute Force    : 2.0
Akar dengan Divide Conquer : 2.0
==================================
```

Link Repository : https://github.com/zidnafaz/Praktikum-Algoritma-Struktur-Data