LAPORAN TUGAS PRAKTIKUM ALGORITMA DAN STRUKTUR DATA JOBSHEET 5

`

NAMA : RADITYA RIEFKI
KELAS : TI 1E
ABSEN : 23

5.2 Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

KODE PROGRAM

Faktorial

```
public class faktorial23 {

    int faktorialBF (int n){
        int faktor = 1;
        for (int i = 1; i <= n; i++) {
            faktor = faktor * i;
        }
        return faktor;
    }
    int faktorialDC (int n){
        if ( n == 1){
            return 1;
        }else{
            int faktor = n * faktorialDC(n - 1);
            return faktor;
        }
    }
}
```

Faktorial Main

```
import java.util.Scanner;
public class mainFaktorial23 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Masukkan nilai : ");
        int nilai = sc.nextInt();

        faktorial23 fk = new faktorial23();
        System.out.println("Nilai faktorial " + nilai + " Menggunakan BF : "
+fk.faktorialBF(nilai));
        System.out.println("Nilai Faktorial " + nilai + "Menggunakan DC : "
+ fk.faktorialDC(nilai));


    }



}
```

OUTPUT

```
Masukkan nilai : 5
Nilai faktorial 5 Menggunakan BF : 120
Nilai Faktorial 5Menggunakan DC : 120
PS D:\CollegeFile\SMT 2\ALSD>
```

5.2.3. Pertanyaan

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

- Pada bagian n==1 untuk menghentikan rekursi

Else untuk pemanggilan rekursif untuk menghitung factorial utama

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

-Berikut adalah pembuktian menggunakan while loop

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

3. Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1);

- fakto *=i merupakan iteratif dan mengalikan nilai fakto secara bertahap

Sebaliknya int fakto = n * faktorialDc(n-1) merupakan rekursif dan menyelesaikan masalah secara terpecah atau menjadikannya sub lebih kecil lagi

4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!

- seperti yang dijelaskan diatas faktorialBF() adalah iteratif lebih efisien untuk input besar

Dan faktorialDC() merupakan rekursif kurang efisien untuk nilai n besar

5.3 Menghitung Hasil Pangkat dengan Algoritma Brute Force dan Divide and Conquer

KODE PROGRAM

Pangkat

```
public class Pangkat23 {
    int nilai , pangkat;

    Pangkat23(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));
            }
        }
    }
}
```

Pangkat Main

```
import java.util.Scanner;
public class mainPangkat23{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Masukkan jumlah elemen : ");
        int elemen = sc.nextInt();

        Pangkat23 [] png = new Pangkat23[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 elemen ke-" + (i+1) + "
: ");
            int pangkat = sc.nextInt();
            png[i] = new Pangkat23(basis, pangkat);
```

```
        }
        System.out.println("HASIL PANGKAT BRUTEFORCE :");
        for (Pangkat23 p : png){
            System.out.println(p.nilai + " ^ " + p.pangkat + ": " +
p.pangkatBF(p.nilai, p.pangkat));
        }
        System.out.println("HASIL PANGKAT DIVIDE AND CONQUER :");
        for (Pangkat23 p : png){
            System.out.println(p.nilai + " ^ " + p.pangkat + ": " +
p.pangkatDC(p.nilai, p.pangkat));
        }

    }
}
```

OUTPUT

```
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 BRUTEFORCE :
2 ^ 3: 8
4 ^ 5: 1024
6 ^ 7: 279936
HASIL PANGKAT DIVIDE AND CONQUER :
2 ^ 3: 8
4 ^ 5: 1024
6 ^ 7: 279936
PS D:\CollegeFile\SMT 2\ALSD>
```

5.3.3. Pertanyaan

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

- pangkatBF() menggunakan iteratif tidak efisien untuk nilai n yang besar

pangkatDC() menggunakan rekursif lebih efisien untuk nilai n yang besar

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

- Tahap Combine sudah ada dalam kode **pangkatDC()**, yaitu pada baris yang mengalikan hasil dari pangkatDC(a, n/2).

3. Pada method pangkatBF()terdapat parameter untuk melewatkan nilai yang akan dipangkatkan dan pangkat berapa, padahal di sisi lain di class Pangkat telah ada atribut nilai dan pangkat, apakah menurut Anda method tersebut tetap relevan untuk memiliki parameter? Apakah bisa jika method tersebut dibuat dengan tanpa parameter? Jika bisa, seperti apa method pangkatBF() yang tanpa parameter?

-

```
int nilai, pangkat;

    Pangkat23(int n, int p) {
        nilai = n;
        pangkat = p;
    }

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

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

- pangkatBF() menggunakan brute force mengalikan a dan n berulang dan efisien

pangkatDC() menggunakan divide and conquer membagi masalah menjadi lebih kecil lebih cepat jika nilai n lebih besar

5.4 Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

KODE PROGRAM

SUM

```
public class Sum23 {
    double keuntungan[];

    Sum23(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;
    }
}
```

Sum Main

```
import java.util.Scanner;
public class MainSum23 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Masukkan jumlah elemen : ");
      int elemen = sc.nextInt();

      Sum23 sm = new Sum23(elemen);
      for (int i = 0; i < elemen; i++) {
         System.out.print("Masukkan keuntungan ke-" + (i+1) + " : ");
         sm.keuntungan[i] = sc.nextDouble();
      }
      System.out.println("Total keuntungan menggunakan BruteForfce: " + sm.totalBF());
      System.out.println("Total keuntungan menggunakan Divide and Conquer : " + sm.totalDC(sm.keuntungan,
0, elemen-1));
   }
```

OUTPUT

```
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 BruteForfce: 150.0
Total keuntungan menggunakan Divide and Conquer : 150.0
PS D:\CollegeFile\SMT 2\ALSD>
```

5.4.3. Pertanyaan

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

- berfungsi untuk membagi array menjadi dua bagian dalam pendekatan Divide and Conquer**.**

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

- membagi masalah menjadi sub lebih kecil

Menghitung total dari array secara rekursif

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

- untuk mengcombine Kembali dari hasil rekursif

4. Apakah base case dari totalDC()?

- Base case terjadi ketika **l == r**, dan nilai elemen dikembalikan**.**

5. Tarik Kesimpulan tentang cara kerja totalDC()

- memecah array menjadi 2 bagian sampai tersisa 1 elemen

Hitung total bagian secara rekursif

Menggabung hasil perhitungan dari kedua bagian

Mengembalikan total array