LAPORAN HASIL PRATIKUM

ANALISIS DAN STRUKTUR DATA

JOBSHEET 4

Raihan Akbar Putra Prasetyo/244107020087

Kelas: TI-1E

D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

PRAKTIKUM 25

Percobaan 1

- **Code program faktorial.java**

```
package jobsheet5;
public class faktorial24 {
    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;
        }
    }
}
```

- **Code program mainfaktorial.java**

```
package jobsheet5;
import java.util.Scanner;
public class mainFaktorial24 {
    public static void main(String[] args) {
        Scanner input = new Scanner (System.in);
        System.out.print("Masukkan nilai: ");
        int nilai =input.nextInt();

        faktorial24 fk = new faktorial24();
        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 5 menggunakan DC 120
PS D:\kuliah\PRAKTIKUM-ASD>
```

- **Pertanyaan**
  1. **Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!**
     - **Pada faktorialDC(), bagian if (n == 1) adalah base case yang menghentikan rekursi ketika n mencapai 1, sedangkan bagian else adalah recursive case yang terus memanggil dirinya sendiri dengan n-1 hingga mencapai base case.**
  2. **Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan for? Buktikan!**
     - **Ya, perulangan dalam faktorialBF() bisa diubah menggunakan while atau do-while, di mana keduanya tetap melakukan iterasi dari 1 hingga n untuk menghitung faktorial.**
     - **Pembuktian**

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

  3. **Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !**
     - **fakto *= i; digunakan dalam iterasi untuk memperbarui nilai faktorial dalam setiap loop, sedangkan int fakto = n * faktorialDC(n-1); digunakan dalam rekursi untuk menghitung faktorial dengan memanggil fungsi itu sendiri hingga mencapai base case.**
  4. **Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!**
     - **faktorialBF() menggunakan loop untuk menghitung faktorial secara langsung, sedangkan faktorialDC() menggunakan rekursi dengan membagi masalah menjadi submasalah yang lebih kecil, di mana BF lebih efisien dalam penggunaan memori, sementara DC lebih elegan dalam struktur kode.**

PERCOBAAN 2

- **code program**

```
package jobsheet5;
public class pangkat24 {
    int nilai,pangkat;

    pangkat24 (int n, int p){
        nilai = nilai;
        pangkat = p;
    }
    int pangkatBF(int a, int n){
        int hasil = 1;
        for ( int i=0; i<n; i++){
            hasil = hasil * a;
    }
    return hasil;
    }

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

```
package jobsheet5;
import java.util.Scanner;
public class mainpangkat24 {
    public static void main(String[] args) {
        Scanner input = new Scanner (System.in);
        System.out.print("Masukkan Jumlah elemen : ");
        int elemen =input.nextInt();

        pangkat24 [] png = new pangkat24 [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 pangkat24 (basis, pangkat);

    }
    System.out.println("HASIL PANGKAT BRUTEFORCE:");
    for (pangkat24 p : png) {
        System.out.println(p.nilai + "^" + p.pangkat + " = " +
p.pangkatBF(p.nilai, p.pangkat));
    }

    System.out.println("HASIL PANGKAT DIVIDE AND CONQUER:");
    for (pangkat24 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:
0^3 = 0
0^5 = 0
0^7 = 0
HASIL PANGKAT DIVIDE AND CONQUER:
0^3 = 0
0^5 = 0
0^7 = 0
PS D:\kuliah\PRAKTIKUM-ASD>
```

Pertanyaan :

1. **Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!**
   - **Perbedaan pangkatBF() dan pangkatDC() terletak pada pendekatannya, di mana pangkatBF() menggunakan perulangan untuk mengalikan angka sebanyak n kali (kompleksitas O(n)), sedangkan pangkatDC() menggunakan rekursi dengan membagi masalah menjadi lebih kecil hingga mencapai kondisi dasar (kompleksitas O(log n)), sehingga lebih efisien.**
2. **Apakah tahap combine sudah termasuk dalam kode tersebut? Tunjukkan!**
   - **Tahap combine dalam Divide and Conquer sudah ada dalam pangkatDC(), yaitu pada bagian return (pangkatDC(a, n/2) * pangkatDC(a, n/2)) yang menggabungkan hasil dari dua submasalah untuk mendapatkan hasil pangkat akhir.**

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?**
   - **Metode pangkatBF() sebenarnya tidak perlu memiliki parameter karena nilai basis dan pangkat sudah ada dalam atribut kelas, sehingga bisa dibuat tanpa parameter dengan langsung menggunakan this.nilai dan this.pangkat, misalnya dengan metode int pangkatBF() { int hasil = 1; for (int i = 0; i < pangkat; i++) { hasil *= nilai; } return hasil; }.**

4. **Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!**
   - **pangkatBF() bekerja dengan iterasi yang mengalikan angka berulang kali, sedangkan pangkatDC() bekerja dengan rekursi yang membagi masalah menjadi lebih kecil, di mana Divide and Conquer lebih cepat karena hanya membutuhkan O(log n) pemanggilan dibandingkan iterasi langsung O(n).**

Percobaan 3

- **code program**

```
package jobsheet5;
public class sum24 {
    double keuntungan[];

    sum24(int el) {
        keuntungan = new double[el];
    }
    double totalBF() {
        double total = 0;
        for (int i = 0; i < keuntungan.length; i++) {
            total += keuntungan[i];
        }
        return total;
    }
    double totalDC(double arr[], int l, int r) {
        if (l == r) {
            return arr[l];
        }
        int mid = (l + r) / 2;
        double lsum = totalDC(arr, l, mid);
        double rsum = totalDC(arr, mid + 1, r);
        return lsum + rsum;
    }
}
```

```
package jobsheet5;
import java.util.Scanner;
public class mainSum24 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan jumlah elemen: ");
        int elemen = input.nextInt();
        sum24 sm = new sum24(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));


    }
}
```

- **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 Brute Force: 150.0
Total keuntungan menggunakan Divide and Conquer: 150.0
PS D:\kuliah\PRAKTIKUM-ASD>
```

Pertanyaan :

1. **Kenapa dibutuhkan variable mid pada method TotalDC()?**
   mid digunakan untuk membagi array menjadi dua bagian agar bisa dihitung secara rekursif.
2. **Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?**

```
double lsum = totalDC(arr, l, mid);
double rsum = totalDC(arr, mid+1, r);
```

   lsum dan rsum digunakan untuk menghitung total bagian kiri dan kanan array sebelum digabungkan.
3. **Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?**

```
return lsum+rsum;
```

Penjumlahan lsum + rsum diperlukan untuk menggabungkan hasil perhitungan kedua bagian array.

4. **Apakah base case dari totalDC()?**
   Base case terjadi saat l == r, yaitu saat hanya ada satu elemen yang langsung dikembalikan.
5. **Tarik Kesimpulan tentang cara kerja totalDC()**
   totalDC() bekerja dengan membagi, menghitung secara rekursif, lalu menggabungkan hasilnya.