Laporan Praktikum

Algoritma Dan Struktur Data

Nama : Arifah Zhafirah Wikananda

NIM : 244107020188

Kelas : 1E

Program Studi Teknologi

Informasi Jurusan Teknik

Informatika Politeknik Negeri

Malang

2024

1. Code Program

```
public class faktorial04 {
    public  int nilai;

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

```
import java.util.Scanner;

public class faktorianMain04 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan nilai: ");
        int nilai = input.nextInt();

        faktorial04 fk = new faktorial04();
        System.out.println("Nilai faktorial " + nilai + "menggunakan BF: " +
fk.faktorialBF(nilai));
        System.out.println("Nilai Faktorial " + nilai + "menggunakan DC: " +
fk.faktorialDC(nilai));
    }
}
```

Outputnya :

```
C:\Praktikum ASD\jobsheet4> cmd /C ""C:\Users\WINDOWS 11\AppData\Roaming\Code\User\globalStorage\pleiades.java-extension-pack-jdk\java\latest\bin\java.exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp "C:\Users\WINDOWS 11\AppData\Roaming\Code\User\workspaceStorage\46dea9d03715a3fb0cf23a3ed50cff75\redhat.java\jdt_ws\jobsheet4_6bf5a73f\bin" faktorianMain04 "
Masukkan nilai: 5
Nilai faktorial 5menggunakan BF: 120
Nilai Faktorial 5menggunakan DC: 120
```

Pertanyaan:

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!
Jawab: base case if berfungsi digunakan untuk menghentikan rekursif ketika nilai sudah mencapai kondisi dasar. Kalau base case else digunakan untuk memecah besar menjadi sub-masalah yang lebih kecil, kemudian menyelesaikannya secara rekursif.

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

```
public class faktorial04 {
    public  int nilai;

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

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

3. Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !
Jawaban: jika fakto *=i; menggunakan metode Brute Force (BF) dan cara kerjanya menggunakan perulangan (for loop), jika fakto = n * faktorialDC(n-1); memakai metode Divide & Conquer(DC) dan cara kerjanya memanggil fungsi rekursif.

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

- **perbedaan cara kerja method faktorialBF() dan faktorialDC()**

- faktorialBF(): Menggunakan metode **Brute Force (BF)** dengan **iterasi** (for loop).
- faktorialDC(): Menggunakan metode **Divide & Conquer (DC)** dengan **rekursi**.
- faktorialBF(): Menghitung faktorial dengan mengalikan angka dari 1 hingga n secara bertahap dalam **looping**.
- faktorialDC(): Memecah masalah menjadi sub-masalah yang lebih kecil dengan memanggil dirinya sendiri secara **rekursif** hingga mencapai **base case**.
- faktorialBF() : Jika **efisiensi dan kecepatan** lebih penting, gunakan **iteratif (faktorialBF())**.
- faktorialDC() : Jika ingin menggunakan **rekursi untuk konsep Divide & Conquer**, gunakan **rekursif (faktorialDC())**.

2. Code Program

```
public class pangkat04 {
    public int nilai, pangkat;

    pangkat04 (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));
            }
        }
    }
}
```

```
import java.util.Scanner;

public class mainPangkat04 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan jumlah elemen: ");
        int elemen = input.nextInt();

        pangkat04[] png = new pangkat04[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 pangkat04(basis, pangkat);
        }
        System.out.println("HASIL PANGKAT BRUTEFORCE: ");
        for(pangkat04 p : png){
            System.out.println(p.nilai + "^" + p.pangkat + ": " +
p.pangkatBF(p.nilai, p.pangkat));
        }
        System.out.println("HASIL PANGKAT DIVIDE AND CONQUER: ");
        for(pangkat04 p : png){
            System.out.println(p.nilai + "^" + p.pangkat + ": " +
p.pangkatDC(p.nilai, p.pangkat));
        }
    }
}
```

Outputnya:

```
C:\Praktikum ASD\jobsheet4>
C:\Praktikum ASD\jobsheet4> c: && cd "c:\Praktikum ASD\jobsheet4" && cmd /C ""C:\Users\WINDOWS 11\AppData\Roaming\Code\
User\globalStorage\pleiades.java-extension-pack-jdk\java\latest\bin\java.exe" --enable-preview -XX:+ShowCodeDetailsInEx
ceptionMessages -cp "C:\Users\WINDOWS 11\AppData\Roaming\Code\User\workspaceStorage\46dea9d03715a3fb0cf23a3ed50cff75\re
dhat.java\jdt_ws\jobsheet4_6bf5a73f\bin" mainPangkat04 "
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
```

Pertanyaan:

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!
Jawab : pada method pangkatBF() menggunakan loop untuk menghitung pangkat secara iterativ sedangkan untuk method pangkatDC() yang menerapkan divide and conquer menggunakan rekursi.

2. Apakah tahap combine sudah termasuk dalam kode tersebut?Tunjukkan!
Jawab: sudah, tahap combine ditunjukkan oleh sintaks berikut:

```
17      int pangkatDC(int a, int n){
18          if(n == 1){
19              return a;
20          }else{
21              if(n%2==1){
22                  return (pangkatDC(a, n/2)*pangkatDC(a, n/2)*a);
23              }else{
24                  return (pangkatDC(a, n/2)*pangkatDC(a, n/2));
25              }
26          }
```

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?
Jawab:

- tidak terlalu relavan, karena nilai yang akan dipangkatkan dan pangkatnya sudah ada dalam objek. Menggunakan parameter disini hanya menambah kompleksitas yang tidak perlu.

- Ya, bisa. Cukup dengan menggunakan atribut nilai dan pangkat dalam metode tersebut.
- Implementasi pangkatBF() tanpa parameter:

```
8
9       int pangkatBF(){
10          int hasil = 1;
11          for(int i = 0; i<this.pangkat; i++){
12              hasil *= nilai;
13          }
14          return hasil;
15      }
```

4. Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!
Cara kerja pangkatBF() menggunakan metode Brute Force menghitung pangkat dengan melakukan perkalian berulang sebanyak nilai eksponen, cara kerja pangkatDC() menggunakan metode Divide and Conquer membagi masalah menjadi sub-masalah yang lebih kecil, lalu menyelesaikannya secara rekursif.

3.Code Program

```
public class sum04 {
    double keuntungan[];
    sum04(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;
    }
}
```

```
import java.util.Scanner;

public class mainSum04 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan jumlah elemen: ");
        int elemen = input.nextInt();

        sum04 sm = new sum04(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 Bruteforce: " +
sm.totalBF());
        System.out.println("Total keuntungan menggunakan Divide and Conquer: "
+ sm.totalDC(sm.keuntungan, 0, elemen-1));
    }
}
```

Outputnya:

```
C:\Praktikum ASD\jobsheet4> cmd /C ""C:\Users\WINDOWS 11\AppData\Roaming\Code\User\globalStorage\pleiades.java-extension-pack-jdk\java\latest\bin\java.exe" -
-enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp "C:\Users\WINDOWS 11\AppData\Roaming\Code\User\workspaceStorage\46dea9d03715a3fb0cf23a3ed50cff75\
redhat.java\jdt_ws\jobsheet4_6bf5a73f\bin" mainSum04 "
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 Bruteforce: 150.0
Total keuntungan menggunakan Divide and Conquer: 150.0
```

Pertanyaan:

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

Jawab: Variabel mid pada metode totalDC() dibutuhkan untuk menentukan titik tengah dari array yang sedang di proses. Tujuannya adalah membagi array menjadi dua bagian, sehingga algoritma Divide and Conquer bisa bekerja dengan membagi masalah menjadi sub-masalah yang lebih kecil.

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

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

Jawab:

- lsum = totalDC(arr, l, mid);
  - Memanggil totalDC() secara rekursif untuk menghitung jumlah elemen pada bagian kiri array (dari indeks l hingga mid).
- rsum = totalDC(arr, mid+1, r);
  - Memanggil totalDC() secara rekursif untuk menghitung jumlah elemen pada bagian kanan array (dari indeks mid+1 hingga r).

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

```
return lsum+rsum;
```

Jawab: Menyatukan hasil perhitungan dari bagian kiri (**lsum**) dan kanan (**rsum**) dan menjamin hasil akhirnya adalah jumlah total dari semua elemen array.

4. Apakah base case dari totalDC()?

Jawab: Base case adalah kondisi di mana rekursi berhenti dan mulai mengembalikan hasil tanpa memanggil dirinya sendiri lagi. Base case dalam metode totalDC() biasanya terjadi ketika array sudah tidak bisa dibagi lagi, yaitu saat hanya ada satu elemen tersisa.

5. Tarik Kesimpulan tentang cara kerja totalDC()

Jawab:

- Divide and Conquer: Masalah besar (penjumlahan array) dipecah menjadi masalah kecil hingga base case tercapai.
- Rekursi: Fungsi dipanggil berulang kali untuk menghitung total pada setiap bagian.
- Efisien: Lebih optimal dibanding metode iteratif pada skenario tertentu, terutama untuk array besar.
- Struktur yang Jelas: Memisahkan proses divide (pembagian)**,** conquer (penyelesaian)**,** dan combine (penggabungan hasil)**.**