LAPORAN HASIL PRAKTIKUM

Agoritma Struktur Data

JOBSHEET 5

Muhammad Fitra Adhim Nurrochman
2441007020089
TI 1E
PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLINEMA

PERCOBAAN 1

1. Hasil kode program faktorial

```
public class Faktorial {

    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;
        }
    }
}
```

Hasil kode program faktorialmain

```
import java.util.Scanner;

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

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

Hasil Run Kode Program

```
Masukkan nilai: 5
Nilai faktorial 5menggunakan BF: 120
Nilai faktorial 5menggunakan DC: 120
PS C:\Users\noval\OneDrive\Documents\1
```

PERTANYAAN PERCOBAAN 1

1. Pada method DC terdapat bagian if dan else, bagian if sebagai base case untuk menghentikan perulangan rekursif dengan kondisi n==1, dan else sebagai proses pemecahan

permasalahan faktorial dengan memecah n menjadi nilai yang kecil hingga mencapai base case

2. Method BF dapat diubah perulangannya, disini saya menggunakan while hasilnya sebagai berikut

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

3. fakto *= i; proses perulangan yang terjadi dengan cara fakto *= i berjalan secara bertahap perkalian dilakukan dari yang terkecil.
   int fakto = n * faktorialDC(n-1) proses perulangan yang dilakukan dengan cara int fakto = n * faktorialDC(n-1) berjalan dengan cara dari terbesar kemudian ke terkecil.

4. Method BF bekerja secara berathap melakukan proses perkalian faktorial mulai dari ang paling keciil hingga ke n, sedangkan method DC dimulai dari yang pailng besar lalu disederhanakan menjadi yang paling kecil

PERCOBAAN 2

1. Hasil Kode Program Pangkat

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

   Pangkat(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));
         }
      }
   }
}
```

Hasil Kode Program Main

```
import java.util.Scanner;

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

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

Hasil Run kode prgoram

```
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 PERCOBAAN 2

1. Perbedaan dari kedua method tersebut adalah
   pangkatBF melakukan proses perkalian nilai a sampai n kali secara iteratif, sedangkan pangkatDC memecah pangkat n menjadi kebih kecil dengan dibagi 2 hingga n menjadi bentuk yang paling kecil

2. Proses combine sudah termasuk ke dalam kode tersebut yaitu
   If(n%2==1) return(pangkatDC(a, n/2)*pangkatDC(a, n/2*)a)
   Else return(pangkatDC(a, n/2)*apngkatDC(a, n/2))

3. Menurut saya bisa dibuat tanpa parameter jika menggunakan atribut nilai dan oangkat dari class, contohnya

```
int hasil = 1;
    for (int i = 1; i <= pangkat;
i++) {
        hasil *= nilai;
    }
    return hasil;
```

Walaupun bisa dibuat parameter tetapi kurang relevan untuk digunakan

4. BF melakukan perhitungan pangkat nilai a dikalikan dengan nilai a hingga n kali secara iteratiif sedangkan DC melakukan proses perthiungan pangkat n nilai a dengan metode membagi proses menghitung pangkat menjadi kecil dan disatukan kembali

PERCOBAAN 3

1. Hasil kode program sum

```
public class Sum {
    public void Sum() {

    }
    double keuntungan[];

    Sum(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;
    }
}
```

Hasil Kode program mainsum

```
import java.util.Scanner;

public class MainSum {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      System.out.print("Masukan Jumlah Elemen: ");
      int elemen = input.nextInt();

      Sum sm = new Sum(elemen);
      for (int i =0; i < elemen; i++) {
         System.out.print("Masukan 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));
   }
}
```

Hasil Run Kode Program

```
Masukan Jumlah Elemen: 5
Masukan keuntungan ke-1: 10
Masukan keuntungan ke-2: 20
Masukan keuntungan ke-3: 30
Masukan keuntungan ke-4: 40
Masukan keuntungan ke-5: 50
Total keuntungan menggunakan Bruteforce: 150.0
Total keuntungan menggunakan Divide and Conquer: 150.0
```

PERTANYAAN PERCOBAAN 3

1. Karna variabel mid digunakan untuk mengetahui indeks titik tengah pada atribut array keuntungan yang akan digunakan sebagai pembatas dari perhitungan kiri dan sebagai awalan dair perhiutngan kanan
2. double lsum = totalDC(arr,l,mid); berfungsi untuk menghitung penjumlahan array pada indeks 0 – mid (indeks tengah pada array) atau proses penjumlahan pada sisi kiri array secara rekursif.

   double rsum = totalDC(arr,mid+1,r); sedangkan rsum berfungsi untuk menghitung proses penjumlahan array pada indeks tengah (mid+1) hingga batas indeks dari array atau bisa dibilang proses penjumlahan pada sisi kanan array secara rekursif.
3. Dilakukan untuk menggabungkan solusi dari masalah yang dipecah, hasil lsum merupakan solusi dari sub masalah pertama dan hasil rsum merupakan solusi dari sub masalh kedua, untuk menemukan solusi sempurnanya maka keduanya ditambhakan
4. Base case nya yaitu jika posisi indeks l dan indejs r bernilai sama maka akan mengembalikan nilai array pada indeks l
5. Method untuk mengetahui jumlah dari seluruh elemen pada array dengan metode divide and conquer. Pada method totalDC solusi untuk menghitung jumlah seluruh elemen pada array dilakukan dengan membagi proses perhitungan menjadi dua bagian, yaitu sisi kiri dan sisi kanan hingga mencapai basis rekursi, yaitu hingga posisi indeks kiri dan kanan bernilai sama sehingga menunjuk elemen pada indeks yang sama. Setelah itu menemukan hasil dari sisi kiri dan kanan, maka return penjumlahan dari hasil sisi kiri dan sisi kanan untuk mendapatkan hasil akhirnya.