POLITEKNIK NEGERI MALANG

TEKNOLOGI INFORMASI

TEKNIK INFORMATIKA

Mohammad Ariq Baihaqi

244107020161

TI – 1A

16

5.2.1. Langkah-langkah Percobaan

```
package minggu5;
import java.util.Scanner;
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;
        }
    }

    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));
    }
}
```

OUTPUT

```
Masukkan nilai: 6
Nilai faktorial 6 menggunakan BF: 720
Nilai faktorial 6 menggunakan DC: 720
```

Pertanyaan

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!
    - Bagian if(n==1) ini adalah base case dalam rekursi. Jiika nilain sudah mencapai 1, maka faktorialnya adalah 1, dan proses berhenti di sini
    - Bagian else ini adalah recursive case, Dimana fungsi akan terus memanggil dirinya sendiri dengan nilai n-1 hingga mencapai base case
2. Apakah memungkinkan perulangan pada method faktorialBF() diubahselainmenggunakan for? Buktikan!
    - Bisa, dengan menggunakan while loop

```
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 metode faktorialBF(), yang berarti nilai fakto diperbarui dalam setiap iterasi dengan mengalikan nilai sebelumnya dengan i
    - `Int fakto = n * faktorialDC(n-1);` ini adalah bentuk rekusif, di mana nilai factorial dihitung dengan mengalikan n dengan hasil pemanggilan `rekursif faktorialDC(n-1)`
4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!
    - Brute Force(faktorialBF()) lebih efisien dalam penggunaan memori tetapi bisa lebih lambat pada nilai n besr karena banyak iterasi
    - Divide and Conquer(faktorialDC()) lebih elegan tetapi menggunakan lebih banyak memori karena kerusi meyimpan state di stack

5.3.1. Langkah-langkah Percobaan

```
package minggu5;


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;
    }

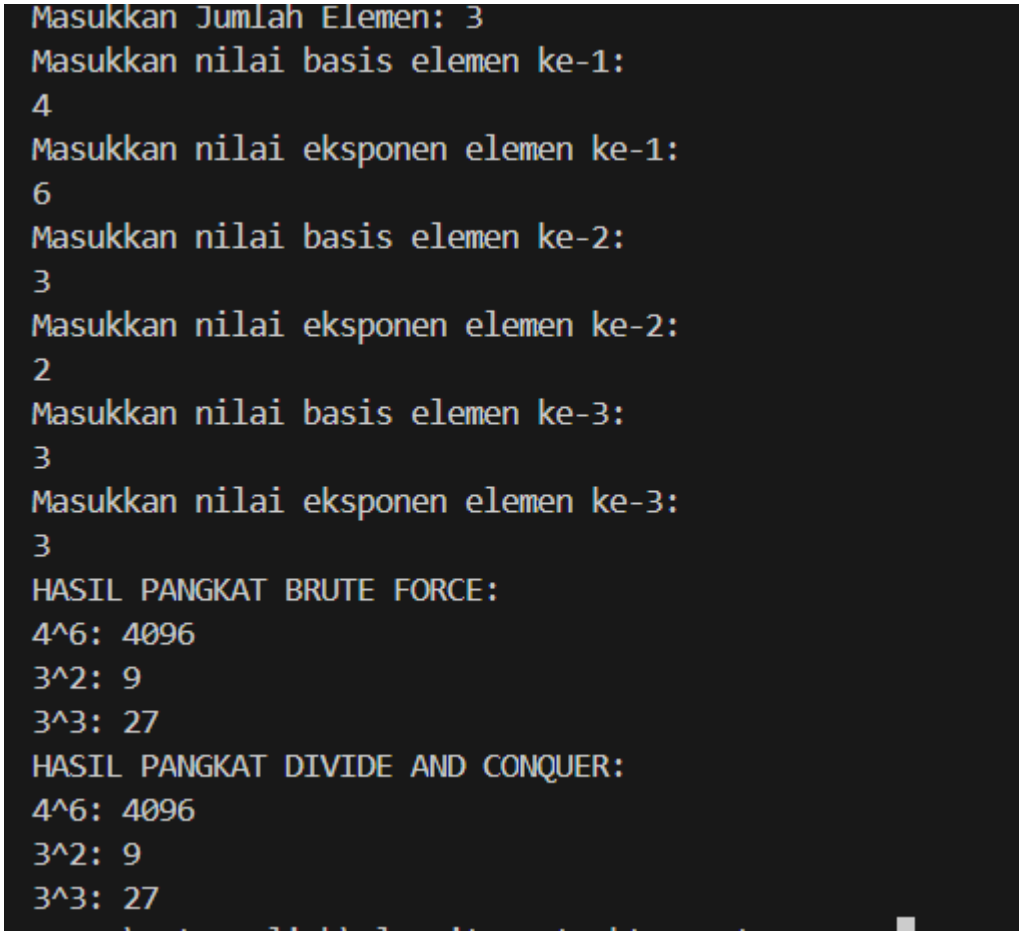
    int pangkatDC(int a, int n){
        if(n==1){
            return a;
        }else{
            if(n%2==1){
                return (pangkatDC(a, n/2)*pangkatDC(a, n/2)*a);
            }else{
                return (pangkatDC(a, n/2)*pangkatDC(a, n/2));
            }
        }
    }
}
```

```
package minggu5;
import java.util.Scanner;
public class MainPangkat {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int elemen = sc.nextInt();

        Pangkat[] png = new Pangkat[elemen];
        for (int i = 0; i < elemen; i++) {
            System.out.println("Masukkan nilai basis elemen ke-" +(i+1)+": ");
            int basis = sc.nextInt();
            System.out.println("Masukkan nilai eksponen elemen ke-" +(i+1)+": ");
            int pangkat = sc.nextInt();
            png[i] = new Pangkat(basis, pangkat);
    }

    System.out.println("HASIL PANGKAT BRUTE FORCE: ");
    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));
    }
}
}
```

OUTPUT



Pertanyaan

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!
    - Kelebihan: pangkatBF() mudah dipahamai dan diimplementasikan, pangkatDC() lebih cepat karena kompleksitasnya O(log n). cocok untuk n besar
    - Kekurangan: pangkatBF() kurang efisien untuk nilai n besar karena kompleksitasnya O(n), pangkatDC() menggunakan lebih banyak memori karena rekursi meyimpan state dalam stack.
2. Apakah tahap combine sudah termasuk dalam kode tersebut?Tunjukkan!
    - Ya, tahap combine sudah ada dalam kode pangkatDC()

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

- Jika ingin mempertahankan fleksibiltas(memungkinkan input nilai selain yang ada di objek), maka tetap relevan
- Ya, bisa di buat tanpa parameter

```
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 mengalikan a dengan dirinya sendiri sebanyak n kali menggunakan perulangan.
    - PangkatDC() bekerja dengan membagi masalah menjadi lebih kecil:
      1.) jika n genap, hasilnya dihitung sebagai kuadrat dari separuh eksponen
      2.) jika n ganjil, hasilnya sama seperti di atas tetapi dikalikan a sekali lagi

5.4.1. Langkah-langkah Percobaan

```
package minggu5;


public class 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;
    }
}
```

```
package minggu5;

import java.util.Scanner;

public class MainSum {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Masukkan jumlah elemen: ");

        int elemen = sc.nextInt();


        Sum sm = new Sum(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 Bruteforce: "+sm.totalBF());

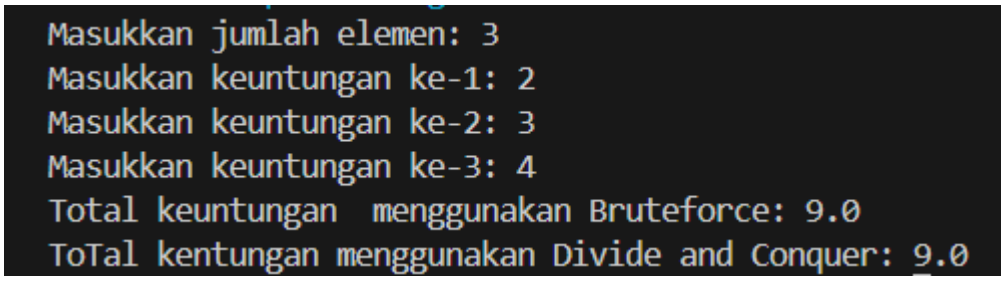
    System.out.println("ToTal kentungan menggunakan Divide and Conquer:
"+sm.totalDC(sm.keuntungan,0,elemen-1));

    }




}
```

OUTPUT



Pertanyaan

1. Kenapa dibutuhkan variable mid pada method TotalDC()?
    - Karena variable mid digunakan untuk membagi menjadi dua bagian dalam algoritma Divide and conqeur
2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?
```
double lsum = totalDC(arr, l, mid);
double rsum = totalDC(arr, mid+1, r);
```
    - Menghitung total keuntungan dari bagian kiri(dari indeks l sampai mid)
    - Menhitung total keuntungan dari bagian kanan(dari indeks mid+1 sampai r)
3. Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?
```
return lsum+rsum;
```
    - Karena setelah array dibagi menjadi dua bagian, hasil dari masing-masing bagianharus dikombinasikan Kembali untuk mendapatkan total keuntungan
4. Apakah base case dari totalDC()?
    - ```
      if(l == r){
          return arr[l];
      }
      ```
5. Tarik Kesimpulan tentang cara kerja totalDC()
    - Membagi array menjadi dua bagian, menghitung totalnya secara rekursif, lalu menggabungkan hasilnya
    - Lebih cepat untuk jumlah data besar
    - Menggunakan lebih banyak memori karena rekursi meyimpan state dalam stack
      ```

4.5Latihan Praktikum

```
package minggu5;


class Mahasiswa16 {
    String nama, nim;
    int tahunMasuk, nilaiUTS, nilaiUAS;


    Mahasiswa16(String nama, String nim, int tahunMasuk, int nilaiUTS, int nilaiUAS)
{
        this.nama = nama;
        this.nim = nim;
        this.tahunMasuk = tahunMasuk;
        this.nilaiUTS = nilaiUTS;
        this.nilaiUAS = nilaiUAS;
    }
}


class NilaiMahasiswa {
    Mahasiswa16[] data;


    NilaiMahasiswa(Mahasiswa16[] data) {
        this.data = data;
    }


    // Menentukan nilai UTS tertinggi menggunakan Divide and Conquer
    int cariMaxUTS(int l, int r) {
        if (l == r) {
            return data[l].nilaiUTS;
        }
        int mid = (l + r) / 2;
        int leftMax = cariMaxUTS(l, mid);
        int rightMax = cariMaxUTS(mid + 1, r);
        return Math.max(leftMax, rightMax);
    }
```

```
// Menentukan nilai UTS terendah menggunakan Divide and Conquer
    int cariMinUTS(int l, int r) {
        if (l == r) {
            return data[l].nilaiUTS;
        }
        int mid = (l + r) / 2;
        int leftMin = cariMinUTS(l, mid);
        int rightMin = cariMinUTS(mid + 1, r);
        return Math.min(leftMin, rightMin);
    }

    // Menghitung rata-rata nilai UAS menggunakan Brute Force
    double hitungRataUAS() {
        int total = 0;
        for (Mahasiswa16 m : data) {
            total += m.nilaiUAS;
        }
        return (double) total / data.length;
    }
}
```

```
package minggu5;
import java.util.Scanner;
public class MahasiswaMain16 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Mahasiswa16[] mahasiswa = {
            new Mahasiswa16("Fika", "220101001", 2022, 78, 82),
            new Mahasiswa16("Alfreda", "220101002", 2022, 85, 88),
            new Mahasiswa16("Nabil", "220101003", 2021, 90, 87),
            new Mahasiswa16("Hanif", "220101004", 2021, 76, 79),
            new Mahasiswa16("Ilham", "220101005", 2023, 92, 95),
    };

    NilaiMahasiswa nm = new NilaiMahasiswa(mahasiswa);

    //menampilkan nilai UTS tertinggi
    int maxUTS = nm.cariMaxUTS(0, mahasiswa.length - 1);
    System.out.println("Nilai UTS tertinggi (Divide and conquer): " + maxUTS);

    //menampilkan nilai UTS terendah
    int minusUTS = nm.cariMinUTS(0, mahasiswa.length-1);
    System.out.println("Nilai UTS terendah (Divide and Conquer): " + minusUTS);

    //menampilkan rata-rata nilai uas
    double rataUAS = nm.hitungRataUAS();
    System.out.println("Rata-rata nilai UAS (Brute Force): " + rataUAS);
}
}
```

OUTPUT

```
Nilai UTS tertinggi (Divide and conquer): 92
Nilai UTS terendah (Divide and Conquer): 76
Rata-rata nilai UAS (Brute Force): 86.2
```

Link Github : https://github.com/Ariqq16/semester2