Laporan Hasil Praktikum Algoritma Struktur Data

Jobsheet 5

Febryan Akhmad Taajuddin

244107020180

Kelas 1E

Program Studi Teknik Informatika

Jurusan Teknologi Informasi

Politeknik Negeri Malang

2024

Percobaan 1

1. Tambahkan method faktorialBF();

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

2. Tambahkan method faktorialDC();

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

3. Buat class MainFaktorial untuk memasukkan input user

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

4. Buat objek dari class Faktorial dan tampilkan hasil pemanggilan method faktorialDC() dan faktorialBF()

```
        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));
```

5. Run kode program

```
Masukkan nilai: 4
Nilai faktorial 4 menggunakan BF: 24
Nilai faktorial 4 menggunakan DC: 24
```

Pertanyaan percobaan 1

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!
2. Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan for? Buktikan!
3. Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !
4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!

Jawaban percobaan 1

1. Perbedaan antara if dan else dalam program tersebut adalah, kode if sebagai pembatas perulangan (base case), sedangkan kode else sebagai perulangan (recursive case).
2. Memungkinkan perulangan diubah menjadi while. Berikut kode programnya.

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

3. fakto *= i digunakan dalam untuk memperbarui nilai faktorial dalam loop, sedangkan int fakto = n * faktorialDC(n-1) digunakan untuk menghitung nilai faktorial dengan memanggil fungsi itu sendiri.
4. Method faktorialBF() menggunakan Brute Force dengan cara perulangan untuk menghitung faktorial dan faktorialDC() menggunakan Divide and Conquer dengan cara memanggil fungsi rekursif untuk menghitung faktorial..

Percobaan 2

1. Tambahkan atribut angka pada class Pangkat

```
int nilai, pangkat;
```

2. Tambahkan konstruktor berparameter

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

3. Tambahkan method PangkatBF

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

4. Tambahkan method PangkatDC

```
    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));
            }
        }
    }
```

5. Buat class MainPangkat untuk memasukkan input jumlah elemen

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

6. Buat instansiasi array of object untuk input nilai yang akan di pangkatkan

```
        Pangkat[] png = new Pangkat[elemen];
        for (int i = 0; i < elemen; i++) {
            System.out.print("Masukan nilai basis elemen ke-"+(i+1)+ ": ");
            int basis = input.nextInt();
            System.out.print("Masukan nilai basis elemen ke-"+(i+1)+ ": ");
            int pangkat = input.nextInt();
            png[i] = new Pangkat(basis, pangkat);
        }
```

7. Buat return value dari method PangkatBF() dan PangkatDC()

```
        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));
        }
```

8. Run kode program

```
Masukkan jumlah elemen: 3
Masukan nilai basis elemen ke-1: 2
Masukan nilai basis elemen ke-1: 3
Masukan nilai basis elemen ke-2: 4
Masukan nilai basis elemen ke-2: 5
Masukan nilai basis elemen ke-3: 6
Masukan nilai basis 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. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!
2. Apakah tahap combine sudah termasuk dalam kode tersebut?Tunjukkan!
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?
4. Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!

Jawaban percobaan 2

1. Method pangkatBF() menggunakan Brute Force yang menggunakan perulangan for untuk menghitung pangkat, sedangkan method Pangkat DC() menggunakan Divide and Conqueror yang menggunakan cara memanggil fungsi untuk menghitung pangkat.
2. Tahap combine sudah termasuk dalam kode tersebut

```
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));
        }
    }
}
```

Pada kode return (pangkatDC(a, n / 2) * pangkatDC(a, n / 2) * a); adalah combine untuk n ganjil, sedangkan kode return (pangkatDC(a, n / 2) * pangkatDC(a, n / 2)); adalah combine untuk n genap.

3. Modifikasi method Pangkat

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

Modifikasi method MainPangkat

```
        for (Pangkat p : png) {
            System.out.println(p.nilai+"^"+p.pangkat+":
"+p.pangkatBF());
        }
```

4. Method pangkatBF() menggunakan Brute Force yang menggunakan perulangan for untuk menghitung pangkat, sedangkan method Pangkat DC() menggunakan Divide and Conqueror yang menggunakan cara memanggil fungsi untuk menghitung pangkat.

Percobaan 3

1. Buat class Sum dan tambahkan konstruktor pada class Sum

```
    double keuntungan[];

    Sum(int el){
        keuntungan = new double[el];
    }
```

2. Tambahkan method TotalBF untuk menghitung total nilai array dengan iterative

```
    double totalBF(){
        double total = 0;
        for (int i = 0; i < keuntungan.length; i++) {
            total = total+keuntungan[i];
        }
        return total;
    }
```

3. Tambahkan method totalDC() untuk implementasi perhitungan nilai total array menggunakan algoritma Divide and Qonquer

```
    double totalDC(double arr[], int I, int r){
        if (I==r) {
            return arr[I];
        }
        int mid = (I+r)/2;
        double Isum = totalDC(arr, I, mid);
        double rsum = totalDC(arr, mid+1, r);
        return Isum+rsum;
    }
```

4. Buat class MainSum untuk input jumlah elemen

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

5. Tampilkan hasil perhitungan Brute Force dan Divide and Qonquer

```
        Sum sm = new Sum(elemen);
        for (int i = 0; i < elemen; i++) {
            System.out.print("Masukkan keuntungan ke-"+ (i+1)+": ");
            sm.keuntungan[i] = input.nextDouble();
        }
```

6. Tampilkan hasil perhitungan melalui objek yang telah dibuat

```
        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));
```

7. Run kode program

```
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 percobaan 3

1. Kenapa dibutuhkan variable mid pada method TotalDC()?
2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?

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

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

```
return lsum+rsum;
```

4. Apakah base case dari totalDC()?
5. Tarik Kesimpulan tentang cara kerja totalDC()

Jawaban percobaan 3

1. Variable mid pada method TotalDC() diperlukan untuk membagi array menjadi 2 bagian, hal ini sesuai degan konsep Divide and Conquer yaitu membagi masalah menjadi submasalah yang lebih kecil.
2. Kode double Isum = totalDC(arr, I, mid); adalah untuk memproses bagian kiri array, sedangkan kode double rsum = totalDC(arr, mid + 1, r); adalah untuk memproses bagian kanan array.
3. Untuk menggabungkan hasil dari 2 bagian array kanan dan kiri.
4. Base case pada kode ini adalah

```
double totalDC(double arr[], int I, int r){
    if (I==r) {
        return arr[I];
    }
    int mid = (I+r)/2;
    double Isum = totalDC(arr, I, mid);
    double rsum = totalDC(arr, mid+1, r);
    return Isum+rsum;
}
```

5. Cara kerja method totalDC() adalah membagi array menjadi 2 bagian, lalu memproses setiap bagian secara rekursif dan menggabungkan hasilnya.