POLITEINIK NEGERI MALANG

TEKNOLOGI INFORMASI

TEKNIK INFORMATIKA

Nama	: Muhammad Nuril Huda
Kelas	: TI-1A
No	: 19
Mata Kuliah	: Algoritma dan Struktur Data

5.2 Menghitung Nilai Faktorial dengan Algoritma Brute Force dan DivideandConquer

5.2.1 Kode Program

```
package Pertemuan5;

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;

        }

    }

}
```

```
package Pertemuan5;

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();


        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));

    }

}
```

5.2.2 Hasil Percobaan

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

5.2.3 Pertanyaan

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!
    - If adalah base case dari rekursi. Ketika n sudah mencapai 1, proses rekursi akan berhenti dan mengembalikan nilai 1, karena factorial 1 adalah 1. Sedangkan else adalah recursive case, dimana akan terus mengalikan n dengan hasil pemanggilan rekursif faktorialDC(n-1), yang berarti akan menjadi lebih kecil hingga mendekati base case.
2. Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan for? Buktikan!
    - Iya memungkinkan. Bisa menggunakan While.
    - Kode Program

```
package Pertemuan5;

public class Faktorial {

    int faktorialBF(int n) {

        int fakto = 1;

        int i = 1;

        while (i <= n) {

            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);!
    - fakto *= i;
      Ini adalah operasi dalam perulangan yang akan terus mengalikan nilai fakto dengan I di setiap iterasi.
    - int fakto = n * faktorialDC(n-1);
      Ini adala pemanggilan rekursif, dimana n dikalikan dengan hasil pemanggilan faktorialDC(n-1). Proses ini akan terus berjalan sampai mencapai base case.

4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!
    - FaktorialBF() menggunakan perulangan for dan while, sedangkan faktorialDC() menggunakan pemanggilan rekursif.
    - Penggunaan faktorialBF() lebih simple dibandingan dengan faktorialDC().
    - FaktorialBF lebih cepat dan hemat memori dibandikan faktorialDC yang banyak melakukan pemanggilan rekursi.

5.3 Menghitung Hasil Pangkat dengan Algoritma Brute Force dan DivideandConquer

5.3.1 Kode Program

```
package Pertemuan5;

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 Pertemuan5;

import java.util.Scanner;

public class MainPangkat {

    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));

        }

    }

}
```

5.3.2 Hasil Percobaan

```
Masukkan jumlah elemen: 3
Masukkan nilai basis elemen ke-1: 2
Masukkan jumlah elemen: 3
Masukkan nilai basis elemen ke-1: 2
Masukkan nilai basis elemen ke-1: 2
Masukkan nilai pangkat elemen ke-1: 3
Masukkan nilai basis elemen ke-2: 4
Masukkan nilai basis elemen ke-2: 4
Masukkan nilai pangkat elemen ke-2: 5
Masukkan nilai pangkat elemen ke-2: 5
Masukkan nilai basis elemen ke-3: 6
Masukkan nilai pangkat elemen ke-3: 7
Masukkan nilai pangkat elemen ke-3: 7
HASIL PANGKAT BRUTEFORCE:
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
```

5.3.3 Pertanyaan

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!
    - pangkatBF mengunakan perulangan for yaitu dengan cara mengkalikan a sebanyak n.
    - pangkatDC menggunakan perulangan rekursif dan menggunakan Teknik divide and conquer yaitu membagi pangkat menjadi dua.
2. Apakah tahap combine sudah termasuk dalam kode tersebut?Tunjukkan!
    - Iya sudah

    ```
    return (pangkatDC(a, n/2)*pangkatDC(a, n/2)*a);

    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? Jikabisa, seperti apa method pangkatBF() yang tanpa parameter?
    - Bisa

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

4. Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!
    - pangkatBF() menggunakan perulangan for, sedangkan pangkatDC() menggunakan rekursif.
    - pangkatBF() mengkalikan a sebanyak n, sedangkan pangkatDC() membagi pangkat menjadi 2.
    - pangkatBF() lebih mudah dipahami dan lebih hemat sedangkan pangkatDC() lebih cepat untuk pangkat besar.

5.4 Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

5.4.1 Kode Program

```
package Pertemuan5;

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 Pertemuan5;

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 keuntungan menggunakan Divide and Conquer:
"+sm.totalDC(sm.keuntungan,0,elemen-1));

    }


}
```

5.4.2 Hasil Percobaan

```
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
```

5.4.3 Pertanyaan

1. Kenapa dibutuhkan variable mid pada method TotalDC()?
    - Variabel mid digunakan untuk membagi array menjadi dua bagian, yaitu bagian kiri dan bagian kanan.
2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?

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

    - lsum digunakan untuk memanggil rekursi unruk menjumlahkan bagian kiri dari array, sedangkan rsum digunakan untuk memanggil rekursi untuk menjumlahkan bagian kanan dari array.
3. Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?

```
return lsum+rsum;
```

- Karena setelah array dipecah menjadi dua bagian, perlu digabung kembali hasil dari dua bagian tersebut yang berfungsi untuk memberikan jumlah total semua elemen dalam array

4. Apakah base case dari totalDC()?

```
if (l==r){
    return arr[l];
}
```

5. Tarik Kesimpulan tentang cara kerja totalDC()
    - Base case terjadi ketika hanya tersisa satu elemen (l == r), di mana metode langsung mengembalikan elemen tersebut.
    - Menggunakan pendekatan Divide & Conquer dengan cara membagi array menjadi dua bagian, menghitung total dari masing-masing bagian, lalu menggabungkannya kembali.

4.5 Latihan Praktikum

```
package Pertemuan5;

public class NilaiMahasiswa {

    String nama;

    String nim;

    int tahunMasuk;

    int uts;

    int uas;

    NilaiMahasiswa(String nama, String nim, int tahunMasuk, int uts, int uas) {

        this.nama = nama;

        this.nim = nim;

        this.tahunMasuk = tahunMasuk;

        this.uts = uts;

        this.uas = uas;

    }

    double hitungRataRataUAS(NilaiMahasiswa[] mahasiswa) {

        double total = 0;

        for (int i = 0; i < mahasiswa.length; i++) {

            total += mahasiswa[i].uas;

        }

        return total / mahasiswa.length;

    }

    NilaiMahasiswa cariUTSTertinggiDC(NilaiMahasiswa[] mahasiswa, int l, int r) {

        if (l == r) {

            return mahasiswa[l];

        }

        int mid = (l + r) / 2;

        NilaiMahasiswa maxKiri = cariUTSTertinggiDC(mahasiswa, l, mid);

        NilaiMahasiswa maxKanan = cariUTSTertinggiDC(mahasiswa, mid + 1, r);

        return (maxKiri.uts >= maxKanan.uts) ? maxKiri : maxKanan;

    }

    NilaiMahasiswa cariUTSTerendahDC(NilaiMahasiswa[] mahasiswa, int l, int r) {

        if (l == r) {

            return mahasiswa[l];

        }

        int mid = (l + r) / 2;

        NilaiMahasiswa minKiri = cariUTSTerendahDC(mahasiswa, l, mid);

        NilaiMahasiswa minKanan = cariUTSTerendahDC(mahasiswa, mid + 1, r);

        return (minKiri.uts <= minKanan.uts) ? minKiri : minKanan;

    }

}
```

```java
package Pertemuan5;

import java.util.Scanner;

public class MainNilaiMahasiswa {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        NilaiMahasiswa[] mahasiswa = {

            new NilaiMahasiswa("Ahmad", "220101001", 2022, 78, 82),

            new NilaiMahasiswa("Budi", "220101002", 2022, 85, 88),

            new NilaiMahasiswa("Cindy", "220101003", 2021, 90, 87),

            new NilaiMahasiswa("Dian", "220101004", 2021, 76, 79),

            new NilaiMahasiswa("Eko", "220101005", 2023, 92, 95),

            new NilaiMahasiswa("Fajar", "220101006", 2020, 88, 85),

            new NilaiMahasiswa("Gina", "220101007", 2023, 80, 83),

            new NilaiMahasiswa("Hadi", "220101008", 2020, 82, 84)

        };


        NilaiMahasiswa nm = new NilaiMahasiswa("", "", 0, 0, 0);


        NilaiMahasiswa tertinggi = nm.cariUTSTertinggiDC(mahasiswa, 0, mahasiswa.length - 1);

        NilaiMahasiswa terendah = nm.cariUTSTerendahDC(mahasiswa, 0, mahasiswa.length - 1);

        double rataRataUAS = nm.hitungRataRataUAS(mahasiswa);


        System.out.println("Mahasiswa dengan Nilai UTS Tertinggi\t: "+tertinggi.nama+"
("+tertinggi.nim+") - UTS: "+tertinggi.uts);

        System.out.println("Mahasiswa dengan Nilai UTS Terendah\t: "+terendah.nama+"
("+terendah.nim+") - UTS: "+terendah.uts);

        System.out.printf("Rata-rata nilai UAS\t\t\t: "+rataRataUAS);

    }

}
```

```
Mahasiswa dengan Nilai UTS Tertinggi    : Eko (220101005) - UTS: 92
Mahasiswa dengan Nilai UTS Terendah     : Dian (220101004) - UTS: 76
Rata-rata nilai UAS                     : 85.375
```

Link Git Hub: https://github.com/nurilhuda05/Algoritma-dan-Struktur-Data/tree/410fbbe02a14a4fd3863689c29917838095a4ea3/Pertemuan5