

JOBSHEET V

BRUTE FORCE DAN DIVIDE CONQUER

5.1 Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Mahasiswa mampu membuat algoritma bruteforce dan divide-conquer
2. Mahasiswa mampu menerapkan penggunaan algoritma bruteforce dan divide-conquer


5.2 Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

Perhatikan Diagram Class berikut ini:

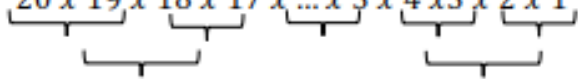
Faktorial
faktorialBF(): int faktorialDC(): int

Berdasarkan diagram class di atas, akan dibuat program class dalam Java. Untuk menghitung nilai faktorial suatu angka menggunakan 2 jenis algoritma, Brute Force dan Divide and Conquer. Jika digambarkan terdapat perbedaan proses perhitungan 2 jenis algoritma tersebut sebagai berikut :

Tahapan pencarian nilai faktorial dengan algoritma Brute Force:

$$20! = 20 \times 19 \times 18 \times 17 \times \dots \times 5 \times 4 \times 3 \times 2 \times 1$$


Tahapan pencarian nilai faktorial dengan algoritma Divide and Conquer:

$$20! = 20 \times 19 \times 18 \times 17 \times \dots \times 5 \times 4 \times 3 \times 2 \times 1$$


5.2.1. Langkah-langkah Percobaan

1. Buat folder baru bernama **Jobsheet5** di dalam repository **Praktikum ASD**
2. Buatlah class baru dengan nama `Faktorial`
3. Lengkapi class `Faktorial` dengan atribut dan method yang telah digambarkan di dalam diagram class di atas, sebagai berikut:



- a) Tambahkan method `faktorialBF()`:

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

- b) Tambahkan method `faktorialDC()`:

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

4. Coba jalankan (Run) class `Faktorial` dengan membuat class baru `MainFaktorial`.

- a) Di dalam fungsi `main` sediakan komunikasi dengan user untuk memasukkan nilai yang akan dicari faktorialnya

```
Scanner input = new Scanner(System.in);
System.out.print(s:"Masukkan nilai: ");
int nilai = input.nextInt();
```

- b) Kemudian 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));
```

- d) Pastikan program sudah berjalan dengan baik!

5.2.2. Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

```
Masukkan nilai: 5
Nilai faktorial 5 menggunakan BF: 120
Nilai faktorial 5 menggunakan DC: 120
```



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!
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()` !

5.3 Menghitung Hasil Pangkat dengan Algoritma Brute Force dan Divide and Conquer

Pada praktikum ini kita akan membuat program class dalam Java, untuk menghitung nilai pangkat suatu angka menggunakan 2 jenis algoritma, Brute Force dan Divide and Conquer. Pada praktikum ini akan digunakan **Array of Object** untuk mengelola beberapa objek yang akan dibuat, berbeda dengan praktikum tahap sebelumnya yang hanya berfokus pada 1 objek faktorial saja.

5.3.1. Langkah-langkah Percobaan

1. Buatlah class baru dengan nama **Pangkat**, dan di dalam class **Pangkat** tersebut, buat atribut angka yang akan dipangkatkan sekaligus dengan angka pemangkatnya

```
int nilai, pangkat;
```

2. Tambahkan konstruktor berparameter

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

3. Pada class **Pangkat** tersebut, 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. Pada class **Pangkat** juga 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. Perhatikan apakah sudah tidak ada kesalahan yang muncul dalam pembuatan class Pangkat
6. Selanjutnya buat class baru yang di dalamnya terdapat method main. Class tersebut dapat dinamakan MainPangkat. Tambahkan kode pada class main untuk menginputkan jumlah elemen yang akan dihitung pangkatnya.

```
Scanner input = new Scanner(System.in);
System.out.print(s:"Masukkan jumlah elemen: ");
int elemen = input.nextInt();
```

7. Nilai pada tahap 5 selanjutnya digunakan untuk instansiasi array of objek. Di dalam Kode berikut ditambahkan proses pengisian beberapa nilai yang akan dipangkatkan sekaligus dengan pemangkatnya.

```
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 pangkat elemen ke-"+(i+1)+" : ");
    int pangkat = input.nextInt();
    png[i] = new Pangkat(basis, pangkat);
}
```

8. Kemudian, panggil hasil nya dengan mengeluarkan return value dari method PangkatBF() dan PangkatDC().

```
System.out.println(x:"HASIL PANGKAT BRUTEFORCE:");
for (Pangkat p : png) {
    System.out.println(p.nilai+"^"+p.pangkat+": "+p.pangkatBF(p.nilai, p.pangkat));
}
System.out.println(x:"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. Verifikasi Hasil Percobaan

Pastikan output yang ditampilkan sudah benar seperti di bawah ini.



```
Masukkan jumlah elemen: 3
Masukan nilai basis elemen ke-1: 2
Masukan nilai pangkat elemen ke-1: 3
Masukan nilai basis elemen ke-2: 4
Masukan nilai pangkat elemen ke-2: 5
Masukan nilai basis elemen ke-3: 6
Masukan 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
```

5.3.3. Pertanyaan

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 melewati 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()` !

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

Di dalam percobaan ini, kita akan mempraktekkan bagaimana proses *divide*, *conquer*, dan *combine* diterapkan pada studi kasus penjumlahan keuntungan suatu perusahaan dalam beberapa bulan.

5.4.1. Langkah-langkah Percobaan

1. Buat class baru yaitu class `Sum`. Tambahkan pula konstruktor pada class `Sum`.

```
double keuntungan[];

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

2. Tambahkan method `TotalBF()` yang akan menghitung total nilai array dengan cara *iterative*.

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



3. Tambahkan pula method `TotalDC()` untuk implementasi perhitungan nilai total array menggunakan algoritma Divide and Conquer

```
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;
}
```

4. Buat class baru yaitu `MainSum`. Di dalam kelas ini terdapat method `main`. Pada method ini user dapat menuliskan berapa bulan keuntungan yang akan dihitung. Dalam kelas ini sekaligus dibuat instansiasi objek untuk memanggil atribut ataupun fungsi pada class `Sum`

```
Scanner input = new Scanner(System.in);
System.out.print(s:"Masukkan jumlah elemen: ");
int elemen = input.nextInt();
```

5. Buat objek dari class `Sum`. Lakukan perulangan untuk mengambil input nilai keuntungan dan masukkan ke atribut keuntungan dari objek yang baru dibuat tersebut!

```
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 untuk kedua cara yang ada (Brute Force dan Divide and Conquer)

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

5.4.2. Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

```
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()` ?
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()`

4.5 Latihan Praktikum

1. Sebuah kampus memiliki daftar nilai mahasiswa dengan data sesuai tabel di bawah ini

Nama	NIM	Tahun Masuk	Nilai UTS	Nilai UAS
Ahmad	220101001	2022	78	82
Budi	220101002	2022	85	88
Cindy	220101003	2021	90	87
Dian	220101004	2021	76	79
Eko	220101005	2023	92	95
Fajar	220101006	2020	88	85
Gina	220101007	2023	80	83
Hadi	220101008	2020	82	84

Tentukan:

- Nilai UTS tertinggi tertinggi menggunakan Divide and Conquer!
- Nilai UTS terendah menggunakan Divide and Conquer!
- Rata-rata nilai UAS dari semua mahasiswa menggunakan Brute Force!