

## Jobsheet 5

### Algoritma dan Struktur Data

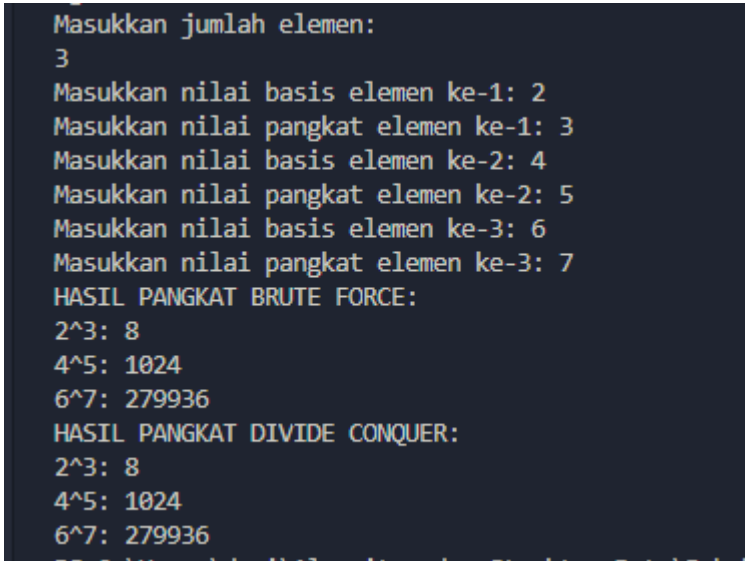
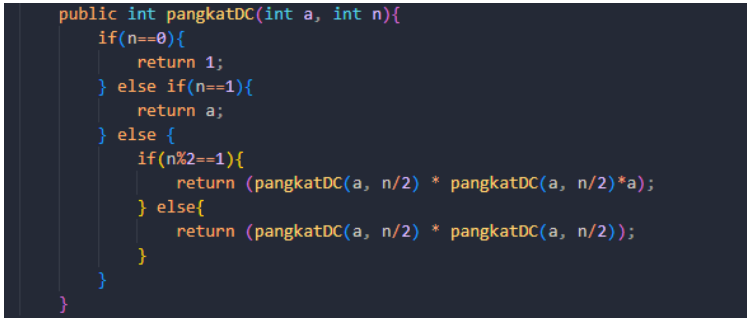
Hernanda Rizka Utami / 244107060075 / SIB – 1B

#### Percobaan 1

No.	Pertanyaan	Jawaban
	Compile n run	<pre>PS C:\Users\dani\Algoritma dan Struktur Data\Jobsheet5&gt; jav Masukkan nilai: 5 Nilai faktorial 5 menggunakan Brute Force: 120 Nilai faktorial 5 menggunakan Devide conquer: 120 PS C:\Users\dani\Algoritma dan Struktur Data\Jobsheet5&gt; █</pre>
1.	Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!	<ul style="list-style-type: none"><li>• Bagian if(n == 1) Base case dari rekrusi. Ketika n mencapai 1, fungsi langsung mengembalikan nilai 1.</li><li>• Bagian else Bagian rekursif case, yang menjalankan perhitungan utama. faktorialDC(n - 1) memanggil dirinya sendiri secara rekursif untuk menghitung faktorial dari angka sebelumnya (n-1). Nilai yang dikembalikan dari rekursi kemudian dikalikan dengan n, sehingga hasil akhirnya adalah n!. proses berlangsung hingga mencapai base case.</li></ul>
2.	Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan for? Buktikan!	<p>Iya, perulangan pada method faktorialBF dapat diubah menggunakan perulangan lain seperti while atau do-while.</p> <pre>public int faktorialBFWhile(int n) {     int fakto = 1;     int i = 1;     while (i &lt;= n) {         fakto *= i;         i++;     }     return fakto; }</pre> <pre>System.out.println("Nilai faktorial " + nilai + " menggunakan Devide conquer: " + 1 System.out.println("Hasil Faktorial dengan WHILE loop: " + fk.faktorialBFWhile(nil</pre>

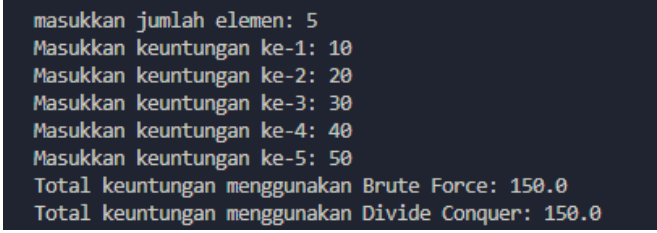
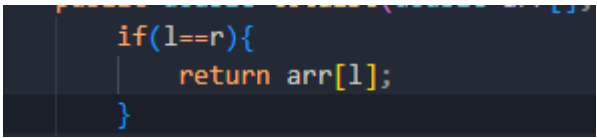
		<p>Nilai faktorial 5 menggunakan Brute Force: 120          Nilai faktorial 5 menggunakan Devide conquer: 120          Hasil Faktorial dengan WHILE loop: 120          PS C:\Users\dani\Algoritma dan Struktur Data\Jobsheet5</p>	
3.	Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !	<ol style="list-style-type: none"> <li>Fakto *= i, digunakan untuk menghitung factorial menggunakan perulangan for atau while.</li> <li>Int fakto = n * faktorialDC(n-1), digunakan untuk metode rekursif (divide dan conquer) untuk menghitung factorial dengan pemanggilan rekrusi.</li> </ol>	
4.	Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!	<ol style="list-style-type: none"> <li>FactorialBF               <ul style="list-style-type: none"> <li>Menggunakan perulangan (for atau while) untuk menghitung faktorial.</li> <li>Memulai dengan fakto = 1, lalu dikalikan secara bertahap hingga n.</li> </ul> </li> <li>faktorialDC               <ul style="list-style-type: none"> <li>Menggunakan <b>rekursi</b>, yaitu metode yang memanggil dirinya sendiri sampai mencapai base case (n == 1).</li> <li>Setiap pemanggilan n * faktorialDC(n-1) menghasilkan pemecahan masalah secara bertahap hingga dasar perhitungan tercapai.</li> </ul> </li> </ol> <p>Kesimpulannya, Jika ingin solusi yang lebih cepat dan efisien dalam penggunaan memori, gunakan <b>faktorialBF()</b> (iteratif). Jika ingin solusi yang lebih elegan dan sesuai dengan konsep rekursi, gunakan <b>faktorialDC()</b> (rekursif).</p>	

## Percobaan 2

No.	Pertanyaan	Jawaban
	Compile n run	 <pre> 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 BRUTE FORCE: 2^3: 8 4^5: 1024 6^7: 279936 HASIL PANGKAT DIVIDE CONQUER: 2^3: 8 4^5: 1024 6^7: 279936 </pre>
1.	Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!	<ol style="list-style-type: none"> <li>Method brute force Menggunakan perulangan untuk mengalikan angka sebanyak angka pangkatnya. Tetapi, tidak efisien ketika angka pangkatnya besar.</li> <li>Method divide conquer Membagi masalah menjadi bagian yang lebih kecil dan menyelesaikan secara rekursif. Jika pangkatnya genap, cukup menghitung setengahnya dan mengkuadratkan hasilnya. Tetapi, jika ganjil menambahkan satu kali perkalian ekstra.</li> </ol>
2.	Apakah tahap <i>combine</i> sudah termasuk dalam kode tersebut? Tunjukkan!	<p>Tahap combine sudah ada pada method pangkatDC</p>  <pre> public int pangkatDC(int a, int n){     if(n==0){         return 1;     } else 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));         }     } } </pre>
3.	Pada method pangkatBF() terdapat parameter untuk melewati nilai yang akan dipangkatkan dan	<p>Tidak, karena pada method pangkatBF() sudah ada atribut n dan p pada kelas yang menyimpan nilai basis dan pangkat. Method pangkatBF() bisa dibuat tanpa parameter, seperti berikut:</p>

	<p>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?</p>	<pre>public int pangkatBF(){     int hasil = 1;     for(int i=0; i&lt; this.p; i++){         hasil *= this.n;     }     return hasil; }</pre> <pre>Pangkat12 p1 = new Pangkat12(2, 5); System.out.println("Hasil pangkat BF: " + p1.pangkatBF());</pre> <pre>6^7: 279936 Hasil pangkat BF: 32 PS C:\Users\dani\Algoritma dan</pre>
4.	<p>Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!</p>	<p>a. Brute force Menghitung pangkat dengan perulangan. Setiap iterasi, nilai dikalikan dengan basis (n) sebanyak (p) kali. Contoh: <math>2^5</math> Iterasi 1 = hasil <math>1 * 2 = 2</math> Iterasi 2 = hasil <math>2 * 2 = 4</math> Iterasi 3 = hasil <math>3 * 2 = 8</math> Iterasi 4 = hasil <math>4 * 2 = 16</math> Iterasi 5 = hasil <math>5 * 2 = 32</math></p> <p>b. Divide conquer Menghitung dengan membagi masalah menjadi sub-masalah lebih kecil. Jika (p) genap, maka bisa dibagi 2. Jika (p) ganjil, maka perlu dikalian satu basis tambahan (n). Contoh: <math>2^5</math> (2,5) 5 ganjil, maka <math>\text{pangkatDC}(2, 2) * \text{pangkatDC}(2, 2) * 2</math> (2, 2) 2 genap, pecah jadi <math>\text{pangkatDC}(2, 1) * \text{pangkatDC}(2, 1)</math> (2, 1) basis rekursi = return 2 Hasil: (2 1) = 2 (2, 2) = 4 (2, 5) = <math>4 * 4 * 2 = 32</math></p>

### Percobaan 3

No	Pertanyaan	Jawaban
.	Compile n run	 <pre> 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 Brute Force: 150.0 Total keuntungan menggunakan Divide Conquer: 150.0 </pre>
1.	Kenapa dibutuhkan variable mid pada method TotalDC()?	Untuk membagi array menjadi dua bagian, yaitu bagian kiri dan kanan.
2.	Untuk apakah statement di bawah ini dilakukan dalam TotalDC()? <pre>double lsum = totalDC(arr, l, mid); double rsum = totalDC(arr, mid+1, r);</pre>	<ol style="list-style-type: none"> <li>lsum = totalDC(arr, l, mid) = untuk memanggil totalDC untuk menghitung jumlah elemen dari indeks 1 sampai mid. Ini bagian kiri dari array yang sedang diproses.</li> <li>rsum = totalDC(arr, mid+1, r) = untuk menghitung jumlah elemen dari indeks mid+1 sampai r. Ini bagian kanan dari array.</li> </ol>
3.	Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini? <pre>return lsum+rsum;</pre>	<ul style="list-style-type: none"> <li>Merupakan langkah terakhir untuk menyusun kembali solusi dari sub-masalah menjadi jawaban utuh.</li> <li>Tanpa combine divide dan conquer hanya membagi masalah tanpa menghasilkan solusi akhir.</li> <li>Hasil akhir dalam divide dan conquer hanya bisa diperoleh jika semua sub-masalah digabung kembali menjadi satu.</li> </ul>
4.	Apakah base case dari totalDC()?	 <pre> if(l==r){     return arr[l]; } </pre> <p>Iya. Base case terjadi ketika l == r, yaitu hanya satu elemen dalam sub-array. Pada base case, rekursi berhenti dan mengembalikan nilai elemen tersebut.</p>
5.	Tarik Kesimpulan tentang cara kerja totalDC()	<ol style="list-style-type: none"> <li><b>Divide</b> Array dibagi menjadi dua bagian secara rekursif menggunakan nilai tengah (mid). Proses ini terus berlangsung hingga mencapai base case (hanya satu elemen dalam subarray).</li> <li><b>Conquer</b> Setiap bagian yang telah dibagi dihitung totalnya secara individu.</li> </ol>

		<p>Jika sudah mencapai base case, nilai elemen tersebut langsung dikembalikan.</p> <p>c. Combine</p> <p>Setelah mendapatkan hasil dari bagian kiri (lsum) dan bagian kanan (rsum), keduanya dijumlahkan.</p> <p>Hasil akhir adalah total dari seluruh elemen array.</p> <p>Kesimpulannya, totalDC() bekerja dengan rekursi untuk membagi array menjadi bagian kecil hingga mencapai satu elemen. Setelah mencapai base case, hasil dari setiap bagian dijumlahkan kembali secara bertahap.</p>
--	--	--

## Latihan Praktikum

No.	Latihan	Jawaban
	Compile n run	<pre>PS C:\Users\dani\Algoritma dan Struktur Data\Jobsheet5&gt; java Nilai UTS tertinggi menggunakan Divide and Conquer: 92 Nilai UTS terendah menggunakan Divide and Conquer: 76 Rata-rata nilai UAS menggunakan Brute Force: 85.375 PS C:\Users\dani\Algoritma dan Struktur Data\Jobsheet5&gt;</pre>
a.	Nilai UTS tertinggi tertinggi menggunakan Divide and Conquer!	<pre>public class Mahasiswa {     public static int maxUTS(int[] uts, int kiri, int kanan) {         if (kiri == kanan) {             return uts[kiri];         }         int mid = (kiri + kanan) / 2;         int kiriMax = maxUTS(uts, kiri, mid);         int kanantMax = maxUTS(uts, mid + 1, kanan);         return Math.max(kiriMax, kanantMax);     } }</pre>
b.	Nilai UTS terendah menggunakan Divide and Conquer!	<pre>public static int minUTS(int[] uts, int kiri, int kanan) {     if (kiri == kanan) {         return uts[kiri];     }     int mid = (kiri + kanan) / 2;     int kiriMax = minUTS(uts, kiri, mid);     int kanantMax = minUTS(uts, mid + 1, kanan);     return Math.min(kiriMax, kanantMax); }</pre>
c.	Rata-rata nilai UAS dari semua mahasiswa menggunakan Brute Force!	<pre>public static double rataUAS(int[] uas) {     int total = 0;     for (int nilai : uas) {         total += nilai;     }     return (double) total / uas.length; }</pre>

		<pre>public static void main(String[] args) {     String[] nama = {"Ahmad", "Budi", "Cindy", "Dian", "Eko", "Fajar", "Gina", "Hadi"};     int[] uts = {78, 85, 90, 76, 92, 88, 80, 82};     int[] uas = {82, 88, 87, 79, 95, 85, 83, 84};      int maxNilaiUTS = maxUTS(uts, 0, uts.length - 1);     int minNilaiUTS = minUTS(uts, 0, uts.length - 1);      double rataNilaiUAS = rataUAS(uas);      System.out.println("Nilai UTS tertinggi menggunakan Divide and Conquer: " + maxNilaiUTS);     System.out.println("Nilai UTS terendah menggunakan Divide and Conquer: " + minNilaiUTS);     System.out.println("Rata-rata nilai UAS menggunakan Brute Force: " + rataNilaiUAS); }</pre>
--	--	---