Laporan Hasil Pratikum

Algoritma Dan Struktur Data

Jobsheet 5



|  |  |
| --- | --- |
| Nama : | Zacky Rio Orlando |
| NIM : | 244107020086 |
| Kelas : | 1E |

Program Studi D-IV Teknik Informatika

Jurusan Teknologi Informasi

Praktikum

2025

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

Kode Program

Class Faktorial

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;

}

}

}

Class MainFaktorial

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));

}

}

Hasil Output



PERTANYAAN

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan  
perbedaan bagian kode pada penggunaan if dan else!

Jawaban :  
- Penggunaan if sebagai kondisi dasar (base case) untuk menghentikan rekursif, Tanpa kondisi dasar ini, rekursi akan berjalan tanpa henti.

- Penggunaan else sebagai melakukan pemecahan masalah dengan memanggil rekursif. Dengan cara mengalikan nilai n saat ini dengan hasil faktorial dari nilai n sebelumnya yaitu n -1, kemudian melakukan pemanggilan rekursif hingga mencapai kondisi dasar pada if.

2. Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan  
for? Buktikan!

Jawaban :

Bisa diubah dengan menggunakan while atau do-while seperti ini kodenya

Menggunakan while

int faktorialBF(int n) {

int fakto = 1;

int i = 1;

while (i <= n) {

fakto = fakto \* i;

i++;

}

return fakto;

}

Menggunakan do-while

int faktorialBF(int n) {

int fakto = 1;

int i = 1;

do {

fakto = fakto \* i;

i++;

} while (i <= n);

return fakto;

}

3. Jelaskan perbedaan antara fakto \*= i; dan int fakto = n \* faktorialDC(n-1); !

Jawaban :

- fakto \*= i digunakan dalam metode iteratif untuk menghitung faktorial dengan perulangan (for, while, do while) dan setiap perulangan berjalan, nilai fakto akan dikalikan dengan nilai I saat ini.

- fakto = n \* faktorialDC(n-1) digunakan dalam metode rekursif untuk menghitung faktorial secara DC dengan memecah masalah 1 menjadi berbagai sub-masalah hingga menjadi lebih kecil sampai kondisi dasar.

4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!

Jawaban :

- method faktorialBF() menggunakan metode iteratif sedangkan faktorialDC() menggunakan metode rekursif

- faktorialBF() menggunakan perulangan untuk menghitung faktorial sedangkan faktorialDC() pemanggil rekursif untuk memecahkan masalah faktorial

- faktorialBF() lebih efisien karena langsung mengerjakan sedangkan faktorialDC() tidak efisien karena harus memecah menjadi beberapa bagian kecil baru dikerjakan.

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

Kode Program

Class Pangkat

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));

}

}

}

}

Class MainPangkat

import java.util.Scanner;

public class MainPangkat {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

int elemen = input.nextInt();

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);

}

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) {

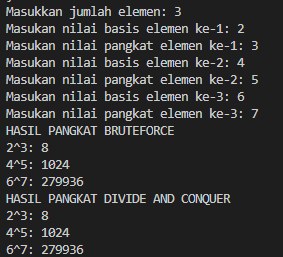
System.out.println(p.nilai + "^" + p.pangkat + ": " + p.pangkatDC(p.nilai, p.pangkat));

}

}

}

Hasil Output



PERTANYAAN

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!

Jawaban :

Method pangkatBF() menggunakan teknik brute force dengan perulangan sebanyak pangkat kali. Method ini sederhana tetapi lambat untuk pangkat besar. Sedangkan method pangkatDC() menggunakan teknik **Divide and Conquer** dengan membagi masalah menjadi lebih kecil secara rekursif. Method ini lebih cepat dan efisien, terutama untuk pangkat besar.

2. Apakah tahap combine sudah termasuk dalam kode tersebut?Tunjukkan!

Jawaban :

Iya sudah termasuk, tahap combine sudah termasuk dalam kode metode pangkatDC() pada kelas Pangkat. Tahap **combine** pada metode **Divide and Conquer** berarti menggabungkan hasil dari submasalah menjadi satu hasil akhir.

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?

Jawaban :

Sebenarnya method pangkatBF() tidak perlu ada parameter karena atribut nilai dan pangkat sudah tersedia di dalam kelas Pangkat. Method tersebut bisa langsung mengakses atribut tersebut tanpa harus menerima parameter.

Method pangkatBF() tanpa parameter

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()!

* Cara kerja method pangkatBF()

1. Inisialisasi variable hasil dengan nilai 1.
2. Lakukan perulangan sebanyak pangkat kali.
3. Pada setiap iterasi, kalikan hasil dengan nilai basis (nilai).
4. Setelah perulangan selesai, kembalikan nilai hasil sebagai hasil perpangkatan.

* Cara kerja method pangkatDC()

1. Jika pangkat sama dengan 1, langsung kembalikan nilai basis (nilai).
2. Jika pangkat ganjil, bagi pangkat menjadi dua, hitung hasil rekursif dua kali, dan kalikan dengan nilai basis. Hasil = pangkatDC(a,n/2) x pangkatDC(a,n/2) x a
3. Jika pangkat genap, bagi pangkat menjadi dua dan kalikan hasil rekursif dua kali. Hasil = pangkatDC(a,n/2) x pangkatDC(a,n/2)
4. Gabungkan hasil dari submasalah (tahap combine) dan kembalikan hasil akhir.

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

Class Sum

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;

}

}

Class MainSum

import java.util.Scanner;

public class MainSum {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

int elemen = input.nextInt();

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();

}

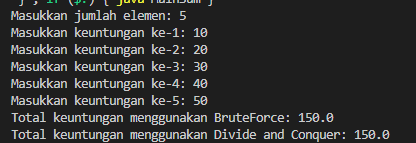
System.out.println("Total keuntungan menggunakan BruteForce: " + sm.totalBF());

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

}

}

Output



PERTANYAAN

1. Kenapa dibutuhkan variable mid pada method TotalDC()?

Jawaban :

Karena digunakan untuk membagi array menjadi dua bagian secara merata. Ini penting dalam teknik **Divide and Conquer** agar masalah besar dapat dipecah menjadi dua submasalah yang lebih kecil. Setelah dihitung secara rekursif, hasil dari kedua bagian tersebut digabungkan menjadi satu total.

2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?



Jawaban :

- lsum menghitung total keuntungan pada bagian kiri dari indeks l hingga mid.

- rsum menghitung total keuntungan pada bagian kanan dari indeks mid+1 hingga r.

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



Jawaban :

Untuk **menggabungkan hasil total dari bagian kiri dan kanan** array yang telah dihitung secara rekursif, sehingga menghasilkan **total keseluruhan** dari seluruh elemen array.  
4. Apakah base case dari totalDC()?

Jawaban :  
Base case dari totalDC()

if (l == r) {

return arr[l];

}

5. Tarik Kesimpulan tentang cara kerja totalDC()

Jawaban :

Method totalDC() bekerja dengan membagi array menjadi dua bagian secara rekursif hingga tersisa satu elemen (base case). Kemudian, hasil dari kedua bagian yaitu kiri dan kanan dijumlahkan kembali untuk mendapatkan total keseluruhan. Meskipun metode ini efektif untuk masalah yang dapat dibagi, pada kasus penjumlahan elemen array, metode ini **kurang efisien** dibandingkan perulangan atau method totalBF().