LAPORAN PRAKTIKUM

Oleh :

MUHAMMAD PEARL OCSHADA

NIM : 244107020064

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2025

NAMA : Muhammad Pearl Ocshada
NIM : 244107020064
KELAS : 1E
TUGAS : Praktikum Algoritma dan Struktur Data

1. Percobaan 1

- Source Code

```
package Praktikum_ASD.ALSD_Jobsheet4;


import java.util.Scanner;


public class mainFaktorial {

   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);

      System.out.print("Masukkan Nilai   :");

      int nilai = sc.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));

   }

}
```

```
package Minggu5;

public class Faktorial {

    int faktorialBF(int n){

        int faktor = 1;

        for (int i = 1 ; i <= n; i++) {

            faktor *= i;

        }

        return fakto;

    }


    int faktorialDC(int n){

        if (n==1) {

            return 1;

        }else{

            int fakto = n * faktorialDC(n-1);

            return fakto;
```

NAMA : Muhammad Pearl Ocshada
NIM : 244107020064
KELAS : 1E
TUGAS : Praktikum Algoritma dan Struktur Data

- Compile And Run

```
Masukkan Nilai    :5
Nilai Faktorial 5 menggunakan BF :120
Nilai Faktorial 5 menggunakan DC :120
```

- Pertanyaan :

1. Dalam metode faktorialDC(), algoritma menggunakan pendekatan rekursif untuk menghitung nilai faktorial.

   - Bagian if digunakan untuk menangani base case yaitu ketika nilai n = 1. Dalam hal ini, fungsi akan langsung mengembalikan nilai 1.
   - Bagian else digunakan untuk menangani recursive case, yaitu ketika n > 1. Dalam hal ini, fungsi akan memanggil dirinya sendiri dengan nilai n - 1 dan mengalikan hasilnya dengan n.

2. perulangan pada method faktorialBF() memungkiankan bisa diubah menggunakan while loop, dan program tetap bisa berjalan dan mengeluarkan hasil yang sama. pembuktian :

```
package Minggu5;

public class Faktorial {

    int faktorialBF(int n){

        int fakto = 1;

        int i =1;

        while (i <= n) {

            fakto *= i;

            i++;

        }

        return fakto;

    }
```

NAMA : Muhammad Pearl Ocshada
NIM : 244107020064
KELAS : 1E
TUGAS : Praktikum Algoritma dan Struktur Data

3. Pada metode faktorialBF(), kode fakto *= i; digunakan dalam perulangan for untuk menghitung nilai faktorial secara bertahap, sedangkan
   Pada metode faktorialDC(), kode int fakto = n * faktorialDC(n-1); digunakan dalam pemanggilan rekursif untuk menghitung faktorial.

   Cara kerja dari fakto *= i; :
   - Memulai dengan nilai fakto = 1.
   - Melakukan perkalian bertahap dengan nilai i yang bertambah dalam setiap iterasi.
   - Nilai akhir dari fakto setelah perulangan selesai adalah hasil faktorial dari n.

   Cara kerja dari int fakto = n * faktorialDC(n-1); :
   - Setiap pemanggilan fungsi akan mengalikan nilai n dengan hasil faktorial dari n-1.
   - Pemanggilan ini berlanjut hingga mencapai base case yaitu ketika n = 1, di mana fungsi mengembalikan nilai 1.
   - Setelah mencapai base case, proses perkalian mulai dihitung dari nilai yang dikembalikan.

4. Metode Brute Force (BF) lebih efisien dalam penggunaan memori, sementara Divide and Conquer (DC) lebih elegan secara konsep tetapi dapat memiliki overhead dalam penggunaan stack rekursi.

   faktorialBF() (Brute Force):
   - Menggunakan perulangan untuk menghitung faktorial.
   - Lebih sederhana dalam implementasi.
   - Tidak memerlukan penggunaan stack rekursi, sehingga lebih efisien dalam penggunaan memori.

   faktorialDC() (Divide and Conquer):
   - Menggunakan rekursi untuk menghitung faktorial.
   - Konsep Divide and Conquer digunakan dengan membagi masalah menjadi sub-masalah yang lebih kecil.
   - Memerlukan tambahan memori untuk menyimpan stack rekursi.

NAMA : Muhammad Pearl Ocshada
NIM : 244107020064
KELAS : 1E
TUGAS : Praktikum Algoritma dan Struktur Data

2. Percobaan 2

- Source Code

```
package Praktikum_ASD.ALSD_Jobsheet4;
import java.util.Scanner;

public class mainPangkat20 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Masukkan Jumlah Elemen :");
      int elemen = sc.nextInt();
      pangkat20[] png = new pangkat20[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 Pangkat basis elemen ke-"+(i+1)+": ");
         int pangkat = sc.nextInt();
         png[i] = new pangkat20(basis, pangkat);
      }
      System.out.println("Hasil Pangkat BruteForce");
      for (pangkat20 p : png) {
         System.out.println(p.nilai +"^"+p.pangkat+": "+p.pangkatBF(p.nilai,
p.pangkat));
      }
      System.out.println("Hasil Pangkat Devide and Conquer");
      for (pangkat20 p : png) {
         System.out.println(p.nilai +"^"+p.pangkat+": "+p.pangkatDC(p.nilai,
p.pangkat));
      }
   }
}
```

NAMA : Muhammad Pearl Ocshada
NIM : 244107020064
KELAS : 1E
TUGAS : Praktikum Algoritma dan Struktur Data

```
 package Praktikum_ASD.ALSD_Jobsheet4;

public class pangkat20 {
   int nilai, pangkat;
   public pangkat20(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)*pangkatBF(a, n/2);
         }
      }
   }
}
```

NAMA : Muhammad Pearl Ocshada
NIM : 244107020064
KELAS : 1E
TUGAS : Praktikum Algoritma dan Struktur Data

- Compile And Run

```
Masukkan Jumlah Elemen :2
Masukkan nilai basis elemen ke-1: 2
Masukkan Pangkat basis elemen ke-1: 5
Masukkan nilai basis elemen ke-2: 2
Masukkan Pangkat basis elemen ke-2: 6
Hasil Pangkat BruteForce
2^5: 32
2^6: 64
Hasil Pangkat Devide and Conquer
2^5: 32
2^6: 64
```

- Pertanyaan :

1. Perbedaan antara method pangkatBF() dan pangkatDC():

   pangkatBF() (Brute Force / Iteratif)
   -Menggunakan perulangan for untuk melakukan perkalian berulang sebanyak n kali.
   -Kompleksitas waktu: O(n) (linear time complexity).
   -Lebih mudah dipahami dan diimplementasikan tetapi tidak optimal untuk nilai pangkat yang besar.

   pangkatDC() (Divide and Conquer / Rekursif)
   -Menggunakan teknik rekursif dan strategi pembagian masalah menjadi submasalah yang lebih kecil.
   -Kompleksitas waktu: O(log n) (logarithmic time complexity), jauh lebih efisien dibandingkan metode brute force.

2. Tahap combine sudah terdapat dalam metode pangkatDC(),
   Combine merupakan Menggabungkan hasil dari submasalah untuk mendapatkan solusi akhir/ penggabungan hasil dari submasalah.
   bagian yang terdapat combine :

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

NAMA : Muhammad Pearl Ocshada
NIM : 244107020064
KELAS : 1E
TUGAS : Praktikum Algoritma dan Struktur Data

3. - Pada method pangkatBF(), terdapat parameter a (basis) dan n (pangkat), padahal atribut nilai dan pangkat sudah ada di dalam class Pangkat07.
   menggunakan parameter dalam method ini sebenarnya tidak terlalu diperlukan, karena nilai basis dan pangkat sudah dapat diakses langsung dari atribut kelas.
   - pangkatBF() bisa dibuat tanpa parameter dengan langsung menggunakan atribut instance nilai dan pangkat dari kelas Pangkat07.

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

4. Cara kerja pangkatBF() (Iteratif / Brute Force) :
   - Memulai dengan hasil = 1.
   - Menggunakan perulangan for untuk melakukan perkalian nilai sebanyak pangkat kali.
   - Mengembalikan hasil akhir

   Cara kerja pangkatDC() (Rekursif / Divide and Conquer) :
   - Jika n == 1, langsung mengembalikan a sebagai hasil.
   - Proses ini berlanjut secara rekursif hingga mencapai base case (n == 1).

NAMA : Muhammad Pearl Ocshada
NIM : 244107020064
KELAS : 1E
TUGAS : Praktikum Algoritma dan Struktur Data

3. Percobaan 3

- Source Code

```
package Praktikum_ASD.ALSD_Jobsheet4;

public class sum20 {
   double keuntungan [];
   sum20(int el){
      keuntungan = new double[el];
   }
   double totalBF(double arr[], int l, int r){
      double total = 0;
      for (int i = 0; i < keuntungan.length; i++) {
         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 = totalBF(arr, mid+l,r);
      return lsum+rsum;
   }
}
```

NAMA : Muhammad Pearl Ocshada
NIM : 244107020064
KELAS : 1E
TUGAS : Praktikum Algoritma dan Struktur Data

```
package Praktikum_ASD.ALSD_Jobsheet4;

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();


      sum20 sm = new sum20(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(sm.keuntungan,0,2));

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

   }

}
```

- Compile And Run

```
Masukkan Jumlah Elemen :3
Masukkan Keuntungan ke-1: 6
Masukkan Keuntungan ke-2: 4
Masukkan Keuntungan ke-3: 5
Total Keuntungan menggunakan BruteForce :15.0
Total Keuntungan menggunakan Devide and Conquer :21.0
```

NAMA : Muhammad Pearl Ocshada
NIM : 244107020064
KELAS : 1E
TUGAS : Praktikum Algoritma dan Struktur Data

- Pertanyaan :

1. Variabel mid digunakan untuk membagi array menjadi dua bagian dalam pendekatan Divide and Conquer.
   Dengan membagi array menjadi dua sub-array yang lebih kecil, algoritma dapat memecah masalah besar menjadi masalah yang lebih kecil hingga mencapai base case

2. Statement ini digunakan untuk menghitung jumlah elemen pada dua bagian array yang lebih kecil.
   memastikan bahwa kita mengakumulasi jumlah elemen array dengan pendekatan rekursif.
   - lsum = totalDC(arr, l, mid); Menghitung total dari bagian kiri array.
   - rsum = totalDC(arr, mid+1, r); Menghitung total dari bagian kanan array.
   Setelah kedua bagian dihitung, hasilnya dijumlahkan dan dikembalikan sebagai hasil akhir

3. Penjumlahan ini diperlukan untuk menggabungkan kembali hasil perhitungan dari dua bagian array yang sudah dipecah sebelumnya.

   - Setelah lsum dan rsum dihitung secara rekursif, kita mengembalikan hasil penjumlahan kedua bagian.
   - Ini merupakan proses penggabungan (conquer) dalam strategi Divide and Conquer.
   - Jika kita tidak menjumlahkan kedua bagian, maka total keuntungan tidak akan dihitung dengan benar.

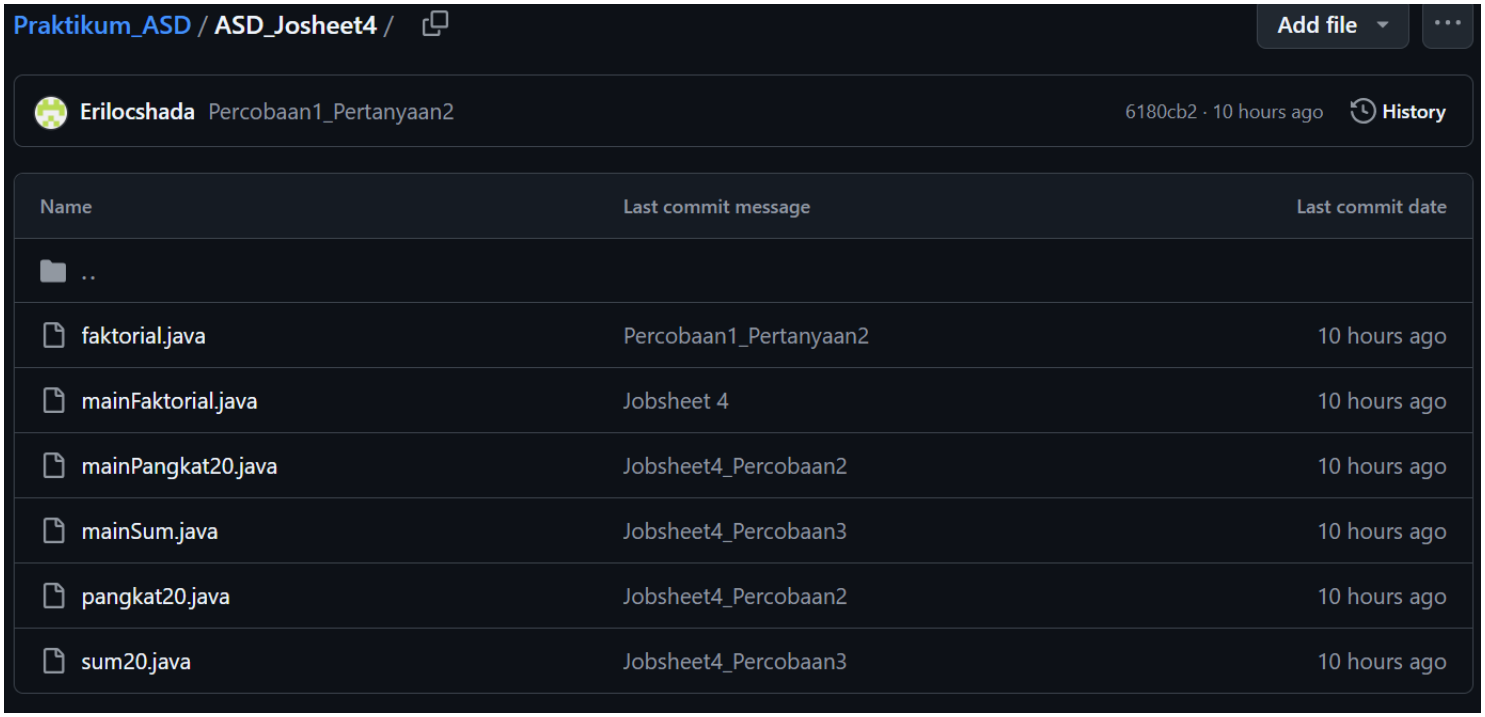
4. Base case dalam metode Divide and Conquer adalah ketika hanya ada satu elemen yang tersisa dalam array.
   - Jika l == r, artinya hanya ada satu elemen yang perlu dikembalikan.
   - Ini adalah kondisi paling dasar dari rekursi, yang menghentikan proses pemecahan lebih lanjut.

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

5. totalDC() menggunakan pendekatan Divide and Conquer untuk menghitung jumlah elemen dalam array secara efisien.
   Divide (Array dibagi menjadi dua bagian dengan menggunakan mid), Conquer (Fungsi totalDC() dipanggil secara rekursif untuk menghitung jumlah setiap bagian),Combine (Hasil dari dua bagian dijumlahkan (lsum + rsum) untuk mendapatkan hasil akhir), Base case (l == r) memastikan rekursi berhenti ketika hanya ada satu elemen yang ters

NAMA : Muhammad Pearl Ocshada
NIM : 244107020064
KELAS : 1E
TUGAS : Praktikum Algoritma dan Struktur Data

Bukti Push Github



NAMA : Muhammad Pearl Ocshada
NIM : 244107020064
KELAS : 1E
TUGAS : Praktikum Algoritma dan Struktur Data