

Objective(s):

- a. To practice problem solving.
- b. Learn how to use recursion, memoization, and dynamic programming (DP) to solve problems.

Create package Lab07\pack.

Task 1 Equal Subsets

Given a list of integers, determine if you can divide it into two subsets where the sum of both subsets is the same. Implement **EqualSubsets.java** with these methods:

- public static boolean canPartition_Recurse(int [] arr)
- public static boolean canPartition_Memoiz(int [] arr)
- public static boolean canPartition_DP(int [] arr)

Example1:

Input -> {1, 5, 11, 5}

Output -> true

Example2:

Input -> {1, 5, 3}

Output -> false

```
static void task_01() {
    int[] a = {1,5,11,5};
    int[] b = {1,5,30};
    int[] c = {1,2,3,5};
    EqualsSubsets sol = new EqualsSubsets();
    System.out.println( sol.canPartition_Recurse(a) );
    System.out.println( sol.canPartition_Memoiz(a) );
    System.out.println( sol.canPartition_DP(a) );
    System.out.println( sol.canPartition_Recurse(b) );
    System.out.println( sol.canPartition_Memoiz(b) );
    System.out.println( sol.canPartition_DP(b) );
    System.out.println( sol.canPartition_Recurse(c) );
    System.out.println( sol.canPartition_Memoiz(c) );
    System.out.println( sol.canPartition_DP(c) );
}
```

Task 2 Grid Paths with Obstacles

A robot starts at the top-left corner of a grid and wants to reach the bottom-right.

- The robot can only move right or down.
- 1 = obstacle, 0 = open space.
- Robot cannot step on obstacles.

Implement **GridPaths.java** with the method

- public int numberOfPaths(int [][] grid)

```
static void task_02() {
    int [][] grid = { {0,0,0,0},
                      {0,1,0,0},
                      {0,0,0,1},
                      {1,0,0,0} };
    GridPaths sol = new GridPaths();
    System.out.println("number of paths: " + sol.numberOfPaths(grid) );
}
```

--- grid paths ---
number of paths: 3

Task 3 Complexity Analysis on Recursion (Do not use master method)3.1 Show time complexity of $T(n) = 2T(n/2) + n$ **Solution**

$$\begin{aligned}
 T(n) &= 2(T(\frac{n}{2})) + n & T(1) &= 1 \\
 &= 2(2T(\frac{n}{2^2}) + \frac{n}{2}) + n & T(n) &= T(1) \\
 &= 2(2T(\frac{n}{2^2})) + n + n & \frac{n}{2^k} &= 1 \\
 &= 2^2T(\frac{n}{2^2}) + 2n & n &= 2^k \\
 &= 2^3T(\frac{n}{2^3}) + 3n & k &= \log_2 n \\
 &= 2^4T(\frac{n}{2^4}) + 4n & T(n) &= 2^kT(\frac{n}{k}) + kn \\
 &\vdots & &= n(T(1)) + n\log_2 n \\
 && (Continue k times) &= n + n\log_2 n
 \end{aligned}$$

$$T(n) = 2^kT(\frac{n}{k}) + kn$$

3.2 $T(n) = T(n - 1) + 1$ **Answer**

$$\therefore T(n) = O(n\log(n))$$

Solution

$$\begin{aligned}
 T(n) &= (T(n - 2) + 1) + 1 & T(0) &= 1 \\
 &= T(n - 2) + 2 & \text{Assume } n - k = 0 \\
 &= T(n - 3) + 3 & \therefore n = k \\
 &= T(n - 4) + 4 & \therefore T(n) = 1 + n \\
 &\vdots & (Continue k times)
 \end{aligned}$$

$$T(n) = T(n - k) + k$$

Answer

$$\therefore T(n) = O(n)$$

Submission:

EqualSubsets_XXYYYY.java, GridPaths_XXYYYY.java and this .pdf file.

Due date: TBA