Lesson:

Dynamic Programming-3

PW SKILLS

Pre Requisites:

- Dynamic Programming

List of concepts involved :

- Catalan Number — Unique BST
- Subset Sum equal K
- Unique Paths - II
- Partition into equal subsets
- Minimum path sum

Catalan Number — Unique BST

Q1. Catalan numbers are defined as a mathematical sequence that consists of positive integers, which can be used to find the number of possibilities of various combinations.

The nth term in the sequence denoted Cn, is found in the following formula: (2n)! / (n + 1)!* n!

The first few Catalan numbers for n = 0, 1, 2, 3, ... are : 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, ...
Given a number n. Print the nth catalan number.

Examples:
Input: n = 6
Output: 132

Input: n = 8
Output: 1430

Solution :
Code : https://pastebin.com/cMcXxWgC

Output :

```
The desired output is : 208012
```

Approach :

- Let's solve this problem. Given a permutation, how many BST are possible for this combination? Let's define the property of BST that the left subtree should have all the members smaller than root and right should have everything greater than root.
- Now let's say you pick k'th element as root then left subtree will contain all element from 1 to k-1 (size k-1) and right subtree will have all the member from k+1 to N (size N-k)
- thus total combination will be f(k-1) * f(n-k) // every possibility of left can go with any possibility of right .
- AS we are free to pick any root node in the first step from 1 to N thus total combination will be
- SUM(f(k-1) * f(N-k)) for all k from 1 to N
- Create an array catalan[] for storing ith Catalan number.
- Initialize, catalan[0] and catalan[1] = 1
- Loop through i = 2 to the given Catalan number n.
- Loop through j = 0 to j < i and Keep adding value of catalan[j] * catalan[i – j – 1] into catalan[i].
- Finally, return catalan[n].

Subset sum equal K

Q2. Given a set of non-negative integers, and a value sum, determine if there is a subset of the given set with sum equal to given sum.

Example:
Input: set[] = {3, 34, 4, 12, 5, 2}, sum = 9
Output: True
There is a subset (4, 5) with sum 9.

Input: set[] = {3, 34, 4, 12, 5, 2}, sum = 30
Output: False
There is no subset that adds up to 30.

Solution :
Code : https://pastebin.com/rGRicYW7

Output :

```
Found a subset with given sum
```

Approach :

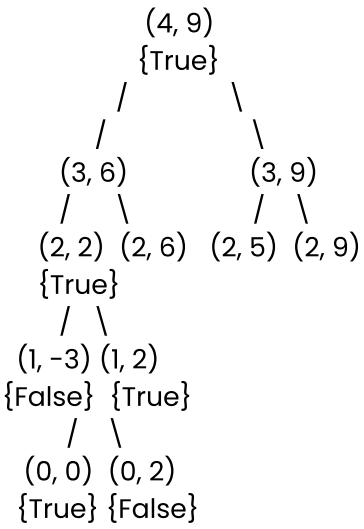
- we will create a 2D array of size (arr.size() + 1) * (target + 1) of type boolean. The state DP[i][j] will be true if there exists a subset of elements from A[0....i] with sum value = 'j'. The approach for the problem is:
  - if (A[i-1] > j)
    - DP[i][j] = DP[i-1][j]
  - else
    - DP[i][j] = DP[i-1][j] OR DP[i-1][j-A[i-1]]
- This means that if current element has value greater than 'current sum value' we will copy the answer for previous cases
- And if the current sum value is greater than the 'ith' element we will see if any of previous states have already experienced the sum='j' OR any previous states experienced a value 'j – A[i]' which will solve our purpose.
- The below simulation will clarify the above approach:

set[]={3, 4, 5, 2}
target=6

	0	1	2	3	4	5	6
0	T	F	F	F	F	F	F
3	T	F	F	T	F	F	F
4	T	F	F	T	T	F	F
5	T	F	F	T	T	T	F
2	T	F	T	T	T	T	T

For example :
set[]={3, 4, 5, 2}

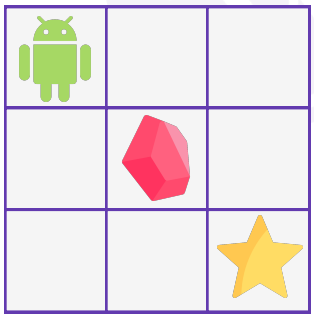
sum=9
(x, y)= 'x' is the left number of elements,
'y' is the required sum



Unique Paths - II :

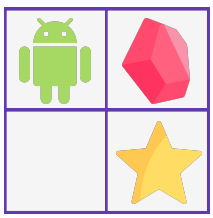
Q3. You are given an m x n integer array grid. There is a robot initially located at the top-left corner (i.e., grid[0][0]). The robot tries to move to the bottom-right corner (i.e., grid[m - 1][n - 1]). The robot can only move either down or right at any point in time.
An obstacle and space are marked as 1 or 0 respectively in the grid. A path that the robot takes cannot include any square that is an obstacle.
Return the number of possible unique paths that the robot can take to reach the bottom-right corner.

Example 1:



Input: obstacleGrid = [[0,0,0],[0,1,0],[0,0,0]]
Output: 2
Explanation: There is one obstacle in the middle of the 3x3 grid above.
There are two ways to reach the bottom-right corner:
1. Right -> Right -> Down -> Down
2. Down -> Down -> Right -> Right

Example 2:



Input: obstacleGrid = [[0,1],[0,0]]
Output: 1

Solution :
Code : https://pastebin.com/gCLDfRyU

Output :

```
3 4
0 1 0 0
0 0 0 1
0 0 0 0
The desired output is : 3
```

Approach :

- In this approach we have checked if the current destination is either out of the grid, or if the value is equal to 1 then no path is possible.
- If we have reached the target cell then we need to return 1.
- We have two paths here, right and down, we have traversed both the paths and stored the result in the dp[i][j] of that cell.
- Next time if we visit the same cell we have the number of paths from that point already available.

Partition into Equal Subsets :

Q4. Given an integer array nums, return true if you can partition the array into two subsets such that the sum of the elements in both subsets is equal or false otherwise.

Example 1:
Input: nums = [1,5,11,5]
Output: true
Explanation: The array can be partitioned as [1, 5, 5] and [11].

Example 2:
Input: nums = [1,2,3,5]
Output: false
Explanation: The array cannot be partitioned into equal sum subsets.

Solution :
Code : https://pastebin.com/zCNwKGjG

Output :

```
4
1 5 11 15
is partition possible : 1
```

Approach :

- Since we have to make two subset both having equal sum then our first condition is to check whether the sum of a given array can be divided in two equal parts which is if total sum is odd then partition is not possible at all and if sum is even then there is chance.
- For example:

a. arr1-> [1,5,11,5] and 2.) arr2 -> [1,5,3,11]

- Both arr1 and arr2 have even sum but 1st can be partitioned into ([1,5,5] & [11]) and 2nd can not.

- In this case we have n elements in array and we have two choices to make whether to keep it in subset1 or subset2 (inclusion in one is direct exclusion in other) and weight of subset will be sum/2.
- So now what our target remains is we have to take care about only one subset because if one subset with weight sum/2 is possible then other subset will surely have the weight sum/2.
- So now using subset sum problem code we have to just check if it's possible to have a subset having sum = totalSum/2;

Minimum Path Sum :

Q5. Given a m x n grid filled with non-negative numbers, find a path from top left to bottom right, which minimizes the sum of all numbers along its path.
Note: You can only move either down or right at any point in time.

Example 1:

1	3	1
1	5	1
4	2	1

Input: grid = [[1,3,1],[1,5,1],[4,2,1]]
Output: 7
Explanation: Because the path 1 → 3 → 1 → 1 → 1 minimizes the sum.

Example 2:
Input: grid = [[1,2,3],[4,5,6]]
Output: 12

Solution :

Code : https://pastebin.com/7uMJHZr1

Output :

```
4 3
2 3 1
4 5 8
3 1 6
4 8 9
The desired minimum sum is : 25
```

Approach :

- we can move either down or right.
- for the grid[0][0] we don't update it.
- so for the 0th row or column we will update the cost by adding the cost before that column :
  - for the 0th row --> grid[i][j] += grid[i][j-1];
  - for the 0th col --> grid[i][j] += grid[i-1][j];
- Now for the other elements we update them adding that block value to MIN of the value from the top element above them and the left element to that block (since we can from either down or right , therefore to reach any element we enter in it either from top of it or left of it).

Upcoming Class Teasers:

- Problems on DP