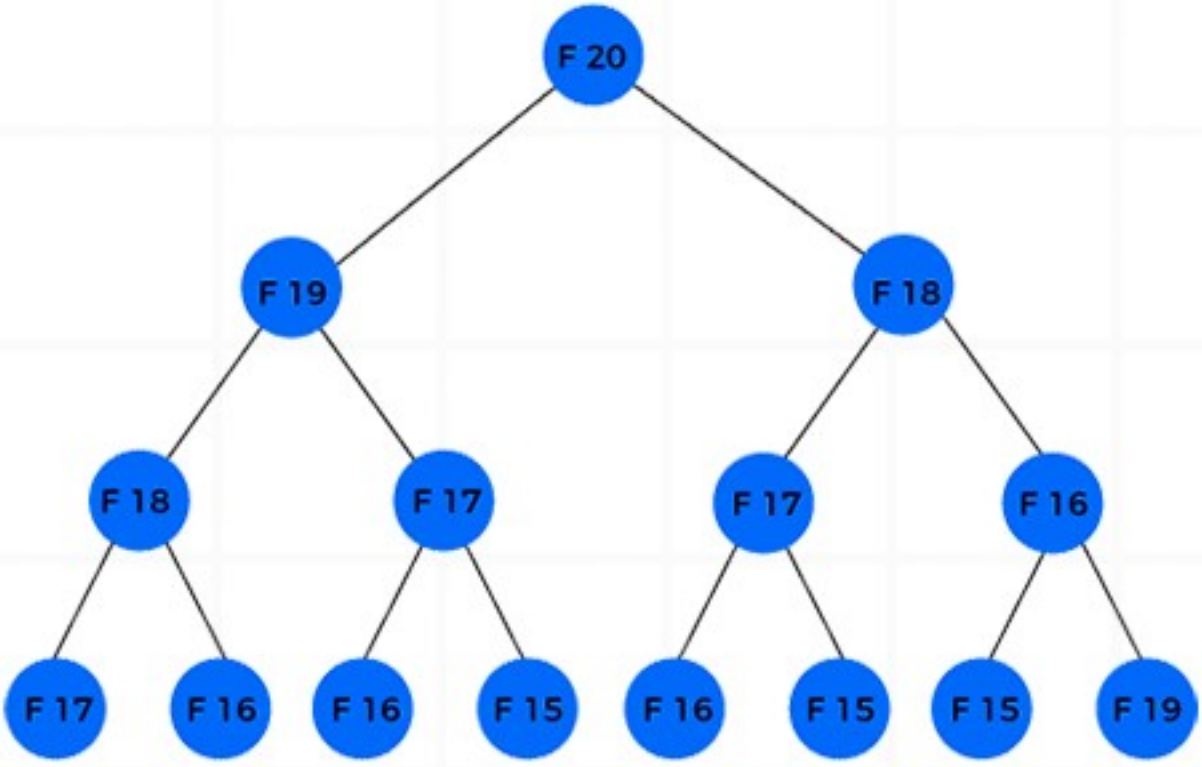
Master

Dynamic Programming

in Just 12 Days!



Challenge accepted?

Advance to PAGE 43 for actual Interview Experience

1-D Dp (Fibonacci Style)

Fibonacci Number

The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is,

F(0) = 0, F(1) = 1
F(n) = F(n - 1) + F(n - 2), for n > 1.

Given n, calculate F(n).

Example 1:

Input: n = 2
Output: 1
Explanation: F(2) = F(1) + F(0) = 1 + 0 = 1.

Example 2:

Input: n = 3
Output: 2

Try problem

Climbing Stairs

You are climbing a staircase. It takes n steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

Example 1:

Input: n = 2
Output: 2
Explanation: There are two ways to climb to the top.
1. 1 step + 1 step
2. 2 steps

Example 2:

Input: n = 3
Output: 3

Try problem

House Robber

Given an integer array nums representing the amount of money of each house, return the maximum amount of money you can rob tonight without alerting the police. Robbing adjacent houses will alert the police via some security system.

Example 1:

Input: nums = [1,2,3,1]
Output: 4
Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3).
Total amount you can rob = 1 + 3 = 4.

Example 2:

Input: nums = [2,7,9,3,1]
Output: 12

Try problem

House Robber 2

Given an integer array nums representing the amount of money of each house, return the maximum amount of money you can rob tonight without alerting the police, given that all houses are arranged in a circle. Robbing adjacent houses will alert the police via some security system.

Example 1:
Input: nums = [2,3,2]
Output: 3
Explanation: You cannot rob house 1 (money = 2) and then rob house 3 (money = 2), because they are adjacent houses.

Example 2:
Input: nums = [1,2,3,1]
Output: 4

Try problem

Knapsack Dp

Equal Sum Partition

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

Try problem

Target Sum

You are given an integer array nums and an integer target. You want to build an expression out of nums by adding one of the symbols '+' and '-' before each integer in nums and then concatenate all the integers.

Return the number of different expressions that you can build, which evaluates to target.

Example:

Input: nums = [1,1,1,1,1], target = 3
Output: 5
Explanation: There are 5 ways to assign symbols to make the sum of nums be target 3.
- 1 + 1 + 1 + 1 + 1 = 3
+1 - 1 + 1 + 1 + 1 = 3
+1 + 1 - 1 + 1 + 1 = 3
+1 + 1 + 1 - 1 + 1 = 3
+1 + 1 + 1 + 1 - 1 = 3

Try problem

Coin Change

You are given an integer array coins representing coins of different denominations and an integer amount representing a total amount of money.

Return the fewest number of coins that you need to make up that amount. If that amount of money cannot be made up by any combination of the coins, return -1. You may assume that you have an infinite number of each kind of coin.

Example 1:

Input: coins = [1,2,5], amount = 11
Output: 3
Explanation: 11 = 5 + 5 + 1

Example 2:

Input: coins = [2], amount = 3
Output: -1

Try problem

Coin Change 2

You are given an integer array coins representing coins of different denominations and an integer amount representing a total amount of money.

Return the number of combinations that make up that amount. If that amount of money cannot be made up by any combination of the coins, return 0. You may assume that you have an infinite number of each kind of coin.

Example 1:

Input: amount = 5, coins = [1,2,5]
Output: 4
Explanation: there are four ways to make up the amount:
5=5
5=2+2+1
5=2+1+1+1
5=1+1+1+1+1

Example 2:

Input: amount = 3, coins = [2]
Output: 0

Try problem

General 1-D Dp

Decode Ways

A message containing letters from A-Z can be encoded into numbers using the following mapping:
'A' -> "1", 'B' -> "2", ..., 'Z' -> "26"

For example, "11106" can be mapped into:
"AAJF" with the grouping (1 1 10 6)
"KJF" with the grouping (11 10 6)

Note that the grouping (1 11 06) is invalid because "06" cannot be mapped into 'F' since "6" is different from "06". Given a string s containing only digits, return the number of ways to decode it.

Example:

Input: s = "12"
Output: 2
Explanation: "12" could be decoded as "AB" (1 2) or "L" (12).

Try problem

Min cost for tickets

You have planned some train traveling one year in advance. The days of the year in which you will travel are given as an integer array days. Each day is an integer from 1 to 365.

Train tickets are sold in three different ways:
a 1-day pass is sold for costs[0] dollars,
a 7-day pass is sold for costs[1] dollars, and
a 30-day pass is sold for costs[2] dollars.
The passes allow that many days of consecutive travel.

For example, if we get a 7-day pass on day 2, then we can travel for 7 days: 2, 3, 4, 5, 6, 7, and 8. Return the minimum number of dollars you need to travel every day in the given list of days.

Example:

Input: days = [1,4,6,7,8,20], costs = [2,7,15]
Output: 11

Try problem

Combination Sum 4

Given an array of distinct integers nums and a target integer target, return the number of possible combinations that add up to target.

Example 1:

Input: nums = [1,2,3], target = 4
Output: 7
Explanation:
The possible combination ways are:
(1, 1, 1, 1)
(1, 1, 2)
(1, 2, 1)
(1, 3)
(2, 1, 1)
(2, 2)
(3, 1)
Note that different sequences are counted as different combinations.

Example 2:
Input: nums = [9], target = 3
Output: 0

Try Problem

Delete and Earn

You are given an integer array nums. You want to maximize the number of points you get by performing the following operation any number of times:

Pick any nums[i] and delete it to earn nums[i] points. Afterwards, you must delete every element equal to nums[i]-1 and every element equal to nums[i]+1.

Return the maximum number of points you can earn by applying the above operation some number of times.

Example:

Input: nums = [3,4,2]
Output: 6
Explanation: You can perform the following operations:
- Delete 4 to earn 4 points. Consequently, 3 is also deleted. nums = [2].
- Delete 2 to earn 2 points. nums = [].
You earn a total of 6 points.

Try problem

Dp on grids

Unique Path

There is a robot on an m x n grid. The robot is 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.

Given the two integers m and n, return the number of possible unique paths that the robot can take to reach the bottom-right corner.

Example:

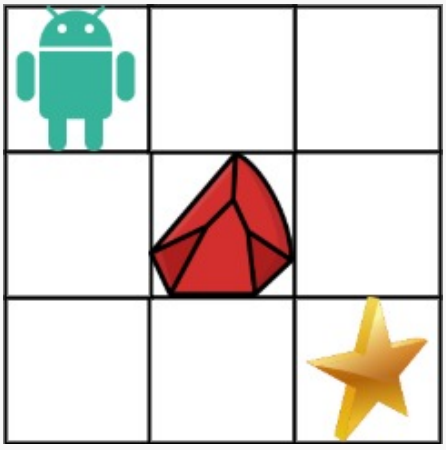
Input: m = 3, n = 7
Output: 28

Unique Paths 2

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 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:



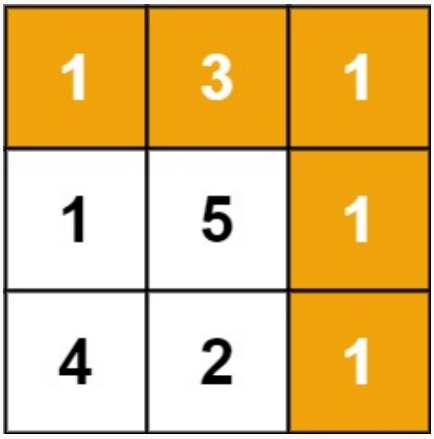
Input: obstacleGrid = [[0,0,0],[0,1,0],[0,0,0]]
Output: 2

Try problem

Minimum Path Sum

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. You can only move either down or right at any point in time.

Example:



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.

Try problem

Triangle

Given a triangle array, return the minimum path sum from top to bottom.

For each step, you may move to an adjacent number of the row below. More formally, if you are on index i on the current row, you may move to either index i or index i + 1 on the next row.

Example:

Input: triangle = [[2],[3,4],[6,5,7],[4,1,8,3]]
Output: 11
Explanation: The triangle looks like:
  2
 3 4
6 5 7
4 1 8 3
The minimum path sum from top to bottom
is 2 + 3 + 5 + 1 = 11 (underlined above).

Try problem

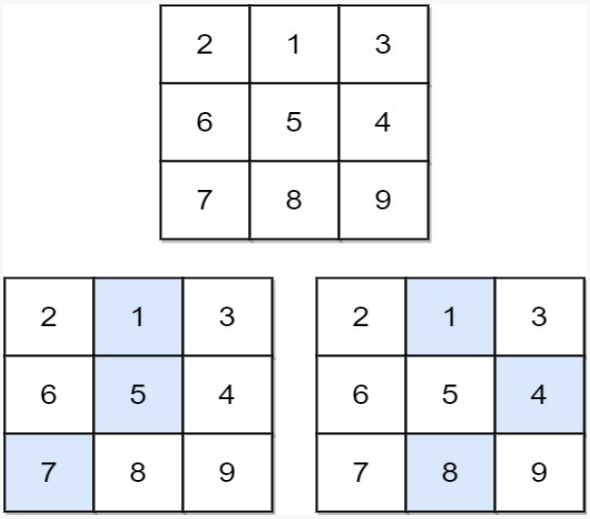
Dp on Grids (Continued)

Minimum Falling Path Sum

Given an n x n array of integers matrix, return the minimum sum of any falling path through matrix.

A falling path starts at any element in the first row and chooses the element in the next row that is either directly below or diagonally left/right. Specifically, the next element from position (row, col) will be (row + 1, col - 1), (row + 1, col), or (row + 1, col + 1).

Example:



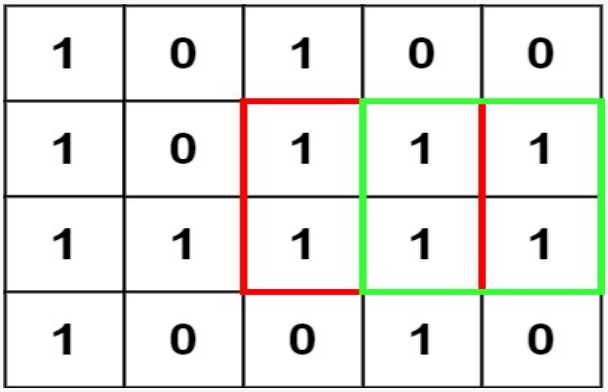
Input: matrix = [[2,1,3],[6,5,4],[7,8,9]]
Output: 13

Try problem

Maximal Square

Given an m x n binary matrix filled with 0's and 1's, find the largest square containing only 1's and return its area.

Example:



Input: matrix = [["1","0","1","0","0"],["1","0","1","1","1"],["1","1","1","1","1"],["1","0","0","1","0"]]
Output: 4

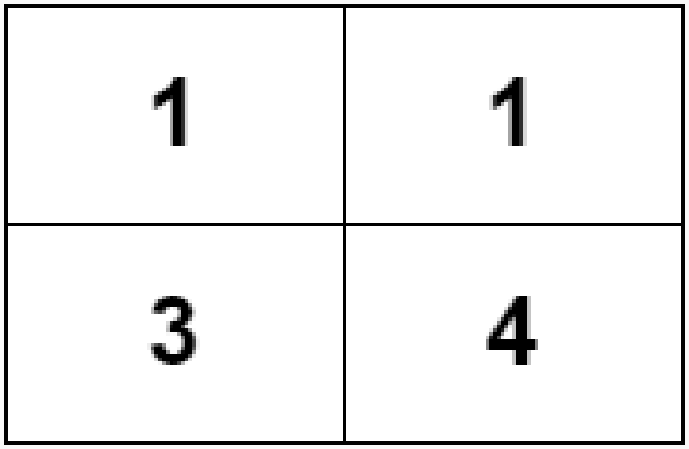
Try problem

Number of increasing paths

You are given an m x n integer matrix grid, where you can move from a cell to any adjacent cell in all 4 directions.

Return the number of strictly increasing paths in the grid such that you can start from any cell and end at any cell. Since the answer may be very large, return it modulo 109 + 7.

Example:



Input: grid = [[1,1],[3,4]]
Output: 8
Explanation: The strictly increasing paths are:
- Paths with length 1: [1], [1], [3], [4].
- Paths with length 2: [1 -> 3], [1 -> 4], [3 -> 4].
- Paths with length 3: [1 -> 3 -> 4].
The total number of paths is 4 + 3 + 1 = 8.

Try problem

Dp on Strings

Longest Common Subsequence

Given two strings text1 and text2, return the length of their longest common subsequence. If there is no common subsequence, return 0.

Example 1:
Input: text1 = "abcde", text2 = "ace"
Output: 3
Explanation: The longest common subsequence is "ace" and its length is 3.

Example 2:
Input: text1 = "abc", text2 = "def"
Output: 0
Explanation: There is no such common subsequence, so the result is 0.

Try problem

Longest Palindromic Subsequence

Given a string s, find the longest palindromic subsequence's length in s.

Example 1:

Input: s = "bbbab"
Output: 4
Explanation: One possible longest palindromic subsequence is "bbbb".

Example 2:
Input: s = "cbbd"
Output: 2
Explanation: One possible longest palindromic subsequence is "bb".

Try problem

Word Break

Given a string s and a dictionary of strings wordDict, return true if s can be segmented into a space-separated sequence of one or more dictionary words.

Note that the same word in the dictionary may be reused multiple times in the segmentation.

Example 1:

Input: s = "leetcode", wordDict = ["leet","code"]
Output: true
Explanation: Return true because "leetcode" can be segmented as "leet code".

Example 2:

Input: s = "catsandog", wordDict = ["cats","dog","sand","and","cat"]
Output: false

Try problem

Min insertions to make a string palindrome

Given a string s. In one step you can insert any character at any index of the string. Return the minimum number of steps to make s palindrome.

Example 1:
Input: s = "zzazz"
Output: 0
Explanation: The string "zzazz" is already palindrome we do not need any insertions.

Example 2:
Input: s = "mbadm"
Output: 2
Explanation: String can be "mbdadbm" or "mdbabdm".

Try problem

Dp on Strings (Continued)

Edit Distance

Given two strings word1 and word2, return the minimum number of operations required to convert word1 to word2.

You have the following three operations permitted on a word:

- Insert a character
- Delete a character
- Replace a character

Example:

Input: word1 = "horse", word2 = "ros"
Output: 3
Explanation:
horse -> rorse (replace 'h' with 'r')
rorse -> rose (remove 'r')
rose -> ros (remove 'e')

Try problem

Distinct Subsequences

Given two strings s and t, return the number of distinct subsequences of s which equals t.

Example:

Input: s = "rabbbit", t = "rabbit"
Output: 3
Explanation:
As shown below, there are 3 ways you can generate "rabbit" from s.

Try problem

Wildcard Matching

Given an input string (s) and a pattern (p), implement wildcard pattern matching with support for '?' and '*' where:

- '?' Matches any single character.
- '*' Matches any sequence of characters (including the empty sequence).

The matching should cover the entire input string (not partial).

Example 1:
Input: s = "aa", p = "a"
Output: false
Explanation: "a" does not match the entire string "aa".

Example 2:

Input: s = "aa", p = "*"
Output: true
Explanation: '*' matches any sequence.

Try problem

Shortest Common Supersequence

Given two strings str1 and str2, return the shortest string that has both str1 and str2 as subsequences. If there are multiple valid strings, return any of them.

Example:

Input: str1 = "abac", str2 = "cab"
Output: "cabac"
Explanation:
str1 = "abac" is a subsequence of "cabac" because we can delete the first "c".
str2 = "cab" is a subsequence of "cabac" because we can delete the last "ac".
The answer provided is the shortest such string that satisfies these properties.

Try problem

LIS Pattern

Longest Increasing Subsequence

Given an integer array nums, return the length of the longest strictly increasing subsequence.

Example 1:
Input: nums = [10,9,2,5,3,7,101,18]
Output: 4
Explanation: The longest increasing subsequence is [2,3,7,101], therefore the length is 4.

Example 2:
Input: nums = [0,1,0,3,2,3]
Output: 4

Try problem

Russian Dolls

You are given a 2D array of integers envelopes where envelopes[i] = [wi, hi] represents the width and the height of an envelope. One envelope can fit into another if and only if both the width and height of one envelope are greater than the other envelope's width and height.

Return the maximum number of envelopes you can Russian doll (i.e., put one inside the other). You cannot rotate an envelope.

Example:

Input: envelopes = [[5,4],[6,4],[6,7],[2,3]]
Output: 3
Explanation: The maximum number of envelopes you can Russian doll is 3 ([2,3] => [5,4] => [6,7]).

Try problem

Largest Divisible Subset

Given a set of distinct positive integers nums, return the largest subset answer such that every pair (answer[i], answer[j]) of elements in this subset satisfies:

- answer[i] % answer[j] == 0, or
- answer[j] % answer[i] == 0

If there are multiple solutions, return any of them.

Example:

Input: nums = [1,2,3]
Output: [1,2]
Explanation: [1,3] is also accepted.

Try problem

Dp + Hashing

Longest Arithmetic Subsequence of Given Difference

Given an integer array arr and an integer difference, return the length of the longest subsequence in arr which is an arithmetic sequence such that the difference between adjacent elements in the subsequence equals difference.

Example 1:

Input: arr = [1,2,3,4,5], difference = 2
Output: 3
Explanation: The longest arithmetic subsequence is [1,3,5].

Example 2:

Input: arr = [1,3,5,7], difference = 1
Output: 1

Try problem

Longest Arithmetic Subsequence

Given an array nums of integers, return the length of the longest arithmetic subsequence in nums.

A sequence seq is arithmetic if seq[i + 1] - seq[i] are all the same value (for 0 <= i < seq.length - 1).

Example 1:

Input: nums = [9,4,7,2,10]
Output: 3
Explanation: The longest arithmetic subsequence is [4,7,10].

Example 2:

Input: nums = [3,6,9,12]
Output: 4

Try problem

Number of Lis

Given an integer array nums, return the number of longest increasing subsequences. Notice that the sequence has to be strictly increasing.

Example 1:
Input: nums = [1,3,5,4,7]
Output: 2
Explanation: The two longest increasing subsequences are [1, 3, 4, 7] and [1, 3, 5, 7].

Example 2:
Input: nums = [2,2,2,2,2]
Output: 5

Try problem

Dp on Stocks

Buy and Sell Stock 1

You are given an array prices where prices[i] is the price of a given stock on the ith day. You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

Example:

Input: prices = [7,1,5,3,6,4]
Output: 5
Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.
Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

Try problem

Buy and Sell Stock 2

You are given an integer array prices where prices[i] is the price of a given stock on the ith day. On each day, you may decide to buy and/or sell the stock.

You can only hold at most one share of the stock at any time. However, you can buy it then immediately sell it on the same day. Find and return the maximum profit you can achieve.

Example:

Input: prices = [7,1,5,3,6,4]
Output: 7
Explanation: Buy on day 2 (price = 1) and sell on day 3 (price = 5), profit = 5-1 = 4.
Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 = 3.
Total profit is 4 + 3 = 7.

Try problem

Buy and Sell Stock 3

You are given an array prices where prices[i] is the price of a given stock on the ith day. Find the maximum profit you can achieve. You may complete at most two transactions.

Note: You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).

Example:

Input: prices = [3,3,5,0,0,3,1,4]
Output: 6
Explanation: Buy on day 4 (price = 0) and sell on day 6 (price = 3), profit = 3-0 = 3.
Then buy on day 7 (price = 1) and sell on day 8 (price = 4), profit = 4-1 = 3.

Try problem

Dp on Stocks (Continued)

Buy and Sell Stock 4

You are given an integer array prices where prices[i] is the price of a given stock on the ith day, and an integer k. Find the maximum profit you can achieve. You may complete at most k transactions: i.e. you may buy at most k times and sell at most k times.

Example:

Input: k = 2, prices = [2,4,1]
Output: 2
Explanation: Buy on day 1 (price = 2) and sell on day 2 (price = 4), profit = 4-2 = 2.

Try problem

Buy and Sell Stock with Cooldown

You are given an array prices where prices[i] is the price of a given stock on the ith day. Find the maximum profit you can achieve. You may complete as many transactions as you like (i.e., buy one and sell one share of the stock multiple times) with the following restrictions:

After you sell your stock, you cannot buy stock on the next day (i.e., cooldown one day).

Example:

Input: prices = [1,2,3,0,2]
Output: 3
Explanation: transactions = [buy, sell, cooldown, buy, sell]

Try problem

Buy and Sell Stock with Transaction Fee

You are given an array prices where prices[i] is the price of a given stock on the ith day, and an integer fee representing a transaction fee. Find the maximum profit you can achieve. You may complete as many transactions as you like, but you need to pay the transaction fee for each transaction.

The transaction fee is only charged once for each stock purchase and sale.

Example:

Input: prices = [1,3,2,8,4,9], fee = 2
Output: 8
Explanation: The maximum profit can be achieved by:
- Buying at prices[0] = 1
- Selling at prices[3] = 8
- Buying at prices[4] = 4
- Selling at prices[5] = 9
The total profit is ((8 - 1) - 2) + ((9 - 4) - 2) = 8.

Try Problem

Partition Dp

Palindrome Partitioning 2

Given a string s, partition s such that every substring of the partition is a palindrome. Return the minimum cuts needed for a palindrome partitioning of s.

Example 1:
Input: s = "aab"
Output: 1
Explanation: The palindrome partitioning ["aa","b"] could be produced using 1 cut.

Example 2:
Input: s = "a"
Output: 0

Try Problem

Burst Balloons

You are given n balloons, indexed from 0 to n - 1. Each balloon is painted with a number on it represented by an array nums. You are asked to burst all the balloons.

If you burst the ith balloon, you will get nums[i - 1] * nums[i] * nums[i + 1] coins. If i - 1 or i + 1 goes out of bounds of the array, then treat it as if there is a balloon with a 1 painted on it. Return the maximum coins you can collect by bursting the balloons wisely.

Example:

Input: nums = [3,1,5,8]
Output: 167
Explanation:
nums = [3,1,5,8] --> [3,5,8] --> [3,8] --> [8] --> []
coins = 3*1*5 + 3*5*8 + 1*3*8 + 1*8*1 = 167

Try problem

Scramble String

We can scramble a string s to get a string t using the following algorithm:

- If the length of the string is 1, stop.
- If the length of the string is > 1, do the following:
- Split the string into two non-empty substrings at a random index, i.e., if the string is s, divide it to x and y where s = x + y.
- Randomly decide to swap the two substrings or to keep them in the same order. i.e., after this step, s may become s = x + y or s = y + x.
- Apply step 1 recursively on each of the two substrings x and y.

Given two strings s1 and s2 of the same length, return true if s2 is a scrambled string of s1, otherwise, return false.

Example 1:
Input: s1 = "great", s2 = "rgeat"
Output: true

Example 2:
Input: s1 = "abcde", s2 = "caebd"
Output: false

Try problem

BONUS SECTION

🚀 How to Approach a Dp Problem in an Interview?

Problem:
We are given an array arr with n positive integers and an integer k. We need to find the number of subsets whose sum is equal to k.

Example:

Input: arr = [1, 3, 3, 4], k = 4
Output: 3
Explanation:
The possible ways are:
[1, 3]
[1, 3]
[4]
Hence the output will be 3. Please note that both 3 present in arr are treated differently.

Approach to solve it

1 First give the recursive solution to the interviewer along with time and space complexities.

```cpp
int solve(vector<int>&arr, int n, int k){
    if(k == 0) return 1;
    if(n == 0) return 0;


    int include = 0;
    if(k >= arr[n-1])
        include = solve(arr, memo, n-1, k-arr[n-1]);


    int exclude = solve(arr, memo, n-1, k);
    return include + exclude;
}


int countSubsets(vector<int>&arr, int k) {
   return solve(arr, arr.size(), k);
}


// Time Complexity - O(2^N)
// Space Complexity - O(N)
```

2 Then memorize the solution (top-down dp). Tell the interviewer the improved time and space complexities.

```cpp
int solve(vector<int>&arr, vector<vector<int>>&memo, int n, int k){
    if(k == 0) return 1;
    if(n == 0) return 0;
    if(memo[n][k] != -1) return memo[n][k];


    int include = 0;
    if(k >= arr[n-1])
        include = solve(arr, memo, n-1, k-arr[n-1]);


    int exclude = solve(arr, memo, n-1, k);
    return memo[n][k] = include + exclude;
}

int countSubsets(vector<int>&arr, int k) {
   int n = arr.size();
   vector<vector<int>>memo(n+1, vector<int>(k+1, -1));
   return solve(arr, memo, n, k);
}


// Time Complexity - O(N*K)
// Space Complexity - O(N*K) + O(N)
```

3 Then present the tabulation or bottom-up dp solution. Don't forget to tell the interviewer time and space complexities.

```cpp
int countSubsets(vector<int>&arr, int k) {
    int n = arr.size();
    vector<vector<int>>dp(n+1, vector<int>(k+1, 0));


    for(int i = 0; i <= n; i++){
        for(int j = 0; j <= k; j++){
            if(j == 0) dp[i][j] = 1;
            else if(i == 0) dp[i][j] = 0;
            else{
                int include = 0;
                if(j >= arr[i-1])
                    include = dp[i-1][j-arr[i-1]];
                int exclude = dp[i-1][j];
                dp[i][j] = include + exclude;
            }
        }
    }

    return dp[n][k];
}


// Time Complexity - O(N*K)
// Space Complexity - O(N*K)
```

4 Finally, apply space optimization if possible. Tell the interviewer the improved space complexity.

```cpp
int countSubsets(vector<int>&arr, int k) {
    vector<int> prev(k+1, 0), curr(k+1, 0);


    for(int i = 0; i <= n; i++){
        for(int j = 0; j <= k; j++){
            if(j == 0) curr[j] = 1;
            else if(i == 0) curr[j] = 0;
            else{
                int include = 0;
                if(j >= arr[i-1])
                    include = prev[j-arr[i-1]];
                int exclude = prev[j];
                curr[j] = include + exclude;
            }
        }
        prev = curr;
    }


    return prev[k];
}
// Time Complexity - O(N*K)
// Space Complexity - O(K)
```

Struggling with DP?

You're not alone. These are the types of questions that can make or break your MAANG interviews.

Dont't let tough coding questions stump you. With HeyCoach, you're not just prepared you're MAANG - ready!

1000+ engineers chose HeyCoach to land a job at a top product company.

Engineers land an average CTC of **27 LPA** with HeyCoach

Engineers got an average of **300% hike** through HeyCoach

100% placement assistance

Signup Now

(link in caption)