1.

64. Minimum Path Sum

Solved

Medium | Topics | Companies

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]]`

```
class Solution {
    public int minPathSum(int[][] grid) {
        int m = grid.length, n = grid[0].length;
        for (int j = 1; j < n; j++) {
            grid[0][j] += grid[0][j - 1];
        }
        for (int i = 1; i < m; i++) {
            grid[i][0] += grid[i - 1][0];
        }
        for (int i = 1; i < m; i++) {
            for (int j = 1; j < n; j++) {
                grid[i][j] += Math.min(grid[i - 1][j], grid[i][j - 1]);
            }
        }

        return grid[m - 1][n - 1];
    }
}
```

2.

45. Jump Game II

Solved

Medium | Topics | Companies

You are given a **0-indexed** array of integers `nums` of length `n`. You are initially positioned at `nums[0]`.

Each element `nums[i]` represents the maximum length of a forward jump from index `i`. In other words, if you are at `nums[i]`, you can jump to any `nums[i + j]` where:

- `0 <= j <= nums[i]` and
- `i + j < n`

Return *the minimum number of jumps to reach* `nums[n - 1]`. The test cases are generated such that you can reach `nums[n - 1]`.

Example 1:

> **Input:** nums = [2,3,1,1,4]
> **Output:** 2
> **Explanation:** The minimum number of jumps to reach the last index is 2. Jump 1 step from index 0 to 1, then 3 steps to the last index.

```java
class Solution {
    public int jump(int[] nums) {
        int near = 0, far = 0, jumps = 0;

        while (far < nums.length - 1) {
            int farthest = 0;
            for (int i = near; i <= far; i++) {
                farthest = Math.max(farthest, i + nums[i]);
            }
            near = far + 1;
            far = farthest;
            jumps++;
        }

        return jumps;
    }
}
```

3.

49. Group Anagrams

Solved

Medium | Topics | Companies

Given an array of strings `strs`, group the anagrams together. You can return the answer in **any order**.

Example 1:

> **Input:** `strs = ["eat","tea","tan","ate","nat","bat"]`
>
> **Output:** `[["bat"],["nat","tan"],["ate","eat","tea"]]`
>
> **Explanation:**
>
> - There is no string in strs that can be rearranged to form `"bat"`.
> - The strings `"nat"` and `"tan"` are anagrams as they can be rearranged to form each other.
> - The strings `"ate"`, `"eat"`, and `"tea"` are anagrams as they can be rearranged to form each other.

```java
class Solution {
    public List<List<String>> groupAnagrams(String[] strs) {
        Map<String, List<String>> ans = new HashMap<>();

        for (String s : strs) {
            char[] chars = s.toCharArray();
            Arrays.sort(chars);
            String key = new String(chars);
            if (!ans.containsKey(key)) {
                ans.put(key, new ArrayList<>());
            }
            ans.get(key).add(s);
        }

        return new ArrayList<>(ans.values());
    }
}
```

4.

91. Decode Ways

Solved

Medium | Topics | Companies

You have intercepted a secret message encoded as a string of numbers. The message is **decoded** via the following mapping:

`"1" -> 'A'`

`"2" -> 'B'`

`...`

`"25" -> 'Y'`

`"26" -> 'Z'`

However, while decoding the message, you realize that there are many different ways you can decode the message because some codes are contained in other codes (`"2"` and `"5"` vs `"25"`).

For example, `"11106"` can be decoded into:

- `"AAJF"` with the grouping `(1, 1, 10, 6)`
- `"KJF"` with the grouping `(11, 10, 6)`
- The grouping `(1, 11, 06)` is invalid because `"06"` is not a valid code (only `"6"` is valid).

Note: there may be strings that are impossible to decode.

```java
class Solution {
    public int numDecodings(String s) {
        int strLen = s.length();

        int[] dp = new int[strLen + 1];
        dp[0] = 1;

        if (s.charAt(0) != '0') {
            dp[1] = 1;
        } else {
            return 0;
        }

        for (int i = 2; i <= strLen; ++i) {
            if (s.charAt(i - 1) != '0') {
                dp[i] += dp[i - 1];
            }
            if (s.charAt(i - 2) == '1' || (s.charAt(i - 2) == '2' && s.charAt(i - 1)
<= '6')) {
                dp[i] += dp[i - 2];
            }
        }

        return dp[strLen];
    }
}
```

5.

122. Best Time to Buy and Sell Stock II

Solved

Medium | Topics | Companies

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 1:

> **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.

Example 2:

> **Input:** prices = [1,2,3,4,5]
> **Output:** 4
> **Explanation:** Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4.
> Total profit is 4.

```java
class Solution {
    public int maxProfit(int[] prices) {
        int profit = 0;

        for (int i = 1; i < prices.length; i++) {
            if (prices[i] > prices[i - 1]) {
                profit += prices[i] - prices[i - 1];
            }
        }

        return profit;
    }
}
```