63. Unique Paths II
We will use 2-d dp. TC is O(m*n), SC is O(m*n)

```
class Solution:
    def uniquePathsWithObstacles(self, obstacleGrid: List[List[int]]) -> int:
        if not obstacleGrid or not obstacleGrid[0]:
          return 0
        if obstacleGrid[0][0] == 1 or obstacleGrid[-1][-1] == 1:
          return 0
        rows = len(obstacleGrid)
        cols = len(obstacleGrid[0])

        dp = [[0 for j in range(cols)] for i in range(rows)]
        dp[0][0] = 1
        for j in range(1, cols):
          if obstacleGrid[0][j] == 1:
            dp[0][j] = 0
          else:
            dp[0][j] = dp[0][j - 1]

        for i in range(1, rows):
          if obstacleGrid[i][0] == 1:
            dp[i][0] = 0
          else:
            dp[i][0] = dp[i - 1][0]

        for i in range(1, rows):
          for j in range(1, cols):
            if obstacleGrid[i][j] == 1:
              dp[i][j] = 0
            else:
              dp[i][j] = dp[i - 1][j] + dp[i][j - 1]
        return dp[-1][-1]
```

64. Minimum Path Sum
We will use 2d dp to get minimum path sum. TC is O(m*n), SC is O(m*n)

```
class Solution:
    def minPathSum(self, grid: List[List[int]]) -> int:
        m = len(grid)
        n = len(grid[0])
        for i in range(1, m):
            grid[i][0] += grid[i - 1][0]

        for j in range(1, n):
```

```
            grid[0][j] += grid[0][j - 1]

        for i in range(1, m):
            for j in range(1, n):
                grid[i][j] += min(grid[i - 1][j], grid[i][j - 1])

        return grid[-1][-1]
```

62. Unique Paths

```
class Solution:
    def uniquePaths(self, m: int, n: int) -> int:
        m_map = [[1 for j in range(n)] for i in range(m)]
        for i in range(1, m):
            for j in range(1, n):
                m_map[i][j] = m_map[i][j - 1] + m_map[i - 1][j]
        return m_map[-1][-1]
```

120. Triangle

```
class Solution:
    def minimumTotal(self, triangle: List[List[int]]) -> int:
        rows = len(triangle)
        result = float('inf')
        for i in range(1, rows):
            for j, num in enumerate(triangle[i]):
                if j == 0:
                    triangle[i][j] += triangle[i - 1][j]
                elif j == i:
                    triangle[i][j] += triangle[i - 1][j - 1]
                else:
                    triangle[i][j] += min(triangle[i - 1][j - 1], triangle[i - 1][j])
        return min(triangle[-1])
```

214. Shortest Palindrome

Brute force

```
var shortestPalindrome = function(s) {
    const r = s.split('').reverse().join('');
    for (let i = 0; i < s.length; i++) {
      if (s.startsWith(r.substring(i, r.length))) {
        return r.substring(0, i) + s;
      }
    }

    return r + s;
```

```
};
```