1. class Solution:

    def findMinArrowShots(self, points: List[List[int]]) -> int:

        points.sort(key=lambda x:x[1])

        res, curEnd = 1, points[0][1]

        for start,end in points:

            if start>curEnd:

                curEnd = end

                res += 1

        return res

2. class Solution:

    memo = []

    def uniquePaths(self, m: int, n: int) -> int:

        self.memo = [[0] \* n for \_ in range(m)]

        return self.dp(m - 1, n - 1)

    def dp(self, x: int, y: int) -> int:

        if x == 0 and y == 0:

            return 1

        if x < 0 or y < 0:

            return 0

        if self.memo[x][y] > 0:

            return self.memo[x][y]

        self.memo[x][y] = self.dp(x - 1, y) + self.dp(x, y - 1)

        return self.memo[x][y]

3. class Solution:

    def longestCommonSubsequence(self, text1: str, text2: str) -> int:

        memo = [[0 for \_ in range(len(text2) + 1)] for \_ in range(len(text1) + 1) ]

        for i in range(len(text1)):

            for j in range(len(text2)):`

                if text1[i] == text2[j]:

                    memo[i+1][j+1] = memo[i][j] + 1

                else:

                    memo[i+1][j+1] = max(memo[i+1][j], memo[i][j+1])

        return max((max(row) for row in memo))