DYNAMIC PROGRAMMING

DP CONCEPTS

- State
  - Unit information, e.g. cost, steps, counts,...etc
- Function
  - Relations about state transition
- Initialization
  - Initialized state

PROBLEM LIST

- No. 53 Maximum Subarray (Medium)
- No. 62 Unique Paths (Medium)
- No. 63 Unique Paths II (Medium)
- No. 64 Minimum Path Sum (Medium)
- No. 70 Climbing Stairs (Easy)
- No. 91 Decode Ways (Medium)
- No. 96 Unique Binary Search Trees (Medium)
- No. 120 Triangle (Medium)
- No. 139 Word Break (Medium)
- No. 152 Maximum Product Subarray (Medium)
- No. 198 House Robber (Easy)
- No. 213 House Robber II (Medium)
- No. 221 Maximal Square (Medium)
- No. 309 Best Time to Buy and Sell Stock with Cooldown (Medium)
- No. 322 Coin Change (Medium)
- No. 357 Count Numbers with Unique Digits (Medium)
- No. 368 Largest Divisible Subset (Medium)
- No. 375 Guess Number Higher or Lower II (Medium)
- No. 377 Combination Sum IV (Medium)
- No. 416 Partition Equal Subset Sum (Medium)
- No. 467 Unique Substrings in Wraparound String (Medium)
- No. 472 Concatenated Words (Medium)
- No. 474 Ones and Zeroes (Medium)
- …etc (hard and locked)

PROBLEM CATEGORIES

- Matrix type
- Sequence type
  - Single sequence
  - Multiple sequences
- Backpack type

MATRIX DP PROBLEMS

- State
  - Matrix is usually represented as 2D array, say, matrix[y][x]
  - State[y][x] is usually represented as the cost/steps/counts accumulated from initial state[0][0] to state[y][x]
- Function
  - Strategies/policies about how we can reach (x,y) from the previous state, e.g. (x-1, y-1)
- Initialization
  - Initial state

NO. 64

MINIMUM PATH SUM

PROBLEM DESCRIPTION

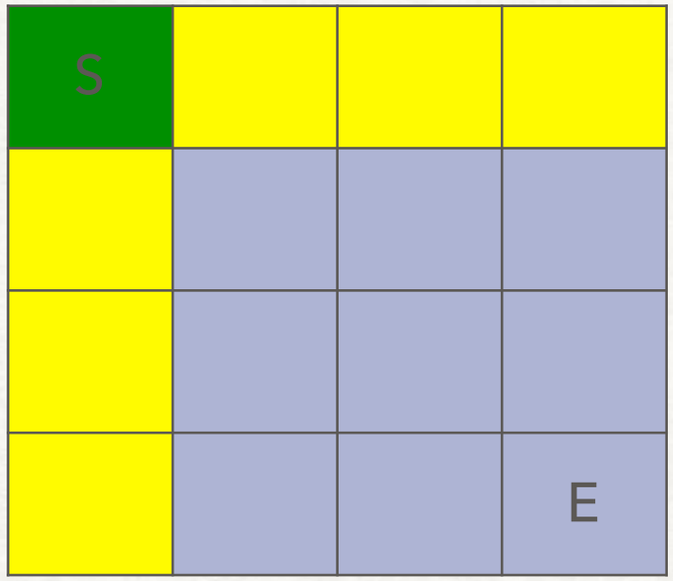
- Given a m x n grid filled with non-negative numbers, find a path from top left to bottom right which minimizes the sum of all numbers along its path.
- Note: You can only move either down or right at any point in time.

IDEAS

- Non-negative cost
  - We don’t have to consider go-back path
- No diagonal moving, e.g. state[y-1][x-1] to state[y][x]
- Initialization : init state[0][0] = cost[0][0]
- State: accumulated state cost
- Formula: state[y][x] = cost[y][x] + min(state[y-1][x], state[y][x-1])

TRICKS

- Matrix type questions
  - Green
    - Set to init cost
  - Yellow
    - To achieve yellow grids, there is only one path
      - Go straight right or go straight down
      - Calculate the states of these yellow grids first!
- Purple
  - To achieve purple grids, there could be multiple paths
    - state[y][x] = grid[y][x] + min(state[y-1][x], state[y][x-1])
    - DP!



SOLUTION

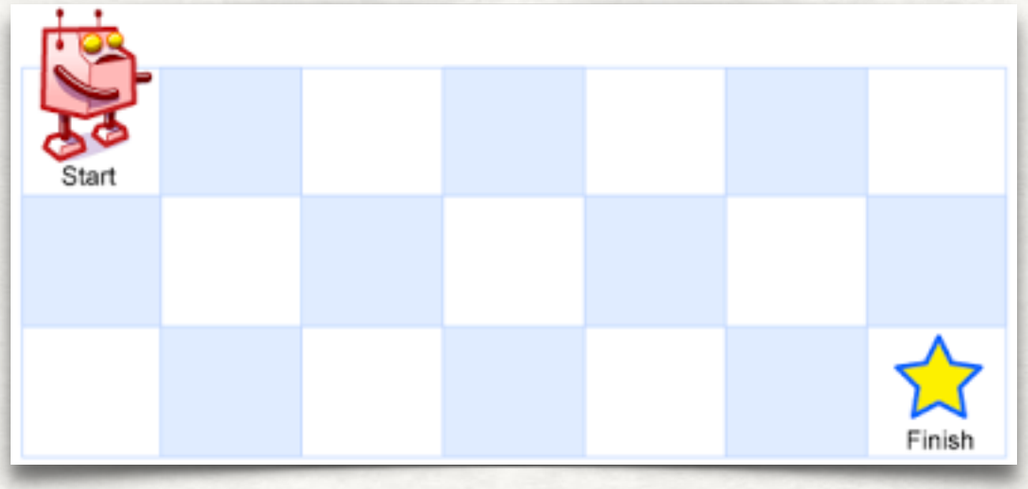
- https://github.com/Brady31027/leetcode/tree/master/64_Minimum_Path_Sum

NO. 62

UNIQUE PATHS

PROBLEM DESCRIPTION

- A robot is located at the top-left corner of a m x n grid (marked 'Start' in the diagram below). The robot can only move either down or right at any point in time. The robot is trying to reach the bottom-right corner of the grid (marked 'Finish' in the diagram below). How many possible unique paths are there?



IDEAS

- Similar to 64. Minimum Path Sum except
  - State[y][x] = state[y][x-1] + state[y-1][x]
- Steps
- Init origin, matrix[o][x], and matrix[y][0] to “1”
- Evaluate other states by DP

1	1	1
1	2	3

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/62_Unique_Paths

NO. 63

UNIQUE PATHS II

PROBLEM DESCRIPTION

- Now consider if some obstacles are added to the grids. How many unique paths would there be? An obstacle and empty space is marked as 1 and 0 respectively in the grid. For example, There is one obstacle in the middle of a 3x3 grid as illustrated below.

- [[0,0,0], [0,1,0], [0,0,0]]

- The total number of unique paths is 2.

- Note: m and n will be at most 100.

IDEAS

- Similar to 62. Unique Paths except
  - Overwrite those grids to “0” if they are unachievable

SOLUTION

- https://github.com/Brady31027/leetcode/blob/master/63_Unique_Paths_II/