Name: Haoze He
NYU ID: hh2537

EL9343 Homework 10

(Due Nov 29th, 2021)

No late assignments accepted

All problem/exercise numbers are for the third edition of CLRS text book

---

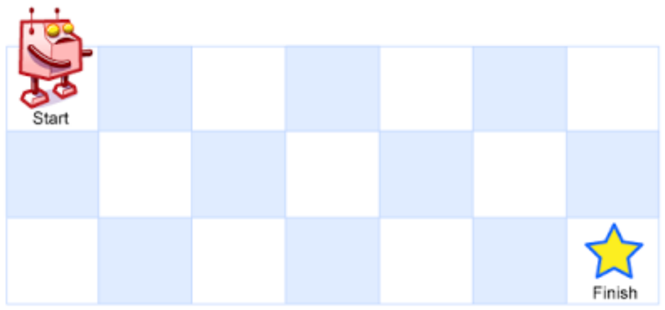
1. A critical water treatment plant is composed of n subsystems. Each subsystem's chance for failure is independent of all the other subsystems, therefore the probability that the plant functions correctly on a given day $(r_1 \cdot r_2 \cdot \dots \cdot r_n)$. Given that the plant is critical for water distribution in the state, each subsystem has a number of backup subsystems, denoted as $b_i - 1$. So, you can see that the probability of the subsystem failing and not being able to be replaced is $(1 - r_i)^{b_i}$. Therefor the overall change of failure is $1 - \prod_{i=1}^{n}(1 - (1 - r_i)^{b_i})$.

   Given a budget B, subsystem costs $c_1, \dots, c_n$ and a new plant to build (in a different state), that is the optimal number of backup $(b_1 \dots b_n)$ subsystems for your new plant.

   Instead of providing an algorithm:
   - Provide the recurrence formula
   - Prove that the problem has optimal substructure

(Pseudocode is not required for this question)

2. 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?



3. Suppose that in a 0-1 knapsack problem, the order of the items when sorted by increasing weight is the same as their order when sorted by decreasing value. Give an efficient algorithm to find an optimal solution to this variant of the knapsack problem and argue that your algorithm is correct.

4. You are given an integer array nums. You are initially positioned at the array's first index, and each element in the array represents your maximum jump length at that position.

Write pseudocode to show your algorithm, return true if you can reach the last index, or false otherwise.

Q1: provid the recurrence formula: prove:

Our Problem: under budget B.

$$\begin{cases} \prod_{i=1}^{n} C_i b_i \le B \\ \min \left(1 - \prod_{i=1}^{n} (1-(1-r_i)^{b_i})\right) \\ \max \prod_{i=1}^{n} (1-(1-r_i)^{b_i}) \end{cases} \Rightarrow$$

Use DP to Build an array $\text{solution}[1,2,\cdots,j,\cdots,B]^T \in R^{B\times(n+1)}$

for each $1 \le j \le B$: $\text{solution}[j] \in R^{1\times(n+1)}$ stores the number of subsystems for each part. $\text{solution}[j][i]$ = under j budget, the number of ith subsystem when $i \ge 2$

$$\text{solution}[j][1] = \prod_{i=1}^{n} (1-(1-r_i)^{b_i})$$

Hence, the formula:

$$\text{solution}[j][1] = \max_{1 \le i \le n} \left(\text{solution}[j-C_i][1] \cdot \frac{1-(1-r_i)^{b_i+1}}{1-(1-r_i)^{b_i}}\right)$$

$\text{Solution}[1][1,2,\cdots,n] = 0$ <

when $j - C_i \ge 0$, build i subsystem to optimize.

Optimal Structure:

Take C_i cost from the beginning, then it becomes $B - C_i$ subproblem for the remaining.

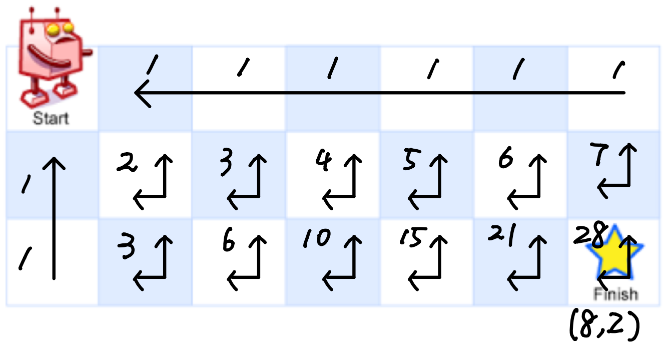
$$\text{Solution}[j][K] = \begin{cases} \text{Solution}[j-C_i][K] & \text{if } K \ne i \\ \text{Solution}[j-C_i][K]+1 & \text{if } K = i \end{cases}$$

$n \ge K \ge 2$

Time Complexity $= O(Bn^2)$

Q2. My answer contains general recurrence formula and specific solution from the graph.

① specific solution from the graph: answer = 28

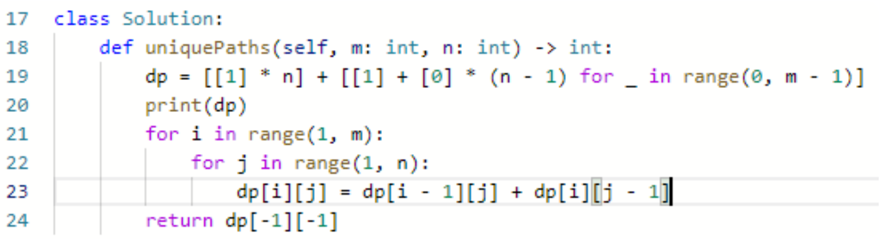


② General recurrence formula

$C[i][j]$ present numbers of unique path from (i,j) to $(0,0)$

$$C[i][j] = \begin{cases} 1 & \text{if } i=0 \text{ or } j=0, \\ C[i-1][j] + C[i][j-1] \end{cases}$$

Python Code:



```python
class Solution:
    def uniquePaths(self, m: int, n: int) -> int:
        dp = [[1] * n] + [[1] + [0] * (n - 1) for _ in range(0, m - 1)]
        print(dp)
        for i in range(1, m):
            for j in range(1, n):
                dp[i][j] = dp[i - 1][j] + dp[i][j - 1]
        return dp[-1][-1]
```

Q3. My general recurrence formula is:

Set of n Items: $S_n = \{<v_1, w_1>, \cdots, <v_i, w_i>, \cdots, <v_n, w_n>\}$. Let $B[k, w]$ be the solution of 0-1 knapsack problem over items sorted by decreased order of value (also increase order of weight from 1 to k in S_n under budget w.

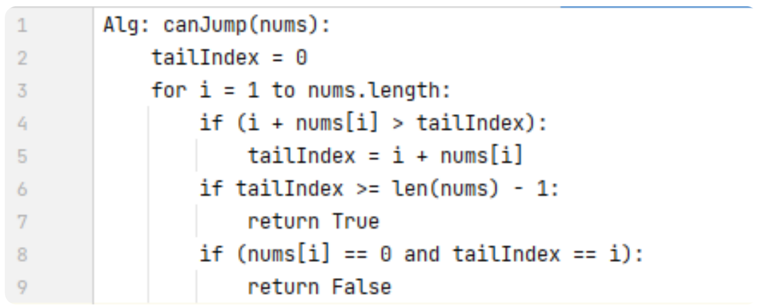
$$B[k,w] = \begin{cases} B[1, w] & \text{if } w_k > w \\ B[k-1, w-w_k] + v_k & \text{if } w_k \le w \text{. put it in the b} \end{cases}$$

prove: Assume there are two items $<v_1, w_1>$, $<v_2, w_2>$, $V_1 > V_2$ and $W_1 < W_2$. We will choose 1 instead of 2. Hence, we should always choose the more valuable item. If we choose item with less value/more weight, it will have contradiction.

Q4.

Use greedy algorithm to solve this problem

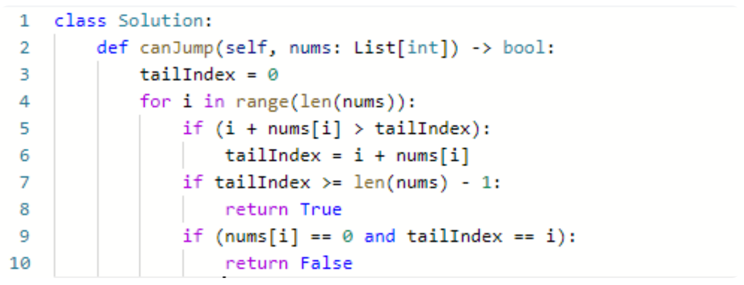
pseudo code:



```
Alg: canJump(nums):
    tailIndex = 0
    for i = 1 to nums.length:
        if (i + nums[i] > tailIndex):
            tailIndex = i + nums[i]
        if tailIndex >= len(nums) - 1:
            return True
        if (nums[i] == 0 and tailIndex == i):
            return False
```

Time Complexity: $O(n)$. Space complexity: $O(1)$

Python Code:



```python
class Solution:
    def canJump(self, nums: List[int]) -> bool:
        tailIndex = 0
        for i in range(len(nums)):
            if (i + nums[i] > tailIndex):
                tailIndex = i + nums[i]
            if tailIndex >= len(nums) - 1:
                return True
            if (nums[i] == 0 and tailIndex == i):
                return False
```