Project 1 - Dynamic Programming

November 9, 2022

4. Project Questions.

    1. What is the recurrence you are using for this problem?.

        $$\mathbf{DP}[s][r] = \mathbf{DP}[s][r-1] + \mathbf{DP}[s-1][r] - \mathbf{DP}[s-1][r-h-1] \tag{1}$$

        Where $\mathbf{DP}[s][r]$ represents total combinations of putting **r** robots in **s** stacks.
        The negative part in the equation if for removing duplicate scenarios where we put maximum numbers of robots in one stack.

    2. What is the base case of your recurrence?

        (a) If robots=0 then $\mathbf{DP}[s][0] = 1$.

        (b) If robots<0 then $\mathbf{DP}[s][r] = 0$.

        (c) If stacks=0 then $\mathbf{DP}[0][r] = 0$.

    3. What are the time and space complexity of your algorithm?

        Time Complexity: $O(b * n) \sim O(n^2)$.
        Space Complexity: $O(b * n) \sim O(n^2)$.

    4. Pseudo-code

        Iterative Approach

```
import numpy as np

def combinations(robots, stacks, height):
    #Create a 2d array with 0 values
    DP = np.zeros((stacks+1)*(robots+1),1), int)

    for i in range(stacks+1):
        for j in range(robots+1):
            if j == 0:
                DP[i][j] = 1
            else if i == 0:
                DP[i][j] = 0
            else:
                DP[i][j] = DP[i-1][j] + DP[i][j-1]

                #Edge case
                if j-height >= 1:
                    DP[i][j] -= DP[i][j-height-1]

    return DP[stacks][robots]
```

5. What are the time and space complexity of your algorithm?

   Time Complexity: $O(b * n) \sim O(n^2)$.
   Space Complexity: $O(b * n) \sim O(n^2)$.