Project 1 – Dynamic Programming

COT 6405 – Fall 2023

Due 10/31/2023 by 11:59pm

1 Overview

You are required to solve a dynamic programming problem and write a memoized version using python.

2 Description

Robots are trying to climb over a wall, however, due to some questionable programming choices the robots are bound by certain rules. To climb over the wall, they can only stand on the top of each other's shoulders, they can only stack so many on top of each other, and they can only create so many stacks. Given the number of stacks (*n*), and the max number of robots that can go in each stack (k) your task is to figure out how many ways a given number of robots (*b*) can distribute themselves into the stacks. Your solution **must** use *dynamic programming*.

3 Code

You must implement the project in python. The only rules for your program are that it must take a text file as input and write the results of the program to the command line. You must have a main python file called RobotStack.py. The name of the input file need to be given to the program as a command line parameter. The following command will be run.

python RobotStack.py input.txt

3.1 Input file format

Each line has 3 comma separated values that detail an instance, and represent *b*, *n*, *k* in that order.

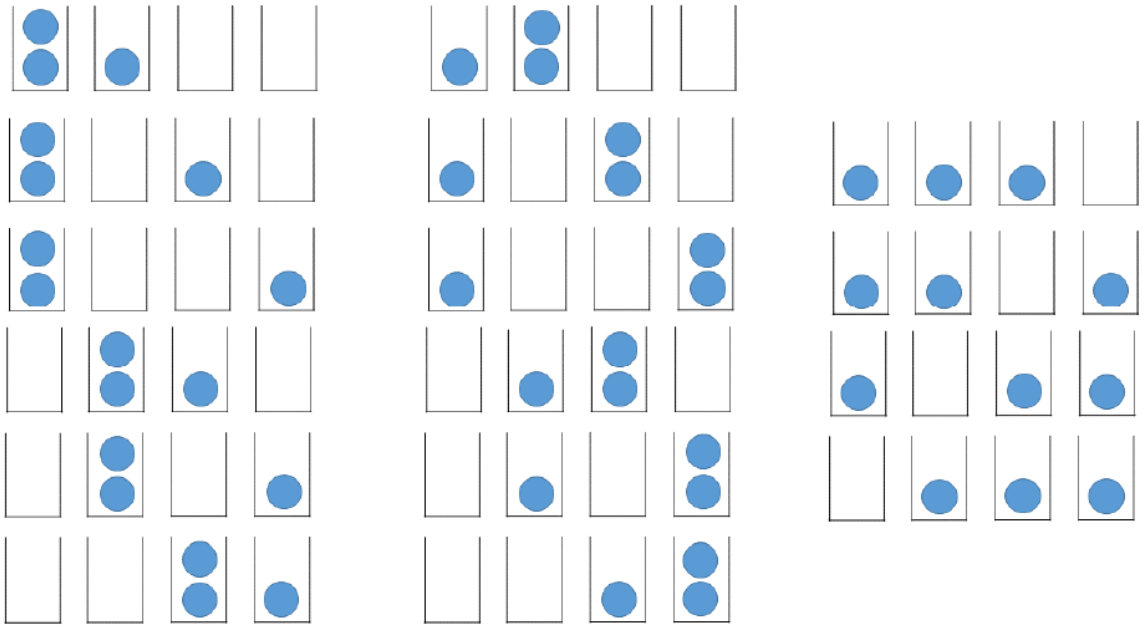
3.2 Output format

Your output format must look like the following screenshot. Note those are examples from the input.txt file on Canvas.

```
>python RobotStack.py input.txt
(3,4,2) = 16
(10,10,5) = 85228
(10,5,4) = 381
```

3.3 Example

Given *b=3, n=4, and k=2* there are 16 different ways the robots can distribute themselves. *HINT: this problem is* $\binom{n}{b}$*when k=1 and* $\binom{n+b-1}{b}$*when k=b. Note the solution increases as k move from 1 to b.*



4 Project Questions

Answer in a PDF of your answers to the following questions.

a. What is the recurrence you are using for this problem?
b. What are the base cases of your recurrence?
c. What are the time and space complexities of your algorithm?
d. Write pseudo-code for an iterative approach to this algorithm. Your pseudo-code needs to be detailed and clear so that a programmer could implement it in their programming language of choice. **NOTE: You only need pseudo-code for the iterative approach. You don't need to write actual code.**
e. What are the time and space complexities for your iterative approach?

5 Submission

Submit a zip archive containing (1) all scripts needed to run your program; and (2) a PDF of your answers to the project questions.

6 Grading

Algorithm implementation - 70%
Project questions – 30%