PROGRAM 1: Glass Falling

A) Describe the optimal substructure/recurrence that would lead to a recursive solution

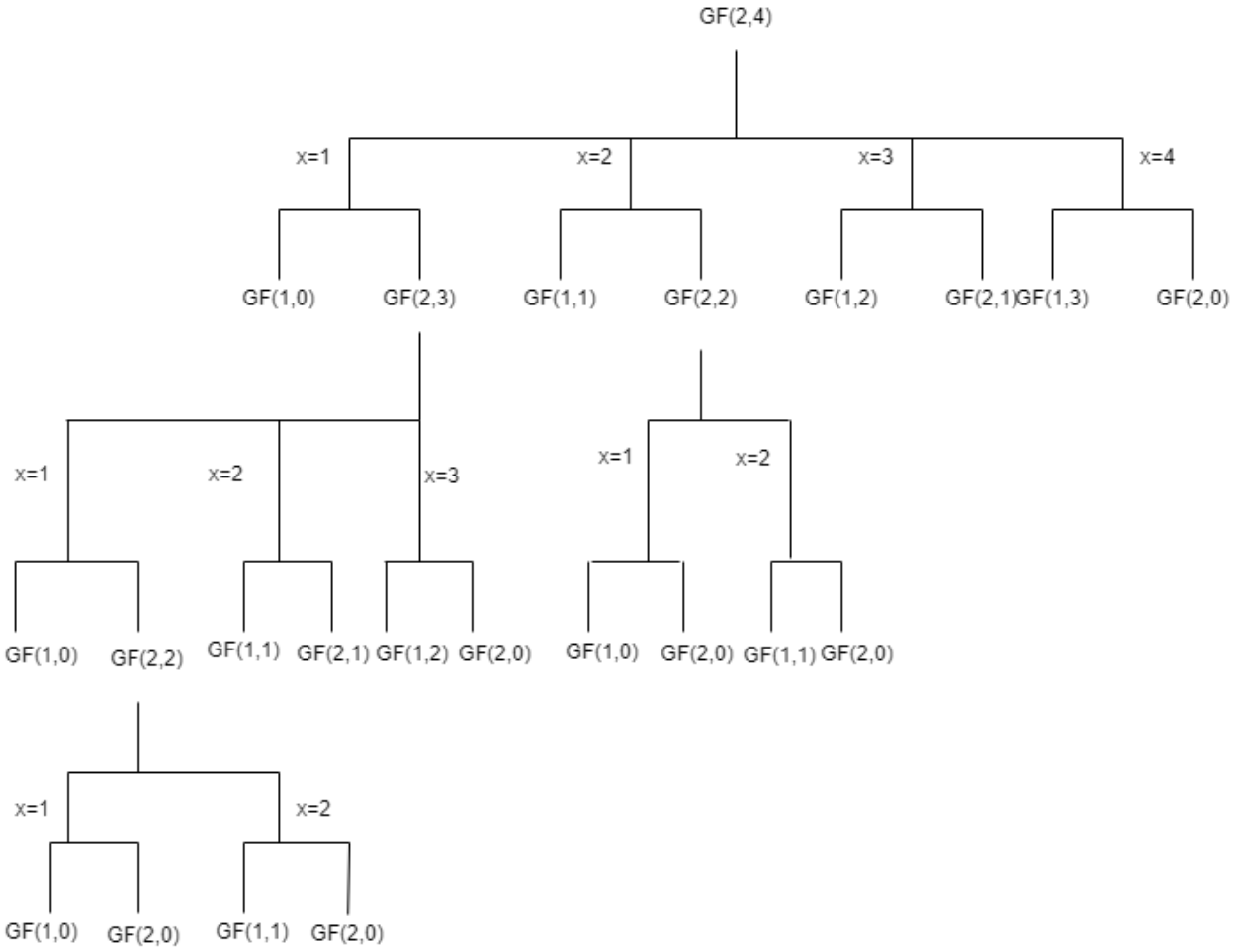
The problem is divided into two optimal sub-problems. For example, if we drop the glass sheet from floor let x, then there would be two cases;

- If the glass sheet breaks after dropping from floor x, then floors lower than x to be checked next ignoring the upper portions. So, the problem reduces to x-1 floors and sheets -1.
- If it does not break, then floors upper than x to be checked next ignoring the lower portions. So, the sheet remains the same and floors – x.

B) Draw recurrence tree for given (floors = 4, sheets = 2)

The above function computes the same sub-problems again and again. There will many repeated sub-problems when you draw the complete recursion tree even for small values of m and n.

Recursion tree for 2 glass sheets and 4 floors.



D) How many distinct subproblems do you end up with given 4 floors and 2 sheets?

12 distinct sub-problems in the case of 4 floors and 2 sheets.

E) How many distinct subproblems for n floors and m sheets?

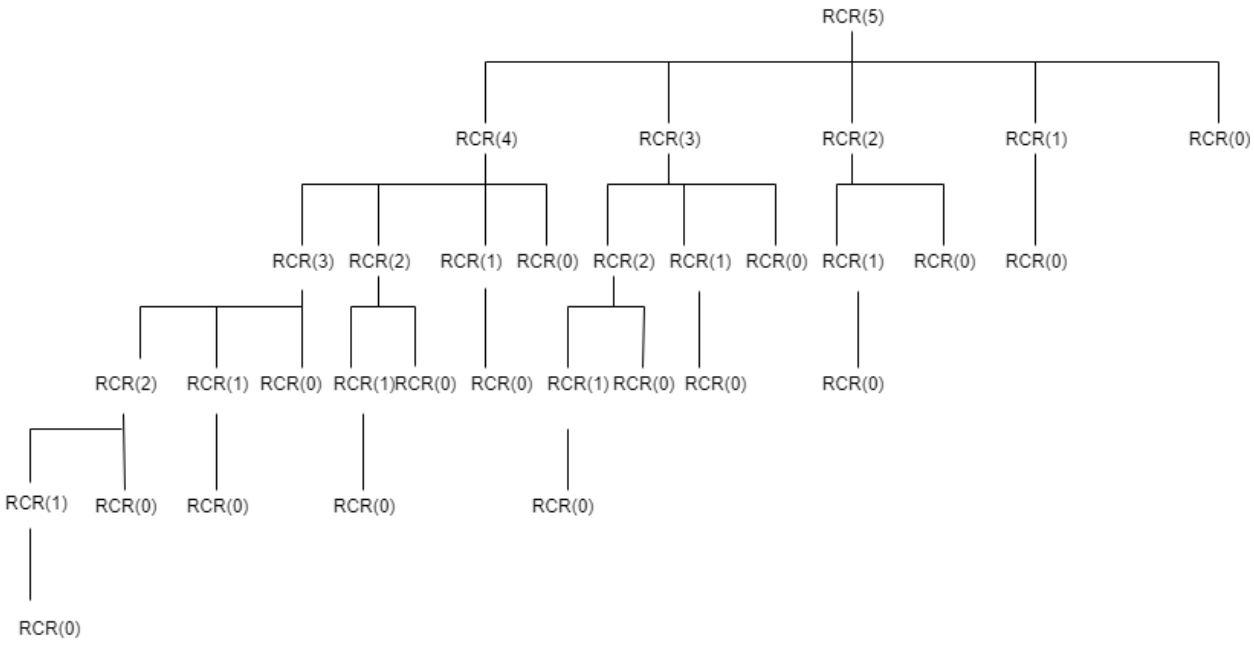
The amount of distinct subproblems would be (n*m)

F) Describe how you would memoize GlassFallingRecur

GlassFallingRecur can be memoize simply by adding 2D array sheetFloor [][] to the function GlassFallingRecur(). sheetFloor [][] saves the computation result of subproblems so that there would no repetition of similar problems. The GlassFallingRecur first checks sheetFloor[][], if the problem is solved then it simply returns the answer and if it is not solve it solves it.

PROGRAM 2: Rod Cutting

a) Draw the recursion tree for a rod of length 5



(b) On page 370: answer 15.1-2 by coming up with a counterexample, meaning come up with a situation / some input that shows we can only try all the options via dynamic programming instead of using a greedy choice.

Let price1 = 0, price2 = 4, price3 = 7 and length = 4. The dynamic programming selects the length 2 rod as it yields the highest revenue of (4+4= 8). So, the greedy strategy would not be better for this and if we increase the length of for example length = 50, then dynamic programming is far better than greedy strategy because of repetition it takes a lot of time to solve the problem.