Jonathan Arenson April 2, 2019

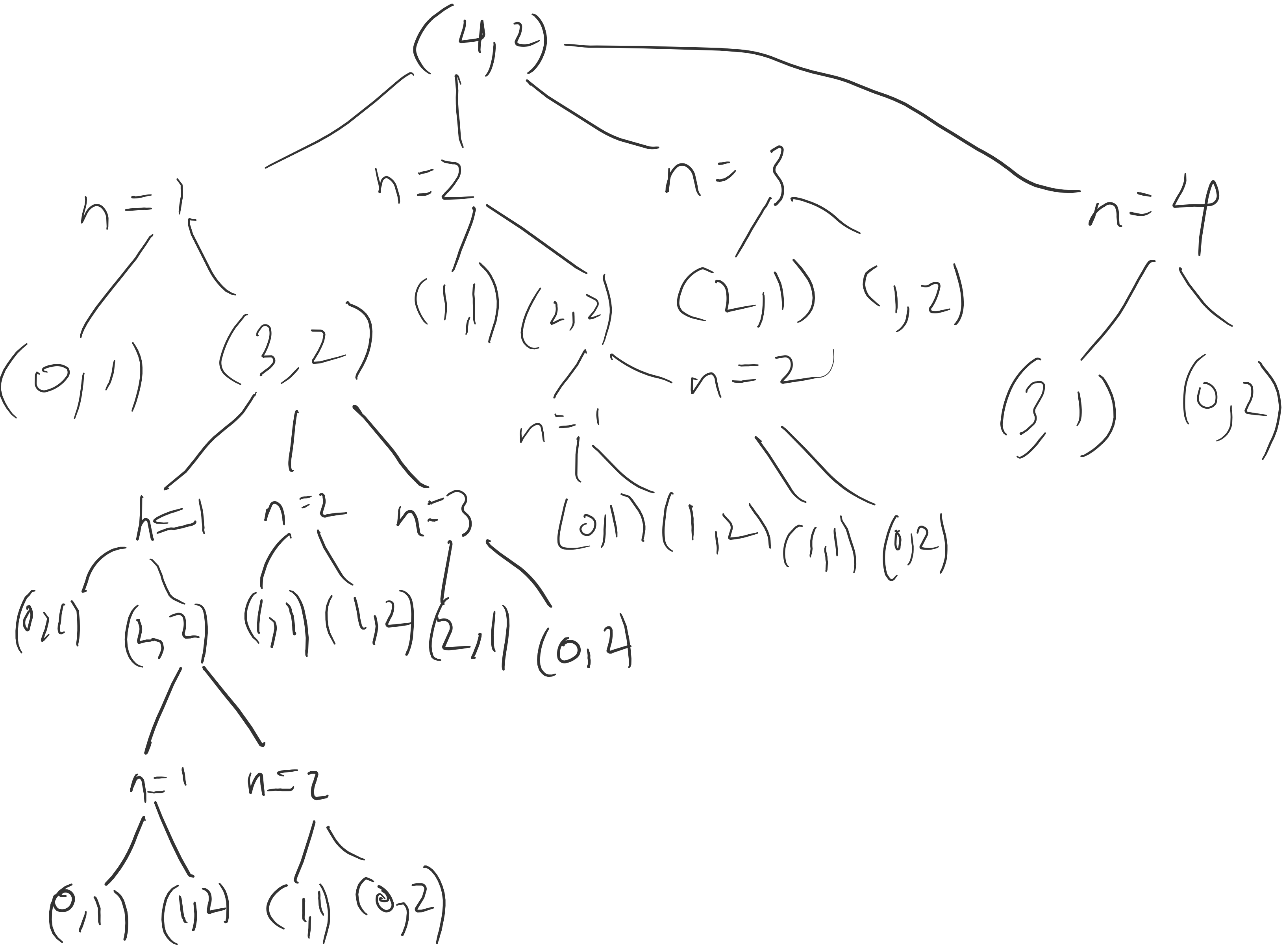
Dynamic Programming Assignment

1. Falling Glass

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

The optimal substructure that would lead to a recursive solution is with two cases. The first being that you drop glass from the nth floor and the glass breaks so you have to go to a floor lower than n and with one less sheet of glass. The second case is that you drop the glass from a floor higher than n and the glass doesn't break and you have the same amount of glass.

b. Draw a recurrence tree for given (floors = 4, sheets = 2)



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

There are 8 distinct subproblems and they are: (0,1),(0,2),(1,1),(1,2),(2,1),(2,2),(3,1),(3,2).

e. How many distinct subproblems for n floors and m sheets?

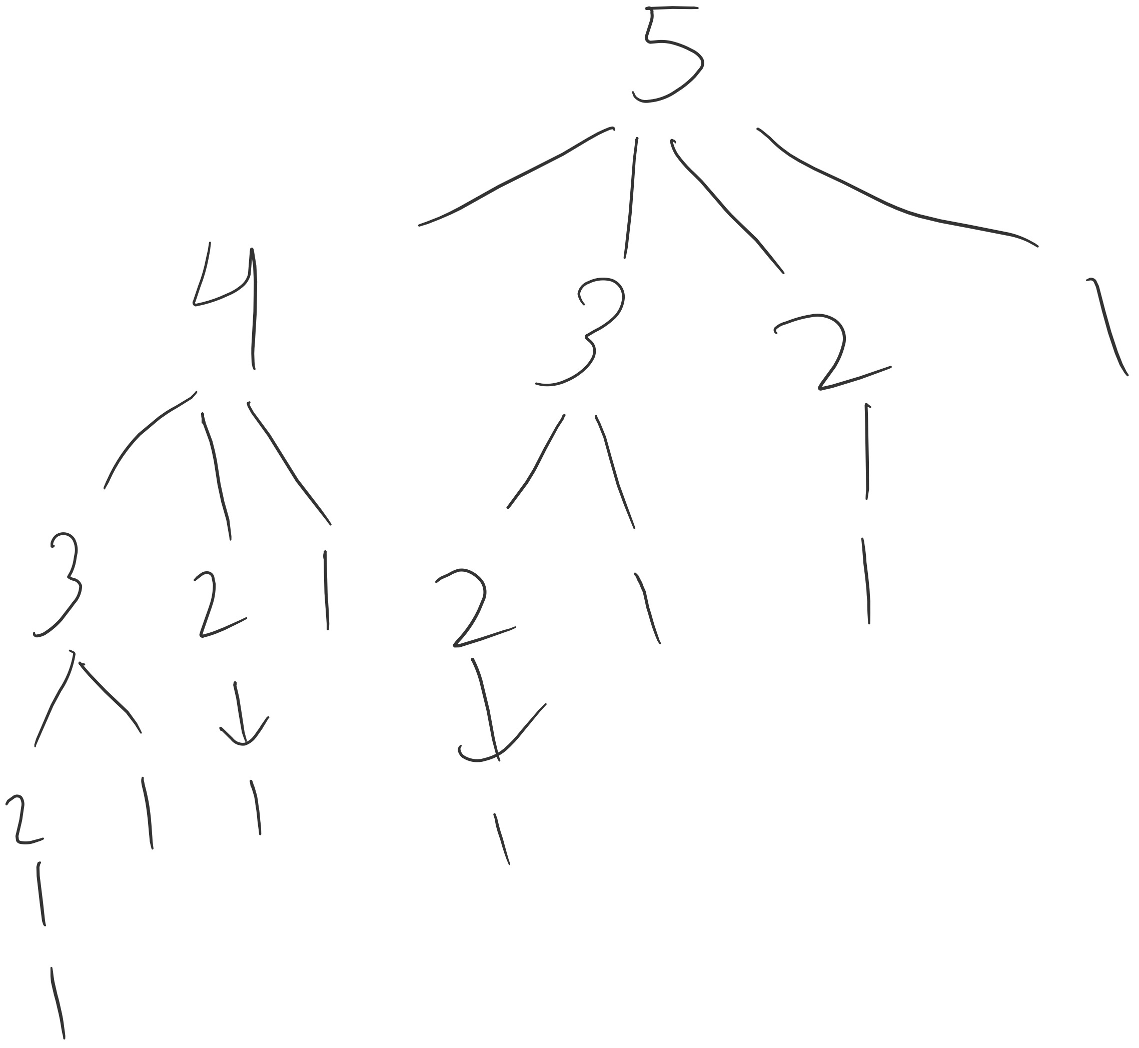
It creates (n*m) distinct problems.

f. Describe how you would memorize GlassFallingRecur.

It would have an array that checks on all base cases of the amount of sheets and if it's floor zero or floor one. If it has an answer solved within the array it will return that in order to not have to recalculate.

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.

Length:	1	2	3	4
Price:	1	20	33	36

A counterexample is that we have a rod of length 4 and by greedy strategy, we first cut the rod of length 3 which is price 33. That's a remainder of length 1 and price 1, which is a total of 34. While, the optimal way to cut the rod is into two lengths of two, which would result in a price of 40.