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

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

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

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

(f) Describe how you would memoize GlassFallingRecur

(a)

Recursion: drop the glass pane from each floor 1 to f and find minimum number of trials assuming the worst-case scenario.

Two cases: Dropping glass pane from floor = x. total floors = f, total glass panes = g

Shatters

- Check floors lower than x: x-1 floors
- Glass pane is gone: g –1 glass panes

Intact

- Check floors above x: f – x floors
- Glass doesn't break: g stays the same

Thus,

Base cases

- 0 or 1 floors = 0 or 1 trials respectively
- 1 egg = f trials

Recursion:

glassFalling(floors, glass) = 1 + minimum of [maximum of {shatters(x-1, g-1) or intact(f-x, g)}]

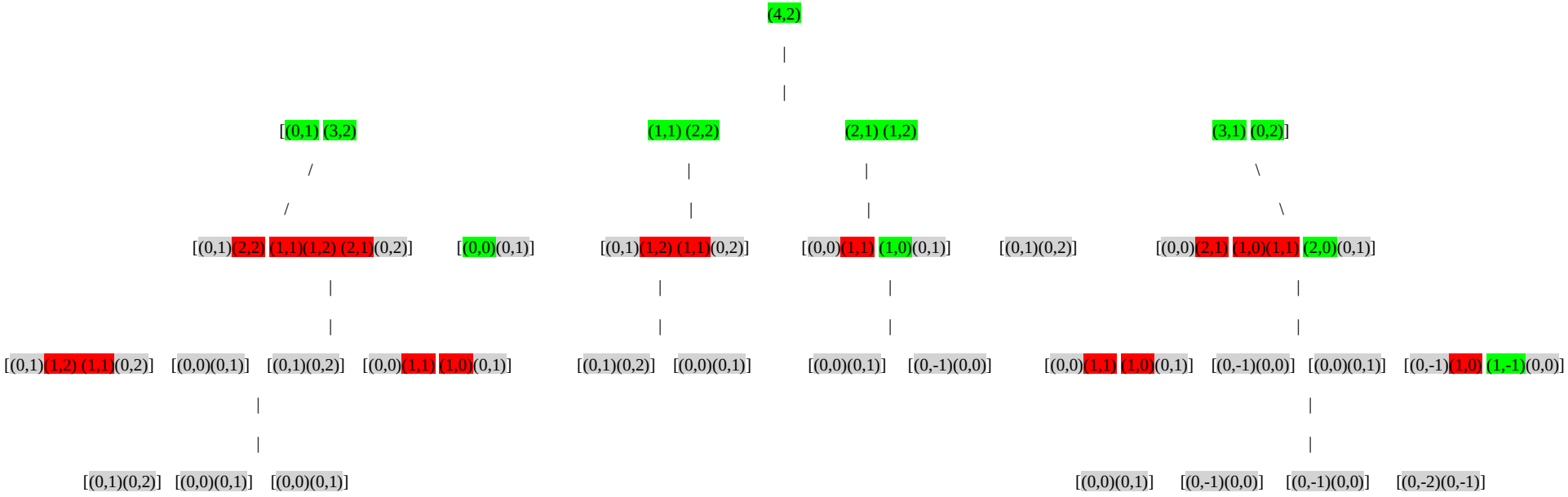
- +1 from dropping the glass pane
- Find maximum of either case because we are assuming the worst case
- Find the minimum of that to get least amount of trials

(b)

Green = first occurrence

Red = multiple occurrence

Grey = end of recursion and multiple occurrence



(d)

11 distinct subproblems

78 total subproblems

(e)

n-floors and m-sheets --> (n+1) * (m+1) possible distinct subproblems

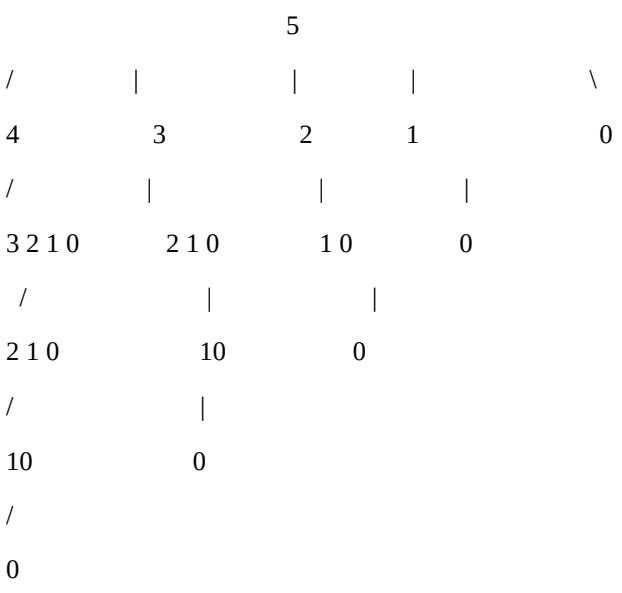
(f)

Create an 2 d array and fill each cell for each subproblem calculation. Same as the recursion function.

(a) Draw the recursion tree for a rod of length 5
(Above is just an example, you could be given anything where length i = 1 to N and price is any integer)

(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.

(a)



(b)

length	1	2	3	4	5
price	1	18	30	44	45
p/i	1	9	10	11	9

Show, by means of a counterexample, that the following “greedy” strategy does not always determine an optimal way to cut rods. Define the ***density*** of a rod of length i to be pi=i, that is, its value per inch. The greedy strategy for a rod of length n cuts off a first piece of length i, where 1 <= i <= n, having maximum density. It then continues by applying the greedy strategy to the remaining piece of length n - i.

- The greedy algorithm would select to cut a rod of length 5 into 4 and 1 with a total price of 45.
- The optimal algorithm would select to cut a rod of length 5 into 3 and 2 with a total price of 48.