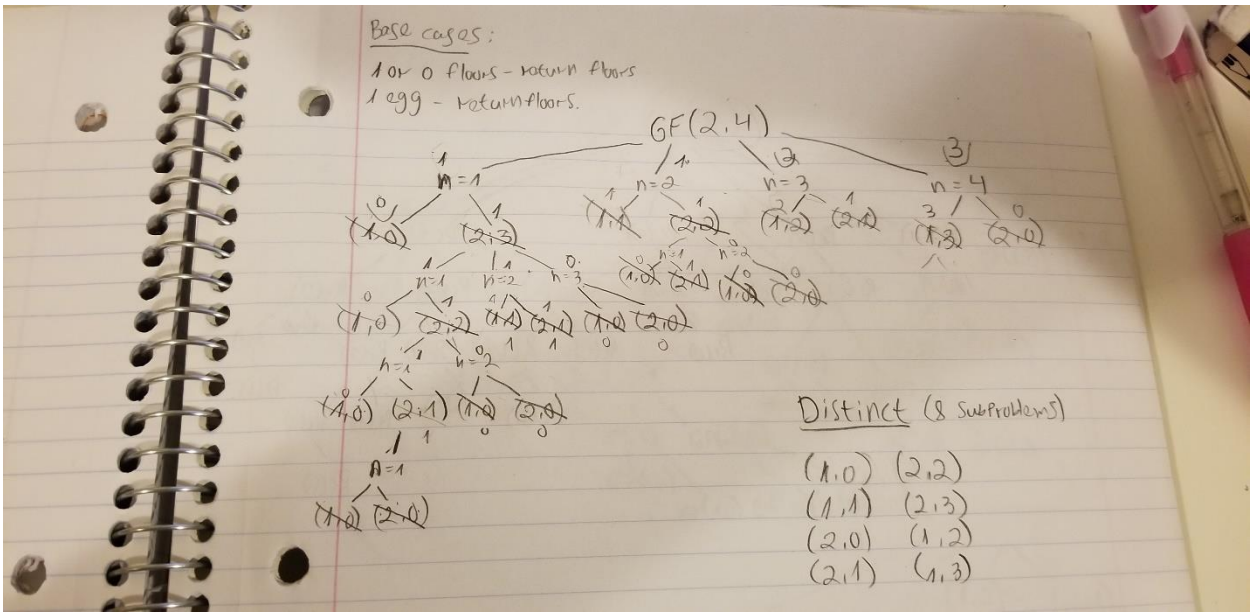
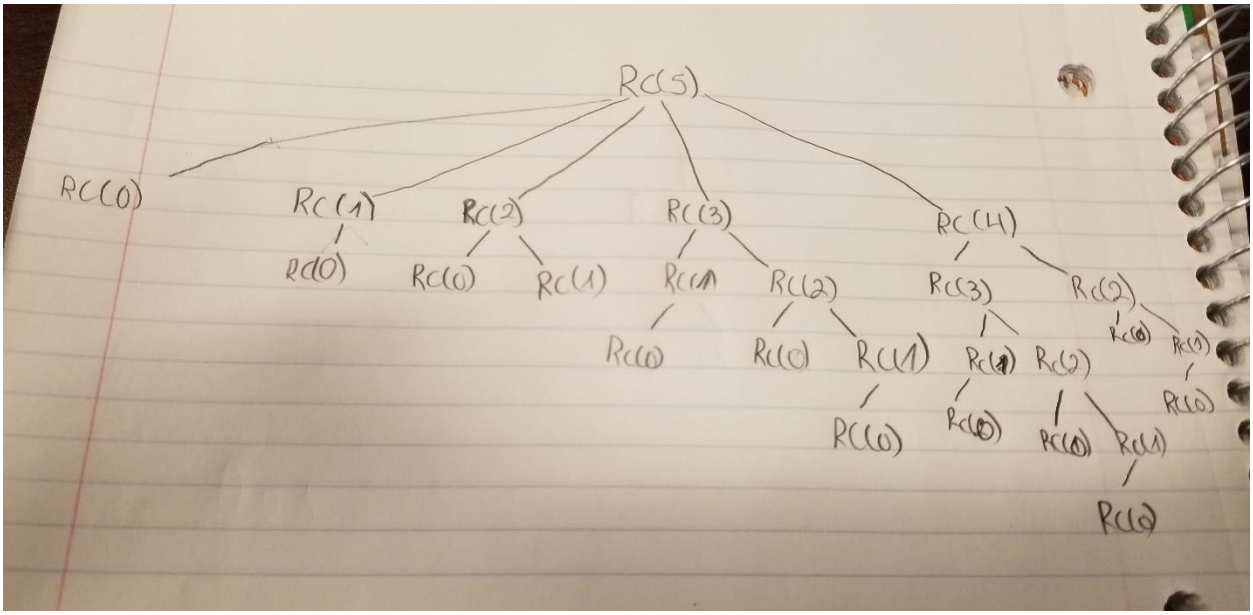
1) a. Optimal substructure: We need to consider two cases when dropping a glass from a certain level: either it shatters or it does not.
If the glass shatters when dropped at a level n then we only need to check the levels that are lower than n. so the problem has now shrunk to n-1 levels and m-1 glass sheets.
If the glass does not shatter at nth level then we only need to check for levels higher than n. so now the problem has reduced to TotalNumOfFloors – n and m glass sheets.
We will have to take the maximum of the two above cases for each floor since we are considering the worst-case number of trials. Then we find the minimum number of trials by identifying the floor which gives the minimum num of trials.

b.



d. 8 distinct subproblems. (see variations in picture above)
e. m X n distinct subproblems (which are all the unique permutations of m and n).
f. Since this problem has overlapping subproblems (as we can see in the recursion tree drawn above) we can avoid solving the same subproblems again by populating an array with all the computations in a bottom-up approach. Once we have that set up, we would go through the array and find lest number of trials out of all the floors and return that result.

2. a.



b. Using the same values you gave us:

length | 1 2 3 4 5 6 7 8

price | 1 5 8 9 10 17 17 20

Where the length of the rod is 5.

If we were to use a Greedy Algorithm approach here then we would first pick a length 5 which has the highest price/length ratio (at 1/2). At this point we reached maximum capacity yet the value we got is 10. But we can get a higher value (and thus a better solution) by using DP, for example lengths 2 + 3 yields a value of 5 + 8 = 13. So for this particular case the Greedy Algorithm approach does not yield an optimal solution.