Siheng Li

(a) Optimal substructure:

When a glass pane is dropped from floor n, there are only two possible outcomes, i.e. (1) The glass breaks. (2) The glass doesn't break

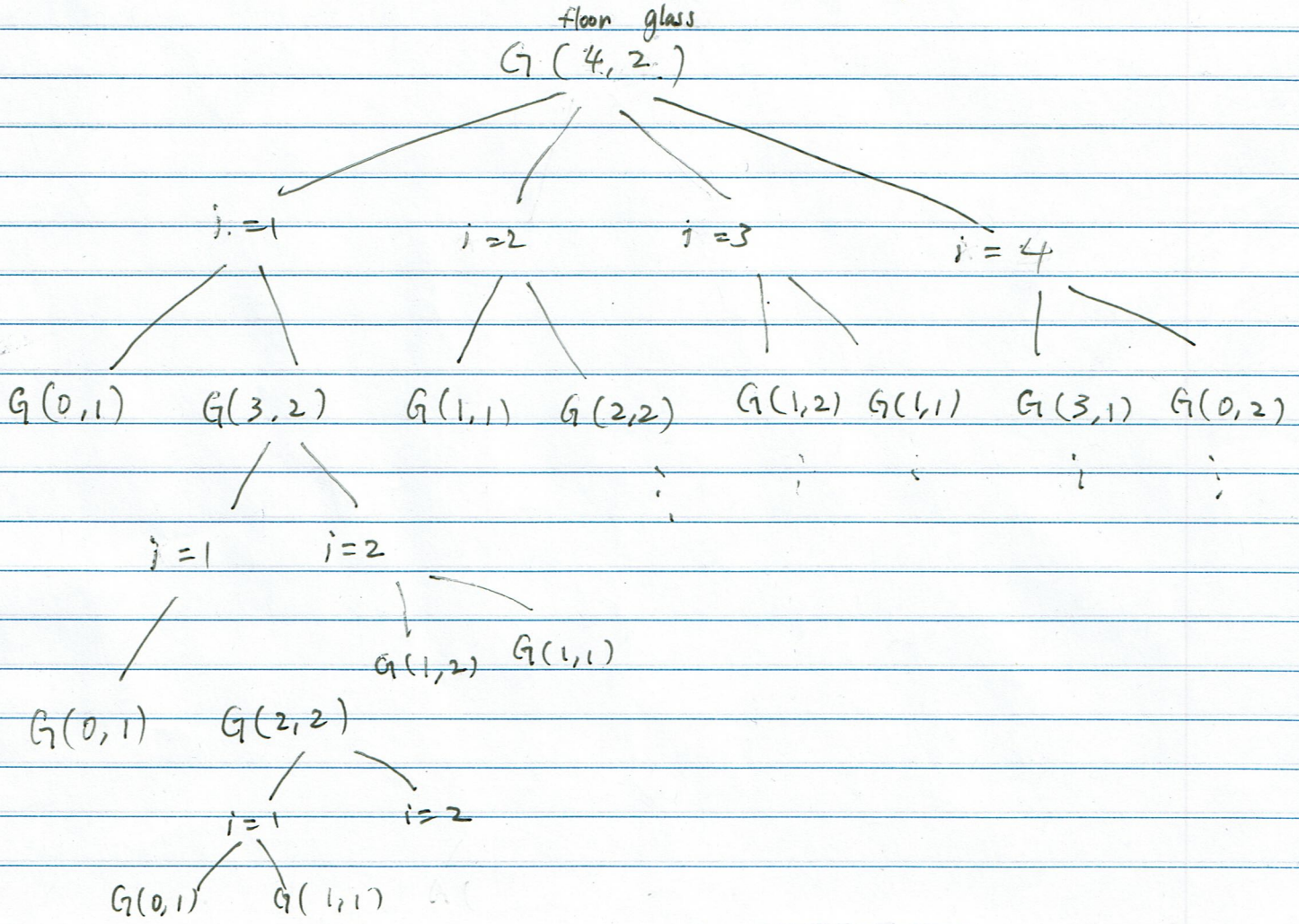
(1) If the glass breaks after dropping from the i^{th} floor, we need to try the floors lower than n with remaining glasses, the problem is then reduced to i−1 floor and m−1 glasses.

(2) If the glass doesn't break after dropping from the i^{th} floor, we then need to try the floors higher than n, the problem then reduces to n−i floors and m glasses.

Variable n is # of floors and m is # of glasses

```
glassFalling(n,m) { Minimum number of trials needed to find the critial floor in worst case
                    for i from 1 to n: (1+min{max(glassFalling(i-1,m-1), glassFalling(n-i,m))
```

(b)



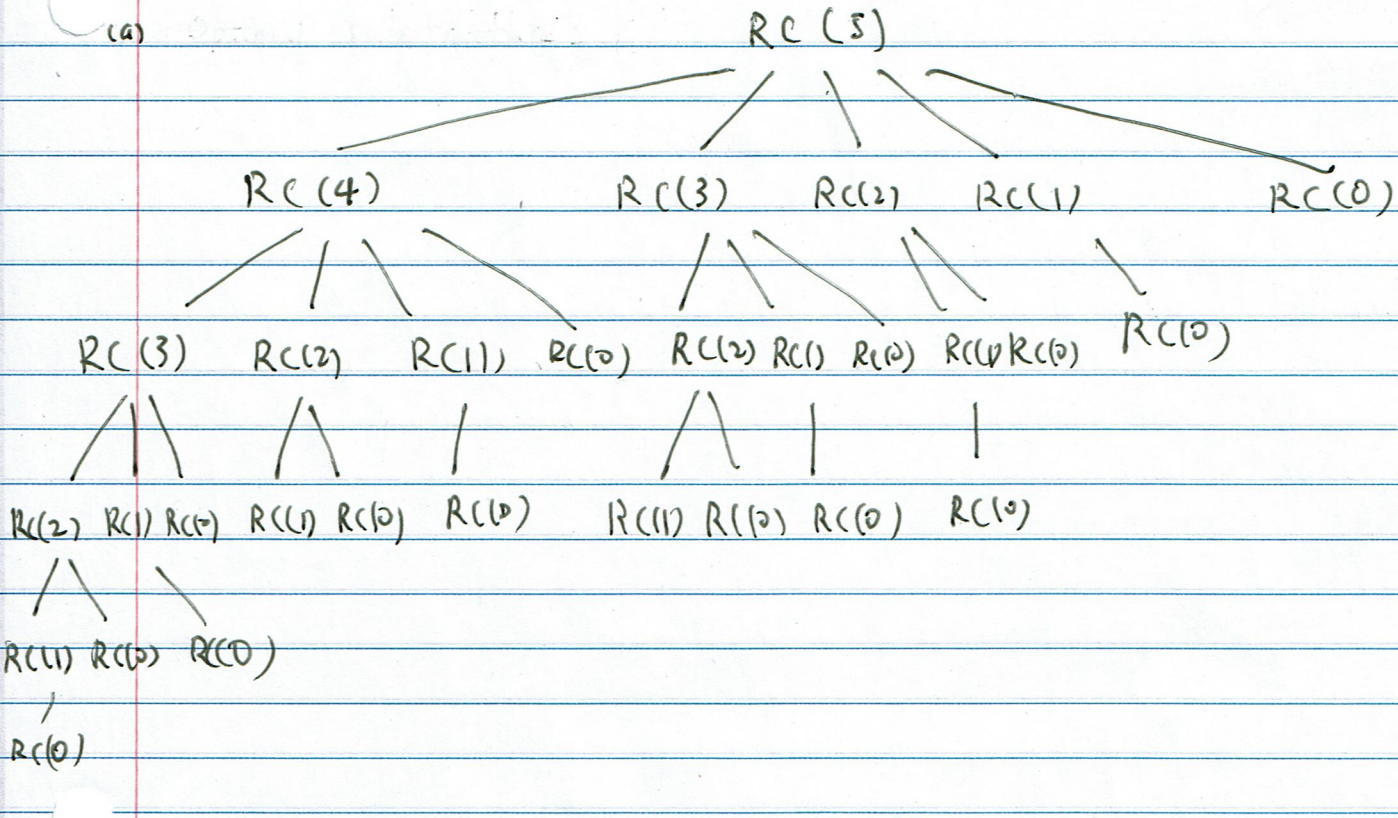
(d) There are k distinct subproblems

(e) There are $m \times n$ distinct subproblems

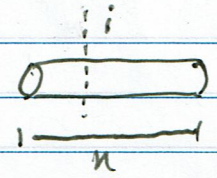
(f) Have a global array called memo[][] that stores the result of all the pairs of subproblem and check if they are already calculated at the begining of each recursive iteration.

Sihang Li

(a)



(b)



length = n $1 \le i \le n$

Let the length of the rod be 5, and the price be [1, 4, 7,]
If we use the stragedy considering the density, we would cut the rod as 2+2+1 and the max density is 4/2 = 2, which has value 4+4+1=9
If we use dynamic programming solution, then we cut the rod as 2,3 which has value 4+7 = 11. Hence greedy strategy using density doesn't work here