Ting Ting Ye

A)Optimal Structure
First, we have

sheets= the number of tested glass that we have
floors= the number of floors that has to be tested

For example, (4,2) is having 4 floors that has be tested with 2 sheets.

There can only be two cases in this problem, one being which the sheet breaks after dropping from i^{th} floor. In that case we would have to double check the floors that are lower with the other sheets which will give us i-1 floors and sheets-1 sheets. Another case is if the sheets doesn't break after the i^{th} floor which allows us to use the same sheets and check the floors that are higher up. So we will have floors-x and sheets.

The initial state of this process is (floors, sheets) which gives us the amount of sheets and floors that needs to be tested. The recursive function will terminate either when there is only one sheet or when there is either 0 or 1 floor.

Now, we can see

(floors, sheets) = is the minimum amount of trails under the worst given scenario

it can also be shown as

(floors, sheets) = 1 + min(max((floor-1, sheets-1),(floors-x, sheets)) where x={1,2,…,k} n={2,3,…,n} k= {2,3,….,k}

NOTE: that if there is only one sheet the amount of floors will be the trails because if you only have one sheet you would have to start from the lowest floor. And also, for when there is 0 or 1 floor there will only be zero or one trail.

B)

```
                                        GlassFalling(4,2)
               /                   |                    |                    \
              x=1                 x=2                  x=3                  x=4
              / \                 / \                  / \                  / \
          (1,0) (3,2)         (1,1) (2,2)          (2,1) (1,2)          (3,1) (0,2)
                 / | \              /      \
       x=1   x=2     x=3          x=1       x=2
    /\          /\         /\           /\         /   \
(1,0)(2,2)  (1,1) (1,2)  (2,1)(0,2) | (0,1)(1,2)  (1,1)(0,2)
      /  \
  x=1        x=2
  /  \         / \
(0,1) (1,2) (1,1) (0,2)
```

D) there are 8 distinct sub problems

E) I think there are O(mn) amount of sub problems dependent on m sheets and n floors.

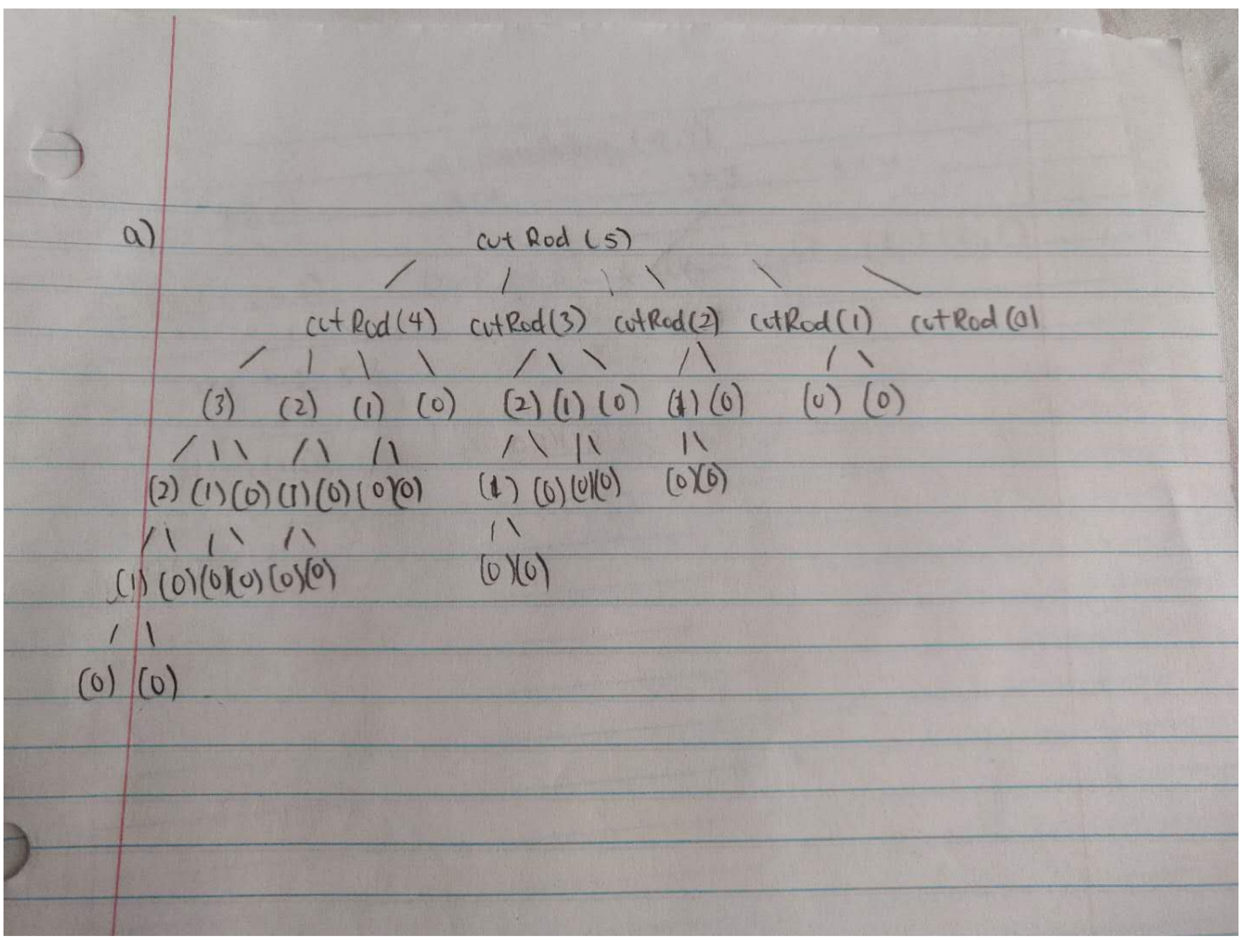
F) For memoized GlassFallingRecur, I passed in an array that will record all the minimum trails that was already calculated. There will also be an if statement to check if the recursive call has already been calculated in the Array. If it does exist, then the if statement will just return the trails recorded in the Array. So it wouldn't have to recalculate everything.

Rod Cutting

A)

```
                                        RodCutting(5)
            /                            |                 |              |      \
          RC(4)                        RC(3)            RC(2)          RC(1)  RC(0)
        /    |   |   \                /   |   \           /\              |
      (3)   (2)  (1) (0)            (2)  (1) (0)        (1)(0)           (0)
      /|\        /\   |              /\    |              |
   (2) (1)(0) (1)(0)(0)           (1)(0) (0)            (0)
   / |\   |       |
(1)(0) (0) (0)   (0)
 |
(0)
```

RC(0)



B)
One counter argument that will show that the greedy strategy will not give the optimal answer is when rod of length 1 cost \$2, length 2 cost \$38, length 3 cost \$60, length 4 cost \$120, length 5 cost \$10. When the rod length is 5.

Length	1	2	3	4	5
Price	\$2	\$38	\$60	\$120	\$10
Price/length	\$2	\$19	\$20	\$30	\$2

In this case greedy will take the length 5 rod and cut a 4 out making a profit of \$30 and leaving length 1 selling for \$2 creating a total profit of \$32

However, if you cut the rod into a 3 and 2. It will give you a profit of \$20 + \$19 = \$38 earning a higher profit then greedy strategy.