Fall glass problem

- Shock proof landing mat that catch the fall of the glass at a height
- Guarntee up to # of floor glass not break
- find height floor X where glass will not break

* Restriction : set amount of glass panes

If glass Survive
use again in other trial

If Shatter
remove

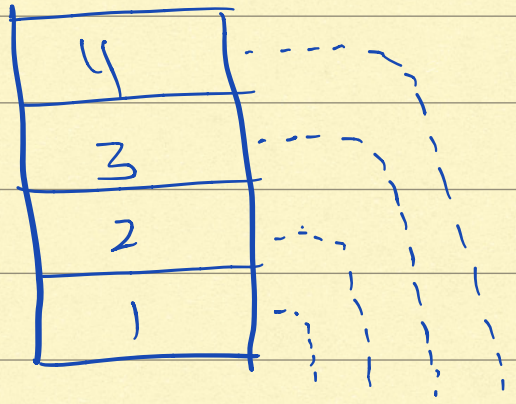
If Shatter at set floor
then Shatter > set floor

If glass Survive at Set floor
then Survive < set floor

No gurantee if thrown from 1st floor glass will or will not break nor last n^{th} floor

e.g If only one glass must start

first and move up.
worst case try all



Check all floors since only 1

Actually trying decide floors which glass sheet should be dropped so total # of trials ar minimized

Given n floors, m sheets

a) Describe optimal substructure/recurrence that will lead to recursive solution

We know for sure for base case

- if we have 0 floor we need do 0 trials [No drop]
- if we have 1 flour we need to do 1 trial [1 drop]
- if we have 1 sheet we cannot min

So we must go thru all floors so we return # of floors

* what if we have 1 floor and 0 sheets

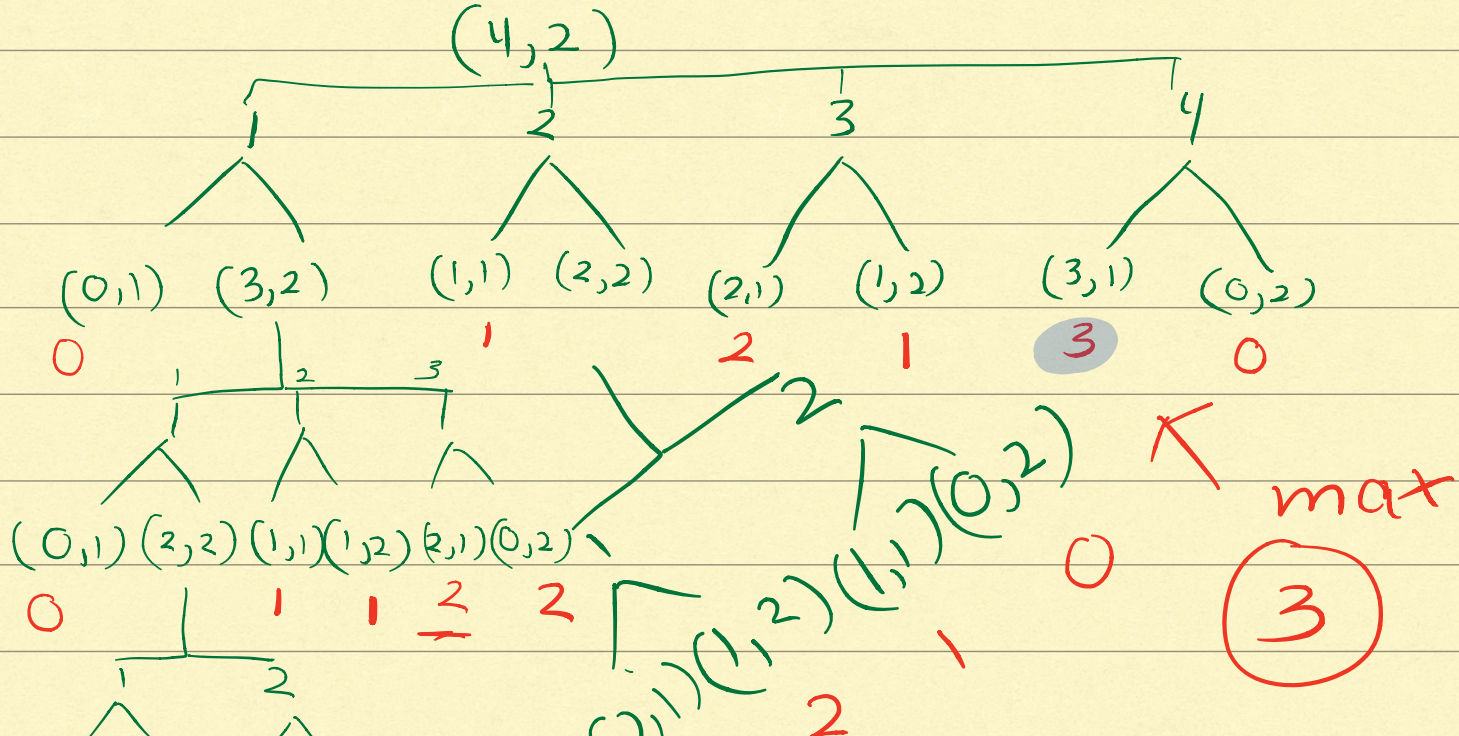
Recursion Part:

For every floor.

Case 1: Shatter at given floor
So return 1 less sheet
and check next floor

Case 2: Does not Shatter at given floor
return same # sheet
and check next floor

b) Recurrence Tree



(0,1)(1,2)(1,1)(0,2)

0 1 1 0

Each floor check both cases

- egg breaking
- egg not breaking

min

(4,2) 2 + 1 = 3 ✓

0 2 1 2 2 1 3 0

(0,1)(3,2) (1,1)(2,2) (2,1)(1,2) (3,1)(0,2)

2 2 2 3

min → 1 + 1

0 2 (3,2) 1 1 2 0

(0,1)(2,2) (1,1)(1,2) (2,1)(0,2)

2 1 2

takes min → 1 + 1

(2,2)

0 1 1 0

(0,1)(1,2) (1,1)(0,2)

1 1 ← not at each

1 for each floor to consider drop

* I noticed # of trial is

like geometric series.

So $\frac{(t)(t-1)}{2}$ = floors

gives a good estimate

$\lceil 2.37 \rceil \simeq 3$

c) Coded

d) How many distinct subproblems?

(3,2) (2,2) (1,2) (0,2)
(3,1) (2,1) (1,1) (0,1)

8 distict subproblems

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

= n(n-1) distinct

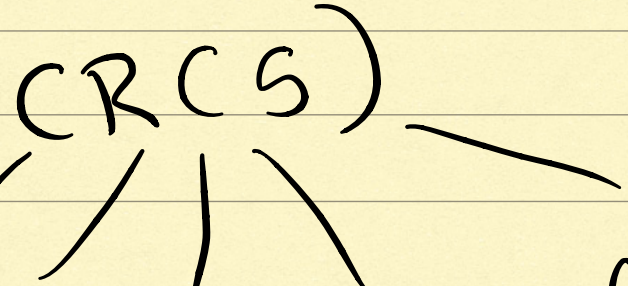
5) Since there is alot of repeated Subproblems. I would use a 2D array so it would save on computing so many subproblems. This would save me computation time as I would save the result of each subproblem. So my program would check if prev sub was solve and return those to solve current subproblem and so on.

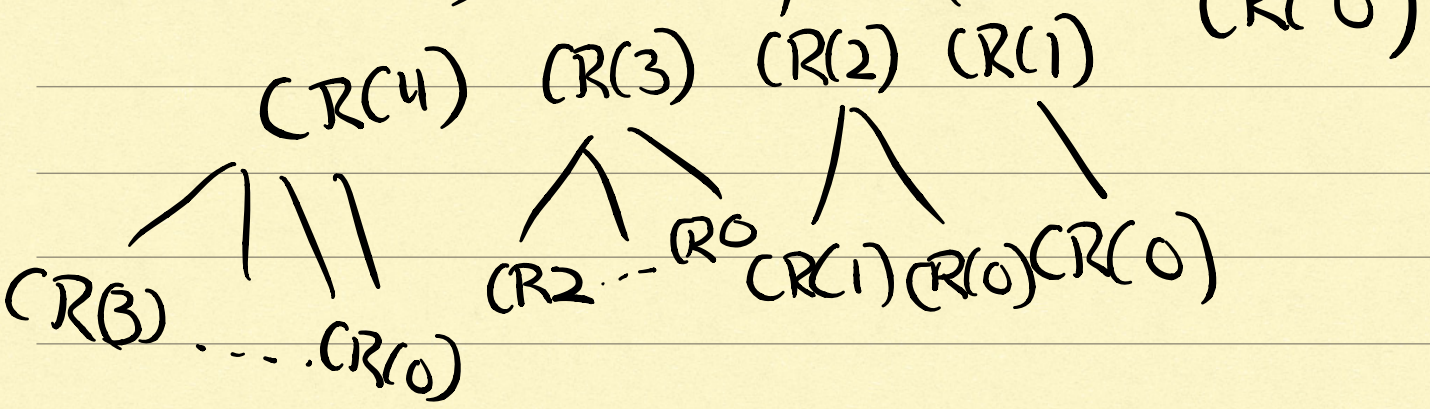
Rod Cutting

Given a rod length N and table of prices like so

length	1	2	3	4	5	6	7	8
price	1	5	8	9	10	17	17	20

a) Drew Recursion tree rod of length 5





b) Counter example for 15.1-2
Situation that only DP work instead of greedy

* Rod cutting with greed it will choose the cut that gives the largest value

Example	1	2	3	4	Length
	1	14	15	20	Price

Greedy would Choose to cut the rod for maximum value which cut at length 3 for price of 15 and then leave rod of length 1 for price of 1. which gives price of 16. But the best way to cut the rod would be to cut it

of length 2 twice for total price of 28.