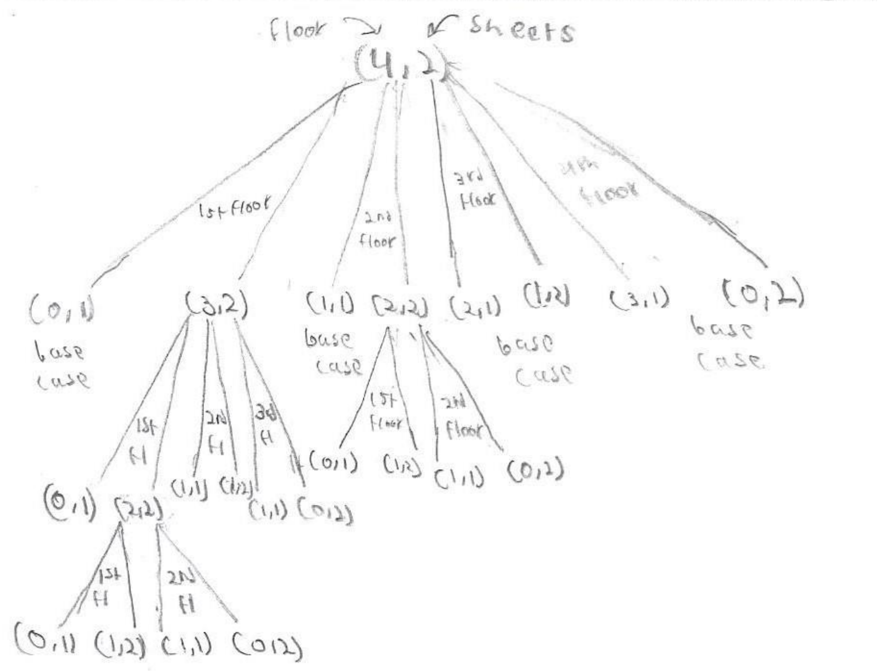
1A) The goal is to the find the floor that has the minimum number of trials needed. When a sheet is dropped two things can happen:

1) The glass breaks
2) The glass doesn't break

If the glass break after being dropped, we would then only check the floors lower than that floor. If the glass doesn't break, we would than only check the floors above that floor. We check each case for every floor and then compare them to see which has the higher number of trials. The value of the case chosen will then be stored as the max. The current minimum will then be compared with the max. If the max is lower, then the min then the min will be replaced with the max.

1B)



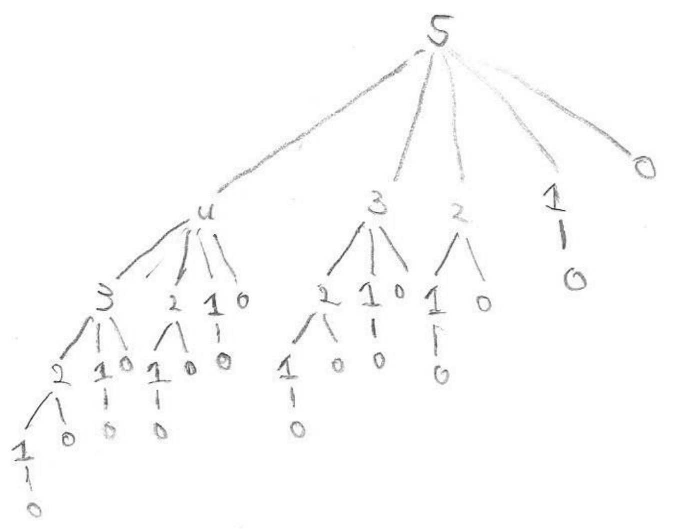
1D) If there are 4 floors and 2 sheets there are 8 distinct subproblems

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

1E) If there are n floors and m sheets then there are (n*m) distinct subproblems

1F) To memoize GlassFallingRecur a 2D array will first need to be initialized in order to store the results of the subproblems. A helper function will also be needed to find the number of trials recursively. The helper function will check the array to see if the result for the sub problem has already been found. If a result exists, the function will simply return it. If it doesn't then the subproblem will be solved. By avoiding problems that have already been solved the algorithm should speed up substantially

2A)



2B)

Length	Value	Density
1	1	1
2	10	5
3	16	5.33
4	12	3

Assume n=4 then according to the greedy algorthim we would first take a rod of length 3 and then a rod of length 1 which would result in the total value of 17 which is not the optimal solution. The optimal solution would be take two rods of length 2 which would be a total value of 20