Kirsten Pevidal CSC323: DP ASSIGNMENT

I. GLASS FALLING

a) Optimal substructure

When a glass pane is dropped from a floor n, there can be two cases

1) The glass pane breaks
2) The glass pane does not break

1) if the glass pane breaks after dropping from nth floor, then we only need to check for floors lower than n with remaining glass panes : so the problem reduces to n-1 floors and n-1 glass panes.

2) If the glass pane does not break after dropping from the nth floor, then we only need to check for floors higher than n; so the problem reduces from k-n floors and n glass panes.

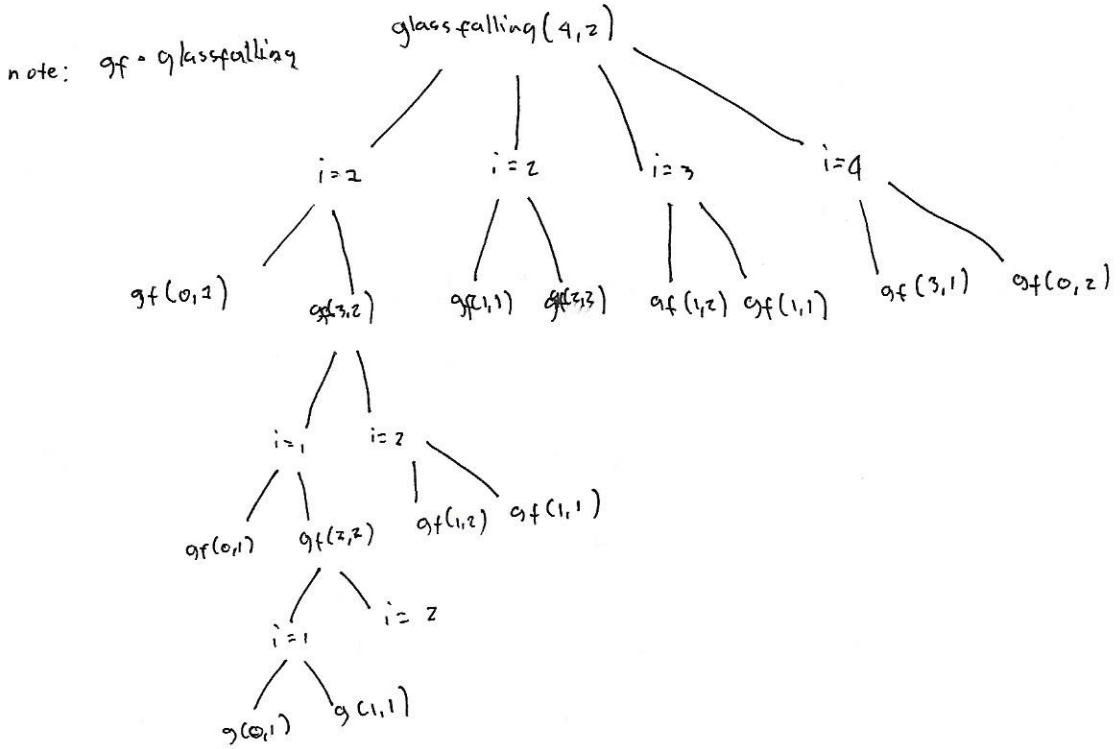
Since we need to minimize the # of trials in worst case, we take the maximum of two cases. we consider the max of above two cases for every floor & choose the floor which yields minimum number of trials

glassFalling (floors, sheets) → minimum number of trials needed to find the critical floor in worst case

glassFalling (floors, sheets) → 1 + min (max (glassfalling (floors-1, sheets-1), glassfalling (floors-i,

resources: geeksforgeeks.org

b) Draw recurrence tree (floors = 4, sheets = 2)



resource: tutorialhorizon.com | dynamic programming egg dropping problem

c) code for glass Fall Recur

→ check glassFalling.java

d) 1 floor × 2 sheets = 2 : how many distinct subproblems for question b

We have 8 distinct subproblems ~~for~~ given 9 floors and 2 sheets

e) how many distinct subproblems for n floors and m sheets?

distinct subproblems for glass Falling = m sheets × n floors
= m × n

f) Describe how you would memorize Glass Falling Recur

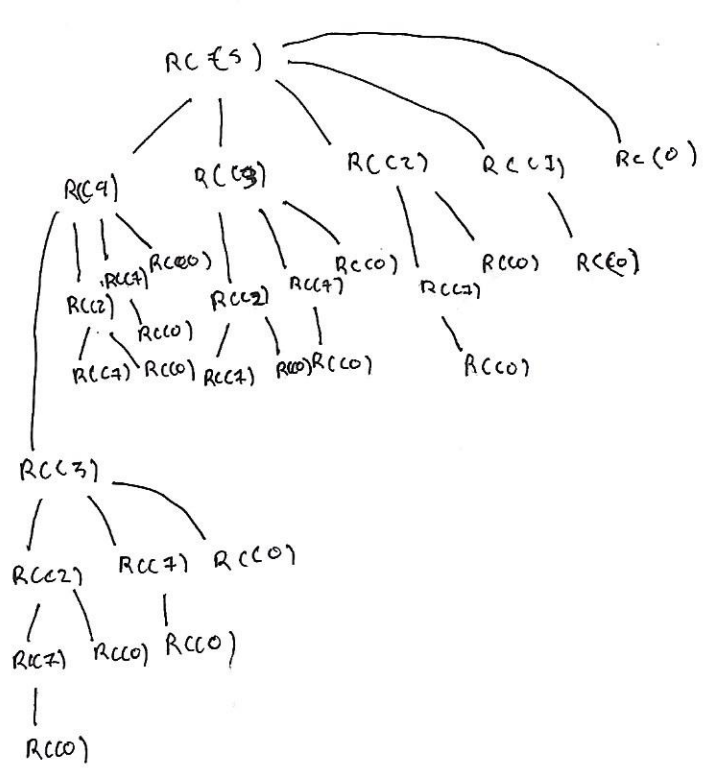
Create an array, say we call it memoTable[][], which stores all the results for all pairs of subproblem. Here, we check ~~the currents pairs from the memoTable for our current pairs~~ if they are already calculated everytime we recurse.

g) code for bottom-up for glass Falling

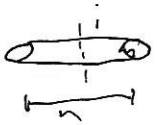
→ check glass Falling Java

II. Rod cutting

a) Draw the recursion tree for rod length = 5



b) page 320.15.1-2, prove by counterexample that shows all options can only be done by dynamic programming isted of using greedy choice.



length = n where $1 \leq i \leq n$

assume the rod length = 5 with price per unit

[1, 3, 5, 7]

greedy approach

The problem insists us to use the "density strategy".

This means we need to cut the rod by

length 2 + length 2 + length 1, since our density is ~~1~~ = 9/2 = 2

= 3 + 3 + 1

= 7

Using ~~the~~ dynamic programming appoach will give us ~~D~~ length 3 + length 2 = 5 + 3 = 8

or

length 4 + length 4 = 7 + 1 = 8

Here, we can say that the dynamic prog. approach ~~prov~~ gives a more optimal way to cut rodes compared to greedy approach