Sarmad Karatela
CS323

Dynamic Programming Assignment 1

Falling Glass:

a) Optimal Substructure:

Bas Case:
//n = floors
//m = glass sheets

Let's think about the two cases that can occur during this experiment

Case 1: Glass breaks
This can only occur if the glass breaks on a certain floor (n). If that happens, we check the floors lower than that. This is how we keep reducing the floors. Consider this case as n-1. As we keep going down the floors until we

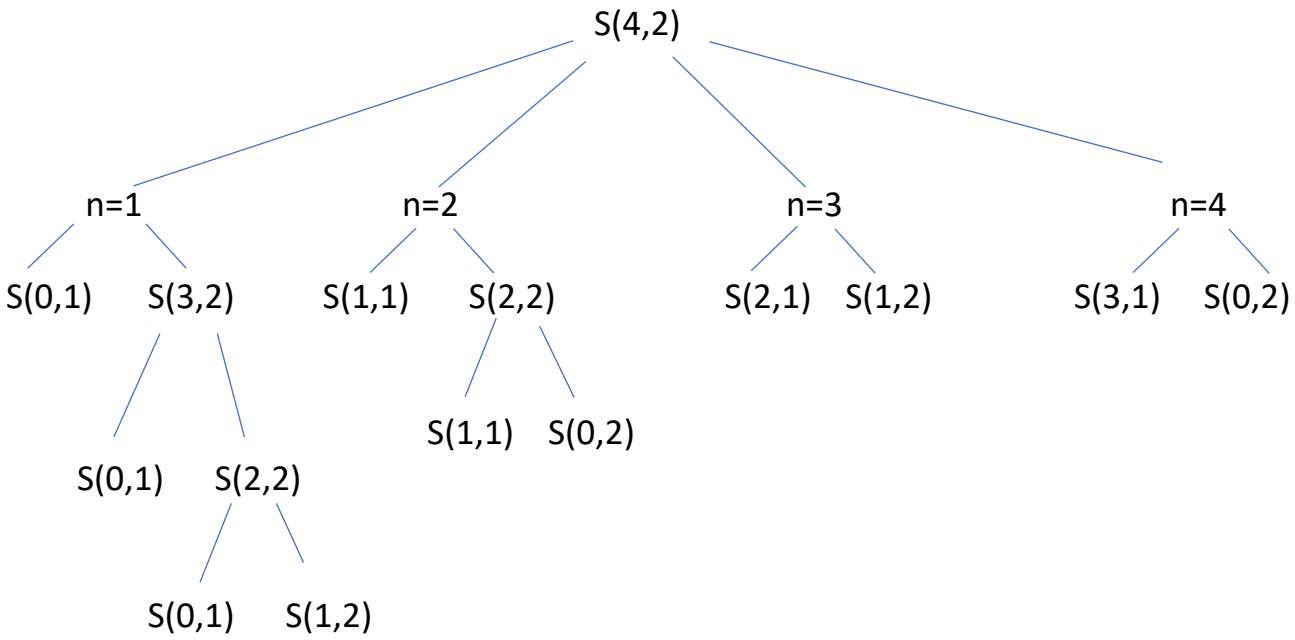
Case 2: Glass doesn't break.
If the glass doesn't break, we are going to check the floors higher than the current floor (n). We will keep checking until we find the floor until it breaks and we are left with K-x floors.

Lets consider this example where we have 2 GlassSheets (m) and 4 Floors (n). If we have 2 glass and 2 floors, if the glass breaks on the first floor, we still have another glass to work with. So we take the max of x-1 and the remaining glass which is k-x.

FallingGlass(n, m): 1 + min{max(FallingGlass(m - 1, n - 1), FallingGlass(m, k - m):
x in {1, 2, ..., k}}

Sarmad Karatela
CS323

b) Recurrence tree: (4 Floors, 2 Sheets)



d) Distinct subproblems for 4 Floors and 2 Sheets = 8
S(0,1), S(1,2), S(2,2), S(3,2), S(2,1), S(3,1), S(0,2)

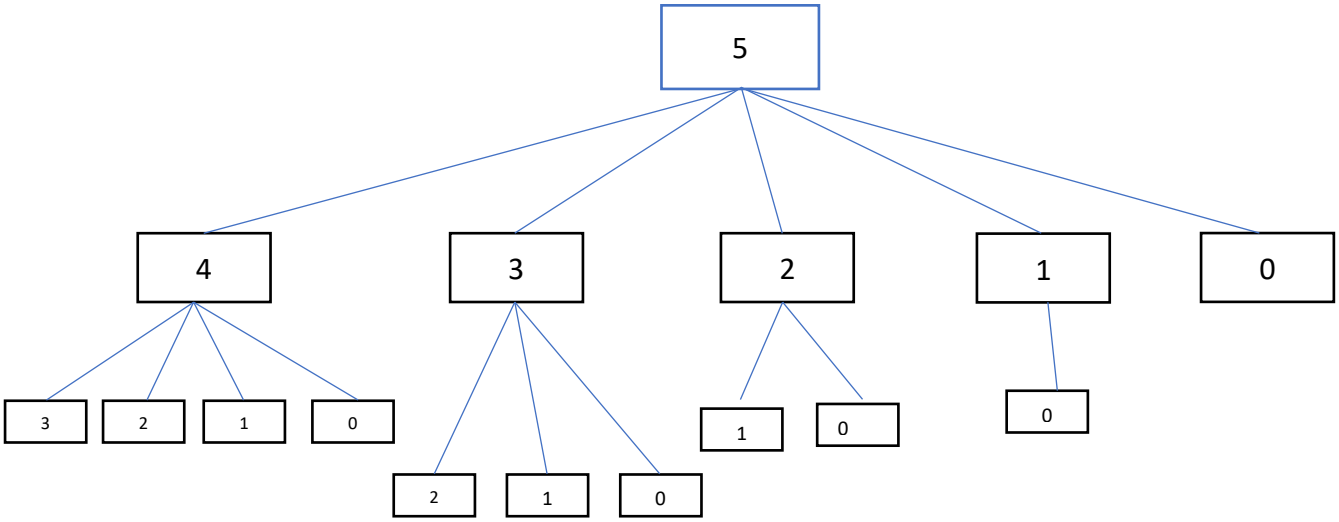
e) Distinct subproblems for n floors and m sheets = n * m

f) Memoize FallingGlassRecur:
Create an array memoize[i][j] and store those repeated subproblems into the array

Sarmad Karatela
CS323

Rod Cutting Problem

a) Recursion tree for length 5



B) Counter Example:

Suppose:

p1 =0, p2 = 1, p3 = 5, p4 = 8, p5 = 2, p6 = 1

Rod Length , n = 6

The greedy algorithm will first cut off a piece with the highest density = 9. But a better solution is possible which is 6 = 3+3 and the cost is 5+5 = 10. Therefore, in this case, the greedy algorithm doesn't determine the optimal solution.