Andrew McCann

2/24/2016

CS350

Assignment #3

1. Given a table composed of N x M cells, each containing some number of gold coins. You start from the upper left hand corner and at each step may go down or to the right. At each step you collect all the coins that are in that location. Find the maximum number of coins you can collect.
2. Dynamic Programming solution:

**Input**: A table (matrix), T, of N x M dimension where each entry is could contain some number of ‘coins’.

**Output**: The maximum sum of gold that can be acquired

# Preprocessing

GM <- N+1 x M+1 dimension matrix of all zeros.

func gold\_path(T)

for i <- 1 ... N

for j <- 1 ... M

GM[i,j] += max(GM[i,j-1], GM[i-1,j]) + T[i-1,j-1]

return GM[N, M]

Complexity Class:

We have to iterated over every single cell in the table with this method so we know it will be at least for order of growth, but since we tabulate them in a specific way we don’t even have to recheck the cells, we just return the final value.

Space:

Since GM is allocated based on N and M plus a single row it grows at the same rate at the source table. This puts it in the same space complexity class of:

1. Brief brute force solution:

Because we have two options per square, and we want to visit every path to analyze its gold. To do this we essentially have to traverse every option in the table of N x M cells. The length of the pass cannot be longer then N+M. If we have two options at each cell, we should have roughly 2^(N+M) permutations of the list. It should wash out to order of growth.