**Count Paths**

Write a program to count all the possible paths from top-left to bottom-right of a m X n matrix with the constraints that **from each cell you can move only right, down, or diagonal (right and down).**

Counting the number of paths can be trivially solved with recursive backtracking. Here is one example of how, for a 2D array matrix (with a call of numberOfPaths(0, 0)):

//Returns # of possible paths to reach cell at row m, column n from top-most, left-most cell

int numberOfPaths(int m, int n)

{

//Base case, there is only one possible path if at end cell

if (m == matrix.length - 1 || n == matrix[0].length - 1)

return 1;

return numberOfPaths(m+1, n) + numberOfPaths(m, n+1) + numberOfPaths(m+1, n+1);

}

**Output, for a 3x3 matrix:** 13

However, the time complexity of above recursive solution is exponential. There are many overlapping sub-problems. The recursion tree would be similar to the recursion tree for the Longest Common Subsequence problem.

This problem has both properties (overlapping sub-problems and optimal sub-structure) of a dynamic programming problem. Like other typical DP problems, recomputations of same sub-problems can be avoided by constructing a temporary matrix and solving in "bottom up" manner.

Recursion, as you should know by now, solves problems in a "top down" fashion, breaking the problem into smaller and smaller sub-problems until the base case is reached, allowing the more complicated problems to then be solved in turn.

DP avoids re-computation by solving from the bottom up, *starting* with the most trivially easy problem and building *next* solution(s) from previous solution(s), until done.

Try to establish the base case(s) and recurrence relation on your own first; if you get stuck, download this document and change the font color of the below:

**Base Case:** There is only one way to reach any cell in either the first row or the first col­umn when starting from index (0, 0).

**Recurrence Relation:** You can reach any cell from 3 dif­fer­ent ways: from the left, from above, of from above-left (diag­o­nal). The total number of paths to reach a cell will be sum of all the paths to reach to left, top, and diag­o­nal cells.