Rat Path

Given a matrix of bits (values 0 and 1), a rat must find a path from index [0][0] to [n-1][n-1]. The rat can only travel to the right or down, and can only travel on 0 values.  
  
Input: Matrix of elements with values either 0 or 1  
Output: Array of two-item arrays indicating the path.

# Example

Input: [[0, 0, 0, 1],

[0, 1, 0, 1],

[0, 1, 0, 0],

[0, 0, 1, 0]]

=> [[0, 0], [0, 1], [0, 2], [1, 2], [2, 2], [2, 3], [3, 3]]

# Constraints

Time Complexity: O()  
Auxiliary Space Complexity: O()

If not path found, return the

following path: [[-1,-1]]

# Solution

1. Create a scope variable called ‘results’ and initialize to an empty array (dynamic array) as follows: [[-1, -1]]
2. Create a helper method that takes in two indices i and j and an array of arrays path
   1. The first base case if *i* or *j* are out of bound or the value at [i][j] equals 1 return false
   2. The second base case if *i* or *j* are at the bottom right corner, save the path to the ‘results’ scope variable and return true
   3. Perform a recursive call going down (i+1, j, copy of path with [i+1, j] pushed in)
   4. Perform a recursive call going right (i, j+1, copy of path with [i, j+1] pushed in)
   5. If either recursive call is true, return true, otherwise return false
3. Call the helper method with the initial parameters (0, 0, [[0,0]])
4. Return ‘results’

**JavaScript Solution:**

function ratPath(maze){

var results = [[-1,-1]];

function findPath(i, j, path){

if(i >= maze.length || j >= maze[0].length || maze[i][j] === 1){

return false;

}

if(i === maze.length-1 && j === maze[0].length-1){

results = path;

return true;

}

return findPath(i+1,j, path.concat([[i+1, j]])) || findPath(i, j+1, path.concat([[i, j+1]]));

}

findPath(0,0,[[0,0]])

return results;

}

# Notes

This can be solved using dynamic programming starting from the end and working the way back in O(N2) time.

# Resources

http://www.geeksforgeeks.org/backttracking-set-2-rat-in-a-maze/