# 520 第八周学习总结

解决课上问题：写出63题，最短路径II的状态转移方程。

算法过程：（参考中国站解法）

1. 如果第一个格点 obstacleGrid[0,0] 是 1，说明有障碍物，那么机器人不能做任何移动，我们返回结果 0
2. 否则，如果 obstacleGrid[0,0] 是 0，代表这个位置可以走，我们初始化这个值为 1 然后继续算法
3. 遍历第一行，如果有一个格点初始值为 1 ，说明当前节点有障碍物，没有路径可以通过，设值为 0 ；否则设这个值是前一个节点的值 obstacleGrid[i,j] = obstacleGrid[i,j-1]
4. 遍历第一列，如果有一个格点初始值为 1 ，说明当前节点有障碍物，没有路径可以通过，设值为 0 ；否则设这个值是前一个节点的值 obstacleGrid[i,j] = obstacleGrid[i-1,j]
5. 现在，从 obstacleGrid[1,1] 开始遍历整个数组，如果某个格点初始不包含任何障碍物，就把值赋为上方和左侧两个格点方案数之和 obstacleGrid[i,j] = obstacleGrid[i-1,j] + obstacleGrid[i,j-1]
6. 如果这个点有障碍物，设值为 0 ，这可以保证不会对后面的路径产生贡献

按照这个思路完成代码：

class Solution {

public int uniquePathsWithObstacles(int[][] obstacleGrid) {

int row = obstacleGrid.length;

int col = obstacleGrid[0].length;

if(obstacleGrid[0][0] == 1) {

return 0;

} else {

obstacleGrid[0][0] = 1;

}

for(int i = 1; i < row; i++) {

if(obstacleGrid[i][0] == 0 && obstacleGrid[i-1][0] == 1) {

obstacleGrid[i][0] = 1;

} else {

obstacleGrid[i][0] = 0;

}

}

for(int i = 1; i < col; i++) {

if(obstacleGrid[0][i] == 0 && obstacleGrid[0][i-1] == 1) {

obstacleGrid[0][i] = 1;

} else {

obstacleGrid[0][i] = 0;

}

}

for(int i = 1; i < row; i++) {

for(int j = 1; j < col; j++) {

if(obstacleGrid[i][j] == 1) {

obstacleGrid[i][j] = 0;

} else {

obstacleGrid[i][j] = obstacleGrid[i-1][j] + obstacleGrid[i][j-1];

}

}

}

return obstacleGrid[row-1][col-1];

}

}

本周着重回顾了动态规划的高级部分（更加完善地讲解了动态规划的使用方法），总结了比如爬楼梯问题、最短路径问题、路径数量问题等典型问题的DP方程，讲解了之前动态规划那一章节留的高级DP的问题，重点是编辑距离这个题，我曾经看到腾讯面试问到了这道题，而且在实际工作中也比较常用。我个人感觉最短路径的问题很多都是自顶向下来考虑DP方程的。

此外还介绍了字符串相关的很多算法，说实话这部分体量可以说很重，实际工作中也有很多很多地方都要和字符串打交道，字符串结合DP的问题更是有趣，开阔了眼界。

非常感谢超哥、班主任、助教老师和同学们的陪伴，就像任何一所大学毕业典礼的时候讲的，毕业，只是日后新征程的起点。况且很多题都没刷完，或者没有过完5遍！！

毕业，只是新的开始，数据结构和算法，需要的时间远不止2个月，甚至远不止2年。