# 13. 机器人的运动范围

[牛客网](https://www.nowcoder.com/practice/6e5207314b5241fb83f2329e89fdecc8?tpId=13&tqId=11219&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking&from=cyc_github)

## 题目描述

地上有一个 m 行和 n 列的方格。一个机器人从坐标 (0, 0) 的格子开始移动，每一次只能向左右上下四个方向移动一格，但是不能进入行坐标和列坐标的数位之和大于 k 的格子。

例如，当 k 为 18 时，机器人能够进入方格 (35,37)，因为 3+5+3+7=18。但是，它不能进入方格 (35,38)，因为 3+5+3+8=19。请问该机器人能够达到多少个格子？

## 解题思路

使用深度优先搜索（Depth First Search，DFS）方法进行求解。回溯是深度优先搜索的一种特例，它在一次搜索过程中需要设置一些本次搜索过程的局部状态，并在本次搜索结束之后清除状态。而普通的深度优先搜索并不需要使用这些局部状态，虽然还是有可能设置一些全局状态。

private static final int[][] next = {{0, -1}, {0, 1}, {-1, 0}, {1, 0}};
private int cnt = 0;
private int rows;
private int cols;
private int threshold;
private int[][] digitSum;

public int movingCount(int threshold, int rows, int cols) {
 this.rows = rows;
 this.cols = cols;
 this.threshold = threshold;
 initDigitSum();
 boolean[][] marked = new boolean[rows][cols];
 dfs(marked, 0, 0);
 return cnt;
}

private void dfs(boolean[][] marked, int r, int c) {
 if (r < 0 || r >= rows || c < 0 || c >= cols || marked[r][c])
 return;
 marked[r][c] = true;
 if (this.digitSum[r][c] > this.threshold)
 return;
 cnt++;
 for (int[] n : next)
 dfs(marked, r + n[0], c + n[1]);
}

private void initDigitSum() {
 int[] digitSumOne = new int[Math.max(rows, cols)];
 for (int i = 0; i < digitSumOne.length; i++) {
 int n = i;
 while (n > 0) {
 digitSumOne[i] += n % 10;
 n /= 10;
 }
 }
 this.digitSum = new int[rows][cols];
 for (int i = 0; i < this.rows; i++)
 for (int j = 0; j < this.cols; j++)
 this.digitSum[i][j] = digitSumOne[i] + digitSumOne[j];
}