## 概要

很多问题的求解需要经过多步决策，每种决策对应一种状态，经过多步决策才能到达终态。未达到终态前，一个状态可能会生成多个新的状态。将状态看做树的节点，那么问题求解的过程就是建立并搜索一棵树的过程，终态就是叶节点。有些状态已经意味着不可能生成需要的解了，那么就不必要再生成它的子节点并求解下去，应该及时将该种情况略去，这就是剪枝。一般为保证搜索过程中不会重复搜索已搜索的节点，需使用一个bool型的vis数组来标记已决策过的状态，若为树形dp，还需用一个dp数组来记录已决策过的状态的值。

搜索一般使用递归来实现，函数参数就标记着一种决策状态。

dfs用得比较多，很多dp问题都要用dfs求解，dfs比较灵活，方便回溯，它是一次性从根节点搜索到底，即叶节点，然后回溯到上一个节点，若它还有未搜索的节点，则从它再搜索到底，否则再回溯到上一个节点，如此重复。回溯的方法是，对当前状态生成的子状态都进行dfs，判断返回值，根据返回值判断是否要将本状态“修改了的值”（例如vis数组）“还原”，这样本节点的父节点（若有）虽生成了本节点并进行了决策，修改了值，但最后又改回来了，不会影响其他子节点的决策。当然，有时会无论如何都应恢复状态。

bfs对树一层一层地搜索，向底部“蔓延”，需使用队列实现，即先将根节点放入队列，然后当队列非空时，不断取出队头元素，生成它的子节点并加入队列，当然若已经为叶节点就没有子节点了。显然这样搜索，所在层数越小的节点越先被搜索，于是可以方便地求解路的权值相同时的最短路问题。

## 求连通子图节点数

高h宽w的矩阵，由 @ . # 组成一个人由@出发，每次只能走上下左右的方向，#为障碍物， .可走，问最多能走多少格。

将可走方向的@和.连成一张图，节点数即为所求。dfs建成一棵树，建树过程中统计节点数即可，搜索过的节点置为#防止再搜。

## 求连通子图数

题目和上面的差不多，有#@\*三种字符，求的是在一块的块数。

从矩阵的每一个元素出发dfs，只要该点可走，计数就加一，然后将连在一块的节点都置为已访问即可。

## 下落的球

编号1-10的球下落，每次可以控制让它落在左边或右边的管道里，但该球编号必须大于该管道顶部球的编号。现给出10个球的下落顺序编号，问能否按规则放好。

第一个球放左右都无所谓，接下来总是有两种选择，dfs加剪枝即可。

## 冰壶游戏

被Poj坑惨了，调了我整整一上午没找出错在哪，下午才发现，g++能过，C++就一直WA……

又是一个矩阵，0为空地，1为石头，2为出发点，3为终点。冰壶停在出发点，可以向上下左右4个方向运动（相邻格子有石头就不行），冰壶运动后就不会停下，除非撞到石头才会停下（该运动方向前面有石头，冰壶停在相邻格子），然后石头会破掉变成空地，此时冰壶可以再次选择向4个方向运动。从静止到运动算1步，问最少多少步能到达终点，到达不了或者超过10步则输出-1。

虽说是最短路，但比较复杂，bfs不好写，还是使用dfs更灵活方便。状态设置当前坐标、是否为运动态、运动方向、当前已走步数。这样可以简化成一格一格移动。设置全局变量MIN，每次到达终态将已走步数和它比较，求最小值。不必标记已走，因为每走一步都会改变道路状态，走过的不可能再走，但因为会改变状态，每一个状态结束后都应该恢复状态。

## 老鼠吃奶酪

还是一个矩阵，由.XS数字组成，.为能走道路，X为障碍物，X为工厂，当然，除了障碍物都是能走的，老鼠从S出发可以向上下左右4个方向移动，因为种种原因，需要从S到1，再从1到2……直到n，问总的最小移动数。

可以分开来求每一次的最短路，最后求和。使用bfs，每个节点状态记录着当前坐标和已走步数，最先到达终态的节点，步数即为所求。

## 7数码问题

小时候玩的那啥，缺一块，可以移来移去的那个 。

一开始打乱了，问最少移多少次可以移成有序。

缺的那块编号为0，显然拼接过程0在不断移动，每次移动，编号排列就是一种状态。

那就成了一个搜索题，相当于从初态转移到末态的最短路，用bfs。

测试样例多，但每次数据量小，不如倒过来，先预处理，从有序的末态开始记录下每个初态，最后直接输出 。

因为移动过程中可能出现重复状态，所以用map存下每个状态对应的移动次数以备后用。