

算法基础——搜索

邓丝雨

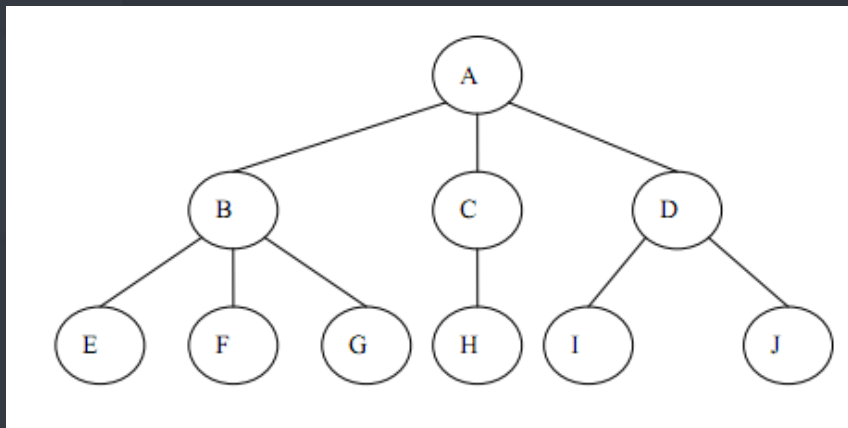


复习：栈和队列的概念

- 栈是限定仅在表头进行插入和删除操作的线性表（先进后出）
- 队列是只允许在表的前端（front）进行删除操作，而在表的后端（rear）进行插入操作。

复习：树

- 树是一种数据结构，它是由 n ($n \geq 1$) 个有限节点组成一个具有层次关系的集合。把它叫做“树”是因为它看起来像一棵倒挂的树，也就是说它是根朝上，而叶朝下的。它具有以下的特点：
- 每个节点有零个或多个子节点
- 没有父节点的节点称为根节点
- 每一个非根节点有且只有一个父节点；除了根节点外，每个子节点可以分为多个不相交的子树





搜索

- 通过不停的**试探**去寻找解的一种算法。
- 与其说是一种算法，不如说是一种方法。
- 基础的方法有暴力的搜索法，深搜，广搜三种。
- 更高级的有IDDFS，DBFS，A*，IDA*等等

深搜



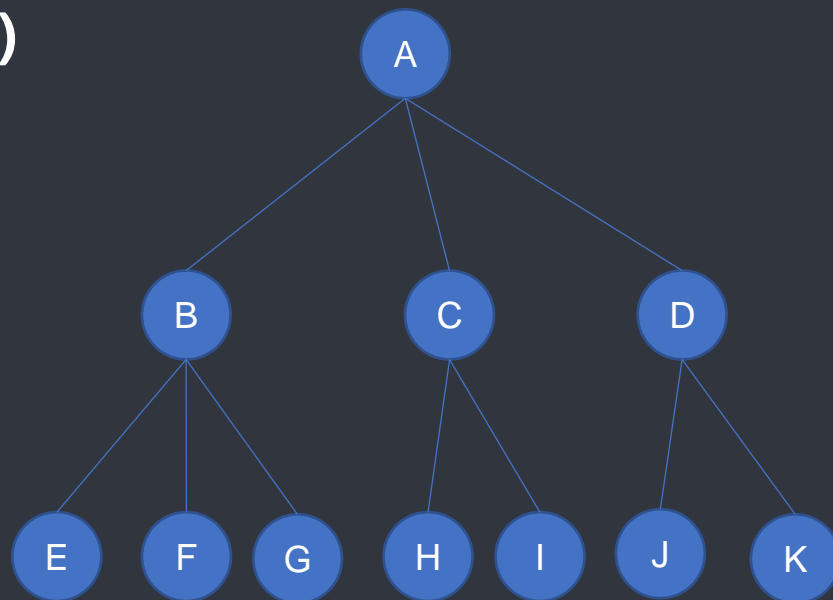
深度优先搜索 (dfs)

“一条道走到黑” “走不了了再倒回去”

算法过程：

VOID DFS(状态 A)

1. 判断当前的状态是否合法。合法则继续执行，否则则回到上次调用。
2. 先下走一层，也就是调用DFS(状态 $A + \Delta$)



例1：输出n个数的全排列

- 1,2,3 组成的全排列为：
- 123
- 132
- 213
- 231
- 312
- 321

可以写n重for循环
也可以用stl里面的next_permutation



- 递归的来想这个问题，以1,2,3,4为例
- 如果第一个数固定为1的话
- 后面就跟着2,3,4的全排列——所以就相当于是原问题的子问题的求解
- 考虑用递归解决



- Void dfs (已经固定了前 $k-1$ 个数剩下的数的全排列，是哪 $k-1$ 个数通过标记该数字用没用过来显示，当前这一步的任务就是把第 k 个数放上去)
- {
 - 如果 已经求出来了 $k > n$ 输出， 返回
 - 否则 i 从1到 n 循环
 - 如果 i 没有用过，那么就将 i 固定在当前位置上，并调用 dfs ($k+1$)
 - 在调用完dfs ($k+1$) 后需要将固定在当前位置上的 i 拿走
- }

```
17 void dfs(int dep)
18 {
19     if (dep > n) {write(); return; }
20
21     for (int i = 1; i <= n; i++)
22     {
23         if (!vis[i])
24         {
25             vis[i] = 1;
26             a[dep] = i;
27             dfs(dep+1);
28             vis[i] = 0;
29         }
30     }
31 }
```



扩展：输出n个数中选M个的组合

- $N=5$ $M=3$

- 1 2 3

- 1 2 4

- 1 2 5

- 1 3 4

- 1 3 5

- 1 4 5

- 2 3 4

- 2 3 5

- 2 4 5

- 3 4 5



例2：N皇后（8皇后的升级版）

- 在 $N \times N$ 的棋盘上放置 N ($n \leq 13$) 个皇后而彼此不受攻击（即在棋盘的任一行，任一列和任一对角线上不能放置2个皇后），编程求解所有的摆放方法。

	1	2	3	4	5	6	7	8
1						♛		
2			♛					
3					♛			
4							♛	
5	♛							
6				♛				
7		♛						
8								♛

	1	2	3	4	5	6	7	8
1								
2	♛							
3					♛			
4		♛						
5								♛
6			♛					
7							♛	
8				♛				

八皇后的两组解



基本思想

- 由于皇后的摆放位置不能通过某种公式来确定，因此对于每个皇后的摆放位置都要进行**试探**和**纠正**，这就是“回溯”的思想。
- 在N个皇后未放置完成前，摆放第i个皇后和第i+1个皇后的试探方法是相同的，因此完全可以采用**递归**的方法来处理。
- 由于皇后本身的特殊性质，即一行一列只能有一个皇后，所以我们所要做的就是，从第0行开始摆放，一直摆到第n - 1行为止。



关于判断当前皇后可不可以放：

- 我们是一行一行的放置皇后，所以不需要判断行冲突；
- 判断列冲突时，可以通过设置一个布尔数组，如果已经有皇后放在那里，就把布尔值设为1，如果可以放置并且没有冲突（即布尔值为0），就放置当前这个皇后，且设置为1；
- 判断对角线冲突时，有一个特殊的技巧：
- 由于每一条主对角线 $(x-y)$ 的值是一定的，每一条副对角线 $(x+y)$ 的值是一定的！！
- 所以就可以用 $(x + y)$ 的值表示副对角线， $(x - y)$ 的值表示主对角线；
- （于是就和处理列的情况一样了！）
- 假设我们把第 cur 个皇后放在了 $pos[cur]$ （ $pos[cur]$ 储存了这个值），那么只需判断所检查的从前往后数第 k 个皇后有没有冲突就行了。

```
22 void dfs(int dep)
23 {
24     if (dep > n) {write(); return; }
25
26     for (int i = 1; i <= n; i++)
27     {
28         if (!vis[i] && !l[i+dep] && !r[i-dep+n])
29         {
30             vis[i] = 1; l[i+dep] = 1; r[i-dep+n] = 1;
31             a[dep] = i;
32             dfs(dep+1);
33             vis[i] = 0; l[i+dep] = 0; r[i-dep+n] = 0;
34         }
35     }
36 }
```





例3：马踏棋盘

- 给一个 5×5 的国际象棋棋盘，国际象棋的马同样是走“日”字，如图：
- 然后：问，如果一个马，从第一个格子开始走，那么走遍整个 5×5 的棋盘的方案，有多少种？并且输出方案数。



如图：

思路：

1. 把问题抽象出来，就是有一只马要对整个图进行一次遍历，不重不漏。
2. 其次考虑这个马的走法（8种）
3. 接着如何去在程序里面表现出这个棋盘
4. 最后就是套用回溯法的主要结构



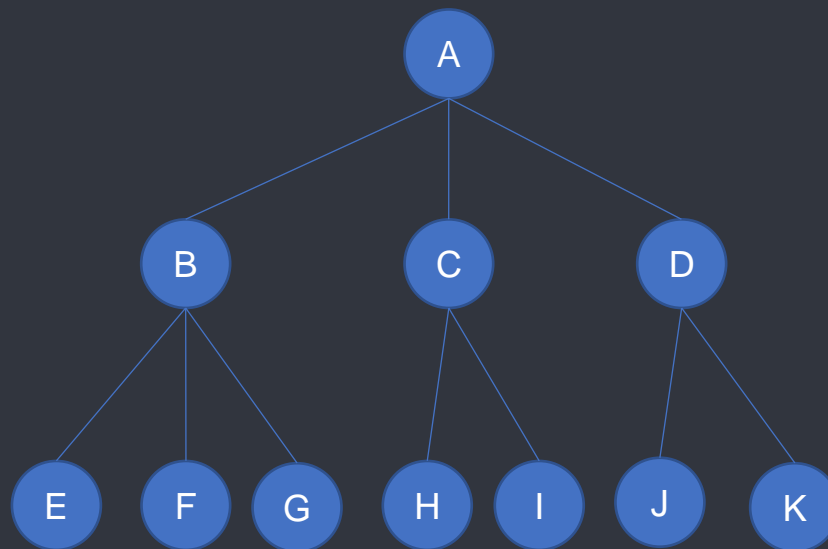
主要过程——DFS 大体框架

- 试探节点A
- A是否满足在这个图（或树）中
- 如果在，标记A如果已经被试探过的话，所影响的各种值；
- 紧接着，去试探所有的A可以达到的节点；
- 等待所有的都执行完之后，还原标记A

广搜

广度优先搜索 (Bfs)

- 一层一层的走!
- 广搜总是每次都把离上一状态最近的状态用一个队列记录下来;
- 记录之后, 检查队列是否为空, 如果不为空, 就讲队首元素弹出, 并且以这个状态为“根节点”进行广度优先搜索。
- 直到整个队列为空为止。





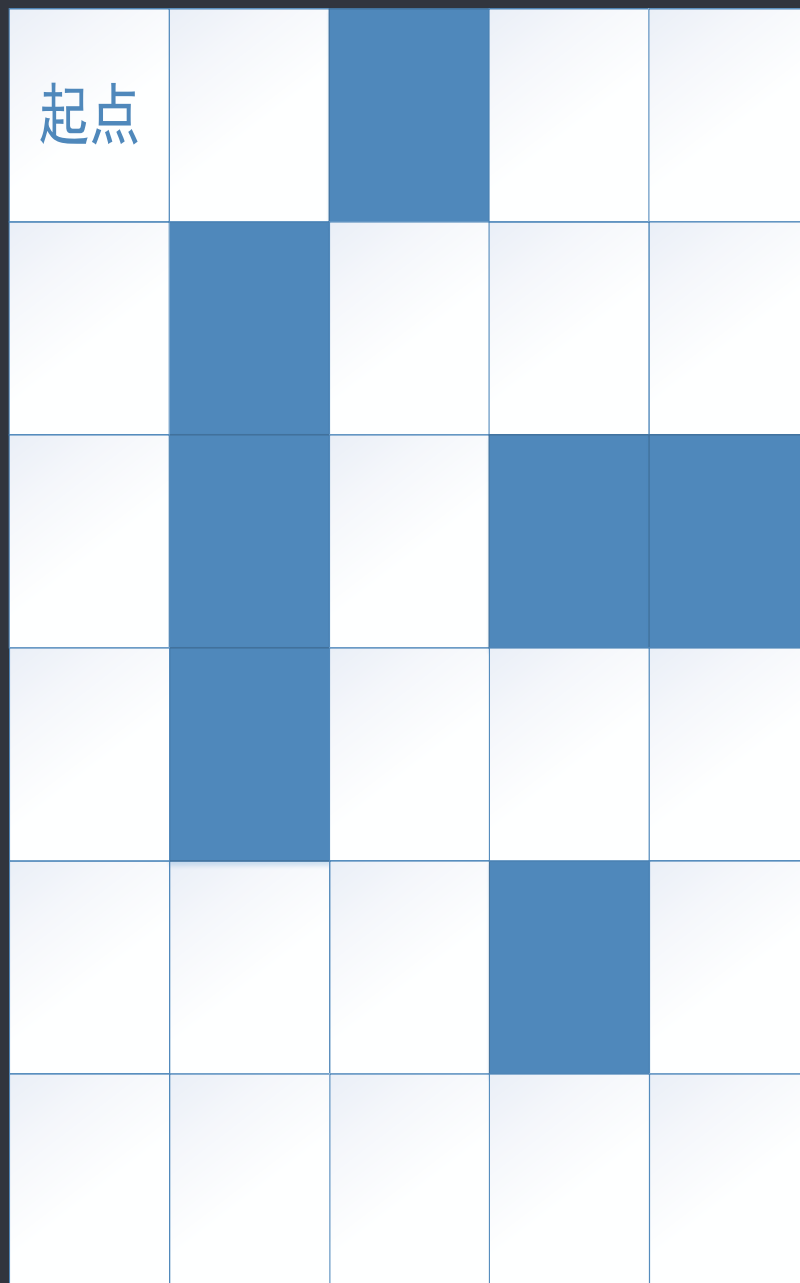
例1：走迷宫

- 设有一个 $n*m$ 方格的迷宫，即 n 行 m 列的迷宫。
- 迷宫格子中分别放有0和1，1表示可通，0表示不能，
- 从某点开始，有上下左右四个方向可走，输入起终点坐标问能否走出去



比如这样一个迷宫：

- 6 5
- 1 1 0 1 1
- 1 0 1 1 1
- 1 0 1 0 0
- 1 0 1 1 1
- 1 1 1 0 1
- 1 1 1 1 1





手工算法

- 从第一节点开始，逐步计算
- 怎么算？
- 建立一个等待处理的节点的队列。
- 先把一步以内能够走到的节点加进来。
- 然后对这个队列的元素进行处理：即从队列中删除这个节点A，然后再把这个节点A的能够一步走到的节点加入队列。
- 直到这个队列空为止。



- 对于坐标的处理方法
- 本来是用两个数来表示的坐标(x, y), 可以用一个数来表示。
- 为什么要这样? 简便呗!
- 第i行第j列的格子编号为 $i*m+j$
- (横纵坐标的起点都是0)
- 反之, 编号为u的节点, 其行号和列号分别为 u / m ; $u \% m$



例2：马的遍历

- 有一个 $n*m$ 的棋盘($1 < n, m \leq 400$), 在某个点上有一个马, 要求你计算出马到达棋盘上任意一个点最少要走几步

搜索剪枝



优化搜索的思路：

- 核心：减小搜索树的大小
- 方法：
 - 一、改变搜索顺序
 - 二、剪枝：最优化剪枝 & 可行性剪枝



例1:

- $N*M$ 的迷宫中给定你起点S，和终点D，问你是否能在 T 时刻恰好到达终点D。
- S.X.
- ..X.
- ..XD



例2：小木棍

- 乔治拿来一组等长的木棒，将它们随机地砍断，使得每一节木棍的长度都不超过50个长度单位。然后他又想把这些木棍恢复到为裁截前的状态，但忘记了初始时有多少木棒以及木棒的初始长度。请你设计一个程序，帮助乔治计算木棒的可能最小长度。每一节木棍的长度都用大于零的整数表示。
- 小木棍的数量 ≤ 64



- 枚举最终小木棍的长度，搜索能不能拼出K根
- 但是会超时.....



- 将所有题目给的棍子的长度按照从大到小的顺序排列，然后按照此顺序进行深搜。



- 如果第 i 个棍子不能拼成假设的长度，则和第 i 个棍子相同长度的棍子也是不可能的，所以可以直接跳过去的！



- 替换第 i 根棍子的第一根木棒是没用的



- 如果某次拼接选择长度为 S 的木棒，导致最终失败，则在**同一位置**尝试下一根木棒时，要跳过所有长度为 S 的木棒。

```
24 bool dfs(int num, int len, int rest)
25 {
26     if((rest == 0) && (num == 0)) return true;
27     if (rest == 0) rest = len;
28     for(int i = 0; i != n; i++)
29     {
30         if (v[i]) continue;
31         if (a[i] > rest) continue;
32         v[i] = true;
33         if(dfs(num - 1, len, rest - a[i])) return true;
34         v[i] = false;
35         if((a[i] == rest) || (len == rest)) break;
36         while (a[i] == a[i + 1]) i++;
37     }
38     return 0;
39 }
```

一点点思考



搜索究竟是解决的什么问题？

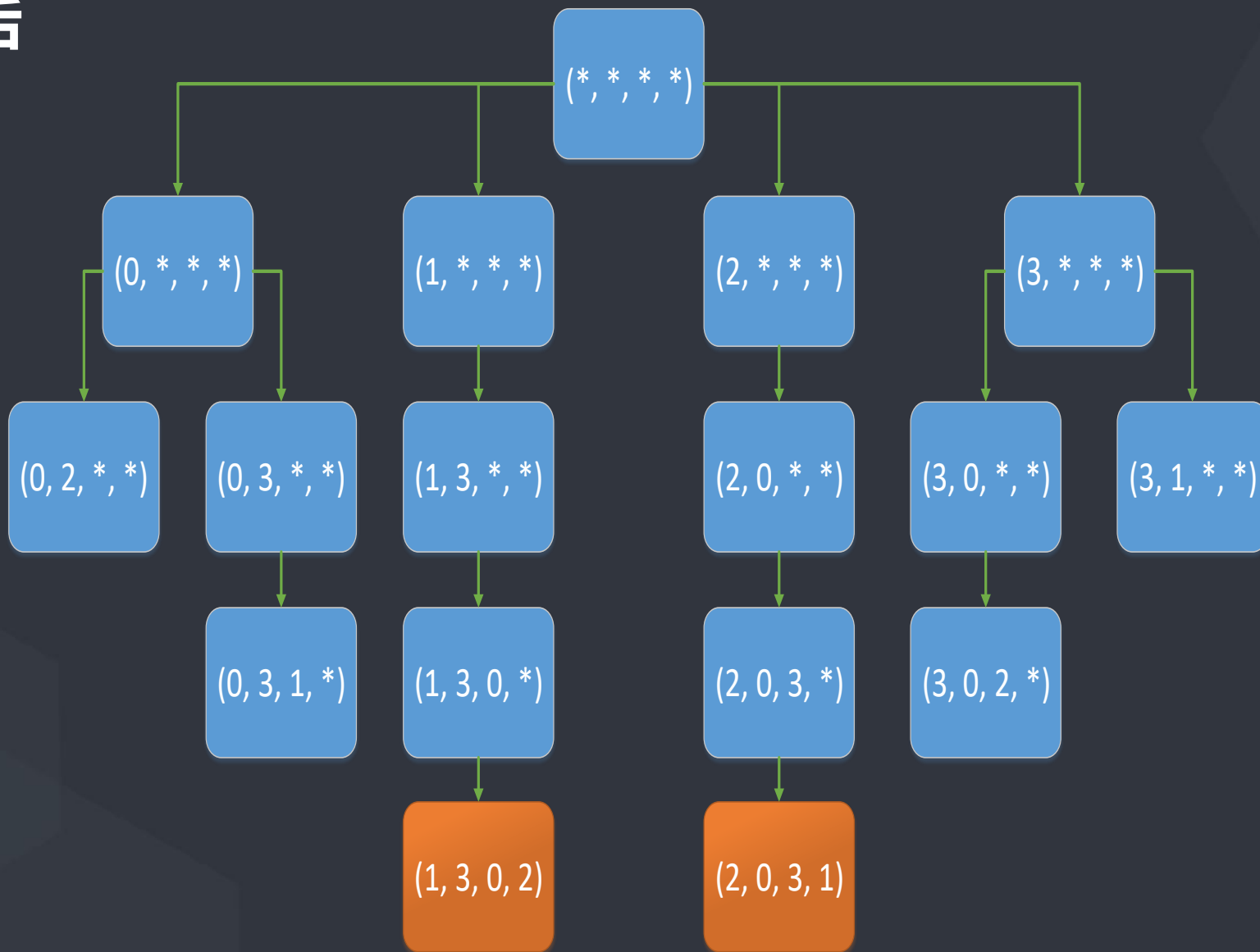
- 在某一个空间里寻找目标
- 空间指的是**解空间**
- 目标指的是**目标状态**
- 那么你就要问我了：解空间和目标状态又是什么？



- 解空间：也就是如果把一个问题的解抽象成一个数学上的向量，那么包含这个向量空间，也就是解空间。
- 状态：用于描述问题或者问题的解的一些量（我是谁？我在哪？）
- 以四皇后为例：
- 解空间**可以**为一个四维的空间，每一维的取值为 0 到 3
- （可以的意思就是，对于问题来说，解空间往往不是唯一的，我如果把这个加一维，他依然是这个问题的解空间）
- 状态则是指的是 (a, b, c, d) 所表示四个的皇后分别放在 $(0, a)$ $(1, b)$ $(3, c)$ $(4, d)$ 四个位置



4皇后



一些题目



例1：NC16591 关押罪犯

- S 城现有两座监狱，一共关押着 N 名罪犯，编号分别为 $1 \sim N$ 。他们之间的关系自然也极不和谐。很多罪犯之间甚至积怨已久，如果客观条件具备则随时可能爆发冲突。我们用“怨气值”（一个正整数值）来表示某两名罪犯之间的仇恨程度，怨气值越大，则这两名罪犯之间的积怨越多。如果两名怨气值为 c 的罪犯被关押在同一监狱，他们俩之间会发生摩擦，并造成影响力为 c 的冲突事件。
- 每年年末，警察局会将本年内监狱中的所有冲突事件按影响力从大到小排成一个列表，然后上报到 S 城 Z 市长那里。公务繁忙的 Z 市长只会去看列表中的第一个事件的影响力，如果影响很坏，他就会考虑撤换警察局长。
- 在详细考察了 N 名罪犯间的矛盾关系后，警察局长觉得压力巨大。他准备将罪犯们在两座监狱内重新分配，以求产生的冲突事件影响力都较小，从而保住自己的乌纱帽。假设只要处于同一监狱内的某两个罪犯间有仇恨，那么他们一定会在每年的某个时候发生摩擦。那么，应如何分配罪犯，才能使 Z 市长看到的那个冲突事件的影响力最小？这个最小值是多少？
- $N \leq 20000, M \leq 100000$





例2: NC51032 八数码问题

- 在 3×3 的棋盘上，摆有八个棋子，每个棋子上标有1至8的某一数字。棋盘中留有一个空格，空格用0来表示。空格周围的棋子可以移到空格中。要求解的问题是：给出一种初始布局（初始状态）和目标布局（为了使题目简单,设目标状态为123804765），找到一种最少步骤的移动方法，实现从初始布局到目标布局的转变。

- | | |
|---------|---------|
| • 1 2 3 | • 1 2 3 |
| • 4 5 6 | • 4 5 6 |
| • 7 X 8 | • X 7 8 |



例3: NC 15665 maze

- $n \times m$ 个格子组成的迷宫，有些格子是陷阱，用'#'表示，小明进入陷阱就会死亡， '.'表示没有陷阱。小明所在的位置用'S'表示，目的地用'T'表示。小明只能向上下左右相邻的格子移动，每移动一次花费1秒。
- 有 q 个单向传送阵，每个传送阵各有一个入口和一个出口，入口和出口都在迷宫的格子里，当走到或被传送到一个有传送阵入口的格子时，小明可以选择是否开启传送阵。如果开启传送阵，小明就会被传送到出口对应的格子里，这个过程会花费3秒；如果不开启传送阵，将不会发生任何事情，小明可以继续向上下左右四个方向移动。
- 一个格子可能既有多多个入口，又有多多个出口，小明可以选择任意一个入口开启传送阵。使用传送阵是非常危险的，因为有的传送阵的出口在陷阱里，如果小明使用这样的传送阵，那他就会死亡。也有一些传送阵的入口在陷阱里，这样的传送阵是没有用的，因为小明不能活着进入。请告诉小明活着到达目的地的最短时间。

例4: NC117187 uva 11573 Ocean Currents

- $n*n$ 的海面上，每个格子都有一个潮水涌动的方向（共8个方向），初始时船在某个位置，每次可以选择顺着水涌动的方向前进，**不消耗能量**。或者消耗一个能量前往任一个方向。问从起始位置到目标位置最少需要消耗多少能量。

例5:NC 111125 CF590C Three States

- 有三个国家，想要相互交通
- 有一个 $n*m$ 的矩阵，#代表墙，表示不能走动有一个
- 1, 2, 3代表国家，之间可以走动
- .代表可开设的路，开设路需要花费1
- 最少花费多少可以让三个国家相互连通相互连通



例6: NC202589 魔法数字

- 牛妹给牛牛写了一个数字 n ，然后又给自己写了一个数字 m ，她希望牛牛能执行最少的操作将他的数字转化成自己的。
- 操作共有三种，如下：
 1. 在当前数字的基础上加一，如：4转化为5
 2. 在当前数字的基础上减一，如：4转化为3
 3. 将当前数字变成它的平方，如：4转化为16
- 返回最少需要的操作数。



例7: NC 204418 新集合

- 集合 s 中有整数 1 到 n ，牛牛想从中挑几个整数组成一个新的集合。现在牛妹给牛牛加了 m 个限制，每个限制包含两个整数 u 和 v ($u \neq v$)，且 u 和 v 不能同时出现在新集合中。
- 请问牛牛能组成的新集合多少种。
- 可以选 0 个数。
- 返回一个整数，即新集合的种类数。