Mirror Box (Codeforces 578F) 题解

南京外国语学校 陈孙立

November 16, 2019

1 题面

1.1 题目描述

有一个长方形黑盒里装满了镜子。这个黑盒被划分成 $n \times m$ 的正方形网格，每个格子都恰好包含一个"\" 型或"/" 型的镜子，且和盒子的边界成 45° 角。现在有一些网格里已经放上了镜子，你需要在剩下的每个格子里选择"\" 型或"/" 型，使得以下两个条件满足：

- 把盒子的边界分成 $2n + 2m$ 个单位小段，并称两小段相邻当且仅当有一个公共点。对于任何一个小段，如果一束垂直于它的光从小段的中点射入黑盒，那么它必须从一个相邻的小段射出。
- 如果按照上述方法从每个小段射入一束光，那么对于盒子里的每个方格的 4 个边界段，都至少有一束光穿过了它。

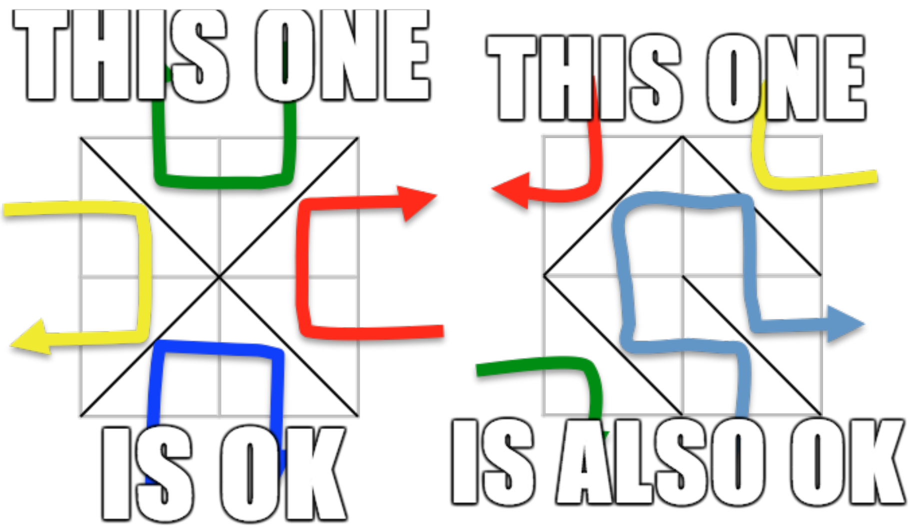


Figure 1: 如上图，左右都是满足条件的放置方式

你试着放了几个镜子，发现有可能有多种放置方法使得以上两个条件被满足。请问总共有多少种放镜子的方法？由于答案很大，你需要模一个质数 MOD 输出。

1.2 输入格式

第一行包含三个整数 n, m, MOD 。

接下来 n 行每行一个长为 m 的字符串，表示盒子中这一行的放置镜子的情况，每个字符如果是"*" 表示未放置，否则是"\" 和"/" 中的一个，分别表示对应形状的镜子。

1.3 输出格式

输出一个整数，表示方案数模 MOD 的结果。

1.4 样例输入

```
2 2 1000000007
*/
/*
```

1.5 样例输出

```
1
```

1.6 样例解释

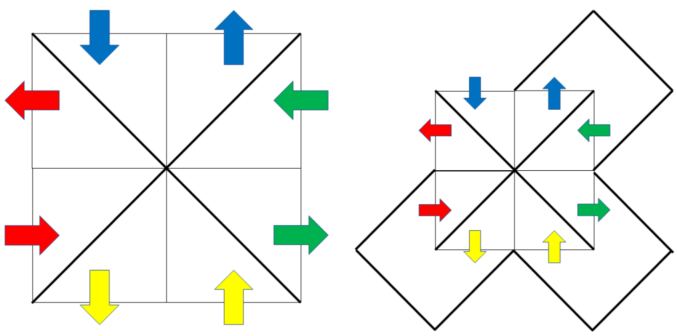
仅有的一种满足条件的方法为题目描述中左图的形状。

1.7 数据范围与约定

保证 $1 \le n, m \le 100$ ， $3 \le MOD \le 10^9 + 7$ ， 且 MOD 是质数。

保证输入中字符"*" 的个数不超过 200 。

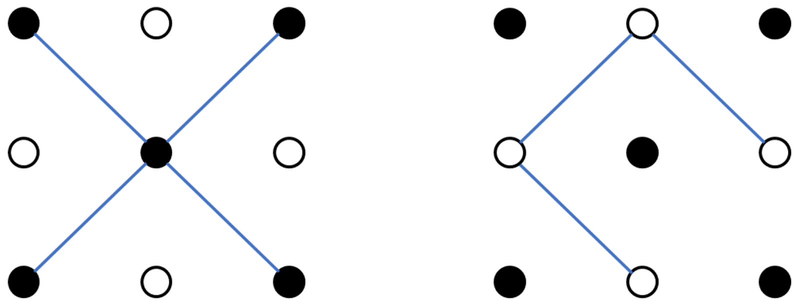
2 解题过程



不妨假设我们是按照上图中左边的方式射入与射出的，其中同色箭头代表同一束光线。那么我们按照右边的方式，新添加一些边，使得其构成一张完整的平面图。注意左上角没有加边是为了让外围的无限大区域也和内部区域联通。

结合题目条件，不难发现这张新图的对偶图就是一个简单环。根据平面图的性质，我们可以知道，对偶图中的简单环可以和原图中的割一一对应，由于对偶图中只有一个环，我们可以知道原图中的割是唯一的。然而，如果我们把原图的点黑白染色，如下图所示，那么每条边只会连接两个同色点，因此黑色、白色即为一个合法的割，且它们必须各自联通，否则会存在更多的割。

在上图中，新加入的边只连接了白点，因此我们可以得到黑点在原图中是联通的（如下图左）。对称地，如果是按照另一种方式射入、射出，就可以得到白点在原图中是联通的（如下图右）。



可以计算图的点数为 $(n+1)(m+1) = nm+n+m+1$ ，原图的边数为 nm ，新加入的边数（三条折线看作一条边）为 $n+m-1$ ，而联通块个数为 2 ，因此有点数 $=$ 边数 $+$ 联通块数，也即这个图无环。这将是解决本题的关键性质。

性质：放置方法满足题目条件的必要条件为：黑点及其之间的边构成树，或者白点及其之间的边构成树。

根据染色方法可以看出，如果某种颜色的点形成了一个环，则另一种颜色的点将会有部分在环内，另一部分在环外，因此不能形成树。由此可得，**上述性质也是充分的**。由于点数和边数的限制，黑点成树时，白点（不考虑新边）将会形成 $n+m$ 个联通块，当 $n,m > 0$ 时不可能是树。

综上所述，合法的放置方法个数为：黑点成树的方案数 + 白点成树的方案数。下面只考虑求黑点成树的方案数。

首先先判断已经确定的黑点之间的边是否形成环，如果成环显然无解。注意到图中最多只有 200 个未确定的位置，因此最多只会添加 200 条边，当黑点的联通块个数大于 201 时必然无解，对于剩下的情况，在缩点之后最多只剩下 201 个点，因此在缩起来的图上求生成树的个数即可。生成树计数可用 Matrix-Tree 定理 [1]，以及行列式来计算。白点成树的计算也是完全类似的。总的复杂度为 $O(nm+t^3)$ ，其中 t 为 * 的个数。

[1]Matrix-Tree 定理可参考维基百科：https://en.wikipedia.org/wiki/Matrix_tree_theorem

Ant Man (Codeforces 704) 题解

南京外国语学校 陈孙立

November 16, 2019

1 题面

1.1 题目描述

Scott Lang（蚁人）正在大战 Darren Cross（反派蚁人）。他们所在的地方有 n 个椅子排列在一条数轴上，从左到右编号为 $1, 2, \ldots, n$ ，第 i 个椅子在数轴上的 x_i 位置。Scott 在第 s 个椅子上，而 Cross 在第 e 个椅子上。Scott 可以在任意两个椅子之间跳跃。他希望从他所在的位置开始，把每个椅子经过恰好一次，最终到达 Cross 所在的第 e 个椅子。

众所周知，Scott 是蚁人，在任意时刻都有“小”（蚂蚁大小）和“大”（普通人大小）两种形态之一，而且他只能在某个椅子上转换形态，而不能在空中跳跃时进行。Scott 在起跳、跳跃和落地时都需要花时间，但是转换形态不花时间。从第 i 个椅子跳到第 j 个椅子的跳跃时间是 $|x_i - x_j|$ 秒。

Scott 在从右往左跳时只能是“小”形态，而在从左往右跳时只能是“大”形态。

从第 i 个椅子起跳时：

- 如果他是“小”形态，起跳时间是 c_i 秒；
- 否则（“大”形态），起跳时间是 d_i 秒。

落到第 i 个椅子时：

- 如果他是“小”形态，落地时间是 b_i 秒；
- 否则（“大形态”），落地时间是 a_i 秒。

形式化地说，当他要从第 i 个椅子跳到第 j 个椅子上时：

- 如果 $j < i$ 花费时间是 $|x_i - x_j| + c_i + b_j$ 秒；
- 否则（$j > i$）花费时间是 $|x_i - x_j| + d_i + a_j$ 秒。

给定所有的 x, a, b, c, d ，找到 Scott 跳到 Cross 的椅子上的最短时间，要求经过每个椅子恰好一次。

1.2 输入格式

第一行包含三个整数 n, s, e 。

接下来一行包含 n 个整数 $x_1, x_2, \ldots, x_n$ 。

接下来一行包含 n 个整数 $a_1, a_2, \ldots, a_n$ 。

接下来一行包含 n 个整数 $b_1, b_2, \ldots, b_n$ 。

接下来一行包含 n 个整数 $c_1, c_2, \ldots, c_n$ 。

接下来一行包含 n 个整数 $d_1, d_2, \ldots, d_n$ 。

1.3 输出格式

输出一行包含最短时间。

1.4 样例输入

```
7 4 3
8 11 12 16 17 18 20
17 16 20 2 20 5 13
17 8 8 16 12 15 13
12 4 16 4 15 7 6
8 14 2 11 17 12 8
```

1.5 样例输出

```
139
```

1.6 样例解释

最优路线之一为：4→2→1→6→5→7→3。

1.7 数据范围与约定

保证 $1 \leq s, e \leq n \leq 5000$ ，$x_i \leq x_{i+1}$

保证输入中的所有整数都在 $[1, 10^9]$ 之间。

2 解题过程

问题相当于给出了一个 n 个点的有向完全图，对任意 $1 \leq i < j \leq n$ ，i 到 j 的边权为 $|x_i - x_j| + d_i + a_j$ ，j 到 i 的边权为 $|x_i - x_j| + c_j + b_i$ ，要求出 s 到 t 的最短 Hamilton 路径长度，也就是一个固定了起点终点的旅行商问题（Travelling Salesman Problem,TSP）。

众所周知，TSP 问题是 NPC 问题，因此要做到题目中的数据范围肯定需要用到边权的性质。事实上，如果我们定义 $A_i = a_i + x_i, B_i = b_i - x_i, C_i = c_i + x_i, D_i = d_i - x_i$ ，那么可以发现当 $i < j$ 时，i 到 j 的边权为 $D_i + A_j$ ，否则为 $C_i + B_j$ 。也就是是，我们可以修改起跳时间和落地时间，并认为在空中不耗费任何时间。这样做的好处是，我们不需要知道每次跳跃的起点和终点，只需要知道落到第 i 个点时是往左还是往右跳，以及从第 i 个点起跳时是往左还是往右跳。

上一段用图论的话说，就是对于一条 TSP 路径，我们只关心每个点的入边和出边的方向（边的两个端点的大小关系），并以这些信息对每个点算出代价，然后相加。对于一种合法的 TSP 路径和某个 x ，我们考虑所有**两个端点都小于** x 的边，它们会形成若干条路径。对于本题这种只关心入边和出边方向的问题中，我们并不需要知道每条路径的内部情况，只需要知道路径的条数即可，这可以通过之后的转移方式看出。

定义 $dp(x, p)$ 为考虑所有**两个端点都小于** x 的边，它们形成了 p 条路径时，前 x 个点的代价和。这里，当 $x > s$ 时有一条路径以 s 开头，当 $x > e$ 时有一条路径以 t 结尾，因此要求 $p \geq [x > s] + [x > t]$ 。当 x 变为 $x + 1$ 时，最多会多出两条边，即 x 的入边和出边，我们就在这里考虑这两条边的方向。大体的转移方式如下：

- 如果入边为从左往右、出边为从右往左，那么就会新产生两条边，且它们连向任意一个小于 x 的点都是等价的，因此会合并两条路径，也就是用 $dp(x, p) + C_i + A_i$ 去更新 $dp(x + 1, p - 1)$ 。
- 如果入边为从右往左、出边为从左往右，那么不会产生任何边，且它们一定会在后面的转移中被考虑到，因此会新加入一条路径，代表着一个孤立点，也就是用 $dp(x, p) + D_i + B_i$ 去更新 $dp(x + 1, p + 1)$ 。
- 如果入边和出边都是从左往右，那么会产生一条新边，并且由于上面的原因，我们仍然不需要关心这条边连到的是哪一个点，因此会把这个点“拼”在一条路径的右侧，故要求至少有一条路径不是以 t 结尾的，即 $path > [x > e]$ ，用 $dp(x, p) + D_i + A_i$ 去更新 $dp(x + 1, p)$ 。
- 如果入边和出边都是从右往左，那么会产生一条新边，因此会把这个点“拼”在一条路径的左侧侧，故要求至少有一条路径不是以 s 开头的，即 $path > [x > s]$ ，用 $dp(x, p) + C_i + B_i$ 去更新 $dp(x + 1, p)$ 。

上述过程只是当 $x \notin \{s, t, n\}$ 时的情况，当 x 是 s 或 t 时只需要考虑一个方向，当 x 是 n 时必须要转移到 $dp(n + 1, 1)$ ，其他和上面类似。这样转移的复杂度为 $O(n^2)$ ，足够通过本题。

此外，本题还有一个可以得到 AC 的 $O(n \log n)$ 的贪心算法，但我无法证明正确性，故没有放在题解中。

《Eating Symbols Hard (atcoder arc099f) 》题解

南京外国语学校 陈孙立

November 16, 2019

1 题面

1.1 题目描述

在 Takahashi 的脑海中，时刻有一个长度为 $2\times10^9+1$ 的整数序列 $A=(A_{-10^9}, A_{-10^9+1}, \ldots, A_{10^9})$ 和一个整数 P 。

一开始，A 中所有的元素和 P 都是 0 。

当 Takahashi 吃下 +->< 四种符号中的一个时，A 和 P 按以下规则改变：

- 吃下 $+$ 时，A_P 的值加一；
- 吃下 -时，A_P 的值减一；
- 吃下 $>$ 时，P 的值加一；
- 吃下 $<$ 时，P 的值减一。

Takahashi 有一个长度为 N 的字符串 S ，每个字符都是 +->< 之一。他会选择整数对 (i,j) 满足 $1\le i\le j\le N$ ，并按从左到右的顺序吃下 S 的第 i 个、第 $i+1$ 个、...、到第 j 个符号。我们听说，在他吃完之后，序列 A 将会和 $i=1, j=N$ 时，也就是吃掉整个 S 之后的结果一样。有多少种满足条件的整数对 (i,j) ？

1.2 输入格式

输入第一行包含一个整数 N ，接下来一行包含一个长为 N 的字符串 S 。

1.3 输出格式

输出满足条件的有序整数对 (i,j) 个数。

1.4 样例输入

```
5
+>+<-
```

1.5 样例输出

```
3
```

1.6 样例解释

满足条件的有 $(1,5)$, $(2,3)$, $(2,4)$ 。

1.7 数据范围与约定

保证 $1 \le N = |S| \le 250000$ ，且 S 中仅包含 +->< 四种字符。

2 解题过程

2.1 主要算法

对整数序列 A 定义分式函数 $A(x) = \sum_{i=-10^9}^{10^9} A_i x^i$ 。虽然这不是多项式，但是由于次数是有限的，仍然可以看作多项式，并正常地进行四则运算。如果从前往后依次加字符，似乎无法避免把 P 带入状态中，因此我们考虑从后往前加字符。

令一个操作序列 S 结束后的分式为 $F(S)$ ，那么不难发现 $F(+S) = F(S) + 1$ ，$F(-S) = F(S) - 1$ ，$F(> S) = xF(S)$ ，$F(< S) = x^{-1}F(S)$ 。这样，判断两个串 S, T 的执行结果是否相等就相当于判断 $F(S)$ 和 $F(T)$ 是否恒等。判断两个类似多项式的分式恒等并不容易，不过我们可以采用哈希的思想，在模大质数 P 下，随机代入若干个 x 。关于哈希冲突的分析会在之后提到。以下假设已经选定了大质数 P 和随机整数 $0 \le x < P$ ，并假设在域 $\mathbb{F}_P$ 下四则运算均为 $O(1)$ 时间。

选定了 x 之后，就可以 $O(|S|)$ 计算 $F(S)$ 了。如果要对于任意 (i, j) 计算 $F(S[i \dots j])$ ，可以利用这些操作的可逆性，即 +-互为逆运算，>< 互为逆运算，先算出 $F(S[1 \dots j])$ 再依次施以 $S_1, \dots S_{i-1}$ 的逆运算。由于 +->< 所对应的都是一次函数，可以用 $f(x) = kx + b$ 的形式来维护，从而方便的计算前述的两个值。

记 $f_i(x)$ 为 $S[1 \dots i]$ 所对应的一次函数，$g_i(x)$ 为 $S[1 \dots i]$ 所对应的一次函数的逆，那么对于数对 (i, j) 它所对应的函数值 $F(S[i \dots j])$ 等于 $g_{i-1}(f_j(0))$ 。那么问题就转化为计算满足 $g_{i-1}(f_j(0)) = f_n(0)$ 的 (i, j) 个数。

当 g_{i-1} 不是常数时，上式可以等价地转化为 $f_j(0) = g_{i-1}^{-1}(f_n(0))$ ，这样左边只和 j 有关，右边只和 i 有关，因此只需要维护每个数的出现次数就能快速计算答案，使用 STL 的 map 或离线排序 + 二分等均可，复杂度为 $O(N \log N)$ 。如果使用哈希表还可以优化到 $O(N)$ 。

2.2 Hash 的冲突分析

本题中的 Hash 冲突指两个串 S, T 满足 $F(S) \ne F(T), F(S, x) = F(T, x)$ ，这相当于不是恒等于零的分式函数 $F(S) - F(T)$ 在 x 处取到零，而这里当 x 非零时可以等价成某个次数大于零的 $\mathbb{F}_P$ 中多项式在 x 处取到零。

假设我们选择的 P 在 10^9 级别。由操作过程可以看出这个多项式的次数不超过 $2N$ ，因此会有至多 $\min(2N,P)=2N$ 个根，于是随机一个数 x 为根的概率不超过 $\frac{2N}{P}$ ，而本题要求的即为任意与 $F(S)$ 不恒等的 $F(S[i\dots j])$ 在 x 处均不相同。如果我们选择了 k 个不同的模数，某对 (i,j) 产生冲突的概率不超过 $\left(\frac{2N}{P}\right)^k$ ，共 $\binom{n}{2}$ 对存在某个产生冲突的概率不超过 $\binom{n}{2}\left(\frac{2N}{P}\right)^k$ （这里仅用 Union Bound 进行估计）。当 $k=6$ 时，这个值不超过 10^{-12} 。

此外，还可能存在某个 g_{i-1} 恒为零的情况，然而由于 g 是一次函数，这发生的概率不超过 $\frac{1}{P}$ ，相对于上述概率基本可以忽略不计。

然而在实际测试中发现，$k=2$ 足以解决本题，这有两个原因，一方面这里的多项式并不是所有的次数都很高，另一方面它们的系数绝对值不会超过 N ，所以产生冲突的概率也会小很多。