NOIP模拟赛 day5

时间：2022 年 11 月 19日 ??:?? ~ ??:??

题目名称	切蛋糕	羊个了羊	彩树	剪刀石头布
题目类型	传统型	传统型	传统型	传统型
输入文件名	cake.in	sheep.in	colorful.in	rps.in
输出文件名	cake.out	sheep.out	colorful.out	rps.out
每个测试点时限	1.0 秒	1.0 秒	1.0 秒	1.0 秒
内存限制	512 MB	512 MB	512 MB	512 MB
测试点数目	10	10	10	10
测试点是否等分	是	是	是	是

提交源程序文件名

对于 C++ 语言	cake.cpp	sheep.cpp	colorful.cpp	rps.cpp

编译选项

对于 C++ 语言	-lm -O2 -std=c++14

注意事项

1. 文件名（包括程序名和输入输出文件名）必须使用英文小写。
2. C++ 中函数 `main()` 的返回值类型必须是 `int`，值必须为 0。
3. 若无特殊说明，输入文件中同一行内的多个整数、浮点数、字符串等均使用一个空格分隔。
4. 若无特殊说明，结果比较方式为忽略行末空格、文末回车后的全文比较。
5. 程序可使用的栈空间内存限制与题目的内存限制一致。
6. 题目不一定按照难度顺序排序，请注意掌握时间。

切蛋糕（cake）

【题目描述】

小 T 有一个矩形的蛋糕。

小 T 认为，切蛋糕时刀必须要平行于矩形的某一边。而且，随意切割会让图案不美观，所以横着只有 $n-1$ 个能够切的位置，竖着只有 $m-1$ 个能够切的位置。

如果直接切 $(n-1)+(m-1)$ 刀，会让这个矩形蛋糕分成 $n\times m$ 个小块。每个小块都有一个美味度，位于第 i 行第 j 列的小块的美味度为 $a_{i,j}$。一块蛋糕的美味度为它所含有的小块的美味度之和。

对于所有 $0\le i\le n-1, 0\le j\le m-1$，你需要让横着恰好切 i 刀，竖着恰好切 j 刀时，切出的 $(i+1)(j+1)$ 块蛋糕中，美味度最小的蛋糕的美味度尽可能大。

【输入格式】

从文件 ***cake.in*** 中读入数据。

第一行两个整数 n,m。

接下来 n 行，每行 m 个整数 $a_{i,j}$，表示第 i 行第 j 列的小块蛋糕的美味度。

【输出格式】

输出到文件 ***cake.out*** 中。

共 n 行，每行 m 个数，第 $i+1$ 行的第 $j+1$ 个数 $ans_{i,j}$ 表示横着恰好切 i 刀，竖着恰好切 j 刀时，美味度最小的蛋糕的美味度最大值。

【样例 1 输入】

```
3 4
1 2 3 4
5 3 2 1
6 3 4 2
```

【样例 1 输出】

```
36 16 8 7
15 6 3 2
10 3 1 1
```

【样例 1 解释】

例如，横着切 1 刀，竖着切 2 刀的一种最优策略为

```
1 | 2 | 3 4
5 | 3 | 2 1
-------------
6 | 3 | 4 2
```

可以发现，此时蛋糕的美味度分别为 $[6,5,10,6,3,6]$，最小值为 3。

【样例 2】

见选手目录下的 ***cake/cake2.in*** 与 ***cake/cake2.ans***。

【样例 3】

见选手目录下的 ***cake/cake3.in*** 与 ***cake/cake3.ans***。

【数据范围与提示】

对于 30% 的数据，$n,m \leq 8$。
对于另外 30% 的数据，$a_{i,j}=1$。
对于 100% 的数据，$1 \leq n,m \leq 14$，$1 \leq a_{i,j} \leq 1000$。

羊个了羊（sheep）

【题目描述】

小 T 有两个长为 n 的序列 a, b。

小 T 很无聊，于是他又找了一个可重集 S。

但是这个可重集 S 比较有个性，它认为自己是高贵的集合，而不是相同元素有一堆导致 .count() 方法复杂度炸裂的垃圾数据结构。

所以一旦当 S 中有三个相同的元素时，它在 10^{-18} 秒后就会把这三个相同的元素一起删掉。

开始时，可重集 S 为空。小 T 每次能做以下两种操作：

- 将序列 a 最开头的未被删除的元素加入可重集 S 中，并把这个元素在序列 a 中删除。
- 将序列 b 最开头的未被删除的元素加入可重集 S 中，并把这个元素在序列 b 中删除。

容易发现，在经过恰好 $2n$ 步操作之后，序列 a 和序列 b 都会被删空。但小 T 发现可重集 S 也被删空了，也就是说，每种值在序列 a 和序列 b 中出现的次数均为 3 的倍数。

小 T 想让这个过程中，可重集 S 的大小在任意时刻都不超过一个阈值 C。他想知道，这个阈值 C 最小能是多少，才能存在一个合法的操作方案。

小 T 想对序列 a 做一些微调，所以对于每个 $1 \le i \le n-1$，你需要回答，假如将 a_i 和 a_{i+1} 交换，那么上面问题的答案（最小的阈值 C）是多少。

注意：每个询问是独立的，也就是说，这次询问的交换并不影响之后的询问。

【输入格式】

从文件 ***sheep.in*** 中读入数据。

第一行两个整数 n，表示序列长度。

第二行 n 个整数 a_i，表示序列 a。

第三行 n 个整数 b_i，表示序列 b。

【输出格式】

输出到文件 ***sheep.out*** 中。

输出 $n-1$ 行，每行一个整数，第 i 行的正整数 C_i 表示，假如将 a_i 和 a_{i+1} 交换，那么最小的阈值 $C = C_i$。

【样例 1 输入】

```
6
1 2 3 2 1 3
4 2 1 4 3 4
```

【样例 1 输出】

```
6
```

```
6
5
7
6
```

【样例 1 解释】

例如，若 $i = 3$，那么交换后的 a 序列为 $[1, 2, 2, 3, 1, 3]$，一种最优地加入可重集 S 的顺序为：$[a_1, a_2, a_3, b_1, b_2, a_4, a_5, b_3, b_4, b_5, b_6, a_6]$。

【样例 2】

见选手目录下的 ***sheep/sheep2.in*** 与 ***sheep/sheep2.ans***。

【样例 3】

见选手目录下的 ***sheep/sheep3.in*** 与 ***sheep/sheep3.ans***。

【数据范围与提示】

对于 20% 的数据，$n \leq 9$。
对于另外 50% 的数据，$n \leq 300$。
对于另外 20% 的数据，$a_i, b_i \leq 2$。
对于 100% 的数据，$3 \leq n \leq 5000$，$1 \leq a_i, b_i \leq n$，保证操作完之后可重集 S 为空。

彩树（colorful）

【题目描述】

小 T 有一张 n 个点 m 条边的简单无向图 G。

小 T 有 k 种颜色，他将第 i 个点染成了颜色 c_i。

小 T 想知道，有多少个无向图 G 的子图 H，使得在 H 中 k 种颜色的点都恰好有一个，且 H 是一棵树。你只需要输出答案对 998244353 取模的结果。

一个无向图 $G(V,E)$ 的子图 $H(V',E')$ 是指选出点集 V 的一个子集 V' 和边集 E 的子集 E'，满足 E' 中边的两个端点都在 V' 中。

【输入格式】

从文件 ***colorful.in*** 中读入数据。

第一行三个整数 n,m,k，分别表示无向图点数，边数，颜色数。

第二行 n 个整数 a_i，表示每个点的颜色。

接下来 m 行，第 i 行两个整数 u_i,v_i，表示一条无向边 (u_i,v_i)。保证图中没有自环或重边。

【输出格式】

输出到文件 ***colorful.out*** 中。

一行一个整数 ans，表示答案对 998244353 取模的结果。

【样例 1 输入】

```
6 8 4
4 2 1 2 3 4
1 2
2 3
3 4
5 3
6 3
1 6
1 5
5 4
```

【样例 1 输出】

```
11
```

【样例 2】

见选手目录下的 ***colorful/colorful2.in*** 与 ***colorful/colorful2.ans***。

【样例 3】

见选手目录下的 ***colorful/colorful3.in*** 与 ***colorful/colorful3.ans***。

【样例 4】

见选手目录下的 ***colorful/colorful4.in*** 与 ***colorful/colorful4.ans***。

【数据范围与提示】

对于 10% 的数据，$n \leq 9$。

对于 30% 的数据，$n \leq 27$。

对于另外 20% 的数据，保证 $k \leq 4$。

对于另外 10% 的数据，$m = n \cdot (n-1)/2$。

对于 100% 的数据，$2 \leq n \leq 200$，$1 \leq m \leq n \cdot (n-1)/2$，$2 \leq k \leq 12$，$1 \leq a_i \leq k$，$1 \leq u_i, v_i \leq n$，保证图中没有自环重边。

剪刀石头布（rps）

【题目描述】

小 T 有一个长为 n 的字符串 s，字符串的字符集为 $\{\mathsf{R},\mathsf{P},\mathsf{S}\}$。

小 T 发现，这个字符串的每两个相邻的字符都不同，也就是说对任意 $1 \le i \le n-1$，$s_i \neq s_{i+1}$。小 T 还发现，这个字符串中 $s_1 = s_n = \mathsf{R}$。

小 T 想改变一下这个字符串，具体来说，小 T 可以对这个字符串做以下两种操作：

1. 找出所有出现在该字符串中的连续子串 RS 和 SR，若不存在则无法操作，否则取其中出现在最左侧（下标最小）的那个子串，将其替换为 R。
2. 找出所有出现在该字符串中的连续子串 SP 和 PS，若不存在则无法操作，否则取其中出现在最左侧（下标最小）的那个子串，将其替换为 S。

若无法进行任何一种操作，则结束，否则必须选择一种可行的操作进行。结束后，小 T 把他进行的操作过程用一个序列 b 记录了下来，$b_i = 1$ 表示进行了第一种操作，$b_i = 2$ 表示进行了第二种操作。

……

小 T 凭借着残存的记忆，告诉了你字符串 s 中某些位置的字符，小 T 想知道，有多少个满足条件的字符串 s 和序列 b。你只需要输出答案对 998244353 取模的结果。

特别地，小 T 会告诉你 $s_1 = s_n = \mathsf{R}$。

【输入格式】

从文件 ***rps.in*** 中读入数据。

一行一个字符串 s'，若 $s'_i = ?$ 表示小 T 忘记了这一位上的字符，否则表示小 T 确定 $s_i = s'_i$。

【输出格式】

输出到文件 ***rps.out*** 中。

一行一个整数 ans，表示答案对 998244353 取模的结果。

【样例 1 输入】

```
R??RSR
```

【样例 1 输出】

```
4
```

【样例 1 解释】

可能的字符串 s 有 RPSRSR 和 RSPRSR，分别对应了 $2, 2$ 种操作序列。

例如，在 $s = \mathsf{RPSRSR}$ 时，操作序列 $[2, 1, 1]$ 代表了如下过程：RPSRSR → RSRSR → RRSR → RRR。

【样例 2】

见选手目录下的 ***rps/rps2.in*** 与 ***rps/rps2.ans***。

【样例 3】

见选手目录下的 ***rps/rps3.in*** 与 ***rps/rps3.ans***。

【数据范围与提示】

对于 20% 的数据，$n \leq 10$。
对于另外 20% 的数据，$s'_i \neq$?。
对于另外 30% 的数据，$n \leq 150$。
对于 100% 的数据，$3 \leq n \leq 200$，$s'_i \in \{\mathsf{R}, \mathsf{P}, \mathsf{S}, ?\}$，保证 $s'_1 = s'_n = \mathsf{R}$。
提示：要相信你算法的常数和现代计算机的速度。