NOIP模拟赛 day2

时间：2022 年 11 月 16 日 ??:?? ~ ??:??

题目名称	矩阵	介值	货币系统	排列
题目类型	传统型	传统型	传统型	传统型
输入文件名	`matrix.in`	`intermediate.in`	`coin.in`	`permutation.in`
输出文件名	`matrix.out`	`intermediate.out`	`coin.out`	`permutation.out`
每个测试点时限	1.5 秒	3.0 秒	1.0 秒	1.0 秒
内存限制	512 MB	512 MB	512 MB	1 GB
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	`matrix.cpp`	`intermediate.cpp`	`coin.cpp`	`permutation.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。

矩阵（matrix）

【题目描述】

小 T 有一个 $n \times m$ 的 01 矩阵 A。

如果矩阵 A 的每一个 2×2 的连续子矩阵中都恰好有一个 1，那么称矩阵 A 是好的。换句话说，对任意的 $1 \le i \le n-1, 1 \le j \le m-1$，都满足 $A_{i,j} + A_{i,j+1} + A_{i+1,j} + A_{i+1,j+1} = 1$。

现在已知这个矩阵中某些位置上的值为 0，请问如何在已知信息的基础上构造一个好的矩阵 A。

数据保证构造方案存在。

【输入格式】

从文件 ***matrix.in*** 中读入数据。

第一行两个整数 n, m。

接下来 n 行，第 i 行有一个长为 m 的字符串 s_i，s_i 的第 j 个字符若为 `0` 表示 $A_{i,j} = 0$，若为 `?` 表示 $A_{i,j}$ 未知。

【输出格式】

输出到文件 ***matrix.out*** 中。

共 n 行，每行一个长为 m 的字符串 t_i，若 $A_{i,j} = 0$ 则 t_i 的第 j 个字符为 `0`，若 $A_{i,j} = 1$ 则 t_i 的第 j 个字符为 `1`。若有多组可能的构造方案，你只需输出任意一组方案即可。

【样例 1 输入】

```
4 4
??0?
0???
?0?0
00?0
```

【样例 1 输出】

```
0101
0000
1010
0000
```

【样例 2 输入】

```
3 3
000
```

```
0??
0??
```

【样例 2 输出】

```
000
010
000
```

【样例 3 输入】

```
3 5
??00?
0?0?0
0?0?0
```

【样例 3 输出】

```
00000
01010
00000
```

【数据范围与提示】

对于 30% 的数据，$n, m \le 4$。

对于另外 10% 的数据，输入数据的所有字符串 s_i 中不包含 `0`。

对于另外 20% 的数据，输入数据的所有字符串 s_i 中一共包含至多 3 个 `0`。

对于 100% 的数据，$2 \le n, m \le 4000$，保证输入字符串 s_i 中仅包含字符 `0` 和 `?`，保证数据有解。

介值（intermediate）

【题目描述】

小 T 学不会微积分。

有 m 种颜色的球，每种颜色的球分别有 k 个。这些球组成了一个序列，记从左到右第 i 个颜色的球为 a_i。

定义一个由球组成的长为 L 的非空序列 b 是好的，当且仅当不存在 $1 \le i < j < k \le L$，使得 $b_i < b_j < b_k$ 或 $b_k > b_j > b_i$。

在序列 a 中，一共有 $2^{mk}-1$ 个非空子序列，你需要求出这些子序列中有多少是好的。你只需要输出答案对 998244353 取模的结果。

【输入格式】

从文件 *intermediate.in* 中读入数据。

第一行两个整数 m,k，表示球的种类数和每种球的个数。

第二行 mk 个整数 a_i，表示第 i 个球的颜色。

【输出格式】

输出到文件 *intermediate.out* 中。

输出一行一个整数，表示答案对 998244353 取模的结果。

【样例 1 输入】

```
3 2
2 1 3 2 1 3
```

【样例 1 输出】

```
49
```

【样例 1 解释】

可以用容斥原理来计算这个样例。

不好的子序列一定要么包含第 $3,4,5$ 个球，要么包含第 $2,4,6$ 个球。

- 没有任何限制的方案数为 $2^6-1=63$。
- 一定包含第 $3,4,5$ 个球的方案数为 $2^3=8$。
- 一定包含第 $2,4,6$ 个球的方案数为 $2^3=8$。
- 一定同时包含第 $3,4,5$ 个球和第 $2,4,6$ 个球的方案数为 $2^1=2$。

综上所述，好的子序列的方案数为 $63-8-8+2=49$。

【样例 2】

见选手目录下的 *intermediate/intermediate2.in* 与 *intermediate/intermediate2.ans*。

【样例 3】

见选手目录下的 *intermediate/intermediate3.in* 与 *intermediate/intermediate3.ans*。

【数据范围与提示】

对于 20% 的数据，$m \times k \leq 20$。
对于另外 10% 的数据，$m, k \leq 50$。
对于另外 20% 的数据，$m \leq 20$。
对于另外 20% 的数据，$k \leq 20$。
对于 100% 的数据，$1 \leq m, k \leq 300$，$1 \leq a_i \leq m$，保证每种颜色的球恰好有 k 个。

货币系统（coin）

【题目描述】

小 T 学不会对大数取模。

小 T 想通过一些特定的硬币面额，来使每个正整数的钱数都能用一些硬币表示。为了方便表示，他认为每个面额都应该是上一个面额的倍数，且不能相差过大。

具体来说，给定参数 m，一个货币系统是一个非空整数序列 a，记 a 的长度为 L，则 a 满足 $a_1 = 1$ 且对任意 $1 \le i \le L-1$，有 $a_{i+1} = k_i a_i$，其中 k_i 是 $[2, m]$ 中的正整数。

小 T 比较关心 n 元钱在这些货币系统里都是如何表示方案的。具体来说，货币系统 a 里的一个 n 元钱的表示方案是一个整数序列 c，记它的长度为 K，则 c 满足以下条件：

- c 中的每个数都在 a 中出现过，即对每个 $1 \le i \le K$，都存在 $1 \le j \le L$，使得 $c_i = a_j$。
- a_m 在 c 中出现过。
- $\sum_{i=1}^{k} c_i = n$。
- 对任意的 $2 \le i \le K$，都有 $c_i | \sum_{j=1}^{i-1} c_j$。

小 T 没有想好表示方案，也没有想好货币系统，所以他想问，在所有货币系统 a 中，表示方案的数量的总和是多少。

两个表示方案不同当且仅当序列 a 或 c 不完全相同。

答案可能很大，但小 T 认为对大数取模会让他手玩的时候十分痛苦，所以你只需要输出答案对 19 取模的结果。

【输入格式】

从文件 ***coin.in*** 中读入数据。

一行两个正整数 n, m，分别表示小 T 关心的钱数和货币系统的参数。

【输出格式】

输出到文件 ***coin.out*** 中。

一行一个整数 ans，表示答案对 19 取模的结果。

【样例 1 输入】

```
5 3
```

【样例 1 输出】

```
6
```

【样例 1 解释】

若货币系统为 $[1]$，则表示方案只有 $[1, 1, 1, 1, 1]$。

若货币系统为 $[1,2]$，则表示方案有 $[2,2,1],[2,1,1,1],[1,1,2,1]$。

若货币系统为 $[1,3]$，则表示方案只有 $[3,1,1]$。

若货币系统为 $[1,2,4]$，则表示方案只有 $[4,1]$。

注意：货币系统为 $[1,2]$ 时，表示方案 $[1,1,1,1,1]$ 是不合法的，因为表示方案必须满足 a_L 在 c 中出现过。

【样例 2】

见选手目录下的 ***coin/coin2.in*** 与 ***coin/coin2.ans***。

【样例 3】

见选手目录下的 ***coin/coin3.in*** 与 ***coin/coin3.ans***。

【样例 4】

见选手目录下的 ***coin/coin4.in*** 与 ***coin/coin4.ans***。

【样例 5】

见选手目录下的 ***coin/coin5.in*** 与 ***coin/coin5.ans***。

【数据范围与提示】

对于 20% 的数据，$n \le 500$。

对于 40% 的数据，$n \le 10^5$。

对于另外 10% 的数据，$m = 2$。

对于另外 10% 的数据，$m = 3$。

对于另外 20% 的数据，$n \le 10^9$。

对于 100% 的数据，$1 \le n \le 10^{10}$，$2 \le m \le 50$。

排列（permutation）

【题目描述】

小 T 学会了对大数取模。

小 T 有一个长为 n 的排列 p。小 T 知道可以通过一些相邻两项交换（邻项交换）的操作把排列 p 变成升序，也就是 $1,2,\ldots,n$。但小 T 发现排列 p 不仅能通过邻项交换变为升序，而且这些操作在每对相邻的位置上都恰好操作了一次。

这让小 T 很感兴趣，所以他把这个交换次序记录了下来。具体来说，小 T 记录了一个长为 $n-1$ 的排列 b。这个排列的第 i 项 b_i 代表了，在第 i 次操作中，小 T 交换了排列 p 中第 b_i 个数和第 b_i+1 个数，换句话说小 T 交换了 p_{b_i} 和 p_{b_i+1}。

在小 T 为自己的发现而高兴，正在打开 qq 和别人分享的时候，他的电脑突然死机了。小 T 在尝试了一番后只能被迫重启，但重启之后排列 p 和排列 b 已经消失不见了。

小 T 凭借着残存的记忆，告诉了你排列 p 中某些元素的值，你能帮他求出，有多少满足条件的排列 p 和排列 b 吗？

由于小 T 已经学会了对大数取模，你只需要告诉他答案对 998244353 取模的结果。

数据保证至少有一组可能的 p 和 b。

【输入格式】

从文件 ***permutation.in*** 中读入数据。

第一行一个整数 n，表示排列 p 的长度。

第二行 n 个整数 p'_i，若 $p'_i=0$ 表示 p_i 不确定，若 $p'_i\neq 0$ 表示确定 $p_i=p'_i$。

【输出格式】

输出到文件 ***permutation.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】

```
5
2 0 0 5 0
```

【样例 1 输出】

```
6
```

【样例 1 解释】

可能的排列 p 有 $[2,3,4,5,1]$ 和 $[2,4,1,5,3]$，它们分别对应的排列 b 的方案数分别为 $1,5$。

例如，$[2,3,4,5,1]$ 可能对应 $[4,3,2,1]$，因为按照这个顺序操作最后可以得到升序排列 $[1,2,3,4,5]$。

【样例 2】

见选手目录下的 *permutation/permutation2.in* 与 *permutation/permutation2.ans*。

【样例 3】

见选手目录下的 *permutation/permutation3.in* 与 *permutation/permutation3.ans*。

【样例 4】

见选手目录下的 *permutation/permutation4.in* 与 *permutation/permutation4.ans*。

【数据范围与提示】

对于 10% 的数据，$n \le 10$。
对于 20% 的数据，$n \le 20$。
对于 30% 的数据，$n \le 50$。
对于 50% 的数据，$n \le 300$。
对于另外 10% 的数据，保证 $p'_i \ne 0$。
对于另外 10% 的数据，保证 $p'_i = 0$。
对于 100% 的数据，$2 \le n \le 5000$，$0 \le p'_i \le n$，保证至少有一组可能的 p 和 b。