IOI2025 中国国家集训队集中培训

第一试

时间：2024 年 12 月 3 日 08:30 ~ 13:30

题目名称	乘积的期望	最大匹配 2	平原
题目类型	传统型	传统型	传统型
输入	标准输入	标准输入	标准输入
输出	标准输出	标准输出	标准输出
每个测试点时限	2.0 秒	3.0 秒	1.0 秒
内存限制	1024 MiB	1024 MiB	512 MiB
子任务数目	7	6	7
测试点是否等分	否	否	否

乘积的期望（expectation）

【题目描述】

有一个长度为 n 的序列 $a_1, a_2, \cdots, a_n$。初始序列的所有元素均为 0。再给定正整数 m、c 和 $(n-m+1)$ 个正整数 $b_1, b_2, \cdots, b_{n-m+1}$。

对序列 $a_1, a_2, \cdots, a_n$ 进行 c 次操作，每次操作为：

- 随机选择整数 $1 \le x \le n-m+1$，其中选到 $y(1 \le y \le n-m+1)$ 的概率为 $\frac{b_y}{\sum_{i=1}^{n-m+1} b_i}$。将 $a_x, a_{x+1}, \cdots, a_{x+m-1}$ 增加 1。

c 次操作中对 x 的随机是独立的。

求操作完成后序列中所有元素的乘积的期望。为了避免浮点数输出，你需要将答案对 998244353 取模。

【输入格式】

从标准输入读入数据。

第一行三个整数 n, m, c，分别表示序列长度、操作区间长度和操作次数。

第二行 $n-m+1$ 个整数 $b_1, \cdots, b_{n-m+1}$，描述随机的权重。

【输出格式】

输出到标准输出。

输出一行一个整数，表示 c 次操作后序列中所有数的乘积的期望。

【样例 1 输入】

```
3 2 2
1 1
```

【样例 1 输出】

```
1
```

【样例 1 解释】

当两次操作选择的 x 不同时，最终序列为 `1 2 1`，序列元素乘积为 2；否则序列为 `2 2 0` 或 `0 2 2`，序列元素乘积均为 0。两次操作选择的 x 不同的概率为 $\frac{1}{2}$，因此输出 $2 \times \frac{1}{2} = 1$。

【样例 2 输入】

```
10 3 10
1 2 3 4 5 6 7 8
```

【样例 2 输出】

```
721023399
```

【样例 3 输入】

```
20 12 98765
9 8 7 6 5 4 3 2 1
```

【样例 3 输出】

```
560770686
```

【子任务】

对于所有测试数据，$2 \leq m \leq n \leq 50$，$1 \leq c < 998244353$，对于 $1 \leq i \leq n-m+1$，$1 \leq b_i \leq 10^5$。

Subtask 1(10%) : $m \leq 8$。

Subtask 2(20%) : $m \leq 16$。

Subtask 3(15%) : $n \leq 20, c \leq n$。

Subtask 4(15%) : $n \leq 30, c \leq n$。

Subtask 5(15%) : $n \leq 40, c \leq n$。

Subtask 6(15%) : $c \leq n$。

Subtask 7(10%) : 无特殊限制。

最大匹配 2（matching）

【题目描述】

给定长度为 n 的整数序列 $a_1, a_2, \cdots, a_n$ 和长度为 n 的 01 序列 $b_1, b_2, \cdots, b_n$。

对于 $1 \le i < j \le n$，称二元组 (i, j) **构成匹配**当且仅当 $b_i = 0$ 且 $b_j = 1$。

定义**极大匹配方案** $S_{\max}$ 为满足以下所有条件的二元组集合：

- 对于任意 $(u, v) \in S_{\max}$，$1 \le u < v \le n$ 且 (u, v) 构成匹配；
- 每一个整数 $1 \le i \le n$ 在 $S_{\max}$ 中出现至多一次；
- 在满足以上条件的前提下，满足 $a_u = a_v$ 的二元组 (u, v) 数量最多，即 $\sum_{(u,v)\in S_{\max}} [a_u = a_v]$ 最大；
- 在满足以上条件的前提下，$|S_{\max}|$ 最大。

给定 m 次修改，每次修改给出 x, p, q，表示将 (a_x, b_x) 修改为 (p, q)。

你需要对于每个 $1 \le i \le m + 1$ 求出：按输入顺序依次进行前 $(i - 1)$ 次操作后，极大匹配方案 $S_{\max}$ 的大小 $|S_{\max}|$。

【输入格式】

从标准输入读入数据。

输入的第一行两个整数 n, m，表示序列长度和操作次数。

第二行 n 个整数 $a_1, a_2, \cdots, a_n$ 描述序列 a。

第三行 n 个整数 $b_1, b_2, \cdots, b_n$ 描述序列 b。

接下来 m 行每行三个整数 x, p, q，描述一次修改。

【输出格式】

输出到标准输出。

输出 $m + 1$ 行，每行一个整数，第 i 行表示按输入顺序依次进行前 $(i - 1)$ 次操作后 $|S_{\max}|$ 的值。

【样例 1 输入】

```
5 5
1 2 1 1 2
0 0 0 0 0
2 2 1
4 2 0
4 2 1
2 2 0
```

```
1 1 1
```

【样例 1 输出】

```
0
1
1
2
1
1
```

【样例 1 解释】

- 初始情况，由于所有的 b_i 都等于 0，因此没有二元组构成匹配，极大匹配方案的大小为 0，故第一行输出 `0`。
- 进行第一次修改后，$b_2 = 1$，极大匹配方案为 $\{(1, 2)\}$，故第二行输出 `1`。
- 进行前三次修改后，序列 a 为 `1 2 1 2 2`，序列 b 为 `0 1 0 1 0`。极大匹配方案为 $\{(1, 2), (3, 4)\}$，故第四行输出 `2`。注意此时 $(4, 5)$ 有 $b_4 = 1$，$b_5 = 0$，并不构成匹配。

【样例 2 输入】

```
10 10
2 1 2 2 2 2 1 2 2 2
1 1 0 0 0 0 1 0 0 0
3 2 0
5 1 0
9 1 1
4 2 1
8 1 1
2 1 0
1 2 1
8 2 0
1 1 1
9 1 0
```

【样例 2 输出】

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

【子任务】

对于所有测试数据，

- $1 \le n \le 2 \times 10^5$，$0 \le m \le 2 \times 10^5$；
- 对于 $1 \le i \le n$，$1 \le a_i \le n$，$0 \le b_i \le 1$；
- 每次修改中有 $1 \le x \le n$，$1 \le p \le n$，$0 \le q \le 1$。

Subtask 1(10%) : $n \le 100, m = 0$。

Subtask 2(15%) : $n \le 2 \times 10^3, m = 0$。

Subtask 3(20%) : $m = 0$。

Subtask 4(15%) : $a_i, p \le 2$。

Subtask 5(20%) : $n, m \le 10^5$。

Subtask 6(20%) : 无特殊限制。

平原（plain）

【题目背景】

“只要你心里的念是真的，只要你心里的念是诚的，高山大海都会给你让路。”——《平原上的摩西》

【题目描述】

这是一道交互题。

在二维平面上有一条**未知**的、不与 y 轴平行的直线 L。设 $L(x_0)$ 表示直线 L 与 $x = x_0$ 的交点纵坐标。

给定值域 V，L 额外满足以下性质：

- 对于任意整数 $0 \le x, y \le V$，$L(x) \ne y$；
- $0 \le L(0), L(V) \le V$。

此时 L 将 $S = \{0, 1, 2, \cdots, V\} \times \{0, 1, 2, \cdots, V\}$ 分割成两个点集 $S_L = \{(x, y) \in S \mid L(x) > y\}$ 和 $\overline{S_L} = \{(x, y) \in S \mid L(x) < y\}$，且它们都不是空集。

你可以向交互库询问至多 limit 次询问。每次询问给出整数 $0 \le x, y \le V$，你可以得到 (x, y) 属于 S_L 还是 $\overline{S_L}$。

你需要找到直线 L'，其不经过 S 中任何一个点，且 L' 和 L 将 S 划分为同样的两个点集。形式化地，L' 需要满足 $\forall (x, y) \in S_L, L'(x) > y$ 且 $\forall (x, y) \in \overline{S_L}, L'(x) < y$。

【实现细节】

请确保你的程序开头有 `#include "plain.h"`。

你不需要也不应该实现主函数。你需要实现以下函数：

- `std::tuple<long long, int, long long, int> Find(int task_id, int V, int limit);`
  - 其中 `task_id` 表示子任务编号，`V` 表示值域，`limit` 表示最大询问次数。
  - 你需要返回四元组 `(k_u, k_v, b_u, b_v)` 表示你给出的直线 L' 的解析式为 $y = \frac{k_u}{k_v} x + \frac{b_u}{b_v}$。
  - 你需要保证 k_u 与 b_u 在 `long long` 范围内，k_v 与 b_v 在 `int` 范围内，且 $k_v, b_v > 0$。你不需要保证给出的分数是既约分数。可以证明在本题的数据范围下，总是存在这样的 (k_u, k_v, b_u, b_v) 满足条件。
  - **在最终测试时，交互库将会调用 $T \le 10^4$ 次 `Find` 函数。**

你可以使用 `std::make_tuple(a,b,c,d)` 来将 `a,b,c,d` 打包成一个 `tuple`。

你可以调用如下函数进行一次询问：

- `bool query(int x,int y);`

- 你需要保证 $0 \le x, y \le V$，在单组测试数据内该函数的调用次数不超过 limit。
- 当 $(x, y) \in S_L$ 时，函数返回 `true`，否则返回 `false`。

保证在满足题目条件和数据范围的情况下，最终测试时交互库的运行时间不会超过 200 ms，运行空间不会超过 64 MiB。

交互库不是自适应的，即 L 是固定的，不会随着交互过程改变。

【测试程序方式】

试题目录下的 `grader.cpp` 是我们提供的交互库参考实现。最终测试的交互库与样例交互库有一定不同，故你的实现不应该依赖样例交互库实现。

你需要在本题目录下使用如下命令编译得到可执行程序：

```
g++ plain.cpp grader.cpp -o plain -O2 --std=c++14 -lm
```

对于编译得到的可执行程序：

- 可执行文件将从标准输入读入以下格式的数据：
  - 第一行三个整数 $\text{task_id}, T, \text{limit}$，分别表示子任务编号，测试数据组数和每组测试数据的询问次数上限。接下来依次输入每组测试数据。
  - 对于每组测试数据输入一行五个整数 V, k_u, k_v, b_u, b_v，其中 V 表示值域，k_u, k_v, b_u, b_v 表示直线 L 的解析式为 $y = \frac{k_u}{k_v}x + \frac{b_u}{b_v}$。
  - 你需要保证 b_u 在 `long long` 范围内，k_u, k_v, b_v 在 `int` 范围内，且 $k_v, b_v > 0$。对于所有测试数据，保证存在这样的 (k_u, k_v, b_u, b_v) 满足题目条件。
- 读入完成之后，交互库将会调用 T 次 `Find` 函数。
- 若每一组测试数据中你都在给定的询问次数内求出了正确的直线，交互库将在标准输出流输出两行，第一行一个字符串 `Correct.`，第二行一个整数，表示 T 组测试数据中 `query` 调用次数的最大值。否则交互库会输出对应错误信息，并立即停止程序。

【样例 1 输入】

```
0 3 20
4 5 11 11 6
4 -11 15 17 5
4 2 15 4 5
```

【样例 1 输出】

```
Correct.
8
```

【样例 1 解释】

task_id = 0 表示该组数据为样例。
这是第一组数据对应的函数图像。

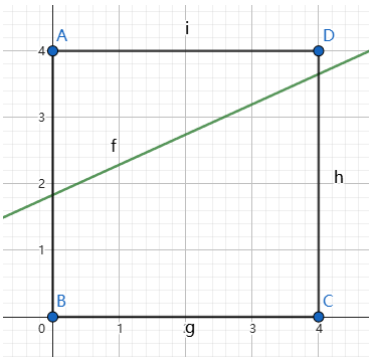


图 1: 第一组数据对应的函数图像

这是第二组数据对应的函数图像。

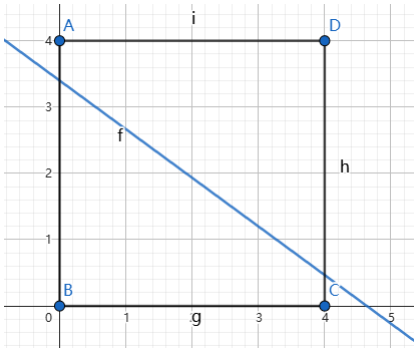


图 2: 第二组数据对应的函数图像

这是第三组数据对应的函数图像。

【子任务】

对于所有测试数据，$2 \le V \le 10^9$，$1 \le T \le 10^4$，$100 \le \text{limit} \le 3,666$。

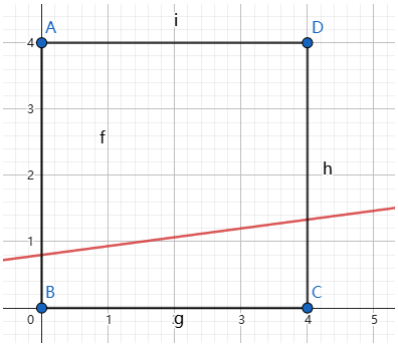


图 3: 第三组数据对应的函数图像

子任务编号	分值	V	T	limit	特殊性质
1	10	$\leq 10^2$	≤ 10	$= 110$	无
2	20	$\leq 10^9$	$\leq 10^3$	$= 180$	A
3	10	$\leq 10^5$	≤ 10	$= 1,111$	无
4	15	$\leq 10^9$	≤ 500	$= 3,666$	
5	20	$\leq 10^5$	≤ 10	$= 10^2$	
6	10	$\leq 10^9$		$= 180$	
7	15		$\leq 10^4$		

特殊性质 A：保证直线 L 是由 $y = \frac{a}{b}x$ 向上平移至多 $\frac{1}{2b}$ 得到的，其中 $0 < b \leq V$。

【评分方式】

本题首先会受到和传统题相同的限制。例如编译错误会导致整道题目得 0 分，运行时错误、超过时间限制、超过空间限制都会导致相应测试点得 0 分。选手只能在程序中访问自己定义的和交互库给出的变量或数据，及其相应的内存空间。尝试访问其他位置空间将可能导致编译错误或运行错误。

当你在每次 `Find` 调用中，程序调用的 `query` 函数次数不超过 limit，且返回的 L' 均满足题目描述中的条件，即通过该测试点，否则该测试点不通过。只有你通过一个子任务的所有测试点时，才能获得该子任务的所有分数。

选手不应通过非法方式获取交互库的内部信息，如试图与标准输入、输出流进行交互。此类行为将被视为作弊。

IOI2025 中国国家集训队集中培训

第二试

时间：2024 年 12 月 4 日 08:30 ~ 13:30

题目名称	阿尔塔尔 2	颠倒歌	路南柯
题目类型	交互型	传统型	传统型
输入	标准输入	标准输入	标准输入
输出	标准输出	标准输出	标准输出
每个测试点时限	1.0 秒	3.0 秒	1.0 秒
内存限制	512 MiB	512 MiB	512 MiB
子任务数目	3	11	2
测试点是否等分	否	否	否

阿尔塔尔 2（altar）

这是一道交互题。

【题目背景】

一年过去了，阿尔塔尔站在曾被他摧毁的祭坛面前。祭坛上的宝石早已失去了原有的光泽。

“解铃还须系铃人，是这样吗?”他喃喃自语。

【题目描述】

祭坛呈正 n 边形，每个顶点上有一个未激活的宝石。我们用 1 至 n 的整数给每种宝石编号。

宝石之间存在着能量流动关系，这使得激活一些宝石之后可以更容易地激活其他宝石。对于每一对宝石 (i,j)（$i \neq j$），要么从 i 到 j 存在流动，要么从 j 到 i 存在流动，即当我们把宝石看作点、从 p 到 q 的流动看作一条从 p 到 q 的有向边，得到的图是一个 n 个点的**竞赛图**。阿尔塔尔并不知道宝石之间的能量流动关系如何。

为了激活祭坛，阿尔塔尔需要激活所有的宝石。为此，阿尔塔尔需要首先选择 n 个宝石中的一个将其初始激活，然后进行若干次**能量传递**。每一次**能量传递**的效果为：若这次**能量传递**前宝石 y 未被激活，且存在一个激活的宝石 x 到 y 有流动，那么**能量传递**后 y 就会被激活。在**能量传递**前就被激活的宝石在**能量传递**之后依然是被激活的。

由于**能量传递**需要耗费大量体力，阿尔塔尔希望找到一个宝石，将其初始激活之后可以使用**最少次数的能量传递**激活所有宝石。为此，阿尔塔尔可以进行若干次**能量感知**：给出 $1 \leq i,j \leq n$ 且 $i \neq j$，一次**能量感知**可以确定宝石 i 和 j 之间的能量流动关系。

你需要帮助阿尔塔尔使用尽可能少的**能量感知**操作，确定初始激活的宝石编号。可以证明总是存在一个宝石，将其初始激活后使用有限次能量传递即可激活所有宝石。

【实现细节】

请确保你的程序开头有 `#include "altar.h"`

你不需要也不应该实现主函数。你需要实现以下函数：

```
int altar(int n);
```

- 其中 n 表示祭坛上的宝石数量。
- 你需要返回一个整数 x，表示阿尔塔尔需要初始激活宝石 x。你需要保证 $1 \leq x \leq n$，且在所有宝石中初始激活 x 可以让阿尔塔尔使用最少次数的**能量传递**激活所有宝石。
- **在最终测试时，交互库会在一次运行中调用 $T = 300$ 次 `altar` 函数。**

你可以调用以下函数进行**能量感知**：

```
bool sense(int i, int j);
```

- 你需要保证 $1 \le i, j \le n$ 且 $i \neq j$。
- 若 i 与 j 的能量流动为 i 向 j，其返回 `true`，否则返回 `false`。
- 你需要保证在一次 `altar` 的调用中 `sense` 的调用次数不超过 4.5×10^4。

在满足题目条件和数据范围的情况下，最终测试时交互库的运行时间不会超过 150 ms，运行空间不会超过 256 MiB。

交互库不是自适应的，即能量流动关系是固定的，不会随着交互过程改变。

【测试程序方式】

试题目录下的 `grader.cpp` 是我们提供的交互库参考实现。最终测试的交互库与样例交互库有一定不同，故你的实现不应该依赖样例交互库实现。

你需要在本题目录下使用如下命令编译得到可执行程序：

```
g++ grader.cpp sample.cpp -o sample -O2 --std=c++14 -lm
```

对于编译得到的可执行程序：

- 可执行文件将从标准输入读入以下格式的数据：
  - 第一行一个整数 n 表示祭坛上的宝石数，你需要保证 $3 \le n \le 300$。
  - 接下来 n 行，第 i 行一个长度为 n 的 `01` 字符串，其中第 $j (j \neq i)$ 个字符为 `1` 表示能量流动从 i 到 j，否则表示能量流动从 j 到 i。你需要保证第 i 行第 j 个字符和第 j 行第 i 个字符间恰好有一个为 `1` 另一个为 `0`，**样例交互库并不会判断输入是否满足条件**。
- 读入完成之后，交互库将调用恰好一次函数 `altar`。
- 在 `altar` 函数退出后，如果你给出了正确的宝石编号，交互库将在标准输出流输出 `Correct. X`，其中 X 表示 `sense` 的调用次数，并在标准错误流输出你返回的宝石编号以及对应的**能量传递**次数；否则交互库将在标准输出流输出 `Wrong Answer` 并在标准错误流输出对应错误信息。

【样例 1 输入】

```
4
0011
1010
0000
0110
```

【样例 1 输出】

```
Correct. 1
You report 2, with number of energy propagations to be 2
```

【样例 1 解释】

该样例输出为下发的样例程序在该组样例下的输出。该样例中，`altar` 返回 1、2、4 均满足条件。

【子任务】

对于所有测试数据，单个测试点中 `altar` 函数的调用次数 $T = 300$，每一次调用都有 $3 \leq n \leq 300$。

子任务 1（10 分，共 1 个测试点）：$n = 300$，两个宝石之间的能量流动方向在两种可能间独立均匀随机。

子任务 2（10 分，共 2 个测试点）：存在一个宝石流向所有其他宝石。

子任务 3（80 分，共 7 个测试点）：没有特殊限制，依赖子任务 1、2。

【评分方式】

本题首先会受到和传统题相同的限制，例如编译错误会导致整道题目得 0 分，运行时错误、超过时间限制、超过空间限制都会导致相应测试点得 0 分。选手只能在程序中访问自己定义的和交互库给出的变量或数据，及其相应的内存空间。尝试访问其他位置空间将可能导致编译错误或运行错误。

对于每个测试点，如果你的程序出现了非法的函数调用，或没有在所有测试数据中都返回正确的宝石，你将获得该测试点 0% 的分数。否则你会按照在 T 组 `altar` 调用中的 `sense` 调用次数的**平均值** X，依据如下公式计算该测试点的得分占比：

- 若 $45000 \geq X \geq 10^4$，你可以获得 $\left(1 + 29 \times \frac{45000 - X}{35000}\right)\%$ 的分数；
- 若 $10^4 > X \geq 2100$，你可以获得 $\left(30 + 30 \times \frac{10^4 - X}{7900}\right)\%$ 的分数；
- 若 $2100 > X \geq 700$，你可以获得 $\left(60 + 20 \times \left(\frac{2100}{X} - 1\right)\right)\%$ 的分数；
- 若 $X \leq 700$，你可以获得该测试点 100% 的分数；

每个子任务的分数为该子任务中所有测试点的得分占比的最小值与该子任务满分的乘积。

选手不应通过非法方式获取交互库的内部信息，如试图与标准输入、输出流进行交互。此类行为将被视为作弊。

颠倒歌（diandao）

【题目描述】

对于树 $T(V,E)$ 和 $S \subseteq V$，记 $f(S,T)$ 表示 T 的对 S 的导出子图（即仅保留 S 中的点和两端都在 S 中的边得到的图）中度数小于等于 1 的点的数量。

对于两棵树 $T_1(V_1,E_1), T_2(V_2,E_2)$，若 $V_1 = V_2$，我们称 $T_1 \preccurlyeq T_2$ 当且仅当对于任意 $S_2 \subseteq V_2$，存在 S_1 满足 $S_2 \subseteq S_1 \subseteq V_1$ 且 $f(S_1,T_1) \leq f(S_2,T_2)$。

称 T_1, T_2 等价当 $T_1 \preccurlyeq T_2$ 且 $T_2 \preccurlyeq T_1$，记作 $T_1 \sim T_2$。该等价关系将 n 个点的有标号树划分成若干等价类。

问：

1. 给定 k 棵 n 个点的树 $T_1, T_2 \ldots, T_k$，求满足 $T \preccurlyeq T_i, \forall 1 \leq i \leq k$ 的**有标号树 T 构成的等价类**数量。
2. 给定 k 棵 n 个点的树 $T_1, T_2 \ldots, T_k$，求满足 $T_i \preccurlyeq T, \forall 1 \leq i \leq k$ 的**有标号树 T** 数量。

注意两问的计数对象不同。两问答案均对 $998,244,353$ 取模。

保证答案取模后非 0。

【输入格式】

从标准输入读入数据。

输入第一行一个整数 p，其中 $p \in \{0,1\}$，$p = 0$ 表示询问第一问，否则表示询问第二问。

接下来一行两个正整数 k, n，分别表示输入树的数量以及点数。

接下来依次输入 k 棵树，对于每棵树输入 $n-1$ 行每行两个正整数 u, v 描述树中的一条边。

【输出格式】

输出到标准输出。

输出一行一个整数表示答案对 $998,244,353$ 取模后的结果。

【样例 1 输入】

```
0
1 4
1 2
1 3
1 4
```

【样例 1 输出】

```
2
```

【样例 1 解释】

可以证明 4 个点的有标号树被划分成了 5 个等价类，所有的链在同一个等价类，而其它分别以每个点为菊花中心对应一个等价类。

可以验证链对应的等价类和该树本身所在的等价类均满足要求，而其他等价类不满足要求。

【样例 2 输入】

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

【样例 2 输出】

```
16
```

【样例 2 解释】

可以验证所有 4 个点的有标号树共 16 个均满足要求。

【样例 3】

见题目目录下的 ***3.in*** 与 ***3.ans***。

【样例 4】

见题目目录下的 ***4.in*** 与 ***4.ans***。

【样例 5】

见题目目录下的 ***5.in*** 与 ***5.ans***。

【样例 6】

见题目目录下的 ***6.in*** 与 ***6.ans***。

【样例 7】

见题目目录下的 ***7.in*** 与 ***7.ans***。

【样例 8】

见题目目录下的 ***8.in*** 与 ***8.ans***。

【样例 9】

见题目目录下的 ***9.in*** 与 ***9.ans***。

【样例 10】

见题目目录下的 ***10.in*** 与 ***10.ans***。

【子任务】

对于所有数据，保证 $p \in \{0,1\}$，$3 \le n \le 5000$，$1 \le k \le 1000$，$1 \le u, v \le n$，答案取模后不等于 0。

子任务编号	p	n	k	树形态	分数
1	$\in \{0,1\}$	≤ 8	≤ 4	无特殊形态	10
2	$=0$	≤ 200	≤ 1	菊花	4
3				无特殊形态	8
4			≤ 2		9
5			≤ 40		10
6		$\le 5,000$	$\le 10^3$		14
7	$=1$	≤ 200	≤ 1	链	7
8				无特殊形态	7
9			≤ 2		10
10			≤ 40		10
11		$\le 5,000$	$\le 10^3$		11

“树形态”中，“菊花”指存在一个向所有点有直接连边的点，“链”指所有点度数不超过 2。

路南柯（nanke）

【题目描述】

称一个 $1 \sim n$ 的排列 $\{p\} = \{p_1, p_2, \cdots, p_n\}$ 是一棵 n 个点、点编号为 1 至 n 的树 T 的**拓扑序列**，当且仅当对于任意 $1 \leq i < n$，恰好存在唯一的 $j > i$ 满足 p_i 与 p_j 之间有连边。

给定树 T，你需要给出尽可能少的该树的拓扑序列 $\{p_1\}, \{p_2\}, \cdots, \{p_k\}$，使得有且仅有树 T 满足 $\{p_1\}, \{p_2\}, \cdots, \{p_k\}$ 均为该树的合法拓扑序列。

【输入格式】

从标准输入读入数据。

本题有多组测试数据。输入第一行一个正整数 T，表示测试数据组数，接下来依次输入每组测试数据。

对于每组数据，输入第一行一个正整数 n，表示给定树的大小。接下来 $n-1$ 行，每行两个正整数 u, v 描述树中存在的一条边。

【输出格式】

输出到标准输出。

对于每组数据，输出第一行一个正整数 k 表示你给出的拓扑序列数量，接下来 k 行，每行输出一个 $1 \sim n$ 的排列，描述你给出的拓扑序列。你需要保证 $1 \leq k \leq n$，且这 k 个拓扑序列均为对应输入的合法拓扑序列，且只有一棵树满足这些拓扑序列都是其合法拓扑序列。

【样例 1 输入】

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

【样例 1 输出】

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

【子任务】

对于所有测试数据，$1 \leq T \leq 100$，$3 \leq n \leq 100$，$1 \leq u, v \leq n$。
本题共有两个测试点。

测试点编号	分值	T	n
1	20	$= 10$	$= 10$
2	80	$= 10^2$	$= 10^2$

特别地，所有测试点中每组数据均为从所有 n 个点的有标号树中等概率随机选择生成得到的。

【评分方式】

对于一个测试点内部的某组数据：

1. 你需要保证 k 以及你输出的序列中每个数都为 $[1, n]$ 范围内的正整数，否则**整个测试点**将会获得 0 分。
2. 若你给出的序列中存在一个不是输入给定树的拓扑序列，那么你**这组数据**的得分比例将会是 0。
3. 若存在多棵树满足其这些序列都是其合法的拓扑序列，那么你**这组数据**的得分比例将会是 0。
4. 否则如果最优解为 K，而你构造的序列数量为 k，那么你**这组数据**的得分比例将为 0.97^{k-K}。

一个测试点的得分比例将会是该测试点内所有数据的得分比例的**平均值**，一个测试点的实际得分值将会是其总分乘以得分比例。

IOI2025 中国国家集训队集中培训

第三试

时间：2024 年 12 月 5 日 08:30 ~ 13:30

题目名称	比赛	绝顶之战	前往何方
题目类型	传统型	传统型	交互型
输入	标准输入	标准输入	标准输入
输出	标准输出	标准输出	标准输出
每个测试点时限	2.0 秒	1.0 秒	3.0 秒
内存限制	512 MiB	512 MiB	512 MiB
子任务数目	6	10	3
测试点是否等分	否	否	否

比赛（match）

【题目描述】

有 n 名选手，编号分别为 $1, 2, \cdots, n$。编号为 i 的选手的实力值为 i。所有选手分为红、蓝两队，其中编号为 i 的选手所在的队伍用字符 s_i 描述。s_i 为 `R` 表示他在红队，s_i 为 `B` 表示他在蓝队。

现在将所有选手排成一个环，每一对相邻且不属于同一队的选手会进行一场比赛，实力值较大的选手获胜，他所在的队伍的得分增加一。

然而蓝队的选手勾结了裁判，如果一场比赛中**红队选手**获胜，且他在蓝队选手的**顺时针方向**，则这场比赛**不计入得分**。

现在你想知道，对于每个 $k = -n, \cdots, -1, 0, 1, \cdots, n$，有多少种将选手排列的方法，使得红队得分恰好比蓝队得分大 k。两种方案不同，当且仅当存在某个选手，使得他顺时针方向的下一个选手在两个方案中不同。

由于答案很大，你只需要输出答案对 $998,244,353$ 取模后的值。

【输入格式】

从标准输入读入数据。

第一行包含一个整数 n，表示选手个数。

第二行包含一个长度为 n 的字符串 $s_1 s_2 \cdots s_n$，表示每个选手所属的队伍。

【输出格式】

输出到标准输出。

输出一行 $2n + 1$ 个整数，分别表示 $k = -n, \cdots, 0, \cdots, n$ 的方案数对 $998,244,353$ 取模后的值。

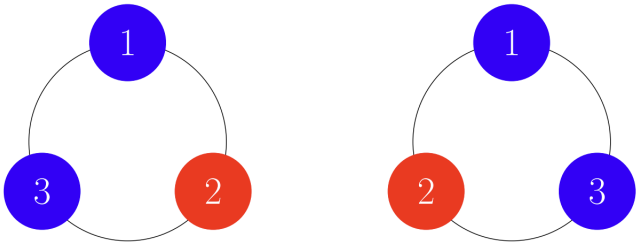
【样例 1 输入】

```
3
BRB
```

【样例 1 输出】

```
0 0 1 1 0 0 0
```

【样例 1 解释】



如图所示，共有两种排列的方法。

第一种排列中，选手 1 与 2 之间的比赛虽然选手 2 获胜，但他属于红队，且在选手 1 顺时针方向，故这场比赛不计入得分。而选手 2 和 3 之间的比赛蓝方获胜。因此红队得分为 0，蓝队得分为 1。

第二种排列中，共进行两场比赛，且均计入得分。红队得分与蓝队得分均为 1。

【样例 2 输入】

```
5
RBBRR
```

【样例 2 输出】

```
0 0 0 0 8 8 8 0 0 0 0
```

【样例 3】

见题目目录下的 ***3.in*** 与 ***3.ans***。

【子任务】

对于所有数据，满足 $3 \leq n \leq 3,000$，s 为由 `B` 和 `R` 构成的字符串。

子任务编号	分值	n
1	10	≤ 17
2		≤ 30
3		≤ 50
4		≤ 200
5	45	≤ 500
6	15	$\leq 3,000$

绝顶之战（struggle）

【题目描述】

有一个长度为 m 的一维空间，还有 n 个物品，第 i 个物品的长度为 a_i。现在按照编号从小到大的顺序依次将物品放入空间中，对于第 i 个物品：

- 如果当前空间中存在一段连续的长度至少为 a_i 的空余，则你**必须**将第 i 个物品放入一段连续的长度为 a_i 的空余空间中。
- 否则，第 i 个物品无法被放入，跳过它。

你需要输出：按照编号从小到大的顺序考虑完所有物品后，所有可能的**空间占用长度**，它的定义是所有被放入空间的物品的长度之和。

【输入格式】

从标准输入读入数据。

输入的第一行两个整数 m, n，分别表示空间长度和物品个数。

第二行 n 个整数 $a_1, \cdots, a_n$，依次表示每个物品的长度。

【输出格式】

输出到标准输出。

输出两行，第一行一个整数 S，表示可能的空间占用长度的数量。

第二行 S 个整数 $b_1, \ldots, b_S$，**从小到大**输出所有可能的空间占用长度。

注意，你需要保证 $b_1, \ldots, b_S$ 不重不漏。

【样例 1 输入】

```
8 4
3 4 1 2
```

【样例 1 输出】

```
4
4 6 7 8
```

【样例 1 解释】

下图分别展示了空间占用长度为 $4, 6, 7, 8$ 的一种可能方式：

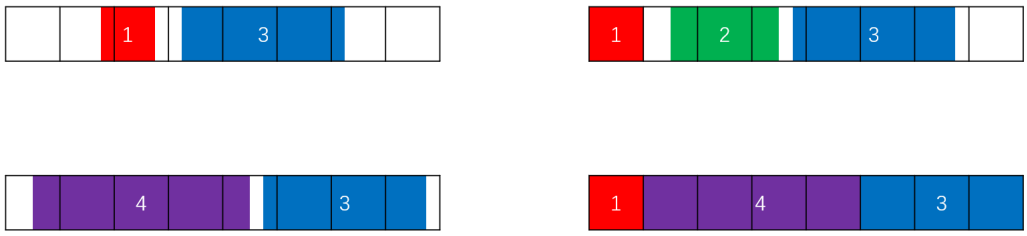


图 1: 样例解释 1

【样例 2】

见题目目录下的 ***2.in*** 与 ***2.ans***。

【子任务】

对于所有测试数据，$1 \le m, a_i \le 10^{16}$， $1 \le n \le 14$。

子任务编号	$n =$	$m, a_i \le$	分数
Subtask 1	5	10	15
Subtask 2	2	10^{16}	5
Subtask 3	3		
Subtask 4	5		
Subtask 5	7		
Subtask 6	9		
Subtask 7	11		10
Subtask 8	12		
Subtask 9	13		
Subtask 10	14		30

前往何方（wheretoreach）

这是一道交互题。

【题目描述】

有一棵 n 个点的树，你不知道其形态。你有一个初始为空的点集 S。你可以进行以下两种操作：

1. 向 S 中加入一个结点。
2. 从 S 中删除一个结点。

每次操作结束后，交互库将会返回树对 S 的导出子图（即仅保留 S 中的点和两端都在 S 中的边得到的图）的最大连通块大小。

你需要使用尽可能少的操作还原树的结构。

【实现细节】

请确保你的程序开头有 `#include "wheretoreach.h"`。

你不需要也不应该实现主函数。你需要实现以下函数：

```
void solve (int n);
```

- 其中 n 表示树的结点个数，结点从 1 至 n 编号。

你可以调用下列由交互库提供的函数：

```
int add (int x);
int remove (int x);
void report (int x,int y);
```

- `add(x)` 用于向 S 中加入一个结点 x，若 x 本身就在 S 中则什么也不做。
  - 你需要保证 $1 \le x \le n$。
  - 该函数返回值为操作后 S 的导出子图最大连通块大小。
- `remove(x)` 用于从 S 中删除一个结点 x，若 x 本身就不在 S 中则什么也不做。
  - 你需要保证 $1 \le x \le n$。
  - 该函数返回值为操作后 S 的导出子图最大连通块大小；
- `report(x,y)` 表示你找到了树的一条边 (x, y)。
  - 你需要恰好调用该函数 $n-1$ 次以提交树的所有边。边的顺序和方向任意。

你调用 `add` 与 `remove` 的总次数不得超过 2×10^6。

保证在满足题目条件和数据范围的情况下，最终测试时交互库的运行时间不会超过 2000 ms，运行空间不会超过 128 MiB。

交互库不是自适应的，即树形态固定，不会随着交互过程改变。

【测试程序方式】

试题目录下的 **implementer.cpp** 是我们提供的交互库参考实现。最终测试的交互库与样例交互库有一定不同，故你的实现不应该依赖样例交互库实现。

你需要在本题目录下使用如下命令编译得到可执行程序：

```
g++ implementer.cpp sample.cpp -o sample -O2 --std=c++14 -lm
```

对于编译得到的可执行程序：

- 可执行文件将从标准输入读入以下格式的数据：
  - 第一行一个整数 n 表示树的点数。你需要保证 $1 \leq n \leq 10^4$。
  - 接下来 $n-1$ 行，每行两个整数 x, y 描述树上的一条边。边的顺序和方向是任意的。
- 读入完成之后，交互库将调用恰好一次函数 `solve`。
- 当你的实现不符合【实现细节】中的要求时，交互库只会在标准输出流输出一行一个整数 `-1`。具体地，以下情况发生时交互库会输出 `-1`：
  - 对 `add` 和 `remove` 的函数调用中传入了不满足 $1 \leq x \leq n$ 的 x；
  - 调用 `add` 和 `remove` 的次数和超过了 2×10^6；
  - `solve` 函数结束时 `report` 函数的调用次数不为 $n-1$。
- 否则，交互库会在标准输出流按照以下格式输出：
  - 第一行两个整数 Q, c，Q 表示你调用 `add` 和 `remove` 的总次数，$c = 1$ 表示你的答案正确，$c = 0$ 表示不正确。
  - 接下来若干行，为你调用 `add`，`remove` 和 `report` 的记录，其格式可以参考样例 1。

【样例 1 输入】

```
3
1 2
2 3
```

【样例 1 输出】

```
4 1
add(2);
add(3);
add(1);
remove(2);
```

```
report(3,2);
report(1,2);
```

【样例 1 解释】

该样例输出为下发的样例程序在该组样例下的输出。

【子任务】

对于所有测试数据，$1 \le n \le 10^4$。

Subtask 1 (10%)：$n = 500$。

Subtask 2 (20%)：$n = 2500$。

Subtask 3 (70%)：$n = 10^4$。

【评分方式】

本题首先会受到和传统题相同的限制，例如编译错误会导致整道题目得 0 分，运行时错误、超过时间限制、超过空间限制都会导致相应测试点得 0 分。选手只能在程序中访问自己定义的和交互库给出的变量或数据，及其相应的内存空间。尝试访问其他位置空间将可能导致编译错误或运行错误。

对于 Subtask 1,2 中的测试点，只要你的交互符合【实现细节】中的规则，且给出了正确答案，则可以获得满分。

对于 Subtask 3 中的测试点，在你的交互符合【实现细节】中的规则，且给出了正确答案的基础上，设你调用 `add` 和 `remove` 的总次数为 Q，你的得分为 $f(Q)$，其中

$$f(Q) = \begin{cases} 70 & (0.0 \times 10^6 \le Q \le 0.6 \times 10^6) \\ 70 - \frac{Q - 0.6 \times 10^6}{5000} & (0.6 \times 10^6 < Q \le 0.7 \times 10^6) \\ 50 - \frac{Q - 0.7 \times 10^6}{15000} & (0.7 \times 10^6 < Q \le 1.0 \times 10^6) \\ 30 - \frac{Q - 1.0 \times 10^6}{25000} & (1.0 \times 10^6 < Q \le 1.5 \times 10^6) \\ 10 - \frac{Q - 1.5 \times 10^6}{50000} & (1.5 \times 10^6 < Q \le 2.0 \times 10^6) \end{cases}$$

是一个连续的分段线性函数。你可以理解为：你的前 0.6×10^6 次调用是免费的；此后的 0.1×10^6 次调用中每 5000 次扣 1 分；随后的 0.3×10^6 次调用中每 15000 次扣 1 分；随后的 0.5×10^6 次调用中每 25000 次扣 1 分；随后的 0.5×10^6 次调用中每 50000 次扣 1 分。

你在每个子任务上获得的分数为在其中所有测试点上分数的最小值。

选手不应通过非法方式获取交互库的内部信息，如试图与标准输入、输出流进行交互。此类行为将被视为作弊。