第3章 百度之星2019

3.1 初赛第一场

Polynomial

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

度度熊最近学习了多项式和极限的概念。现在他有两个多项式 $f(x)$和 $g(x)$，他想知道当 x 趋近无限大的时候，$f(x)/g(x)$收敛于多少。

输入

第一行一个整数 $T(1\leqslant T\leqslant 100)$表示数据组数。对于每组数据，第一行一个整数 $n(1\leqslant n\leqslant 1000)$，$n-1$ 表示多项式 f 和 g 可能的最高项的次数(最高项系数不一定非 0)。接下来一行 n 个数表示多项式 f，第 i 个整数 $f_i(0\leqslant fi\leqslant 1000000)$表示次数为 $i-1$ 次的项的系数。接下来一行 n 个数表示多项式 g，第 i 个整数 $g_i(0\leqslant g_i\leqslant 1000000)$表示次数为 $i-1$ 次的项的系数。数据保证多项式 f 和 g 的系数中至少有一项非 0。

输出

对于每组数据，输出一个最简分数 a/b(a 和 b 的最大公约数为 1)表示答案。如果不收敛，输出 1/0。

输入样例

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

输出样例

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

样例描述

这些多项式分别为

```
f(x) = 2x
g(x) = 1
f(x) = 1
g(x) = 2x
f(x) = 4x + 2
g(x) = 2x + 1
```

关键思路

这道题要比较的是 f 和 g 的次数和最高项系数 f_h，g_h。若 f 的次数＞g 的次数，则答案为 1/0，若 f 的次数＜g 的次数，则答案为 0/1，否则答案为 $\frac{f_h}{\gcd(f_h,g_h)}/\frac{g_h}{\gcd(f_h,g_h)}$。

Game

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

度度熊在玩一个好玩的游戏。游戏的主人公站在一根数轴上，他可以在数轴上任意移动，对于每次移动，他可以选择往左或往右走一格或两格。现在他要依次完成 n 个任务，对于任务 i，只要他处于区间$[a_i,b_i]$上，就算完成了任务。度度熊想知道，为了完成所有的任务，最少需要移动多少次？度度熊可以任意选择初始位置。

输入

第一行一个整数 $T(1\leqslant T\leqslant 10)$表示数据组数。

对于每组数据，第一行一个整数 $n(1\leqslant n\leqslant 1000)$表示任务数。接下来 n 行，第 i 行两个整数 a_i，b_i $(1\leqslant a_i\leqslant b_i\leqslant 1000000)$表示任务对应的区间。

输出

对于每组数据，一行一个整数表示答案。

输入样例

```
1
2
1 10
20 30
```

输出样例

```
5
```

样例描述

选取 10 为起点，经过的轨迹为 10-12-14-16-18-20。

关键思路

对于区间 $[a_i,b_i]$，关键点至多只有四个，分别是 a_i，a_i+1，b_i-1，b_i，从区间外面进来的话，第一下肯定是停留在这些关键点中的某一个，随机可以完成任务。离散所有的关键点，用 $f[i][j]$ 表示完成了前 i 个任务，当前在第 j 个关键点时最少要移动多少次，直接 dp 即可。

Mindis

时间限制：4000/2000 毫秒(Java/其他)

空间限制：131072/131072KB(Java/其他)

题目

平面上有 n 个矩形，矩形的边平行于坐标轴，现在度度熊需要操控一名角色从 A 点走到 B 点。

该角色可以上下左右移动，在恰被 k 个矩形覆盖的区域，该角色的速率为 $k+1$ 个距离/秒(矩形覆盖区域包括边界)。

请求出 A 移动到 B 最快需要多少秒。

输入

第一行一个整数 $T(1\leqslant T\leqslant 5)$表示数据组数。

对于每组数据，第一行输入一个整数 $n(1\leqslant n\leqslant 200)$。

接下来 n 行每行 4 个整数 $x1,y1,x2,y2(0\leqslant x1<x2\leqslant 1000000000,0\leqslant y1<y2\leqslant 1000000000)$，分别表示矩形的左下角和右上角的坐标。

最后一行四个整数 $xa,ya,xb,yb(0\leqslant xa,xb,ya,yb\leqslant 1000000000)$代表 A 和 B 的坐标。

输出

对于每组数据，输出一个小数表示答案。答案保留 5 位小数。

输入样例

```
1
1
5 5 6 6
7 7 8 8
```

输出样例

```
2.00000
```

关键思路

设输入中的点，横坐标集合为 s_1，纵坐标集合为 s_2；s_1 与 s_2 做笛卡儿积，得到一些“关键点”；每个“关键点”向上下左右的第一个“关键点”连边(如果存在)；求 A 点到 B 点的最短路即可。

稍麻烦的地方在求权边。边的端点，有可能被个数不同的矩形覆盖。一种解决办法是在两个相邻的“关键点”之前插入一个点，根据插入的点被矩形覆盖的次数计算边权。点被矩形覆盖次数，可以构造差分数组后做二维前缀和求解。

Str

时间限制：2000/1000 毫秒(Java/其他)

空间限制：131072/131072KB(Java/其他)

题目

对于一个字符串 s，定义 $rev(s)$为它的反串，abc 的反串为 cba。

有 n 个串，第 i 个串为 $s[i]$。

1. 决定一个整数 $k(1\leqslant k\leqslant n)$(这一步有 n 种方案)

2. 有序地从 n 个串中，选出 k 个串，设第 i 个选择的串为 $t[i]$(这一步有 Akn 种方案)

3. 对于每个串我们可以决定是否翻转它，如果翻转第 i 个串，那么 $t[i]=rev(t[i])$。(这一步有 $2k$ 种方案)。

求有多少种方案使得 $str=t[1]+t[2]+\cdots+t[k]$(+表示字符串拼接)是回文串。

输入

第一行输入一个整数 T，代表 $T(1\leqslant T\leqslant 20)$组数据。

对于每组数据，第一行一个数字 $n(1\leqslant n\leqslant 10)$，表示串的个数，接下来 n 行，第 i 行表示字符串 $s[i]$。

输入保证，$n\geqslant 5$ 的数据不超过 5 组，每组数据字符串长度之和不会超过 1000。

输出

输出 T 行，每行一个整数表示答案。

输入样例

```
2
2
a
a
2
a
ab
```

输出样例

```
12
6
```

样例描述

样例 1 有 $s1,rev(s1),s2,rev(s2),s1+s2,s1+rev(s2),rev(s1)+s2,rev(s1)+rev(s2),s2+s1,s2+rev(s1),rev(s2)+s1,rev(s2)+rev(s1)$一共 12 种。

样例 2 有 $s1,rev(s1),s2+s1,s2+rev(s1),s1+rev(s2),rev(s1)+rev(s2)$共 6 种。

关键思路

考虑 dp，我们从两侧向中间一次决策加入什么串以及是否应该翻转。dp[xxx]表示总的方案数，其中 mask 表示已经添加了 mask 集合的串，id 表示未被匹配的字符串编号，len 表示串 id 漏出的长度，0/1 表示漏出的是前缀还是后缀。新加入一个字符串的时候，尝试着拿着这个串去和漏出的部分匹配，这样做可以保证漏出的部分至多来源于一个字符串。

Seq

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

度度熊有一个递推式 $a_n = (\sum_{i=1}^{n-1} a_i * i) \% n$，其中 $a_1 = 1$。现给出 n，需要求 an。

输入

第一行输入一个整数 T，代表 $T(1 \leqslant T \leqslant 100000)$组数据。

接下来是 T 行，每行一个数字 $n(1 \leqslant n \leqslant 10\text{^}12)$。

输出

输出 T 行，每行一个整数表示答案。

输入样例

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

输出样例

```
1
1
0
3
0
```

关键思路

介绍某种打表的思路：

打表输出 a 序列前若干项。定睛一看，存在某个常数 B，使得 $i > B$，$a_i - a_{i-6} = a_{i+6} - a_i$。$B$ 取 500 即可通过。有兴趣的同学证明自己的结论是对的。

Subway

时间限制：6000/4000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

一列地铁，有编号为 1 到 L 的 L 节车厢，这 L 节车厢排成一排，第 i 节车厢在位置 i，容量为 $c[i]$ 人，地铁只在第 1 秒至第 D 秒的时间内允许乘客上车(注：包括第 1 秒和第 D 秒)。所有乘客都要先乘坐手扶电梯到达位置 p，每个乘客每秒钟可以向左或者右走动一个单位的距离，上车的时间不计。

现有两组人：

* 第一组有 n 个人，第 i 个人到达位置 p 的时间，为第 $t1[i]$秒。
* 第二组有 m 个人，第 i 个人到达位置 p 的时间，为第 $t2[i]$秒，他想上第 $y[i]$号车厢，如果他走到

$y[i]$号车厢时 $y[i]$车厢人满了，或者地铁不允许乘客上车，他不会上车。

现在，我们需要给第一组的第 i 个人，决定他打算上的车厢 $x[i]$（如果走到 $x[i]$号车厢时，$x[i]$号车厢人满了，或者地铁不允许乘客上车，他不会上车），从而最大化第一组人中上车的人数。

地铁里的人行色匆匆，每个人都会以最短的时间，走向自己想要上的车厢所在位置。

输入

第一行是一个整数 T，表示 $T(1\leqslant T\leqslant 20)$组数据。

每组数据

* 第一行 5 个整数 $L,p,n,m,D(1\leqslant p\leqslant L\leqslant 100000,1\leqslant D\leqslant 100000,0\leqslant n,m\leqslant 100000)$。
* 第二行 L 个整数，第 i 个表示 $c[i](0\leqslant c[i]\leqslant 100000)$。
* 第三行 n 个整数，表示第一组人下电梯的时间$(1\leqslant t1[i]\leqslant 100000)$。
* 第四行 m 个整数，表示第二组人下电梯的时间$(1\leqslant t2[i]\leqslant 100000)$。
* 第五行 m 个整数，表示第二组人想去的车厢$(1\leqslant y[i]\leqslant L)$。

数据保证，同一时刻，不会有两个人同时到达位置 p。

输出

每组数据，输出一个整数表示第一组中，最多几个人可以上车。

输入样例

```
1
3 2 2 1 10
1 1 1
1 3
2
2
```

输出样例

```
2
```

关键思路

首先，我们可以观察到，第一组中，后来的人能走到的区间，是先来的人能走到的区间的子区间。先假设第一组的人全在睡觉，不进入车厢，我们把第二组全部塞到车厢里面去，每塞一个人车厢容量－1，车厢容量可以减至负数。现在第一组的人不睡觉了，我们进行时光倒流，每当遇到一个第二组的人，我们可以把他赶出来，车厢容量＋1，每当遇到一个第一组的人，我们查询他能走到的区间内有没有容量为正的车厢（可以用线段树或者堆维护极值）。如果有，让他进入一个容量为正的车厢，并让该车厢容量－1，如果没有，那么不让他进入车厢。这个方法是对的，因为比他来得晚的人能走到的位置是他能走到的位置的子集，如果我们对来得比他晚的人进行“调整”，无法找到增广路。

3.2 初赛第二场

度度熊与数字

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

度熊发现，1，3 以及 9 这三个数字很神奇，它们的所有的倍数的每位数字的和一定是自己的倍数。例如，54 是 3 的倍数，同时 5＋4＝9 也是 3 的倍数。在另一个例子 666 是 9 的倍数，同时 6＋6＋6＝18 也是 9 的倍数。

度熊又发现，除了 1，3，9 以外的的正整数，虽然并不满足“所有的倍数的每位数字的和一定是自己的倍数”，但也存在一些数是它们的倍数且各位数字和也是它们的倍数。例如，888 是 12 的倍数，且它的各位数字和 8＋8＋8＝24 也是 12 的倍数。

现在度熊想知道，给你一个正整数 V，是否存在一个数 x，使得 V 是 x 的倍数，同时它的每位数字的和也是 x 的倍数呢？请找出所有这样的数 x。

输入

有多组询问，第一行包含一个正整数 T 代表有几组询问，接着每组测试数据占一行，包含一个正整数 V。

$$1 \leqslant T \leqslant 100, \quad 1 \leqslant V \leqslant 10^9$$

输出

对于每一个询问，输出两行，第一行包含一个正整数 m，m 代表对于该询问的 V，有几个满足条件的 x。第二行输出 m 个数，把所有满足条件的 x 由小到大输出。

输入样例

```
3
1
9
666666
```

输出样例

```
1
1
3
1 3 9
6
1 2 3 6 9 18
```

提示

第一个询问中，1 的各位数和为 1＝1×1，本身等于 1×1 都是 1 的倍数，故 1 确实为 V＝1 的答案。

第三个询问中，666666 的各位数和为 36＝9×4，本身等于 9×7474 都是 9 的倍数，故 9 确实为 V＝666666 的答案，经过仔细计算后能发现，除 9 以外，1，2，3，6，18 也都是答案。

关键思路

难度 0/5

签到题，令 V 的所有位数的数字和为 d，且 V 和 d 的最大公因数为 g，则把 g 的所有因数由小到大输出就是答案。

顺便一提，若 V 的限制改为位数可高达 10^6 也是能做，不过由于是签到题，就不这样搞了。

度度熊与排列

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

度熊有一台机器，这台机器有一个 $1\sim M$ 的排列 $p[1..M]$当作参数，若丢进一个长度为 M 的字符串，此机器会将此字符串重新排列后再输出，重新排列的方式为：原本第 i 个位置的字符会变到第 $p[i]$ 个位置。

举例来说，当 $M=3$，$p[1]=3$，$p[2]=1$，$p[3]=2$，那么丢 abc 进入这个机器后，机器会输出 bca；若丢进的是 ded，那么机器会输出 edd。

某天，度熊不小心忘记这个机器的参数了，只记得参数的长度是 M，于是他丢了 N 长度为 M 的字符串进去，并记录下对于每个字符串机器的输出结果，请你根据这些结果，帮度熊找回这个机器的参数。若有多组参数都满足度熊的记录，请输出字典序最小的排列作为参数。若并不存在任何参数满足度熊的记录，请输出-1。

注：对于两个相异的排列 a：$a[1..M]$和 $b[1..M]$，我们称 a 比 b 小当且仅当存在一个 i，满足对于所有小于 i 的 j 都有 $aj=bj$ 且 $ai<bi$。

输入

有多组询问，第一行包含一个正整数 T 代表有几组询问。

每组询问的第一行包含两个正整数 N，M，分别代表度熊丢进机器的字符串数目以及参数的长度。接下来还有 $2\times N$ 行，每行有一个长度为 M 的字符串，当中的第 $2\times i-1$ 行的字符串代表度熊丢进去机器的第 i 个字符串，而第 $2\times i$ 行的字符串代表机器对于第 i 个字符串的输出结果。

$1\leqslant T\leqslant 100$，$1\leqslant N\leqslant 20$，$1\leqslant M\leqslant 50$，字符串由英文小写字母('a'至'z')组成。

输出

对于每一个询问，输出一行，若不存在任何参数满足度熊的记录，这行只包含一个整数-1。否则这行包含一个排列，代表此机器所有可能的参数中字典序最小的那个。

输入样例

```
4
1 3
abc
bca
2 4
aaab
baaa
cdcc
cccd
3 3
aaa
aaa
bbb
bbb
ccc
```

```
ccc
1 1
a
z
```

输出样例

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

提示

第一组询问中，$p[1]=3, p[2]=1, p[3]=2$ 是唯一的机器可能的参数。

第二组询问中，$p=[2,4,3,1]$和 $p=[3,4,2,1]$都是机器可能的参数，不过[2,4,3,1]的字典序比[3,4,2,1]还小，故必须输出 2,4,3,1。

关键思路

难度 1/5

若 $p[i]=j$，那么把丢进去的 N 个字符串的第 i 个字符按照顺序连接起来的字符串，会和输出的 N 个字符串的第 j 个字符按照顺序接起来的字符串一模一样。

所以对于 i 从 1 至 M，只要依序找出最小的还没配对过且满足上述条件的 j 即可，若某个时刻找不到可配对的 j 就是无解。

度度熊与运算式 1

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

某天度度熊发现了一个由 $n+1$ 个数字 1 组成的运算式如下：

```
1  [op]_1  1 [op]_2  1…1  [op]_n  1
```

其中 op_i（i 的取值为 1 到 n）可能是⊕(按位异或运算)或是?(问号)。

例如当 $n=5$ 时，式子可能长成这样：1⊕1?1⊕1?1?1

现在，度熊想把所有的?取代为+或⊕(贴心提示：加法运算的优先级比按位异或运算还高)。

请问取代完后此运算式可能的最大运算结果为何?

输入

有多组询问，第一行包含一个正整数 T 代表有几组询问，接着每组测试数据占一行，包含一个长度为 n 的字符串，仅由^和?组成，第 i 个字符若是^就代表 $op_i=⊕$，否则 op_i 就是问号。(n 的值不会在数据中出现，请由字符串长度来判断)。

* $1\leqslant n\leqslant 2^{21}-2$
* 所有询问的 n 的总和不超过 2×10^7。

输出

对于每一个询问输出一行包含一个整数代表答案，也就是该算式的问号被取代后可能的最大运算结果。

输入样例

```
4
?
??
^^
^^^
```

输出样例

```
2
3
1
0
```

提示

样例的第一组询问算式为：1?1。取代后有2种可能1+1和1⊕1，其中1+1=2、1⊕1=0，所以最大的可能值是2。

样例的第二组询问算式为：1?1?1。取代后有4种可能1+1+1，1+1⊕1，1⊕1+1和1⊕1⊕1，其中1+1+1=1+1⊕1=1⊕1+1=3、1⊕1⊕1=1，所以最大的可能值是3。

样例的第三组询问算式为：1⊕1⊕1。并不包含问号，只有唯一的运算结果1。

样例的第四组询问算式为：1⊕1⊕1⊕1。并不包含问号，只有唯一的运算结果0。

关键思路

难度3/5

观察1：运算结果的奇偶性和$n+1$的奇偶性一样。

观察2：若把最终的运算式用⊕符号切成很多段，定义一段的长度为该段中1的个数。若某一段长度(假设是k)不是2的幂次方，那么我们可以把该段的一些+号改成⊕，也就是切成更多段，使得每一段长度都是2的幂次方且运算结果不变，例如说：1+1+1+1+1+1+1=1+1+1+1⊕1+1⊕1。(简单来说就是把k用二进制的每个bit所代表的树拆开。)

观察3：假设最终的运算式有多个长度为2^i的段落($i\geq 1$)，那么我们可以只保留当中1个段落，剩下的段落都切成长度为1的小段落，如此变更后，最终的运算结果只会增加不会减少。所以运算结果最大的可能列式中，存在一种列式方法是：对于所有非零的i，至多存在一个长度为2^i的段落，且不存在任何长度为非2的幂次方的段落。

有了这些观察后，一个贪心的做法已经呼之欲出了：枚举i从大到小，尝试能不能切出某个段落长度为2^i。

把原始的输入一样用⊕来分段。假设我们正在枚举i，若存在两个以上段落长度$\geq 2^i$，这代表对于每个$\leq i$的正整数j，我们一定能切出一段长度为2^j的段落，若恰只有一段长度为x满足$x\geq 2^i$，那我们可以先把该段切两段长度分别为2^i和$x-\geq 2^i$，其中2^i那段之后已经不需要再考虑，接着我们就继续枚举$i-1$重复同样的步骤。

度度熊与组题

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

沃老师在出比赛的题目时遇到麻烦啦！遇到的麻烦如下：

现在沃老师手上有 $2n$ 道题，题目编号由 $1 \sim 2n$，已知第 i 道题难度为 a_i，这些题的难度还满足当 $i < j$ 时 $a_i \leqslant a_j$。现在沃老师想把这些题目分在两套比赛上，每套比赛会被分到 n 道题，每道题都会恰出现在其中一场比赛中。假设分配完后，第一套题难度第 i 小的题的难度为 c_i(第 i 小是指不去重的的第 i 小，例如有四道题难度分别是 1,1,2,3 时，难度第 3 小的题是难度为 2)，第二套题难度第 i 小的题为 d_i，沃老师定义两场比赛的难度相似度为 $\sum_{i=1}^{n}(c_i - d_i)$，且沃老师希望分配完题目后，两场比赛的难度相似度尽可能的小。

看到这你可能会觉得这算什么麻烦，难度相似度的最小值不就是 $\sum_{i=1}^{n}(a_{2i} - a_{2i-1})$ 吗？

是的，光是要使难度相似度最小并不构成沃老师的麻烦，但沃老师是个好奇宝宝，他还想知道，有多少种分配题目的方式，能使难度相似度最小呢？这个问题可能就没那么简单了。

于是沃老师就来拜托聪明的度熊帮他解决心中的困惑，各位也帮忙算算吧。

请输出分配题目的方式数量除以 10^9+7 的余数。

当且仅当某道题出现在 A 的第一套比赛中，却没有出现在 B 的第一套比赛中，则称两种分配方式 A 和 B 不同。举例来说，当 $n=1$ 时，第一套比赛含有题目 1 第二套比赛含有题目 2 和，这和第一套比赛含有题目 2 第二套比赛含有题目 1 是不同的。

输入

有多组询问，第一行包含一个正整数 T 代表有几组询问，接着每组测试数据占 2 行，第一行包含一个正整数 N，第二行包含 $2N$ 个正整数 $a_1, a_2, \cdots, a_{2N}$。

* $1 \leqslant T \leqslant 2 \times 10^4$，$1 \leqslant N \leqslant 10^5$，$1 \leqslant a_i \leqslant 2 \times N$，对于不同的正整数 i, j，若 $i < j$，则 $a_i \leqslant a_j$。
* 所有询问的 N 的总和不超过 10^6。

输出

对于每一个询问。输出一个非负整数代表答案除以 10^9+7 的余数。

输入样例

```
5
2
1 2 2 4
2
1 2 3 4
1
1 1
1
1 2
6
9 9 9 10 10 10 11 11 11 12 12 12
```

输出样例

```
6
4
2
2
324
```

提示

令 day1 是第一套比赛的题目题号集合，day2 是第二套比赛的题目题号集合。

在第一组询问中，全部六种组合难度相似度都是 3，故答案为 6。

在第二组询问中难度相似度最小为 2，有四种可能分配方式如下：

```
1. day1: {1,3},day2:{2,4}
2. day1: {1,4},day2:{2,3}
3. day1: {2,3},day2:{1,4}
4. day1: {2,4},day2:{1,3}
```

关键思路

难度 3/5

引理：在满足沃老师对两套题的要求下，对于任意整数 v，在第一套题中难度$\leqslant v$ 的题数与在第二套题中难度$\leqslant v$ 的题数差的绝对值不超过 1。

证明：若题数差的绝对值超过 1，那么只要把难度$\leqslant v$ 的题目比较多的那套题中，任选一个最高难度且难度值$\leqslant v$ 的题，和题目比较少的那套题中，任选一个最低难度且难度值$>v$ 的题交换，即可找到更优的组题方式，故产生矛盾。

有了这个引理后，我们只要采用动态规划，状态 $dp[v]$代表已经决定好所有难度$\leqslant v$ 的题要放在哪一套的方法数。转移可分为四种情形：难度$\leqslant v-1$ 的题数的奇偶性搭配难度$\leqslant v$ 的题数的奇偶性。

举例来说：若难度$\leqslant v-1$ 的题数和难度$\leqslant v$ 的题数都是偶数，且前者比后者少 $2k$ 题，那么转移就是 $dp[v]=dp[v-1]\times\binom{2k}{k}$。

若难度$\leqslant v-1$ 的题数是偶数，但难度$\leqslant v$ 的题数都是奇数，且前者比后者少 $2k+1$ 题，那么转移就是 $dp[v]=dp[v-1]\times 2\times\binom{2k+1}{k}$。

剩下两种情况请大家自己尝试看看。

度度熊与整数集合

时间限制：4000/2000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

度度熊在用一个由最小的 N 个正整数组成的集合(也就是$\{1,2,\cdots,N\}$)和一个下标为 $1\sim N$ 的数组 a 玩游戏。

游戏开始时数组 a 的所有数字都是 0，接着共有 $N-1$ 次操作，每次操作由以下步骤组成：

1. 选取一个元素个数大于1的集合 S。

2. 对于 S 中所有元素 i，都把 $a[i]$ 的值加1。

3. 选取一个正整数 x，这个 x 必须满足 S 中小于等于 x 的数的数量以及大于 x 的数的数量都不为0。

4. 把集合 S 分成两个集合，第一个包含 S 中所有小于等于 x 的元素，第二个包含 S 中所有大于 x 的元素，产生出两个集合后，集合 S 就消失了。

$N-1$ 次操作结束后就会产生出 N 个集合，这 N 个集合恰是对于所有 $i=1\sim N$，大小为1且仅包含数字 i 的集合。

游戏结束后，度熊忘记了 $N-1$ 次操作的过程，只知道操作完后 $a[1]\sim a[N]$ 的值，请根据这些值来还原操作，仅需依序输出第 i 次操作所选的 x 值即可。若有多组解，请输出字典序最小的解。

提示：我们称数组 $c_1,c_2,\cdots,c_n$ 字典序比数组 $d_1,d_2,\cdots,d_n$ 小，当且仅当存在某个 i 满足对于所有 $j<i$ 都有 $c_j=d_j$，且 $c_i<d_i$。

输入

有多组询问，第一行包含一个正整数 T 代表有几组询问，接着每组测试数据占2行，第一行包含一个正整数 N，第二行包含 N 个正整数 $a[1],a[2],\cdots,a[N]$。

$1\leqslant T\leqslant 20000$，$2\leqslant N\leqslant 10^5$，$1\leqslant a[i]\leqslant N-1$，所有询问的 N 的总和不超过 3×10^6。

输出

对于每一个询问。若有可能还原操作，则输出两行，第一行包含一个字符串 Possible，第二行包含 $N-1$ 个正整数，依序是第 i 次操作所选的 x 值（若有多组可能，请输出字典序最小的）；若不可能还原，仅需输出一行包含一个字符串 Impossible。

输入样例

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

输出样例

```
Possible
1
Impossible
Possible
2 1 3
Possible
1 2
```

提示

对于第一组询问来说，当 $n=2$ 时，可能的游戏过程只有一种，也就是在唯一一次的操作中，令 $x=2$，

即可把集合{1,2}拆成两个集合{1}和{2}。游戏结束后数组 $a=[1,1]$ 刚好就是这组询问，所以答案为 1。

对于第二组询问来说，没有任何一种游戏方式可以让 a 变成[1,1,2]，答案为 Impossible。

对于第三组询问来说，虽然存在 2,1,3 和 2,3,1 两种可能的游戏过程，但 2,1,3 的字典序比较小，所以必须输出 2,1,3。

关键思路

难度 3/5

首先，我们能发现游戏过程可以对应到一个 n 个叶子的二元树，且每个非叶子节点都恰有两个子节点，由左至右数来第 i 个叶子的深度就是 $a[i]$，游戏的每个步骤，恰好对应到某个节点，把子树的所有叶子分到左子树和右子树。实际上，每个合法的输入可以对应到唯一的二元树，若我们能得到对应的二元树，就可以用 dfs，先往左子树走来得出字典序最小的游戏步骤。

思考过程略过，就结论而言，我们可以借助 stack 用 $O(n)$ 的时间复杂度来找到输入对应到的二元树。依序把叶子由编号小至大加入 stack，stack 里保持点的深度为非递减数列。若要加进的叶子深度比 stack 顶端的深度还要小，就代表 stack 当前顶部的两个点是同一个节点的子节点，且深度必须一样，若不一样就代表无解，否则我们就把该两点移除 stack，并把它们的父节点试图加入 stack（使用试图两个字，是因为要把它们的父节点加入 stack 时，可能引起其他点也再度合并的连锁反应），直到最后 stack 里还会剩一些点，再持续把顶部的两个点合并最后会只剩下一个点，也就是树根，如此一来我们就还原出二元树了。

此方法时间复杂度为 $O(n)$。

度度熊与运算式 2

时间限制：8000/4000 毫秒(Java/其他)

空间限制：262144/262144KB(Java/其他)

题目

某天度熊发现了一个由 $n+1$ 个数字 1 组成的算式如下：$1\ op_1\ 1\ op_2\ 1\cdots 1\ op_n\ 1$，其中 op_i 可能是＋(加法运算符号)或是?。

例如当 $n=5$ 时，式子可能长成这样：1＋1?1＋1?1?1

现在，度熊想把所有的? 取代为＋或⊕(按位异或运算)。

(贴心提示：加法运算的优先级比按位异或运算还高)

请问取代完后此运算式可能的最大运算结果为何? 同时度熊也想知道，有多少取代方式能达到最大的运算结果?

对于两种取代方式，只要存在至少一个 i 使得 op_i 是问号且在两种取代方式被取代为不同运算符号，就视为不同。

(特别的贴心提示：程序语言中取余数的操作(%)比加减法或乘法慢很多，请尽量减少余数的操作)

输入

有多组询问，第一行包含一个正整数 T 代表有几组询问，接着每组测试数据占一行，包含一个长度为 n 的字符串，仅由＋和? 组成，第 i 个字符就是 op_i。(n 的值不会在数据中出现，请由字符串长度来判断。)

$1 \leqslant n \leqslant 2^{21}-2$，所有询问的 n 的总和不超过 2×10^7。

输出

对于每一个询问输出一行包含两个数字间用一个空格做分隔，第一个数字代表最大的可能运算结果，第二个数字代表有多少种不同取代方式可以达到相同结果。由于答案可能很大，请输出答案除以 10^9+7 的余数。

输入样例

```
6
?
??
+?
???
????????+????
??????????????
```

输出样例

```
2 1
3 3
3 2
4 1
15 66
15 75
```

提示

样例的第一组询问算式为：1?1。取代后有 2 个可能 1+1 和 1⊕1，其中 $1+1=2$、$1 \oplus 1=0$，所以最大的可能值是 2 且只有一种取代方式能达到该值。

样例的第二组询问算式为：1?1?1。取代后有 4 种可能 1+1+1，1+1⊕1，1⊕1+1 和 1⊕1⊕1，其中 $1+1+1=1+1 \oplus 1=1 \oplus 1+1=3$、$1 \oplus 1 \oplus 1=1$，所以最大的可能值是 3 且有 3 种取代方式能达到该值。

关键思路

难度 4/5

这题是出题者自认为能排进至今出过的题中前十名的好题之一。标程的做法是常数极小的 $O(n \log n)$。

首先，算式的最大值一定是 $n+1$，只要把所有问号取代为加号即可，并且由于按位异或其实就是不进位的加法，所以含有⊕的算式得到的结果一定不会高于全部都是加号。

接着不难观察出，所有使得算式达到最大值的问号取代方式一定满足以下条件：

令由左边数来第 i 个⊕符号左边的 1 的个数有 a_i 个，其中共有 m 个⊕符号，并且令 $a_{m+1}=n+1$，那么对于任意 $\ll m$ 的正整数 i 都必须满足 $a_i \& a_{i+1}=a_i$（这里的 & 是按位与符号），换句话说，序列 a 中相邻的两个数都必须满足：前一个数的二进制表示法中，是 1 的位元必须是后一个数的子集。

于是至此我们可以把问题转化为：给你一个所有元素范围在 1 至 n 的正整数集合 A，请问它有多少个子集满足：把子集元素再加上 $n+1$ 由小到大排序后，相邻两个数中，前一个数的二进制表示法中的 1 的位置是后一个数的子集。

令 $dp[x]$代表有多少满足上述条件的子集，其中最大的数字是 x，那么此问题列出的动态规划关系式为 $dp[0]=1$，对于 $x>0$ 且 $x\leqslant n+1$，若 x 不在集合中且 $x\neq n+1$，$dp[x]=0$，否则 $dp[x]=\sum j\geqslant 0 \wedge (j\&x)=j dp[j]$。

于是我们得到了一个直接计算上述式子＋枚举位元子集时间复杂度为 $O(3^k)$ 的方法（k 为 $n+1$ 的最高非 0 的 bit 位置）。

接着我们来优化这个 dp 式。

优化方式其实和高维前缀和的概念是几乎一模一样的，差别只在于，流传于大众的高维前缀和可以只使用 $O(n)$ 的空间，但这题估计是一定得用 $O(n\log n)$ 的空间才能完成。

我们设计新的动态规划状态如下：

首先定义计划 $S(x,k)$。

假定 $x=2^{b_1}+2^{b_2}2^{b_3}+\cdots+2^{b_m}$，$(0\leqslant b_1<b_2<b_3<\cdots<b_m)$ 也就是说 x 的二进制表示法有 k 个 bit 是 1。

若 $y\in S(x,k)$ 当且仅当 y 能表示为 $\sum_{i=1}^{m} t^i\times 2^{bi}$，若 $i>k$，则 $t_i=1$，否则 t_i 可以是 0 或 1。

举例来说，若 $x=21=(10101)2$，那么

$$S(x,0)=\{21=(101012)\}$$
$$S(x,1)=\{20=(10100)2,21=(101012)\}$$
$$S(x,2)=\{16=(101012),17=(101012),20=(101012),21=(101012)\}$$

接着我们基于前面所定义的动态规划状态，在新定义 $dp2[x][k]=\sum_{y\in s(x,k)} dp[y]$ 中可以得到以下关系式：

（1）当 $k=0$ 时，若 $x\in A$ 或 $x=n+1$，有 $dp2[x][k]=\sum_{i=1}^{m} dp2[x-2^{bi}][i]$（$bk$ 就是 x 由最低位数来第 k 个是 1 的 bit 的位置，m 是 1 的 bit 数）否则 $dp2[x][k]=0$。

（2）当 $k>0$ 时，$dp2[x][k]=dp2[x][k-1]++dp2[x-2^{bk}][k-1]$

最终答案就会是 $dp2[n+1][0]$。

3.3 初赛第三场

最短路 1

时间限制：2000/1000 毫秒（Java/其他）

空间限制：32768/32768KB（Java/其他）

题目

有一张 n 个点的完全无向图，点的标号是 $1,2,\cdots,n$，其中边 (i,j) 的长度是 i xor j，现在你需要求出点 1 到点 n 的最短路的长度。

输入

第一行一个正整数 T，表示数据组数 $1\leqslant T\leqslant 100$。

对于每组数据：第一行一个正整数 n 表示点数（$2\leqslant n\leqslant 10^5$）。

输出

输出 T 行，每行一个整数表示点 1 到点 n 的最短路。

输入样例

```
1
3
```

输出样例

```
2
```

关键思路

很明显可以证明走$(1,n)$是最短的，因为这张图的一条路径(u,v)的本质是把 u 每次异或上一个值 w，最后要求变成 v，然后长度之和就是 w 的和，异或显然满足三角不等式，所以答案是 1 xor n。

最短路 2

时间限制：6000/4000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

小 A 是社团里的工具人，有一天他的朋友给了他一个包含 n 个点，m 条边的正权连通无向图，要他计算所有点两两之间的最短路。

作为一个工具人，小 A 熟练掌握着 Floyd 算法，设 $w[i][j]$为原图中(i,j)之间的权值最小的边的权值，若没有边则 $w[i][j]$=无穷大。特别地，若 $i=j$，则 $w[i][j]=0$。

Floyd 的 C++实现如下：

```
for(int k = 1;k < = p;k++)
   for(int i = 1;i < = n;i++)
      for(int j = 1;j < = n;j++)
          w[i][j] = min(w[i][j],w[i][k] + w[k][j]);
```

当 $p=n$ 时，该代码就是我们所熟知的 Floyd，然而小 A 为了让代码跑得更快点，所以想减少 p 的值。

令 $D_{i,j}$ 为最小的非负整数 x，满足当 $p=x$ 时，点 i 与点 j 之间的最短路被正确计算了。

现在你需要求 $\sum n_i=1\sum n_j=1D_{i,j}$，虽然答案不会很大，但为了显得本题像个计数题，你还是需要将答案对 998244353 取模后输出。

输入

第一行一个正整数 $T(T\leqslant 30)$表示数据组数。

对于每组数据：

第一行两个正整数 $n,m(1\leqslant n\leqslant 1000,m\leqslant 2000)$，表示点数和边数。

保证最多只有 5 组数据满足 $\max(n,m)>200$。

接下来 m 行，每行三个正整数 u,v,w 描述一条边权为 w 的边(u,v)，其中 $1\leqslant w\leqslant 10^9$。

输出

输出 T 行，第 i 行一个非负整数表示第 i 组数据的答案。

输入样例

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

输出样例

```
6
```

关键思路

根据 Floyd 算法的原理易知：$D_{i,j}$ 是 i 到 j 的使得路径最大点标号(不含端点)最小的最短路中，标号最大的点。所以枚举 i，搞个最短路图后，在最短路图上进行 DP 就可以算出所有 $D_{i,j}$。

时间复杂度：$O(nm\log n)$

算　术

时间限制：2000/1000 毫秒(Java/其他)

空间限制：262144/262144KB(Java/其他)

题目

定义一个函数 $\mu(x)$：如果 x 等于 k 个不同的质数的乘积，则 $\mu(x)=(-1)k$，否则(即 x 有大于 1 的平方因子)$\mu(x)=0$

定义 $lcm(a,b)$ 为 a,b 的最小公倍数，给定 n,m，你需要求：$\sum_{i=1}^{z}\sum_{j=1}^{m}\mu(lcm(i,j))$

输入

第一行一个正整数 $T(T\leqslant 10)$表示数据组数。

接下来 T 行，每行两个正整数 $n,m(1\leqslant n,m\leqslant 10^6)$，表示一次询问。

输出

输出 T 行，每行一个整数表示该组数据的答案。

输入样例

```
2
2 4
5 5
```

输出样例

```
-2
-2
```

关键思路

显然有 $\mu(lcm(i,j))=\mu(i)\mu(j)\mu(\gcd(i,j))$

所以

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\mu(lcm(i,j))=\sum_{i=1}^{n}\mu(i)\sum_{j=1}^{m}\mu(j)\mu(\gcd(i,j))$$

$$=\sum_{d=1}^{\min(n,m)}\sum_{i=1}^{n/d}\sum_{j=1}^{m/d}\mu(id)\mu(jd)\mu(d)[\gcd(i,j)=1]$$

因为 $[x=1]=\sum_{d|x}\mu(d)$

展开得

$$=\sum_{d=1}^{\min(n,m)}\sum_{d_2=1}^{\min(n,m)/d}\sum_{i=1}^{n/dd_2}\sum_{j=1}^{m/dd_2}\mu(idd_2)\mu(jdd_2)\mu(d)\mu(d_2)]$$

令 $T=dd_2$

$$=\sum_{T=1}^{\min(n,m)}\sum_{d|T}\sum_{i=1}^{n/T}\sum_{j=1}^{m/T}\mu(iT)\mu(jT)\mu(d)\mu(T/d)]$$

令 $f(T)=\sum_{d|T}\mu(d)\mu(T/d)$

令 $g(T,K_n,K_m)=\sum_{i=1}^{K_n}\sum_{j=1}^{K_m}\mu(iT)\mu(jT)=\left(\sum_{i=1}^{K_n}\mu(iT)\right)\left(\sum_{j=1}^{K_m}\mu(jT)\right)$ 则答案就是：$\sum_{T=1}^{\min(n,m)}f(T)g(T,n/T,m/T)$

可以发现 $g(T,n/T,m/T)$ 是可以在 $O(n/T+m/T)$ 时间内计算的。

所以时间复杂度是 $O(Tn\log n)$，通过预处理 $\sum_{i=1}^{K_n}\mu(iT)$ 可以做到 $O(n\log n+Tn)$

反向传播

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

一共有 2^{m-1} 个函数 $f_i(x_1,\cdots,x_{2^{m-1}})\approx\approx\sim(1\leqslant i\leqslant 2^m-1)$，其中对于 $2^{m-1}\leqslant i\leqslant 2^m-1$，有 $f_i(x)=x_{i-2^{m-1}+1}$。

对于 $1\leqslant i\leqslant 2^{m-1}-1$，有 $f_i(x)=f_{2i}(x)\sim op_i\sim f_{2i+1}(x)$，其中 $op_i\in\{+,\times\}$

一开始有 $x_i=i$，接下来一共有 Q 次操作，每次操作是以下两个之一：

$-1\sim i\sim y$：表示令 $x_i=y$。

$-2\sim i$：表示你需要求出：$\lim_{d\to 0}\frac{f_i(add(x,i,d))-f_i(x)}{d}$，其中 $add(x,i,d)$ 表示：将 $x_1\cdots,x_{2^{m-1}}$ 中的 x_i 变成 x_i+d，其他的保持不变，可以发现该操作其实就是需要你求出 $\frac{\partial f_i(x)}{\partial x_i}$。

由于答案可能很大，你只需要输出答案对 998244353 取模后的值。

输入

第一行两个正整数 $m(2\leqslant m\leqslant 20)$，$Q(1\leqslant Q\leqslant 10^5)$。

第二行一个长度为 $2^{m-1}-1$ 的 01 字符串，第 i 个字符为 0 表示 $opi=+$，否则 $opi=\times$

接下来 Q 行，每行两个数或者三个数表示一次操作，保证 $0\leqslant y<998244353$。

输出

对于每次操作 2，输出答案。

输入样例

```
2 6
1
2 1
2 2
1 1 6
1 2 7
2 1
2 2
```

输出样例

```
2
1
7
6
```

关键思路

每次询问的本质是求$\frac{\partial f_1}{\partial f_i}$，而我们可以发现对于任意一个 i 根据链式法则有$\frac{\partial f_1}{\partial f_i}=\frac{\partial f_1}{\partial f_{i/2}}\frac{\partial f_{i/2}}{\partial f_i}$。

所以其实询问可以分解成一条链上每个点的父亲对它的梯度的乘积。

对于任意一个 i，设 $t(i,v)=\frac{\partial f_i}{\partial f_{2i+v}}$，如果 i 的操作是加法，则显然 $t(i,v)=1$，否则 $t(i,0)=f_{2i+1}(x)$，$t(i,1)=f_{2i}(x)$。

所以每次修改和询问我们都可以直接暴力了，因为树高是 $O(m)$ 的所以时间复杂度是 $O(mQ)$。

Min

时间限制：2000/1000 毫秒(Java/其他)

空间限制：131072/131072KB(Java/其他)

题目

给定 n 个数 $a1,2,\cdots,n$，你需要求有几个$\{1,2,\cdots,n\}$的排列 p，满足对于 $1\leqslant i\leqslant n-2$，有 $\min(api,api+1)\leqslant\min(api+1,api+2)$。

由于答案可能过大，你只需要输出答案对 998244353 取模后的值。

输入

第一行一个正整数 $T(1\leqslant T\leqslant 100)$，表示数据组数。

接下来对于每组数据，第一行一个整数 $n(1\leqslant n\leqslant 1000)$。

接下来一行 n 个数字，表示 $a1,2,\cdots,n$，其中 $1\leqslant ai\leqslant n$。

输出

输出 T 行，每行一个整数表示该组数据的答案。

输入样例

```
1
3
1 2 3
```

输出样例

```
4
```

关键思路

可以发现 $\min(a,b)\leqslant\min(b,c)$ 等价于 $\min(a,b)\leqslant c$，所以相当于求 a 有几个排列满足每个数都大于等于前面两个数的 min。

考虑从小到大插数，对于一个空隙有两种状态：可以插入更大的和不能插入更大的。可以发现，若 a,b 之间的空隙能插入更大的数，且插入后，得到的 a,c,b 没有产生新的能插入的空隙，如果插入到 c，b 之间则对于 b 来说不合法，如果插入 a,c 之间则对于 b 不合法。

所以我们的 DP 状态可以维护还剩几个空隙可以插入，然后转移时考虑新插入 k 个相同的数，考虑一下，它们分到几个空隙内，且有几个放到末尾就行了。

时间复杂度：$O(n^2)$

权　值

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

给定数组 $b_1,b_2,\cdots,n$ 和 $c_1,c_2,\cdots,n$，求有几个整数数列 $a_1,a_2,\cdots,n$，满足 $0\leqslant a_i<260$，且 a_i xor $ab_i\leqslant c_i$。

由于答案可能过大，你只需要输出答案对 998244353 取模后的值。

输入

第一行一个正整数 $3\leqslant n\leqslant 10^5$。

第二行 n 个整数表示 $b_1,b_2,\cdots,n$，满足 $1\leqslant b_i\leqslant n$，且 $b_i\neq i$。

第三行 n 个整数表示 $c_1,c_2,\cdots,n$，满足 $0\leqslant c_i<260$。

输出

输出答案

输入样例

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

输出样例

```
416046766
```

关键思路

题目等价于：给定一群环套树，要给点赋点权，使得每条边的边权（指两个端点的点权异或起来）不超过某个给定的值，求方案数。

首先可以把环套树变成环，因为对于每个叶子来说，如果它的父亲的点权已经定了，且它到父亲的边权不能超过 c，那么它的点权的方案数是 $c+1$，分别对应边权为 $0\cdots c$，所以我们可以一直去掉叶子，最后变成环上的问题。我们可以这样去算方案：把边权给定下来，然后定一下某个点的点权，就可以递推出。剩下的所有点权，所以点权的方案数等于定边权的方案数除 2^{60}。而定边权的话条件只有两个：每条边权有个上限，且边权的异或和要是 0。

所以现在问题变成了，给定 $b_1,b_2,\cdots,n$，求有几个 $a_1,a_2,\cdots,n$ 满足 $a_i\leqslant b_i$，且 a 的异或和为 0。我们可以枚举一下从第 j 位开始，存在一个 i 使得 a_i 和 b_i 在这一位不同，也就是说比 j 更高的位上 a_i 和 b_i 都相同。那么这样搞有一个好处：因为 a_i 和 b_i 在第 j 位不同了，所以 a_i 的 $0\cdots j-1$ 位都是可以任选的，不会违反 $a_i\leqslant b_i$ 的条件。所以可以令其他的 a_i 的 $0\cdots j-1$ 在不不违反 $a_i\leqslant b_i$ 的条件下任选，然后最后让 a_i 取个合适的值使得 $0\cdots j-1$ 位的异或为 0 即可。所以就枚举 j，然后 dp 一下，状态记录一下是否已经有 i 使得 a_i 和 b_i 在第 j 位不同即可。

时间复杂度：$O(n\log a_i)$

3.4 初赛第四场

Strassen

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

在本题中，我们只有两种方法计算两个 $n\times n$ 的矩阵的乘积，第一种为定义法，需要 n^3 次乘法和 $(n-1)n^2$ 次加法。第二种为 Strassen 分治法，仅当 n 为偶数时可以使用，需要 $18(n/2)^2$ 次加法以及再计算 7 次大小为 $(n/2)\times(n/2)$ 的矩阵的乘积。这 7 次更小矩阵的乘积也可以选择两种方法之一计算。现假设计算机计算一次加法需要 a 单位时间，计算一次乘法需要 b 单位时间，其他任何操作不花费时间，问计算两个 $n\times n$ 的矩阵的乘积至少需要多少时间。输出答案模 10^9+7 的余数。

输入

第一行一个正整数 t 表示数据组数($1\leqslant t\leqslant 20$)。

每组数据包含一行三个正整数 n,a,b($1\leqslant n\leqslant 2^{32}$，$n$ 是 2 的幂，$1\leqslant a\leqslant 10^9$，$1\leqslant b\leqslant 10^9$)。

输出

每组数据输出一行，包含一个整数表示答案模 10^9+7 的余数。

输入样例

```
1
16 1 1
```

输出样例

```
7872
```

关键思路

按照矩阵的大小1,2,4,8,…计算所需的时间即可。计算大小为2k的矩阵乘法所需时间时,根据大小为k的矩阵乘法所需时间,再从两种计算方法中选择一个时间较短的。要注意mod溢出。

杏杏和正方形和矩形

时间限制：2000/1000毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

在二维坐标平面上,有一个正方形,它的四个顶点是(1,1),(0,0),(0,1),(1,0)。现在杏杏随便选出了四个正方形边界上的点。对于每一个点,你都可以沿着正方形的边界按照顺指针或逆时针方向移动任意距离。你可以不移动任意一个点,也可以依次移动多个点。你的目标是：为每个点选择正方形的一条边上的一个位置,且为不同的点选择不同的边,使得这四个点组成一个矩形的四个顶点。问四个点的移动距离的和至少是多少(两条邻边上的点所选的位置可以重合在这两条邻边共用的顶点。矩形的面积可以为0,此时要求矩形的四个顶点去重后恰好有两个点)。

输入

第一行一个整数n表示数据组数($1\leqslant n\leqslant 100$)。

接下来每组数据四行,每行两个实数x和y表示一个点的坐标。

所有实数最多有7位小数,所有点均保证在题目中的正方形的边界上(允许多个点的初始位置重合或者处于正方形的同一条边上)。

输出

对于每组数据输出一行一个实数表示答案,保留12位小数。

输入样例

```
2
0 0
1 1
0 1
1 0
0.12 1.00
.91 .0
1. .08
0 .89
```

输出样例

```
0.000000000000
0.060000000000
```

关键思路

可以证明构成的矩形要么是正方形,此时正方形的重心在(0.5,0.5)；要么是四条边斜率均为1或−1的矩形。两种情况都可以枚举关键点再枚举每个点去哪条边暴力解决。

五教练的教学

时间限制：5000/3000毫秒(Java/其他)

空间限制：131072/65536KB(Java/其他)

题目

在思源湖底流行一种游戏，其中用到 1,2,…,n 的每种数字牌各 4 张。游戏玩家先从这 $4n$ 张牌中选择 $3k+1$ 张(k 是非负整数)，再获取一张随机的牌 x(随机的牌的数字也是 1,2,…,n 之一，即使玩家已经有某种数字牌 4 张，也可能再随机到这张牌)。如果这 $3k+2$ 张牌可以不重不漏地划分为一组对子和 k 组面子，玩家就胜利了。对子是两张数字相同的牌。面子可以是三张数字相同的牌，也可以是数字为 x，$x+1$，$x+2$ 的三张数字牌。

现在五教练要给新手举例子。给定玩家选择的 $3k+1$ 张牌，请输出最后有几种不同的牌 x 可以使他胜利。

输入

第一行一个正整数 t 表示数据组数($1\leqslant t\leqslant 10000$)。

每组数据第一行为两个非负整数 n，k($1\leqslant n\leqslant 100000$，$0\leqslant k\leqslant 100000$)，接下来一行有 $3k+1$ 个 1 到 n 之间(含)的正整数表示玩家选择的牌。保证每种牌最多出现 4 次。保证所有数据的 k 的和不超过 210000，所有数据 n 的和不超过 1100000。

输出

对于每组数据输出行，包含一个 0 到 n 之间(含)的整数，表示最后有几种不同的牌 x 使得玩家胜利。

输入样例

```
4
9 1
6 6 6 6
2 1
1 1 2 2
9 1
2 2 2 3
9 4
1 1 1 2 3 4 5 6 7 8 9 9
```

输出样例

```
1
2
3
9
```

关键思路

枚举胜利的牌，再枚举数值跨越这张牌的面子(如这张牌是 x，形如($x-1$，x，$x+1$)和(x，$x+1$，$x+2$)的面子就“跨越”了 x)，两侧的牌就没有联系了。所以我们预处理 dp[i][j][k][l]代表用已经选定的牌中数字为 1,2,…,i 的牌，其中数字为 $i-1$ 的牌只剩 j 张，数字为 i 的牌只剩 k 张，能否恰好组成若干面子(若 l 为 1 则再加一组对子)。从后往前再预处理一遍，枚举胜利的牌和跨越这张牌的面子数量后调用两侧预处理的结果即可。

唯一指定树

时间限制：5000/3000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

给定一个连通的带权无向图 G(可能有重边,但没有自环)。对于它的每一条边 e,问是否存在图 G 的生成树 T,使得 e 在 T 中出现,T 中的每条边的权值(重复算多次)的中位数恰为边 e 的权值。

输入

第一行一个整数 n 表示数据组数($1\leqslant n\leqslant 10$)。

每组数据的第一行有两个正整数 n 和 m 表示图中的点数和边数($1\leqslant n\leqslant 100000$,$n$ 是偶数,$n-1\leqslant m\leqslant 200000$,保证输入的图连通)。

接下来 m 行每行三个正整数 a,b,w 分别表示一条边的两个端点和它的权值($1\leqslant a$,$b\leqslant n$,$1\leqslant w\leqslant 10^9$)。

输出

对于每组数据输出一行 01 串,其中第 i 个字符为 1 表示对输入中第 i 条边的询问的答案为真,否则表示为假。

输入样例

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

输出样例

```
01011
```

关键思路

先假设各边权值不同。现在我们尝试确定边权为 x 的一条边可否成为生成树的中位数。必要条件是,边权比 x 小的所有边能选出边数至少是 $(n-1)/2$ 的森林。边权比 x 大的边亦然。可以证明这两个必要条件同时满足,这条边权为 x 的边就可以是生成树的中位数:任意生成一个包含这条边权为 x 的边的树。我们称权值小于 x 的边为小边,权值大于 x 的边为大边。如果树中小边数量多于大边,则大边数量小于 $(n-1)/2$。我们找到另一条和树中现有的大边不成环的大边。由于大边中可以选出 $(n-1)/2$ 条组成森林,这一条新的大边一定能找到。在树中找到连接这条新的大边的路径。路径上必定有一条小边,因为否则这条新的大边就和树中已有的大边成环了。删去树上的这条小边,加入新的大边。树仍然是一棵树,且大小边数量的差减小了。重复直到大小边数量相同即可。所以同时满足两个必要条件就是能成为中位数边的充要条件。我们称这个充要条件为 P。

如果有权值相同的边,只要其中一条边可以成为中位数边,其他所有权值和它相同的边都可以。证明:假设边 e 和边 f 权值相同,且有一棵生成树使得 e 为中位数边。考虑树上连接 f 的两个端点的路

径。在这条路径上随意去掉一条边换成 f。因为在一组中位数为 x 的数中，去掉任意一个数再补充一个 x 中位数必定还是 x，所以 f 成为了中位数边。

如果权值为 x 的边都可以成为中位数边，那么将这些权值相同的边任意指定顺序(剩下的边还是按照权值排序)，当做权值互不相同的情况，总有一条边满足刚才的条件 P。证明：任取一棵中位数是 x 的树。树上的边按照我们指定的顺序，正中间的一条就满足条件 P。

有了上面的结论，按照边权排序并预处理前 i 条边最多能选出几条边的森林，再从后往前预处理一次，就可以快速判断一条边是否满足刚才的充要条件了。最后同一权值的边只要有一个满足就全部都可以。

wls 的树

时间限制：6000/3000 毫秒(Java/其他)

空间限制：131072/131072KB(Java/其他)

题目

wls 有一棵有根树，其中的点从 1 到 n 标号，其中 1 是树根。每次 wls 可以执行以下两种操作之一：

(1) 选定一个点 x，将以 x 为根的子树变成一条按照编号排序的链，其中编号最大的作为新的子树的根(成为原来 x 的父亲节点的儿子，如果原来 x 没有父亲节点则新的子树的根也没有父亲节点)。

(2) 查询两个点之间的最短路径上经过了多少边。

输入

第一行一个整数 t 表示数据组数($t\leqslant 10$)。

每组数据第一行一个正整数 n 表示树上的点数($1\leqslant n\leqslant 100000$)。

接下来 $n-1$ 行每行两个 1 到 n 之间的正整数表示一条树边。

接下来一行一个正整数 q 表示询问的个数($1\leqslant q\leqslant 200000$)。

接下来 q 行每行表示一个操作。第一种操作格式为 $1x$，其中 x 为指定的树根。第二种操作格式为 $2xy$，表示查询从 x 到 y 的路径。

输出

对于每个第二种操作，输出一行一个正整数表示答案。

输入样例

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

输出样例

```
3
```

关键思路

每次询问时，路径其实是由两端的两段被拉抻成链的树以及中间的原来的树组成。因为一旦被拉

抻成链就再也变不回来了，可以用并查集维护各个变成链的子树（以及它们的根）。这样两端链的长度就是子树内权值小于某个值的节点数，可以用 dfs 序线段树，加离线排序询问的方法来实现。中间的长度就是 LCA。

Totori's Switching Game

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

我们知道 Shannon's Switching Game 是一个在图(graph)上玩的游戏，它的必胜策略和图的两个不相交的生成树有关。你不需要了解这个游戏。

本题中我们考虑 Totori's Switching Game(架空)。它的定义很简单，在一个图(可能有重边，但是没有自环)中，若存在 k 个生成树，且它们的边互不相同，则玩家胜利，否则玩家失败。

输入

第一行一个整数 t 表示数据组数($1\leqslant t\leqslant 20$)。

接下来每组数据的第一行包含三个正整数 n，m，k，依次表示图的点数边数和题面里的 k($2\leqslant n\leqslant 300$，$1\leqslant m\leqslant 300$，$1\leqslant k\leqslant 300$)。

接下来 m 行每行两个不同的正整数 a，b 表示图中一条连接 a 和 b 的边($1\leqslant a$，$b\leqslant n$)。

输出

对每组数据输出一行。若玩家胜利，则输出 Yes，否则输出 No。

输入样例

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

输出样例

```
Yes
No
```

关键思路

做法：维护 k 个森林(最后希望它们都变成树)，将输入的边任意排序，每次尝试加一条边。

尝试加边的方法：我们建一个图。每条 k 个森林中现有的边都看做点，新加的边 x 也是一个点。如果一条边 e 能加入某个它此刻不在的森林，并且成环，环上有另一条边 f，就从 e 到 f 画一条弧(弧代表我们构造的图中的边，为了和题目中的图区分)。我们希望在构造的图中找到从 x 开始到某条边 e 的路径，使得 e 可以加入某个它不在的森林并且不成环。这样的路径称为目标路径。若找到任意目标路径就可以沿着这条路径更新所有遇到的边的归属(每条边都加入它下一条边所属的森林并且把下一条

边踢掉，最后一条边直接加入最后的森林），找不到则直接抛弃新边 x。最后看总共加入了多少条边，如果每个森林都加满了则答 Yes，否则答 No。

正确性证明：

引理 1：首先我们证明，如果现在 k 个森林中共有 m 条边，且新边 x 按照上面的方式加不进来，则 k 个森林上现有的边加上 x 不可能分配在 k 个森林中。设新边 x 按照上面的方式找不到路径。考虑集合 S，为在我们构造的图中从 x 可以到达的边的集合。称我们维护的 k 个森林为森林 1，……，森林 k。称森林 i 和 S 的交为 $S[i]$。我们证明对于任意的 i，$S[i]$都是 S（将 S 考虑为一个点集为所有点，边集为 S 的图）的一个最大生成森林（一个图的最大生成森林是它每个连通块任取一个生成树的并）。若不然，假设 $S[i]$可以加入 e 而不成环，e 是 $S-S[i]$中的一条边。如果 e 可以加入森林 i 且不成环，那我们找到了一条目标路径（因为 e 可以从 x 到达），矛盾。如果 e 加入森林 i 会成环，由于 e 加入 $S[i]$不成环，e 的加入一定可以把森林 i 除去 $S[i]$中的一条边 f 踢掉。这样 f 应该属于 S，矛盾。所以 $S[i]$是 S 的最大生成森林。这对 $i=1,\cdots,k$ 都成立，所以我们有了 S 的 k 个互不相交的最大生成森林和一条额外的边 x。那么 S 不可能分配到 k 个森林中。

上面的引理证明了我们的算法等价于：维护一个可以分成 k 个森林的边的集合 E，每次判断加入边 x 后 E 是否仍然可以分为 k 个森林，若可以则加，否则不加。

引理 2：接下来我们证明，如果题目的答案为 Yes，则对于任意可以划分为 k 个森林的边集合 E，总存在 k 个不相交的生成树，使得它们的并包含 E。证明：如果 E 可以划分为 k 棵树，则不需证明。否则任选 k 个不相交的生成树（称为 $T[1],\cdots,T[k]$）。设 E 可以分为 k 个森林（称为 $L[1],\cdots,L[k]$）。因为 E 中的边数已满，总有一个 $T[i]$比 $L[i]$多至少一条边。一定可以从 $T[i]$中选择一条边加入 $L[i]$且不成环（生成树的性质）。就将这条边加入 $L[i]$。如果这条边在 $L[j](j\neq i)$中都没有出现过，那么 E 的大小增大了 1（且包含原来的 E）。如果它在某个 $L[j]$里出现过，就把它从 $L[j]$里删掉（这样 $L[1],\cdots,L[k]$依然互不相交）。$L[i]$和 $T[i]$的交增大了 1 且集合 E 不变。由于 E 的大小有限，$L[i]$和 $T[i]$的交的大小也有限，E 最终必将成为包含初始的 E 的 k 棵不相交的树的并。

最后证明算法的正确性。假设答案为 Yes。维护一个可以分成 k 个森林的边的集合 E。称一组 k 个不相交的生成树为一组合法解。算法依次判断每条边，我们只考虑包含当前 E 且由 E 和当前未判断过的边组成的合法解。用归纳法证明这样的合法解一直存在：初始时 E 为空，未判断的边为所有边，由于答案为 Yes，这样的合法解存在。每次判断加入边 x 后 E 是否仍然可以分为 k 个森林，如果不可以，则由引理 2，不存在包含 $E\cup\{x\}$且由 E，x 和未判断过的边组成的合法解。由归纳假设，存在包含 E 且由 E 和除 x 以外的未判断的边组成的合法解。所以可以安全地忽略 x。如果可以加入 x，则由引理 2，存在包含 $E\cup\{x\}$且由未判断的边组成的合法解。最终剩下的未判断过的边集为空，即 E 包含一组合法解，E 自身就是合法解，算法输出 Yes。如果答案为 No，算法显然无法构造出合法解，会输出 No。

3.5 复赛

Diversity

时间限制：2000/1000 毫秒（Java/其他）

空间限制：65536/65536KB（Java/其他）

题目

给你一棵 n 个点的树，对于节点 i，你要给它标上一个[li，ri]的数，要求所有边两端节点上标的数字的差的绝对值的总和最大。

输入

第一行一个整数 T（$1 \leqslant T \leqslant 5$）表示数据组数。对于每组数据格式如下。

第一行一个正整数 n（$2 \leqslant n \leqslant 10^5$）。

接下来 $n-1$ 行，每行两个正整数 u，v（$1 \leqslant u$，$v \leqslant n$），表示一条边。

接下来 n 行，第 i 行两个正整数 l_i，r_i（$1 \leqslant l_i \leqslant r_i \leqslant 10^9$）。

输出

对于每组数据，一个整数表示答案。

输入样例

```
1
5
1 2
2 3
3 4
4 5
1 5
2 7
7 9
5 8
3 4
```

输出样例

```
16
```

关键思路

考虑对于一个点来说，要和周围的点的绝对值的和最大，那么一定取最大值或者最小值，也就是说对于点 u，取值一定是 l_u 或者 r_u。所以可以考虑 $dp_{u,0/1}$ 表示 u 这个点取 l_u 或者 r_u，子树的价值之和，做一个简单的树形 dp 即可。

时间复杂度 $O(n)$。

Transformation

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

给你一个二元组(a,b)，支持 AB 两种操作，分别是将其变成$(a,2b-a)$和$(2a-b,b)$。问能否通过大于等于零次操作将其变成(c,d)。

输入

第一行一个正整数 T（$T \leqslant 8 \times 10^4$）表示数据组数。

接下来 T 行，每行 4 个整数 a，b，c，d（$-10^{18} \leqslant a$，b，c，$d \leqslant 10^{18}$）。

输出

对于每组数据，如果有解，首先输出一行 Yes，然后输出一行由 AB 构成的字符串，表示一系列操作。如果有多解，输出长度最短的解，如果有多个长度最短的解，输出字典序最小的。如果无解，输出 No。

输入样例

```
3
0 1 2 3
0 1 0 1
0 1 -1 3
```

输出样例

```
No
Yes
Yes
BA
```

关键思路

考虑操作的逆操作，也就是将(l,r)变成$\left(l,\frac{l+r}{2}\right)$或者$\left(\frac{l+r}{2},r\right)$。所以就是用类似于线段树查询的方法就行了。注意特判一下区间长度为 0 之类的特殊情况。

时间复杂度$O(\log n)$。

Quasi Binary Search Tree

时间限制：4000/2000 毫秒(Java/其他)

空间限制：65536/65536KB(Java/其他)

题目

给你一个 n 个点的二叉树，每个点被标上了 1 到 n 中不同的标号。一棵树是伪二叉树当且仅当对于每个节点，它的左子树的所有节点的标号都小于它本身，且它的右子树的标号都大于它本身，或者它的左子树都大于它且它的右子树都小于它。

现在有一棵树，它的标号都被擦去了，问能否将其标上号使得这棵树是一个伪二叉树。如果有多解，输出字典序最小的解，即比较 1 号点的标号，再比较 2 号点的标号，依次类推。

输入

第一行一个正整数 $T(T\leqslant 7.5\times 10^4)$表示数据组数。

对于每组数据，第一行一个正整数 $n(n\leqslant 10^5,\ \sum n\leqslant 2.5\times 10^6)$。

接下来 n 每行两个整数 l_i,r_i 表示 i 号节点的左右儿子，如果不存在左儿子或者右儿子，那么对应的数字为 0，存在唯一的节点不是其他所有点的儿子。

输出

对于每组数据，令 u 点的标号为 pu，那么输出 $\sum_{u=1}^{n}(p_u\oplus u)*233^u$ 对 10^9+7 取模的值，这里的 $\oplus$ 为异或符号。

输入样例

```
1
5
0 0
0 0
0 4
2 0
1 3
```

输出样例

```
114911413
```

样例解释

实际上答案应该为(1,3,5,4,2)。

关键思路

容易将问题转化成,对于每个点,你都要确定是左儿子小还是右儿子小。

接着考虑贪心,我们考虑从根开始确定。如果根不是这个子树里面最小的节点,那么我们肯定选择将最小的节点所在的子树作为小的那边。否则根节点就是整个子树里面最小的节点,那么我们看两个儿子的 size 的大小,因为根节点标号就是小的子树的 size+1。如果两个节点的 size 不同,那么我们也能直接确定。否则对于根节点来说标号都一样,那么我们就看这两个子树里面的最小值哪个更小。

我们需要一开始预处理一个每个子树的最小值,处理完之后按上述方法贪心即可。

时间复杂度 $O(n)$。

Maximum or Sum

时间限制：2000/1000 毫秒(Java/其他)

空间限制：65536/65536KB(Java/其他)

题目

给你一个正整数数组 $c_1,c_2,\cdots,c_{n-1}$,求 n 元组$(a_1,a_2,\cdots,a_n)$的个数,满足 a_i 是正整数,且对于每个 $i(1\leqslant i\leqslant n-1)$都有 $\max(a_i,a_{i+1})=c_i$ 或者 $a_i+a_{i+1}=c_i$。

由于答案很大,输出对 10^9+7 取模的值。

输入

第一行一个整数 $T(1\leqslant T\leqslant 7)$表示数据组数。对于每组数据格式如下。

第一行一个正整数 $n(2\leqslant n\leqslant 3\times 10^3)$,接下来一行 $n-1$ 个正整数 $c_i(2\leqslant c_i\leqslant 10^9)$。

输出

对于每组数据,一个整数表示答案。

输入样例

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

输出样例

```
56
```

关键思路

考虑最暴力的 dp 就是 $dp_{i,j}$ 表示第 i 个数是 j 的方案数，可以通过前缀和优化到 $O(nm)$，但这个题里面 m 是 10^9，所以是无法通过的。

所以我们可以用类似于分段函数的方法维护这个 dp 值，就是把 dp 值看成若干个 (l_i,r_i,x_i) 的三元组相加，即 $dp_j=\sum[l_i\leqslant j\leqslant r_i]x_i$。

注意到 dp 值具有线性性，也就是说可以分别算每个三元组对下一个 dp 值的贡献，那么对于 (l_i,r_i,x_i)，如果下一位是 c（不妨设 $r_i\leqslant c$），那么如果是加法就贡献到 $(c-r_i,c-l_i,x_i)$ 这样一段 dp 值。如果是 max 操作，那么对于不是 c 的位置，下一个位置必须得是 c，否则就贡献到 $(1,c)$ 这么一段。

所以这么转移，只会在上一次的基础上增加 $(c,c,*)$ 和 $(1,c,*)$ 这样两个三元组，也就是说每个位置的 dp 数组的段数是 $O(n)$ 的，所以总段数是 $O(n^2)$ 的。

时间复杂度 $O(n^2)$。

Expected Remainder

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

在 $[a,b]$ 中均匀随机选取一个实数 x，在 $[c,d]$ 中均匀随机选取一个实数 y，请问 $x \bmod y$ 的期望，也就是 $x-\lfloor xy\rfloor y$ 是多少。

容易证明当 a,b,c,d 是正整数且区间长度大于 0 时，答案是有理数 r，那么请问 r 模 10^9+7 的值是多少。

对于一个有理数 $r=n/m$，其中 n,m 互质，那么我们定义它模素数 p 的值为，如果 $p\mid m$，那么为 0，否则为 $l(0\leqslant l\leqslant p)$，满足 $ml\equiv n(\bmod p)$。

输入

第一行一个正整数 $T(T\leqslant 2\times 10^5)$ 表示数据组数。

对于每组数据，第一行四个正整数 $a,b,c,d(1\leqslant a<b\leqslant 10^6,1\leqslant c<d\leqslant 10^6)$。

输出

对于每组数据，输出一行一个整数表示答案。

输入样例

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

输出样例

```
833333340
500000005
```

关键思路

$x \bmod y = x - \left\lfloor \frac{x}{y} \right\rfloor y$。所以 $IE[x \bmod y] = IE[x] - IE\left[\left\lfloor \frac{x}{y} \right\rfloor y\right]$。

前面部分比较好算，后面部分一个想法就是枚举$\left\lfloor \frac{x}{y} \right\rfloor$等于什么，然后算对应的期望与概率。但是因为它是一个矩形，所以直接算可能会分很多段，这么算起来会十分麻烦。

所以考虑 x 在 $[0,b]$ 上的随机，y 在 $[0,a]$ 之间随机这么一个问题，然后使用差分算出原问题的解。令 $k = \left\lfloor \frac{b}{a} \right\rfloor$，我们可以分考虑$\left\lfloor \frac{x}{y} \right\rfloor$小于，等于和大于 k 的情况分别讨论。通过一些可能比较简单但是烦琐的数学推导得出它可能等于

$$\frac{1}{6ab}\left(b^3\left(\frac{1}{k+1} - \frac{\pi^2}{6} + \sum_{i=1}^{k+1}\frac{1}{i^2}\right) + a^3 k + (b-ka)^2 a + \left(\frac{(k+2)b}{k+1} - ka\right)\left(a - \frac{b}{k+1}\right)b\right)$$

所以只需要预处理$\frac{1}{i^2}$的前缀和即可。

时间复杂度 $O(n)-O(1)$。

APSP on Cactus

时间限制：2000/5000 毫秒(Java/其他)

空间限制：262144/262144KB(Java/其他)

题目

一个图为仙人掌当且仅当每条边在至多一个简单环内且连通。给你一个仙人掌，每条边带权，对于每个点，求出这个点到其他所有点的最短路的和。

输入

第一行一个整数 $T(1 \leqslant T \leqslant 7)$表示数据组数。对于每组数据格式如下。

第一行两个正整数 $n,m(1 \leqslant n \leqslant 2 \times 10^5, 1 \leqslant m \leqslant 4 \times 10^5)$，表示点数与边数。

接下来 m 行，每行三个整数 $u,v,w(1 \leqslant u \neq v \leqslant n, 1 \leqslant w \leqslant 10^6)$，保证图中没有重边。

输出

对于每组数据，令 u 点的答案为 pu，那么输出 $\sum_{u=1}^{n}(p_u \oplus u) * 233^u$ 对 10^9+7 取模的值，这里的 $\oplus$ 为异或符号。

输入样例

```
1
5 5
5 1 2
5 2 8
5 4 9
3 2 7
4 1 4
```

输出样例

```
354181127
```

样例解释

实际上答案应该为(33,39,60,45,31)。

关键思路

首先对于一个环的情况,可以简单地使用双指针解决,即起点在环上移动时,那么对应的分界点也会跟着移动,可以使用前缀和的方法来算出答案。

对于仙人掌的情况也差不多,我们考虑对于每个起点最短路径树是什么要你管的。本质上来说,对于每个环,都会按分界点删去其中一条边,这些点都是沿着其余的边走出这个环。对于最短路径树来说,每条边的代价就是通过这条边的点的个数乘以这条边的长度,所以我们可以维护这个值。

我们考虑起点在图上移动的时候,如果起点经过的是树边,那么最短路径树不会变化,否则就只有这个点所在的环会发生变化,也就是前面说的对应的分界点也会跟着移动,通过预处理之后这样的东西可以在均摊 $O(1)$ 时间复杂度内算出来。

时间复杂度 $O(n)$。

3.6 决赛

深度学习计算图调度优化

题目

题目背景

- 深度学习模型训练可以被描述为一个有向无环的计算图的执行。
- 计算图中的节点叫做 Operator。
- 节点之间有方向的边代表 Operator 之间的依赖关系,这个依赖关系是由深度学习模型确定的,即一个 Operator 的输出(Tensor)是另一个 Operator 的输入,那么在计算图中就需要有一条射线连接两个 Operator。
- 一个 Operator 可以有多个 Tensor 输入或者多个 Tensor 输出。
- Operator 本身有计算代价,通常是计算 Operator 所需要的时间。
- Operator 可以运行在多种设备上。
  - 同一个设备下在任意时间点只能最多 N 个 Operator 并发执行,如果 N=1,意味着这个设备下执行 Operator 只能串行。
  - 执行在不同设备下的 Operator,可以并发执行。
- 计算图的起始节点可以有多个,结算图的终止节点也可以有多个,整个计算图的节点都执行完算作深度学习模型的一次迭代(Iteration)。

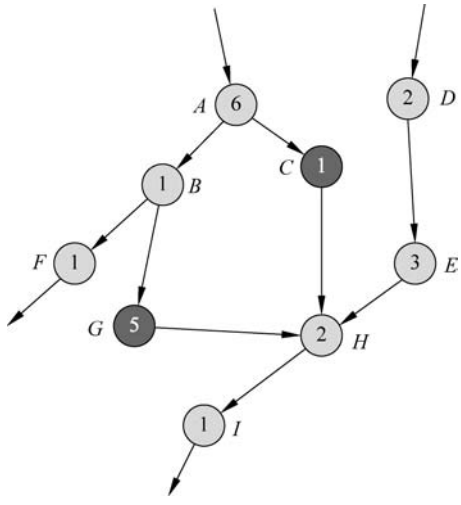

图 3-1 DAG

用一张 DAG 解释以上概念

图 3-1 中有两个颜色的节点,代表在不同设备上执行的 Operator,节点中的数字代表 Operator 的计算代价。假设每个设备只允许一个并发,那么可以给出若干个符合拓扑序的

执行顺序，例如 ABCFGDEHI，执行代价为 16，也可以是 DEACBFGHI，执行代价为 19。

题目

- 本题假定计算图中的 Operator 有两种类型，一种是计算 Operator，一种是通信 Operator。计算 Operator 在 AI 芯片上执行，通信 Operator 主要在网络通信设备上执行。
- 通信 Operator 最多有 3 个并发，计算 Operator 最多有 1 个并发。
- 计算图的总节点数不超过十万。

要求实现一个函数，给定任意计算图，返回一个符合计算图拓扑排序的 Operator 执行顺序，使得整体计算代价尽量小。

数据

- 计算图来源：基于一些计算图生成规则（规则不公开）随机生成的计算图。
- 比赛开始时，会提供 100 个根据赛题系统生成规则随机生成的计算图，以及 10 个真实深度学习模型转换得到的计算图，方便选手进行调试和观察规律。

赛制

三阶段综合排名制，第一阶段为热身阶段，主要目的是熟悉基本工具，不计分。第二阶段选手的平均计算代价占最终得分 10%，第三阶段选手的平均计算代价占最终得分的 90%。

- 第一阶段：选手能够根据比赛平台提供的几个计算图示例，通过程序或者肉眼观察的方式给出计算图的执行顺序，并提交到 Timeline 系统生成 Timeline 文件。Timeline 文件可以用于在 Chrome 浏览器进行可视化，代表计算图的执行时间线。Timeline 在整个比赛都可以提交，推荐选手在 30 分钟内完成。
- 第二阶段：计算图数目 100，由赛题系统生成，提交截止时间是比赛开始后 2.5 小时，到提交时间截止后，系统会运行选手最后一次提交的 CPP 文件作为选手的算法，10 分钟内给出榜单。
- 第三阶段：计算图数目 10000，由赛题系统生成，提交截止时间是比赛开始后 6 小时，随后半小时内系统会给出选手最终排名，比赛结束。

评分规则

- 评估系统会对选手提交的算法进行综合评估，在赛题系统随机生成的题目上进行打分。赛题系统生成题目的方式会参考真实深度学习模型的一些统计指标，通过系统规则生成，这部分规则对选手不公开。
- 得分计算方法：赛题系统会提供基线算法对赛题进行打分，基线算法来源于百度开源深度学习框架飞桨中的图调度引擎的基础算法。选手提交的算法，相对于基线算法的提升会作为选手的最终得分，得分计算公式如下：

$$s_i = \frac{baseline\_cost_i - user\_cost_i}{total\_cost_i}$$

$$s = 0.1 * \frac{1}{N_1}\sum_{p=1}^{N_1} s_p + 0.9 * \frac{1}{N_2}\sum_{p=1}^{N_2} s_q$$

- 其中，baseline_cost 为飞桨的基础调度算法得到计算代价，total_cost 为计算图所有节点的计算代价总和，user_cost 为选手的算法得到的计算代价。
- 关于计算代价：选手提交的程序会被编译为动态库与系统联编，并在集群环境中进行大规模分布式计算，并得到每张图的计算代价值。
- 运行超时的计算代价：在集群环境中，我们会为每次运行设置一个超时时间，长度为 5s。如果

选手提交的程序在同一张计算图下超时达到 3 次，我们认为选手提交的程序运行速度过慢，并给超时的计算图一个超时代价。超时代价为当前计算图所有计算节点计算代价总和。

- 内存超过系统限制的计算代价：系统限制选手的算法在赛题运行过程中，单个计算图执行不能超过 16GB 内存，如果内存超限，系统会给出以内存超限代价返回，内存超限代价为当前计算图所有计算节点计算代价总和。
- 其他系统类错误：运行过程中遇到程序异常退出，视为选手代码有 Bug，给出异常错误的计算代价，异常错误的计算代价为当前计算图所有计算节点计算代价总和。
- 排名计算方法：选手的算法打分会根据上述给定的公式进行计算，系统会根据选手的打分进行排名，评分越高排名越靠前。如果评分相同，则根据算法的执行时间进行排序。算法的执行时间会在选手得分相同的情况下，在公平环境下重新运行得到。

程序接口

- 编程语言采用 C++11 标准，编译器为 gcc 5.4，不允许使用 C++基础库以外的库，编译命令 g++ -O3 -lpthread。
- 计算图的表示格式：计算图通过一组表达式构成，例如在题目介绍部分展示的计算图是通过下面这组表达式构成。

```
B=A
C=A
G=B
F=B
E=D
H=C#E#G
I=H
```

比赛中，每个计算图相关的表达式以及节点的详细信息存放在单独的文件夹下。比如在 task_example 中，这些表达式存放在 expressions.txt 文件中，同时，每个节点的详细信息（比如权重，颜色等）存放在 nodes 文件夹下，节点所在文件的名字为 node_x.txt，其中 x 表示颜色信息。在 task_example 中，计算图中的节点有两种颜色，这两种颜色所对应的节点信息分别存放在 node_0.txt 和 node_1.txt 中。

```
├── task_example
│   ├── expressions.txt
│   └── nodes
│       ├── node_0.txt
│       └── node_1.txt
```

- 选手需要实现的接口：std::vector<std::string> execution_order(const std::string& node_path);
- execution_order 的输入为计算图表达式和节点所在的路径，输出为执行所有节点的拓扑序列，输出的形式为节点名称。

关键思路

整体思路：要明确计算代价是怎么计算的，然后通过启发式算法找到一个相对较优的解，然后再通过计算顺序的扰动来综合评价。难点在于理解题目本身，以及设计高性能的计算方法。

感谢 2019 程序设计大赛冠军张哲宇同学分享的关键思路。

题目大意

给一个有向无环图，结点分为通信结点和计算结点。每个结点需要消耗时间，其中通信结点可以三个结点并行，计算结点只能顺序处理。给出一个总消耗时间尽可能小的拓扑序。

简要分析

读题有点累，原题面里用的是“并发”，感觉不是很准确。

对图的特征一无所知的情况下，先尝试了一个简单的方法：按最长链从长到短处理。

接下来我下载了几组数据，找了一下瓶颈，发现主要时间都浪费在了密集的计算结点上，此时的通信结点空载。

所以我针对计算结点微调了一下，使得计算结点的运行时间分布更加均匀，使得通信结点更有效率，从而缩短了总时间。