NOI 2024 模拟赛 #2

时间：2024 年 3 月 19 日 07:20 ~ 11:50

题目名称	版本回溯	三进制计数	古代城市问题
题目类型	传统型	传统型	交互型
目录	`recall`	`ternary`	`city`
可执行文件名	`recall`	`ternary`	`city`
输入文件名	`recall.in`	`ternary.in`	无输入
输出文件名	`recall.out`	`ternary.out`	无输出
每个测试点时限	7.0 秒	2.0 秒	3.0 秒
内存限制	512 MiB	512 MiB	1024 MiB
测试点数目	20	25	7
测试点是否等分	是	是	否

提交源程序文件名

对于 C++ 语言	`recall.cpp`	`ternary.cpp`	`city.cpp`

编译选项

对于 C++ 语言	`-O2 -std=c++14 -static`

注意事项（请仔细阅读）

1. 文件名（程序名和输入输出文件名）必须使用英文小写。
2. 若无特殊说明，结果的比较方式为全文比较（过滤行末空格及文末回车）。
3. 选手提交的程序源文件必须不大于 100KB。
4. 禁止在源代码中改变编译器参数（如使用 `#pragma` 命令），禁止使用系统结构相关指令（如内联汇编）和其他可能造成不公平的方法。
5. 祝大家好运。

版本回溯 (recall)

题目描述

小 ζ 有一个版本控制系统，他决定利用这个系统整蛊小 ω。

一个文档可以被描述为一个长度为 n 的整数序列 a，小 ζ 会操纵这个文档，每次会给定 l, r, x，使得 $\forall i, i \in [l, r], a_i \leftarrow a_i + x$。

当小 ζ 要整蛊小 ω 时，便会将现在的文档下发给小 ω ，小 ω 需要完成一个任务，它需要根据小 ζ 给他的参数 l, r, x，找到满足 $i \in [l, r], a_i < x$ 的所有 a_i 的**所有历史版本中的最大值**（即初值和每次修改后的值中最大的一个）的最大值。如果小 ω 没有成功回答出来，那么他会被小 ζ 奖励冷水洗头。

小 ω 当然不会这种问题啦！于是他找到了赛场上最聪明的你，希望你能帮助他对付邪恶的小 ζ。

输入格式

从 `recall.in` 读入数据。

第一行两个正整数 sid, n, m，表示测试点编号，序列长度和操作次数。对于样例输入输出，sid 表示它们满足哪一组测试点的特殊性质。

接下来一行，输入 n 个整数，表示初始文档。

接下来 m 行，每行四个整数 op, l, r, x:

- 若 $op = 1$，代表将 $\forall i, i \in [l, r], a_i \leftarrow a_i + x$;
- 若 $op = 2$，代表查询此时满足 $i \in [l, r], a_i < x$ 的 a_i 的历史最大值的最大值。

输出格式

输出到文件 `recall.out` 中。

对于每次 $op = 2$ 的操作，输出一个整数，表示答案。

如果没有数满足小 ζ 的条件，那么输出 `-inf` 。

输入输出样例

样例 1 输入

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

样例 1 输出

```
4
```

样例 1 解释

初始时，$a = [1, 2, 3, 4, 5]$。

第一次操作后，$a = [0, 1, 2, 3, 4]$。

询问时，$i = 1, 2, 3, 4$ 满足条件，此时 a_4 的历史最大值最大，为 4。

样例 2 输入

```
1 5 10
-53 36 41 55 77
1 1 1 10
2 4 4 79
2 3 3 -28
1 1 5 -29
1 1 3 27
2 2 5 88
2 2 3 24
1 2 2 10
1 1 3 -9
1 2 5 -10
```

样例 2 输出

```
55
-inf
77
-inf
```

样例 3 输入 / 输出

见附件中的 `recall3.in/ans` 。

该样例 $sid = 1$，它满足第 1 个测试点的限制条件。

样例 4 输入 / 输出

见附件中的 `recall4.in/ans` 。

该样例 $sid = 4$，它满足第 4 个测试点的限制条件。

样例 5 输入 / 输出

见附件中的 `recall5.in/ans` 。

该样例 $sid = 6$，它满足第 6 个测试点的限制条件。

样例 6 输入 / 输出

见附件中的 `recall6.in/ans` 。

该样例 $sid = 8$，它满足第 8 个测试点的限制条件。

样例 7 输入 / 输出

见附件中的 `recall7.in/ans` 。

该样例 $sid = 10$，它满足第 10 个测试点的限制条件。

样例 8 输入 / 输出

见附件中的 `recall8.in/ans` 。

该样例 $sid = 14$，它满足第 14 个测试点的限制条件。

样例 9 输入 / 输出

见附件中的 `recall9.in/ans` 。

该样例 $sid = 20$，它满足第 20 个测试点的限制条件。

数据范围

对于所有测试数据，保证：$1 \le n, m \le 2 \times 10^5$，$|a_i| \le 10^9$，$1 \le l \le r \le n$，对于 $op = 1$ 满足 $|x| \le 10^9$，对于 $op = 2$ 满足 $|x| \le 10^{18}$。

测试点编号	$n \le$	$m \le$	特殊性质
$1 \sim 3$	5000	5000	
$4 \sim 5$	2×10^5	2×10^5	AB
$6 \sim 7$	2×10^5	2×10^5	A
$8 \sim 9$	2×10^5	2×10^5	B
$10 \sim 13$	2×10^5	2×10^5	C
$14 \sim 15$	2×10^5	2×10^5	D
$16 \sim 17$	2×10^5	2×10^5	E
$18 \sim 19$	2×10^5	2×10^5	F
20	2×10^5	2×10^5	

特殊性质 A：保证 $a_i, x \ge 0$。

特殊性质 B：保证对于所有的 $op = 2$，都有 $x = 10^{18}$。

特殊性质 C：$l = 1, r = n$。

特殊性质 D：$op = 2$。

特殊性质 E：对于 $op = 1$，$l = r$。

特殊性质 F：对于 $op = 2$，$l = r$。

三进制计数 (ternary)

题目描述

小 ζ 发明了基于三进制工作的计算机，据理论推测，该计算机能够存储 n 个比特，从而存储 3^n 个状态。

三进制计算机被认为是一项非常先进的高科技，然而，邪恶的外星人绝对不会允许人类拥有这种设备。于是歌者发布了 m 个诅咒，每个诅咒可以用 (l, r, x) 表示，代表计算机在存储时，必须满足区间 $[l, r]$ 中恰好有 x 种字符，否则计算机就会被夺取控制。

小 ζ 想请你帮忙计算有多少个合法的状态。如果你能正确回答这个问题，他将把这台三进制计算机送给你作为报酬。答案对 $10^9 + 7$ 取模。

输入格式

从 `ternary.in` 读入数据。

本题有多组输入数据。

第一行一个正整数 T，表示数据组数。

对于每组数据，第一行两个正整数 n, m，表示计算机可以存储的比特数和诅咒数。

接下来 m 行，每行三个正整数 l, r, x，表示一条诅咒。

输出格式

输出到文件 `ternary.out` 中。

共 T 行，每行一个整数，表示答案。

输入输出样例

样例 1 输入

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

样例 1 输出

```
3
9
27
18
```

样例 1 解释

对于第 4 组测试用例，所有合法的三进制串为：21000，12000，20100，02100，10200，01200，21011，12011，20111，02111，10211，01211，21022，12022，20122，02122，10222，01222。

样例 2 输入 / 输出

见附件中的 `ternary2.in/ans` 。

该样例满足测试点 2 的性质。

样例 3 输入 / 输出

见附件中的 `ternary3.in/ans` 。

该样例满足测试点 4 的性质。

样例 4 输入 / 输出

见附件中的 `ternary4.in/ans` 。

该样例满足测试点 10 的性质。

样例 5 输入 / 输出

见附件中的 `ternary5.in/ans` 。

该样例满足测试点 14 的性质。

样例 6 输入 / 输出

见附件中的 `ternary6.in/ans` 。

该样例满足测试点 19 的性质。

样例 7 输入 / 输出

见附件中的 `ternary7.in/ans` 。

该样例满足测试点 22 的性质。

数据范围

对于所有测试数据，保证 $1 \le n \le 5000$，$1 \le \sum n \le 1.5 \times 10^4$，$0 \le m \le \min\{n(n-1)/2, 10^6\}$，$1 \le l \le r \le n$，$1 \le x \le 3$。

测试点编号	$n \le$	$\sum n \le$	$m \le$	特殊性质
1	13	13		
$2 \sim 3$	100	100		
$4 \sim 8$	500	1500		
9	2000	2000	4	A
$10 \sim 12$	2000	2000		A
13	5000	5000		B
$14 \sim 16$	2000	2000	4	C
17	2000	2000		C
18	5000	5000	1	
$19 \sim 21$	5000	5000	2	
$22 \sim 25$	5000	1.5×10^4		

特殊性质 A：$x = r - l + 1$。

特殊性质 B：$x = 1$。

特殊性质 C：$x = 3$。

古代城市问题 (city)

题目背景

曾经，传奇英雄达拉崩吧斑得贝迪卜多比鲁翁疯狂爱上了公主米娅莫拉苏娜丹妮谢莉红，但是公主被巨龙昆图库塔卡提考特苏瓦西拉松抓走了。国王答应达拉崩吧，如果他能打败巨龙，就把公主许配给他。

题目描述

巨龙掌握了一座有 n 个城市的邪恶王国，标记为 $0 \sim n-1$，这个王国有 m 条道路，标记为 $0 \sim m-1$。每条道路连接两个不同的城市，每一对城市至多会被一条道路连接。有些道路是御道，专用于巨龙行驶，但这是保密的。达拉崩吧的任务是找出哪些道路是御道，以便他伏击巨龙。

达拉崩吧有一张包括所有城市和所有道路的地图，他不知道哪些道路是御道。但是他可以向知更鸟询问来获得一些信息。

知更鸟首先告诉达拉崩吧，为了方便巨龙统治这个王国，所有御道的集合是一个黄金集合。一个道路的集合是黄金集合，当且仅当：

- 它恰好包含 $n-1$ 条道路；
- 对于每一对城市，仅沿着这个集合中的道路即可从其中一个城市抵达另外一个城市。

然后达拉崩吧可以向知更鸟询问至多 q 个问题，对于每个问题：达拉崩吧会选择一个道路的集合，然后知更鸟会告诉达拉崩吧在所选择集合中有多少条道路是御道。

根据巨龙的巡视力度，知更鸟会限制你询问的集合是否必须为黄金集合（**如果没有限制，那么你可以询问可重集**）。

请帮助达拉崩吧找出所有的御道。

实现细节

你的程序开头应该包含 `#include "city.h"`。

你无需实现 `main()` 函数，你只需要实现下面的函数：

```
std::vector<int> find_roads(int n, bool gold, std::vector<int> u,
std::vector<int> v)
```

其中 n 表示城市的数量，$gold$ 代表你是否只能询问黄金集合，u, v 均为长度为 m 的数据，u_i 和 v_i 表示了第 i 条道路。

该函数需要返回一个长度为 $n-1$ 的数组，其中包括了所有御道的标号（可以以任意的顺序给出）。

你的程序至多只能调用评测工具中的如下函数 q 次：

```
int count_common_roads(std::vector<int> r)
```

其中 r 是任意长度的数组，表示一个集合中的道路标号（可以以任意的顺序给出），当 $gold = 1$ 时，你给出的 r 必须是黄金集合，否则会返回 `Wrong Answer`。

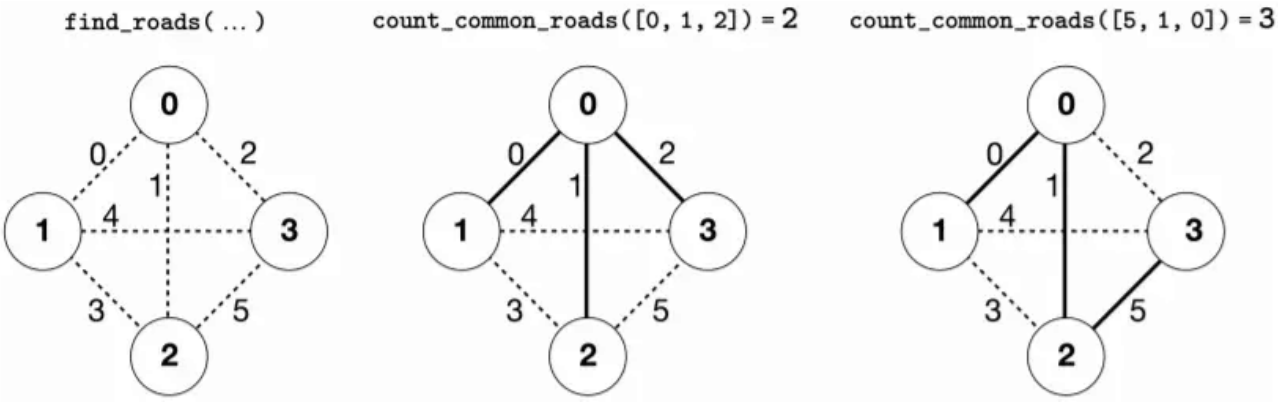
r 中的边必须是合法的，否则也会返回 `Wrong Answer`。

保证当 $gold = 0$ 时，交互库回答一次询问的时间复杂度为 $O(\text{siz}(r))$；$gold = 1$ 时，交互库回答一次询问的时间复杂度为 $O(n \log n)$。

交互范例

该范例是附件中的 `city1.in` 的参考交互过程。

```
find_roads(4, 1, [0, 0, 0, 1, 1, 2], [1, 2, 3, 2, 3, 3])
```



这个例子中有 4 个城市和 6 条道路，我们将连接城市 a 和 b 的道路表示为 (a, b)。假设御道是标号为 0，1 和 5 的道路，即 $(0, 1)$，$(0, 2)$ 和 $(2, 3)$。这样的话：

`count_common_roads([0, 1, 2])` 返回 2。该询问涉及到标号为 $0, 1$ 和 2 的道路，即 $(0, 1)$，$(0, 2)$ 和 $(0, 3)$。其中有两条道路是御道。

`count_common_roads([5, 1, 0])` 返回 3。该询问涉及到所有的御道。

函数 `find_roads` 需要返回 $[5, 1, 0]$ 或任意其他包含这三个元素且长度为 3 的数组。

注意，当 $gold = 1$ 时，下面列出的调用是不允许的，但是 $gold = 0$ 时是允许的：

- `count_common_roads([0, 1])`：这里 r 的长度不是 3。
- `count_common_roads([0, 1, 3])`：这里 r 不是一个黄金集合，因为无法仅沿道路 $(0, 1)$，$(0, 2)$，$(1, 2)$ 就从城市 0 走到城市 3。

此外，$gold = 0$ 允许询问可重集。

输入输出样例

下发文件提供了 `grader.cpp`、`city.h` 和 `city.cpp`，你可以参考 `city.cpp` 来实现你的程序，编译时只需要调用 `g++ grader.cpp city.cpp -o city -O2 -std=c++14` 即可，然后执行 `./city` 来运行你的程序。最终测试时的交互库与下发交互库有所不同，因此你的实现不应该依赖于交互库。

评测工具示例将读入下述格式的输入数据：

- 第一行：$n, m, gold$；
- 第 $2+i$ 行（$0 \le i \le m-1$）：u_i, v_i；
- 第 $2+m$ 行：$s_0, s_1, \cdots, s_{n-2}$，表示答案。

下发的样例提供了 `city1~8.in`，它们均满足 $gold = 0$，你可以自行修改它们来获得 $gold = 1$ 的样例。其中 `city7.in` 满足特殊性质 A。

如果你实现的 `find_roads()` 使用不超过 30000 次询问正确地返回了御道的集合，那么评测工具将输出 `Accepted, use x queries.`，表示你使用了 x 次询问。否则会输出 `Wrong Answer`。

数据范围

对于所有测试数据，保证 $2 \le n \le 500$，$n-1 \le m \le n(n-1)/2$，$0 \le u_i, v_i \le n-1$，$u_i, \ne v_i$，每对城市之间至多连有一条道路，经由这些道路，可以在任意一对城市之间来往，$8000 \le q \le 30000$。

子任务编号	$n \le$	$q =$	$gold =$	特殊性质	分数
1	7	30000	1		12
2	50	30000	1		16
3	240	30000	0		9
4	240	30000	1		17
5	500	12000	1	A	13
6	500	8000	0		18
7	500	8000	1		15

特殊性质 A：任意两个城市之间都有道路相连。