2020~2021 年高中信息学多校联合训练

2021 年省选模拟赛

寒冬送温暖赛

时间：2020 年 12 月 26 日　8:00～12:30

题目名称	作业题	社会实践	旅游
题目类型	传统型	传统型	交互型
目录	count	practice	travel
可执行文件名	count	practice	travel
输入文件名	count.in	practice.in	N/A
输出文件名	count.out	practice.out	N/A
每个测试点时限	1.0 秒	2.0 秒	3.0 秒
内存限制	256 MB	512 MB	256 MB
子任务数目	4	5	3
测试点是否等分	否	否	否

提交源程序文件名

对于 C++　语言	count.cpp	practice.cpp	travel.cpp

编译选项

对于 C++　语言	-lm -O2 -std=c++11

注意事项：

1. 测评时栈空间与内存限制相同。
2. 时间限制保证在标程的两倍以上，具体时限可随实际测评环境调整。
3. 若无特殊说明，输入文件的同一行内的多个整数、浮点数、字符串等均使用一个空格进行分隔。
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. 题目比较简单，请独立完成。

作业题(count)

【题目描述】

小 H 刚刚在离散数学科学习了生成树的知识：一个无向图 $G=(V,E)$ 的生成树 T 为边集 E 的一个大小为 $|V|-1$ 的子集，且保证 T 的生成子图在 G 中连通。

小 H 在做今天的作业是被这样一道题目难住了：

给定一个 n 个顶点 m 条边（点和边都从 1 开始编号）的无向图 G，保证图中无自环（**即可能有重边**）。每一条边有一个非负整数的边权 w_i，对于一棵 G 的生成树 T，定义 T 的价值为：T 所包含的边的边权的异或和，即：

$$val(T)=w_{e_1}\oplus w_{e_2}\oplus\ldots\oplus w_{e_{n-1}}$$

其中 $e_1,e_2,\ldots,e_{n-1}$ 为 T 包含的边的编号，$\oplus$ 为异或操作。

小 H 需要求出 G 的所有生成树 T 的价值之和，她做了很久也没做出来，请你帮帮她。由于答案可能很大，你只需要给出答案对 998244343 取模后的结果。

【输入格式】

从文件 count.in 中读入数据。

第一行两个整数 n,m，表示 G 中点的个数和边的个数。

接下来 m 行，每行三个数 x_i,y_i,w_i，描述 G 中的一条连接 x_i 号点和 y_i 号点的权值为 w_i 的边。

【输出格式】

输出到文件 count.out 中。

输出一行，表示答案对 998244353 取模后的结果。

【样例 1 输入输出】

count.in	count.out
3 3 1 2 4 2 3 15 3 1 1	30

【样例 1 解释】

G 共有三棵生成树。

$T_1=\{(1,2),(2,3)\}$，价值为 $4\oplus 15=11$。

$T_2=\{(1,2),(1,3)\}$，价值为 $4\oplus 1=5$。

$T_3=\{(1,3),(2,3)\}$，价值为 $1\oplus 15=14$。

总和为 30 。

【样例 2】

见选手目录下的 count/count*.in 和 count/count*.ans。

【数据范围和提示】

本题采用捆绑测试。
对于所有数据，满足 $1 \le n \le 60, 1 \le m \le 10^4, 1 \le x_i, y_i \le n, 0 \le w_i < 256$。
子任务见下表：

子任务编号	m	w_i	特殊性质	分值
1	≤ 20	< 256	无	10
2	$\le 10^4$		G 是一棵仙人掌	25
3		$= 1$	无	25
4		< 256		40

子任务 2 保证 G 中没有重边。

社会实践(practice)

【题目描述】

小 H 参加了学校组织的社会实践活动。在活动中，小 H 来到了某个游戏制作公司进行参观学习。

在参观过程中，导游展示了该公司某一项非常出名的作品：汉诺塔游戏。因为与传统的游戏有些不同，所以导游重新介绍了这款游戏。

与传统游戏相似的，该游戏同样是由三个柱子和 n 个**大小不同**的圆盘构成的，但是一开始这 n 个圆盘散落在三个柱子上，且每个柱子上的圆盘**从底到顶**大小递减。

玩家需要做的事则是移动这些圆盘。每一次，玩家可以移动任意一根柱子最顶上的那个圆盘，并移动到其他柱子的最顶上。在移动后，玩家需要保证每个柱子上的圆盘**从底到顶**大小递减。

玩家的最终目标是将这些散落的圆盘全部移动到**某一根**柱子上。

在介绍完后，便进入了游戏体验环节。细心的小 H 发现，全班同学没有任何两个人的游戏初始局面相同，而且每次重开的游戏初始局面也不同。在向导游询问的过程中，小 H 了解到了游戏更深层的制作方式。

为了保证游戏数量足够多且难度不小，设计人员会在后台存放一个长度为 n 的序列 s， s 中每一个数都是 [1,3] 中的整数。对于一个初始局面，可以表示为一段区间 $[l,r]$，该局面的 $n=r-l+1$，并且 1 号圆盘在 s_l 号柱子上， 2 号圆盘在 s_{l+1} 号柱子上， i 号圆盘在 s_{l+i-1} 号柱子上。圆盘的大小**从小到大**依次为 1 号圆盘、2 号圆盘、……、n 号圆盘。不难发现，按照上述方法，一段区间 $[l,r]$ 可以唯一表示一个初始局面。

社会实践结束后，就到了书写报告的时间了，小 H 想让自己的得分高一点。因此，学过竞赛的她打算按照导游告诉她的方法实现这款游戏。

先给定一个长度为 n 的序列 s，其中 $s_i \in [1,3]$。

接下来小 H 要进行 m 个操作，每个操作为以下两种之一：

- 修改操作：给定 x,v，将 s_x 修改成 v。
- 询问操作：给定 l,r，求由区间 $[l,r]$ 构成的游戏的**最优解**的移动次数。在这里，最优解定义为移动次数最少的解法。

为了让自己的实践报告得分不输给小 H，你决定也写一个这样的程序。但是你并不如小 H 优秀，所以你只需要输出移动次数对 998244353 取模后的结果。

【输入格式】

从文件 practice.in 中读入数据。

第一行两个整数 n,m，序列长度和操作次数。

第二行 n 个整数，描述序列 s。

接下来 m 行，每行三个整数，第一个整数为 op。

若 $op=1$，则表示修改操作，接下来输入两个整数 x,v。

若 $op=2$，则表示询问操作，接下来输入两个整数 l,r。

【输出格式】

输出到文件 practice.out 中。
对于每一个询问操作，输出答案对 998244353 取模后的结果。

【样例 1 输入输出】

practice.in	practice.out
10 10 2 2 2 2 1 2 3 3 3 2 1 9 3 2 2 7 1 3 2 2 1 1 2 6 10 2 6 7 1 6 1 1 1 2 1 10 1 2 1 10	23 0 15 1 496

【样例 2】

见选手目录下的 practice/practice*.in 和 practice/practice*.ans。

【数据范围和提示】

本题采用捆绑测试。
对于所有数据，满足 $1 \le n, m \le 3 \times 10^5, 1 \le s_i \le 3$。
对于每一个询问，满足 $op \in \{1,2\}, 1 \le l \le r \le n, 1 \le x \le n, 1 \le v \le 3$。
子任务见下表：

子任务编号	n	特殊性质	分值
1	≤ 3	无	5
2	≤ 12	$m \le 10$	20
3	≤ 2000	无	35
4	$\le 3 \times 10^5$	$l = 1, r = n$	15
5		无	25

旅行(travel)

这是一道交互题。

【题目描述】

抛开繁重的学业，我们一起去没有到过的地方旅游吧。这样想着，小 H 背上了行囊。

小 H 来到了一个特殊的城市，这个城市的道路连接成一棵树。换句话说，这个城市有 n 个分叉路口或关键建筑， $n-1$ 条道路连接着这 n 个点。

这座城市对交通有着非常严格的管制，其中一条规则就是**不能调头**，调头（包括在路口调头）就需要通过特殊方法将车移动到对面车道，这是一个非常麻烦的过程，这里的居民都会尽可能在出行过程中避免调头操作。

在如此奇怪的规则下，该城市开发了适合自己的导航系统。该导航系统有以下两个功能：

- **终点可能性判断**：你可以给导航系统两个点 x,y 和一个非空整数集合 S，集合 S 中的数都属于 $1\sim n$。这相当于你在询问导航，如果从 x 发车，在经过 y 且不调头的情况下，是否可能到达集合 S 中的城市。
- **终点可能性计算**：你可以给导航系统两个点 x,y。这相当于你在询问导航，如果从 x 发车，在经过 y 且不调头的情况下，可能到达的节点数量。

对于终点可能性判断，导航系统可以这样回答你：

① 所有城市都可能到达

② 所有城市都不可能到达

③ 有些城市可能到达

注意，在这两个询问中，每个城市是否可能到达只和 x,y 有关。为了避免歧义，我们给出导航系统的形式化回答：

- **终点可能性判断**：返回 S 中的点 全部在/全部不在/部分在 以 x 为根的 y 的子树中。
- **终点可能性计算**：返回以 x 为根的 y 的子树中的节点个数。

然而，该导航系统刚开发出来不久，所以除了这两个询问功能外什么都没有，即不能显示地图。

刚来到这个城市的小 H 想要快速知道这个城市的地图，但是她手上只有这个导航系统。她打算使用尽量少的次数询问出该城市的地图。

【实现细节】

在本题中，你不需要，也不应该实现主函数，你只需要实现函数 `solve(n,limA,limB)`，这个函数中 n 表示树的节点个数，$limA,limB$ 的意义可以见**【数据范围】**。

你可以调用以下两个询问函数：

- `check(x,y,h)`

  1.该函数为终点可能性判断。在这个函数中，x,y 表示城市中的节点，h 是一个节点集合，在 C++中，使用 `vector` 实现。

  2.这个函数将返回三种值 1,2,3，1 表示 h 中的点都可以被到达，2 表示

h 中的点都不能被到达， 3 表示 h 中的点部分能被到达，部分不能被到达。

3.在调用函数时，你不需要保证 $x \neq y$ ，也不需要保证 h 中的元素没有重复，但你需要保证 $1 \leq x, y \leq n, |h| \leq n$ 。

- `find(x,y)`

1.该函数为终点可能性计算。在这个函数中， x, y 表示城市中的节点。

2.这个函数将会返回一个整数，表示可能到达的节点数量。

3.在调用函数时，你不需要保证 $x \neq y$，但你需要保证 $1 \leq x, y \leq n$ 。

除了以上两个函数，你还可以调用一个提交答案的函数：

- `report(x,y)`

1. 调用此函数的意思是你找到了一条边 x, y 。
2. 你需要调用此函数刚好 $n-1$ 次，报告的边需要没有错误的边，没有重复报告的边。

如果你对实现方式还有任何疑问，那么可以参考选手目录下的 `travel_sample.cpp` ，这应该能解决大部分疑问。

【调试方法】

在选手目录下，还有两个文件 `grader.cpp` 和 `travel.h` ，你需要将你实现的程序命名为 `travel.cpp`，和这两个文件放在同一个文件目录下。

你可以使用该编译命令得到可执行文件：`g++ grader.cpp travel.cpp –o travel –O2 –std=c++11 -lm`

在可执行文件中你可以这样输入：

第一行三个整数 $n, limA, limB$ ，表示节点数，和两个函数的限制。

接下来 $n-1$ 行，每行两个整数 x, y ，描述树的一条边。

如果你在限制次数内正常报告所有边并正常结束程序，那么可执行文件会返回正确信息和函数调用的次数。否则，可执行文件会返回错误信息。

这样你可以调试你的程序。

注意：下发的交互库和最终测试的交互库可能有所不同，请不要依赖交互库写题。

【样例输入输出】

travel.in	travel.out
4 10000 10000 1 2 2 3 2 4	Accept! Ok, you ask 3 times problem A and 1 times problem B.

【样例解释】

以下是一个可能的交互过程：

序号	操作	函数返回值
0	solve(4,10000,10000)	N/A
1	find(1,2)	3
2	check(1,2,{2,3})	1
3	check(1,2,{1,3})	3
4	check(1,2,{1})	2
5	report(1,2)	N/A
6	report(2,3)	N/A
7	report(2,4)	N/A

【数据范围和提示】

本题共有 3 个子任务，每个子任务中包含多个测试点，子任务得分为该子任务中所有测试点的**最低得分**。

子任务不一定按照难度排序。

在本地调试时，$limA$ 表示 check 函数的限制使用次数，$limB$ 表示 find 函数的限制使用次数。

- **子任务 1（30 分）：**

  在该子任务中，$n = 10^3, limA = 3 \times 10^4, limB = 0$ 。

  该子任务中每个测试点只有 0 分和 30 分两种分值。

  当你使用 check 函数次数严格小等于 $limA$，使用 find 函数次数严格小等于 $limB$，并且正确报告所有边才得 30 分，否则得 0 分。

  注意：该子任务中你不允许使用 find 函数。

- **子任务 2（30 分）：**

  在该子任务中，$n = 10^3, limA = 3 \times 10^4, limB = 2 \times 10^4$ 。

  在该子任务中，每个测试点的得分以函数的方式计算。

  如果你使用 check 函数次数大于 $limA$ 或使用 find 函数次数大于 $limB$ 或没有正确报告所有边，那么该测试点得 0 分。

  否则，记你使用了 x 次 check 函数，那么你的得分为：

$$\frac{30}{\log_2\left(\max\left(0,\frac{x-1.6\times10^4}{2800}\right)+1\right)+1}$$

  即你需要将 check 函数的使用次数控制在 1.6×10^4 内才能得满分。

- **子任务 3（40 分）：**

  在该子任务中，$n = 10^3, limA = 2.8 \times 10^4, limB = 999$。

  在该子任务中，每个测试点的得分以函数的方式计算。

  如果你使用 check 函数次数大于 $limA$ 或使用 find 函数次数大于 $limB$ 或没有正确报告所有边，那么该测试点得 0 分。

  否则，记一个得分参数 a，记你使用了 x 次 check 函数，那么：

$$a=\begin{cases}1, & x\le 9100\\ 1.9-\dfrac{x}{10^4}, & 9100<x\le 1.4\times 10^4\\ 1-\dfrac{x}{2.8\times 10^4}, & x>1.4\times 10^4\end{cases}$$

此时，你该测试点的得分为 $40a$ 。

即你需要将 `check` 函数的使用次数控制在 9100 内才能得满分。