NOI2018 湖南省组队选拔赛

第二试试题

一．题目概况

题目名称	游戏	排列	道路
目录	game	perm	road
可执行文件名	game	perm	road
输入文件名	game.in	perm.in	road.in
输出文件名	game.out	perm.out	road.out
每个测试点时限	1 秒	1 秒	1 秒
测试点数目	10	10	20
每个测试点分值	10	10	5
结果比较方式	字符串比较，多行单字符串比较	整数比较，单行单个数字比较	整数比较，单行单个数字比较
题目类型	传统	传统	传统
内存上限	512M	256M	512M

二．提交源程序需加后缀

对于 C++语言	game.cpp	perm.cpp	road.cpp
对于 C 语言	game.c	perm.c	road.c
对于 Pascal 语言	game.pas	perm.pas	road.pas

三．编译命令

对于 C++语言	g++ -o game game.cpp –lm –O2	g++ -o perm perm.cpp -lm	g++ -o road road.cpp -lm
对于 C 语言	gcc –o game game.c –lm –O2	gcc -o perm perm.cpp -lm	gcc -o road road.cpp -lm
对于 Pascal 语言	fpc game.pas –O2	fpc perm.pas	fpc road.pas

注意事项：

（1） 选手必须在自己的工作目录下操作，严禁在其他目录下工作。目录结构请遵从 NOI 规范，即需要在工作目录下再**为每个题目建相应子目录，子目录名为对应题目的英文名**。

（2） 选手最后提交的源程序（.pas 或.c 或.cpp）**必须在自己的工作目录里对应子目录下，**对于缺少文件者，不予测试，该题计零分**。**

（3） 子目录名、源程序文件名和输入输出文件名**必须使用英文小写**。

（4） 特别提醒：评测在 NOI Linux 下进行。

第 1 题：游戏(game)，运行时限 2s，内存上限 512M，100 分。

【问题描述】

一次小 G 和小 H 在玩寻宝游戏，有n个房间排成一列，编号为 1,2,…,n，相邻房间之间都有1道门。其中一部分门上有锁（因此需要对应的钥匙才能开门），其余的门都能直接打开。现在小 G 告诉了小 H 每把锁的钥匙在哪个房间里（**每把锁有且只有一把钥匙**），并作出p次指示：第i次让小 H 从第S_i个房间出发，去第T_i个房间寻宝。但是小 G 有时会故意在指令里放入死路，而小 H 也不想浪费多余的体力去尝试，于是想事先调查清楚每次的指令是否存在一条通路。

你是否能为小 H 作出解答呢？

【程序文件名】

源程序文件名为$game.cpp/c/pas$。

【输入格式】

输入文件名为$game.in$。

第一行三个整数n,m,p，代表共有n个房间，m道门上了锁，以及p个询问。

接下来m行每行有两个整数x,y，代表第x到第$x+1$个房间的门上有把锁，并且这把锁的钥匙被放在了第y个房间里。输入保证x不重复。

接下来p行。其中第i行是两个整数S_i,T_i，代表一次询问。

【输出格式】

输出文件名为$game.out$。

输出m行，每行一个大写的"YES"或"NO"分别代表能或不能到达。（输出不包含引号）

【输入输出样例 1】

game.in	game.out
5 4 5 1 3 2 2 3 1 4 4 2 5 3 5 4 5 2 1 3 1	YES NO YES YES NO

【样例解释 1】

第一个询问$S=2$、$T=5$的一条可行路线是：$2\to3\to2\to1\to2\to3\to4\to5$。

【输入输出样例 2】（此组样例满足特性：$y \le x$恒成立）

game.in	game.out
7 5 4 2 2 3 3 4 2 5 3 6 6 2 1 3 4 3 7 4 5	YES YES NO NO

【样例解释 2】

第一个询问2和1房间之间没有锁所以为一条通路。

【数据范围】

测试点编号	n	m	其他特性
1	≤ 1000	≤ 1000	无
2			
3	$\le 10^5$	$\le 10^5$	$y \le x$恒成立
4			
5			无
6			
7	$\le 10^6$	$\le 10^6$	$y \le x$恒成立
8			
9			无
10			

对于所有数据，保证$1 \le n, p \le 10^6$，$0 \le m < n$，$1 \le x, y, S_i, T_i < n$，保证x不重复。

由于本题输入文件较大，建议在程序中使用读入优化。

【编译命令】

对于 c++语言： g++ -o game game.cpp –lm –O2

对于 c 语言： gcc -o game game.c –lm –O2

对于 pascal 语言： fpc game.pas –O2

提示：本题将开启 02 优化指令。

第 2 题：排列(perm)，运行时限 1s，内存上限 256M，100 分

【问题描述】

给定 n 个整数 $a_1, a_2, \ldots, a_n$，$0 \le a_i \le n$，以及 n 个整数 $w_1, w_2, \ldots, w_n$。称 $a_1, a_2, \ldots, a_n$ 的一个排列 $a_{p[1]}, a_{p[2]}, \ldots, a_{p[n]}$ 为 $a_1, a_2, \ldots, a_n$ 的一个合法排列，当且仅当该排列满足：对于任意的 k 和任意的 j，如果 j<=k，那么 $a_{p[j]}$ 不等于 p[k]。（换句话说就是：对于任意的 k 和任意的 j，如果 p[k]等于 $a_{p[j]}$，那么 k<j。）定义这个合法排列的权值为 $w_{p[1]} + 2w_{p[2]} + \ldots + nw_{p[n]}$。你需要求出在所有合法排列中的最大权值。如果不存在合法排列，输出-1。

样例解释中给出了合法排列和非法排列的实例。

【程序文件名】

源程序文件名为 perm.cpp/c/pas。

【输入格式】

输入文件名为perm.in。

第一行一个整数 n。

接下来一行 n 个整数，表示 $a_1, a_2, \ldots, a_n$。

接下来一行n 个整数，表示$w_1, w_2, \ldots, w_n$。

【输出格式】

输出文件名为perm.out。

输出一个整数表示答案。

【输入输出样例 1】

perm.in	perm.out
3 0 1 1 5 7 3	32

【样例解释 1】

对于 $a_1=0, a_2=1, a_3=1$，其排列有

$a_1=0, a_2=1, a_3=1$，是合法排列，排列的权值是 1*5+2*7+3*3=28；

$a_2=1, a_1=0, a_3=1$，是非法排列，因为 $a_{p[1]}$等于 p[2]；

$a_1=0, a_3=1, a_2=1$，是合法排列，排列的权值是 1*5+2*3+3*7=32；

$a_3=1, a_1=0, a_2=1$，是非法排列，因为 $a_{p[1]}$等于 p[2]；

$a_2=1, a_3=1, a_1=0$，是非法排列，因为 $a_{p[1]}$等于 p[3]；

$a_3=1, a_2=1, a_1=0$，是非法排列，因为 $a_{p[1]}$等于 p[3]。

因此该题输出最大权值 32。

【输入输出样例 2】

perm.in	perm.out
3 2 3 1 1 2 3	-1

【样例解释 2】

对于 $a_1=2,a_2=3,a_3=1$，其排列有：

$a_1=2,a_2=3,a_3=1$，是非法排列，因为 $a_{p[1]}$等于 p[2]；

$a_2=3,a_1=2,a_3=1$，是非法排列，因为 $a_{p[1]}$等于 p[3]；

$a_1=2,a_3=1,a_2=3$，是非法排列，因为 $a_{p[1]}$等于 p[3]；

$a_3=1,a_1=2,a_2=3$，是非法排列，因为 $a_{p[2]}$等于 p[3]；

$a_2=3,a_3=1,a_1=2$，是非法排列，因为 $a_{p[2]}$等于 p[3]；

$a_3=1,a_2=3,a_1=2$，是非法排列，因为 $a_{p[1]}$等于 p[3]。

因此该题没有合法排列。

【输入输出样例 3】

perm.in	perm.out
10 6 6 10 1 7 0 0 1 7 7 16 3 10 20 5 14 17 17 16 13	809

【数据范围】

对于前20% 的数据，$1\le n\le 10$。

对于前40% 的数据，$1\le n\le 15$。

对于前60% 的数据，$1\le n\le 1000$。

对于前80% 的数据，$1\le n\le 100000$。

对于100% 的数据，$1\le n\le 500000$，$0\le a_i\le n$，$1\le w_i\le 10^9$，所有w_i的和不超过1.5×10^{13}。

【编译命令】

对于c++语言： g++ -o perm perm.cpp –lm

对于c语言： gcc -o perm perm.c –lm

对于pascal语言： fpc perm.pas

第 3 题：道路(road)，运行时限 1s，内存上限 512M，100 分。

【问题描述】

W 国的交通呈一棵树的形状。W 国一共有$n-1$个城市和n个乡村，其中城市从1到$n-1$编号，乡村从1到n编号，且1号城市是首都。道路都是单向的，本题中我们只考虑从乡村通往首都的道路网络。对于每一个城市，恰有一条公路和一条铁路通向这座城市。对于城市i，通向该城市的道路（公路或铁路）的起点，要么是一个乡村，要么是一个编号比i大的城市。没有道路通向任何乡村。除了首都以外，从任何城市或乡村出发只有一条道路；首都没有往外的道路。从任何乡村出发，沿着唯一往外的道路走，总可以到达首都。

W 国的国王小 W 获得了一笔资金，他决定用这笔资金来改善交通。由于资金有限，小 W 只能翻修$n-1$条道路。小 W 决定对每个城市翻修恰好一条通向它的道路，即从公路和铁路中选择一条并进行翻修。小 W 希望从乡村通向城市可以尽可能地便利，于是根据人口调查的数据，小 W 对每个乡村制定了三个参数，编号为i的乡村的三个参数是a_i，b_i和c_i。假设从编号为i的乡村走到首都一共需要经过x条未翻修的公路与y条未翻修的铁路，那么该乡村的不便利值为

$$c_i \cdot (a_i + x) \cdot (b_i + y)$$

在给定的翻修方案下，每个乡村的不便利值相加的和为该翻修方案的不便利值。

翻修$n-1$条道路有很多方案，其中不便利值最小的方案称为最优翻修方案，小 W 自然希望找到最优翻修方案，请你帮助他求出这个最优翻修方案的不便利值。

【程序文件名】

源程序文件名为 road.cpp/c/pas。

【输入格式】

输入文件名为road.in。

第一行为正整数n。

接下来$n-1$行，每行描述一个城市。其中第i行包含两个数s_i, t_i。s_i表示通向第i座城市的公路的起点，t_i表示通向第i座城市的铁路的起点。如果$s_i > 0$，那么存在一条从第s_i座城市通往第i座城市的**公路**，否则存在一条从第$-s_i$个乡村通往第i座城市的**公路**；t_i类似地，如果$t_i > 0$，那么存在一条从第t_i座城市通往第i座城市的**铁路**，否则存在一条从第$-t_i$个乡村通往第i座城市的**铁路**。

接下来n行，每行描述一个乡村。其中第i行包含三个数a_i, b_i, c_i，其意义如题面所示。

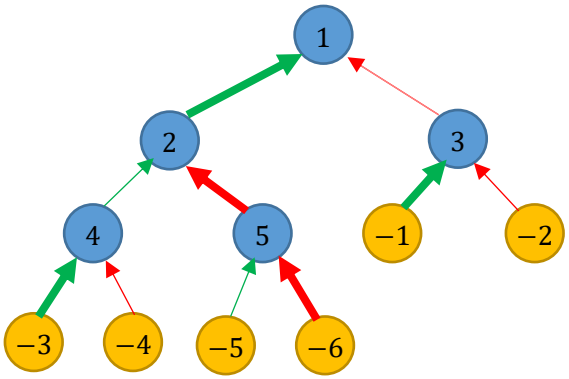
【输出格式】

输出文件名为road.out。

输出一行一个整数，表示最优翻修方案的不便利值。

【输入输出样例 1】

road.in	road.out
6 2 3 4 5 -1 -2 -3 -4 -5 -6 1 2 3 1 3 2 2 1 3 2 3 1 3 1 2 3 2 1	54



【样例解释 1】

如图所示，我们分别用蓝色、黄色节点表示城市、乡村；用绿色、红色箭头分别表示公路、铁路；用加粗箭头表示翻修的道路。

一种不便利值等于54的方法是：翻修通往城市2和城市5的铁路，以及通往其他城市的公路。用$\rightarrow$和$\Rightarrow$表示公路和铁路，用$*\rightarrow$和$*\Rightarrow$表示翻修的公路和铁路，那么：

编号为1的乡村到达首都的路线为：$-1 *\rightarrow 3 \Rightarrow 1$，经过0条未翻修公路和1条未翻修铁路，代价为$3 \times (1+0) \times (2+1) = 9$；

编号为2的乡村到达首都的路线为：$-2 \Rightarrow 3 \Rightarrow 1$，经过0条未翻修公路和2条未翻修铁路，代价为$2 \times (1+0) \times (3+2) = 10$；

编号为3的乡村到达首都的路线为：$-3 *\rightarrow 4 \rightarrow 2 *\rightarrow 1$，经过1条未翻修公路和0条未翻修铁路，代价为$3 \times (2+1) \times (1+0) = 9$；

编号为4的乡村到达首都的路线为：$-4 \Rightarrow 4 \rightarrow 2 *\rightarrow 1$，经过1条未翻修公路和1条未翻修铁路，代价为$1 \times (2+1) \times (3+1) = 12$；

编号为5的乡村到达首都的路线为：$-5 \rightarrow 5 *\Rightarrow 2 *\rightarrow 1$，经过1条未翻修公路和0条未翻修铁路，代价为$2 \times (3+1) \times (1+0) = 8$；

编号为6的乡村到达首都的路线为：$-6 *\Rightarrow 5 *\Rightarrow 2 *\rightarrow 1$，经过0条未翻修公路和0条未

翻修铁路，代价为$1 \times (3+0) \times (2+0) = 6$；

总的不便利值为$9+10+9+12+8+6 = 54$。可以证明这是本数据的最优解。

【输入输出样例 2】

road.in	road.out
9 2 -2 3 -3 4 -4 5 -5 6 -6 7 -7 8 -8 -1 -9 1 60 1 1 60 1 1 60 1 1 60 1 1 60 1 1 60 1 1 60 1 1 60 1 1 60 1	548

【样例解释 2】

在这个样例中，显然应该翻修所有公路。

【输入输出样例 3】

road.in	road.out
12 2 4 5 3 -7 10 11 9 -1 6 8 7 -6 -10 -9 -4	5744902

-12 -5 -2 -3 -8 -11 53 26 491 24 58 190 17 37 356 15 51 997 30 19 398 3 45 27 52 55 838 16 18 931 58 24 212 43 25 198 54 15 172 34 5 524	

【数据范围】

一共20组数据，编号为$1 \sim 20$。

对于编号≤ 4的数据，$n \leq 20$；

对于编号为$5 \sim 8$的数据，$a_i, b_i, c_i \leq 5$，$n \leq 50$；

对于编号为$9 \sim 12$的数据，$n \leq 2000$；

对于所有的数据，$n \leq 20000$，$1 \leq a_i, b_i \leq 60$，$1 \leq c_i \leq 10^9$，s_i, t_i是$[-n, -1] \cup (i, n-1]$内的整数，任意乡村可以通过不超过40条道路到达首都。

【编译命令】

对于c++语言： g++ -o road road.cpp –lm

对于c语言： gcc -o road road.c –lm

对于pascal语言： fpc road.pas