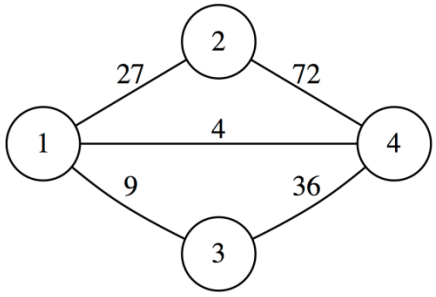


在第一个询问中，一条可行的路径为$1-2-4$。

在第二个询问中，一条可行的路径为$4-3-1-2$。

在第三个询问中，一条可行的路径为$1-4-1-3$。

【数据规模和约定】

所有测试点的数据规模如下：

测试点编号	n	m	q	备注
1	$n \leq 1000$	$m = n-1$	$q \leq 1000$	图为一条链，仅编号为i和$i+1$的顶点之间有边
2		$m \leq 2000$		无
3	$n \leq 50000$	$m = n-1$	$q \leq 50000$	给定的图为一条链
4				
5		$m \leq 100000$		所有边及询问中$a, b \leq 30$
6				所有边及询问中$a = 0$
7				无
8				
9				
10				

对于所有测试数据，$1 \leq n, q \leq 50000$、$1 \leq m \leq 100000$、$0 \leq a, b \leq 10^9$。

【编译命令】

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

对于c语言： gcc -o multiple multiple.c –lm

对于pascal语言： fpc multiple.pas

第 2 题：网络(network)，时间限制 2s，内存限制 128M。

【问题描述】

一个简单的网络系统可以被描述成一棵无根树。每个节点为一个服务器。连接服务器与服

务器的数据线则看做一条树边。

两个服务器进行数据的交互时，数据会经过连接这两个服务器的路径上的所有服务器（包括这两个服务器自身）。由于这条路径是唯一的，当路径上的某个服务器出现故障，无法正常运行时，数据便无法交互。此外，每个数据交互请求都有一个重要度，越重要的请求显然需要得到越高的优先处理权。

现在，你作为一个网络系统的管理员，要监控整个系统的运行状态。系统的运行也是很简单的，在每一个时刻，只有可能出现下列三种事件中的一种：

1. 在某两个服务器之间出现一条新的数据交互请求；
2. 某个数据交互结束请求；
3. 某个服务器出现故障。

系统会在任何故障发生后立即修复。也就是在出现故障的时刻之后，这个服务器依然是正常的。但在服务器产生故障时依然会对需要经过该服务器的数据交互请求造成影响。你的任务是在每次出现故障时，维护**未被影响的请求**中重要度的最大值。注意，如果一个数据交互请求已经结束，则不将其纳入未被影响的请求范围。

【程序文件名】

源程序文件名为 network.cpp/c/pas。

【输入格式】

输入文件名为 network.in。

第一行两个正整数 n,m，分别描述服务器和事件个数。服务器编号是从 1 开始的，因此 n 个服务器的编号依次是 1,2,3,…,n。

接下来 n-1 行，每行两个正整数 u,v，描述一条树边。u 和 v 是服务器的编号。

接下来 m 行，按发生时刻依次描述每一个事件；即第 i 行（i=1,2,3,…,m）描述时刻 i 发生的事件。每行的第一个数 type 描述事件类型，共 3 种类型：

（1）若 type=0，之后有三个正整数 a,b,v，表示服务器 a,b 之间出现一条重要度为 v 的数据交互请求；

（2）若 type=1，之后有一个正整数 t，表示时刻 t（也就是第 t 个发生的事件）出现的数据交互请求结束；

（3）若 type=2，之后有一个正整数 x，表示服务器 x 在这一时刻出现了故障。**对于每个 type 为 2 的事件，就是一次询问**，即询问“当服务器 x 发生故障时，未被影响的请求中重要度的最大值是多少？”

注意可能有某个服务器自身与自身进行数据交互的情况。

【输出格式】

输出文件名为 network.out。

对于每个 type=2 的事件，即服务器出现故障的事件，输出一行一个整数，描述未被影响的请求中重要度的最大值。如果此时没有任何请求，或者所有请求均被影响，则输出 -1。

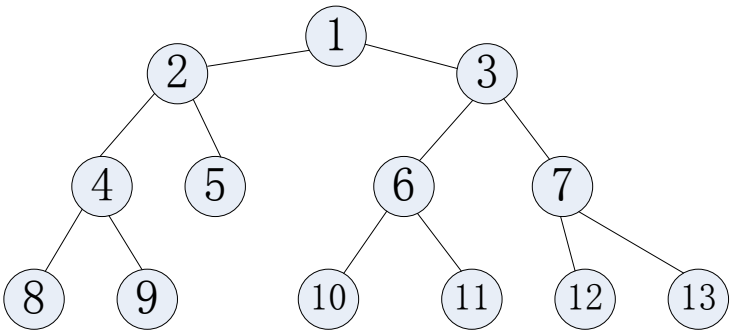
【输入输出样例】

network.in	network.out
13 23 1 2	-1 3

1 3 2 4 2 5 3 6 3 7 4 8 4 9 6 10 6 11 7 12 7 13 **2 1** 0 8 13 3 0 9 12 5 2 9 2 8 2 2 0 10 12 1 2 2 1 3 2 7 2 1 0 9 5 6 2 4 2 5 1 7 0 9 12 4 0 10 5 7 2 1 2 4 2 12 1 2 2 5 2 3	5 -1 1 -1 1 1 3 6 7 7 4 6

【样例解释】

样例给出的树如下所示：



解释其中的部分询问；下面的解释中用(a,b;t,v)表示在 t 时刻出现的服务器 a 和 b 之间的重要度为 v 的请求：

对于第一个询问（在时刻 1），此时没有任何请求，输出 -1。

对于第四个询问（在时刻 6），此时有两条交互 (8,13;2,3),(9,12;3,5)，所有询问均经过 2 号服务器，输出 -1。

对于第五个询问（在时刻 8），此时有三条交互(8,13;2,3), (9,12;3,5),(10,12;7,1)，只有交互 (10,12;7,1) 没有经过 2 号服务器，因此输出其重要度 1。

对于最后一个询问（在时刻 23），此时有三条交互 (9,5;12,6),(9,12;16,4),(10,5;17,7)。当 3 号服务器出现故障时，只有交互 (9,5;12,6) 没有经过 3 号服务器，因此输出 6。

【数据范围】

对于 20%的数据，n, m ≤ 1000。

对于 30%的数据，m ≤ 2000。

另有 20%的数据，第一次事件必定是加入重要度最大的交互，且没有 type=1 的事件。

另有 20%的数据，树是一条链，且满足 i 与 i+1 之间有边。

对于 100%的数据，2 ≤ n ≤ 10^5, 1 ≤ m ≤ 2×10^5，其他的所有输入值不超过 10^9。

为了便于你分辨第三类数据（即第一次事件必定加入重要度最大的交互），规定这条交互的重要度为 10^9。而在其他类型的数据中，任何交互的重要度都小于该值。

【编译命令】

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

对于c语言： gcc -o network network.c –lm

对于pascal语言： fpc network.pas

第 3 题：树(tree)，时间限制 2s，内存限制 128M。

【问题描述】

小 A 想做一棵很大的树，但是他手上的材料有限，只好用点小技巧了。

开始，小 A 只有一棵结点数为 N 的树，结点的编号为 1,2,…,N，其中结点 1 为根；我们称这颗树为模板树。小 A 决定通过这棵模板树来构建一颗大树。构建过程如下：

（1）将模板树复制为初始的大树。

（2）以下(2.1)(2.2)(2.3)步循环执行 M 次