天天爱跑步

【问题描述】

小C同学认为跑步非常有趣，于是决定制作一款叫做《天天爱跑步》的游戏。《天 天爱跑步》是一个养成类游戏，需要玩家每天按时上,，完成打卡任务。

这个游戏的地图可以看作一棵包含***n***个结点和***n****-* 1条边的树，每条边连接两个结 点，且任意两个结点存在一条路径互相可达。树上结点编号为从1到***n***的连续正整数。

现在有***m***个玩家，第***i***个玩家的起点为***S****i，*终点为***T****i*"每天打卡任务开始时，所 有玩家在第0秒同时从自己的起点出发,以每秒跑一条边的速度，不间断地沿着最短 路径向着自己的终点跑去,跑到终点后该玩家就算完成了打卡任务。（由于地图是一棵 树，所以每个人的路径是唯一的）

小C想知道游戏的活跃度，所以在每个结点上都放置了一个观察员。在结点***j***的观 察员会选择在第***W****j*秒观察玩家，一个玩家能被这个观察员观察到当且仅当该玩家在第 ***W****j*秒也正好到达了结点***j*** "小C想知道每个观察员会观察到多少人？

碧丄我们认为一个玩家到达自己的终点后该玩家就会结束游戏，他不能等待一 段时间后再被观察员观察到。即对于把结点***j***作为终点的玩家：若他在第***W****j*秒c到达 终点，则在结点***j***的观察员不能观察到该玩家;若他正好在第***W****j*秒到达终点，则在结 点***j***的观察员可以观察到这个玩家。

【输入格式】

第一行有两个整数***n***和***m*** -其中***n***代表树的结点数量，同时也是观察员的数量， ***m***代表玩家的数量。

接下来***n*** *-* 1行每行两个整数***u***和***v***，表示结点***u***到结点***v***有一条边。

接下来一行***n***个整数，其中第***j***个整数为***W****j*，表示结点***j***出现观察员的时间。

接下来***m***行，每行两个整数***Si***和***T****i*，表示一个玩家的起点和终点。

对于所有的数据，保证1 < ***Si, Ti*** < ***n***，0 < ***W****j <* ***n*** -

【输出格式】

输出1行***n***个整数，第***j***个整数表示结点***j***的观察员可以观察到多少人。

【样例1输入】

6 3

2 3

1 2

4

4 5

4 6

0 2 5 1 2 3

15

13

26

【样例1输出】

0 0 1 1 1

【样例1说明】

对于1号点，***W***1 = 0 ,故只有起点为1号点的玩家才会被观察到，所以玩家1和 玩家2被观察到，共2人被观察到。

对于2号点，没有玩家在第2秒时在此结点，共0人被观察到。

对于3号点，

没有玩家在第5秒时在此结点，共0人被观察到。 玩家1被观察到，共1人被观察到。

玩家1被观察到，共1人被观察到。

玩家3被观察到，共1人被观察到。

对于4号点，

对于5号点，

对于6号点，

【子任务】

每个测试点的数据规模及特点如下表所示。提示：数据范围的个位上的数字可以 帮助判断是哪一种数据类型。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试点编号 | ***n*** | ***m*** | 约定 |
| 1 | *=*991 | *=*991 | 所有人的起点等于自己的终点， |
| 2 | 即 ***S****i =* ***T****i* |
| 3 | =992 | =992 | ***W****j* = 0 |
| 4 |
| 5 | =993 | =993 | 无 |
| 6 |  |  | 树退化成一条链，其中1与2有边，  2与3有边，...，***n*** *-* 1与***n***有边 |
| 7 | =99994 | =99994 |
| 8 |  |  |
| 9 |  |  |  |
| 10 | =99995 | =99995 | 所有的***S****i =* 1 |
| 11 |
| 12 |  |  |  |
| 13 |  |  |  |
| 14 | =99996 | =99996 | 所有的***T****i* = 1 |
| 15 |
| 16 |  |  |  |
| 17 |  |  |  |
| 18 | =99997 | =99997 | 无 |
| 19 |  |  |
| 20 | =299998 | =299998 |  |

【提示】

如果你的程序需要用到较大的栈空间（这通常意味着需要较深层数的递归），请务 必仔细阅读选手目录下的文档***running****/****stack.pdf***,以了解在最终评测时栈空间的限制与 在当前工作环境下调整栈空间限制的方法。

在最终评测时，调用栈占用的空间大小不会有单独的限制，但在我们的工作 环境中默认会有8 MB的限制。这可能会引起函数调用层数较多时，程序发生 栈溢出崩溃。

我们可以使用一些方法修改调用栈的大小限制。例如，在终端中输入下列命 令

ulimit -s 1048576

此命令的意义是，将调用栈的大小限制修改为1 GB。

例如，在选手目录建立如下***sample.cpp***或***sample.pas***

|  |  |
| --- | --- |
| ***sample.cpp*** | ***sample.pas*** |
| void dfs(int a){ | procedure dfs(a: longint); |
| if(a == 0) | var t: longint; |
| return; | begin |
| int t = a; | if a = 0 then |
| dfs(a - 1); | exit; |
| } | t := a; |
| int main(){ | dfs(a - 1); |
| dfs(1000000); | end; |
| return 0; | begin |
| } | dfs(1000000); |

end.

将上述源代码编译为可执行文件***sample***后，可以在终端中运行如下命令运 行该程序

./sample

如果在没有使用命令“ulimit -s 104857 6"的情况下运行该程序,***sample*** 会因为栈溢出而崩溃；如果使用了上述命令后运行该程序，该程序则不会崩溃。

特别地，当你打开多个终端时，它们并不会共享该命令，你需要分别对它们 运行该命令。

请注意，**调用栈占用的空间会计入总空间占用中，和程序其他部分占用的内 存共同受到内存限制**。