沙丘

【问题描述】

根据新出土的一批史料记载，在塔克拉玛干沙漠中的一座沙丘下面，埋藏着一个神秘的地下迷宫。由著名探险家阿强率领的探险队经过不懈的挖掘，终于发现了通往地下迷宫的入口！队员们兴奋不已，急忙钻下去，去寻找那个埋藏已久的秘密。

他们刚钻进迷宫，只听“轰隆”一声巨响，回头一看，入口已与石墙融为一体，无法辨认。他们意识到自己被困在迷宫里了！环顾周围，似乎是一个洞穴。

这座迷宫由很多洞穴组成，某些洞穴之间有道路连接。每个洞穴都有一盏灯，凭借着微弱的灯光，可以看清有多少条道路与这个洞穴相连。每个洞穴的内部是完全相同的，且无法做标记。每条道路也是完全相同的，也无法做标记。

阿强凭借着微弱的灯光，发现了墙壁上的一段文字（事实上，每个洞穴的墙壁上都有这段文字），翻译成现代汉语就是：“陌生人，请把这个迷宫的洞穴数和道路数告诉我，我就会指引你走出迷宫。”

阿强很快镇定了下来，他拿出一个路标，对队员们说：“这个迷宫的危险程度远超出我们的想象，为了安全起见，大家一定要集体行动。我这儿有一个路标，有了它，我们一定能探明迷宫的结构。大家跟我走！”

现在，轮到你扮演阿强了。路标只有一个，可以随身携带，也可以暂时放在某个洞穴中（把路标放在道路上是毫无意义的，因为那里一片漆黑，什么都看不见）。你的任务很简单：用尽量少的步数探明这个迷宫共有多少个洞穴和多少条道路。“一步”是指从一个洞穴走到另一个相邻的洞穴。

【交互方法】

本题是一道交互式题目，你的程序应当和测试库进行交互，而不得访问任何文件（包括临时文件）。测试库提供了若干函数，它们的用法和作用如下：

- init 必须先调用，但只能调用一次，用作初始化测试库；
- look(d, sign)的作用是查看当前洞穴的情况，测试库将从整型变量 *d* 中返回与该洞穴相连的道路的数目，从布尔变量 *sign* 中返回该洞穴内是否有路标，*sign* 为 true 表示有路标，为 false 表示无路标。
- put_sign 的作用是在当前洞穴放上路标。只有当路标随身携带着的时候，才可以调用这个函数。
- take_sign 的作用是把当前洞穴的路标拿走。只有当路标在当前洞穴时，才可以调用这个函数。
- walk(i)的作用是沿着编号为 *i* 的道路走到相邻的洞穴中。这里的编号是相对于当前所在洞穴而言的，并且是暂时的。假设与某洞穴相连的道路有 *d* 条，这些道路按照逆时针顺序依次编号为 0，1，2，……，*d*-1。走第一步时，编号为 0 的道路由库确定。以后的过程，阿强会将他走进这个洞穴的道路编号为 0。
- report(n, m)的作用是向测试库报告结果。*n* 表示洞穴的数目，*m* 表示道

路的数目。当这个函数被调用后，测试库会自动中止你的程序。

【对使用 Pascal 选手的提示】

你的程序应当使用如下的语句引用测试库。

```
uses dune_lib;
```

测试库使用的函数原型为：

```
procedure init;
procedure look(var d: longint; var sign: boolean);
procedure put_sign;
procedure take_sign;
procedure walk(i: longint);
procedure report(n, m: longint);
```

【对使用 C/C++选手的提示】

你应当建立一个工程，把文件 dune_libc.o 包含进来，然后在程序头加一行：

```
#include “dune_lib.h”
```

测试库使用的函数原型为：

```
void init();
void look(int *, int *);
void put_sign();
void take_sign();
void walk(int);
void report(int, int);
```

在 C/C++中，布尔型变量用整型变量代替，0 表示 false，1 表示 true。

【你如何测试自己的程序】

- 在工作目录下建立一个文件叫做 dune.in，文件的第一行包括一个整数 n 为洞穴的数目，洞穴用 1 到 n 的整数编号，以下 n 行描述迷宫的结构。文件的第 i+1 行描述编号为 i 的洞穴的情况，第一个数 d_i 表示与该洞穴相连的道路的数目，其后的 d_i 个数按照逆时针顺序给出了这些道路另一端的洞穴编号。
- 调用 init 函数之后，库将编号为 1 的洞穴作为探险队的起始洞穴，并暂定编号为 0 的道路通向的洞穴编号为文件中第二行的第二个数。比如样例中，初始时，库暂定编号为 0 的道路通向洞穴 4。
- 执行你的程序，此时测试库会产生输出文件 dune.log，该文件中包括了你程序和库交互的记录和最后的结果。
- 如果程序正常结束，dune.log 的最后一行包含一个整数，为你走的步数。
- 如果程序非法退出，则我们不保证 dune.log 中的内容有意义。

【约定】

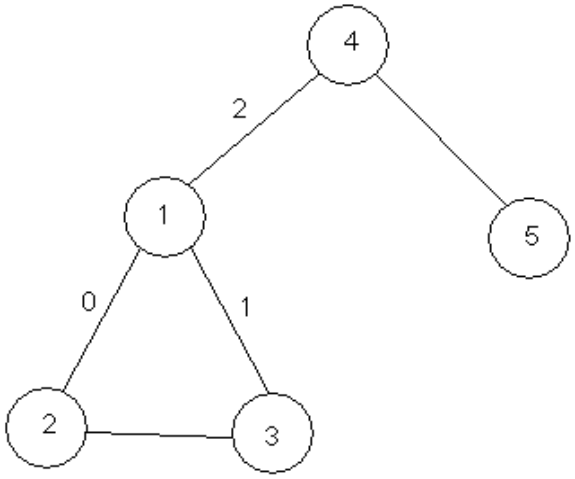
- 洞穴数不超过 100，道路数不超过 4000。
- 迷宫是连通的，即任意两个洞穴都相互可达。
- 两个洞穴之间最多只有一条道路。

- 没有哪条道路连接两个相同的洞穴。

【样例】

dune.in 内容如下

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



探险队初始时站在编号为 1 的洞穴内，编号为 0 的道路通向洞穴 2，编号为 1 的道路通向洞穴 3，编号为 2 的道路通向洞穴 4。

一种可能得满分的调用方案如下：

Pascal 选手的调用方法	C/C++选手的调用方法	说明
`init;`	`init();`	初始化程序
`look(d, sign);`	`look(d, sign);`	返回 d=3，sign=false
`put_sign;`	`put_sign();`	放下路标
`walk(0);`	`walk(0);`	选择编号为 0 的道路
`look(d, sign);`	`look(d, sign);`	返回 d=2，sign=false
`walk(1);`	`walk(1);`	选择编号为 1 的道路
`look(d, sign);`	`look(d, sign);`	返回 d=2，sign=false
`walk(1);`	`walk(1);`	选择编号为 1 的道路
`look(d, sign);`	`look(d, sign);`	返回 d=3，sign=true
`take_sign;`	`take_sign();`	拿起路标
`walk(1);`	`walk(1);`	选择编号为 1 的道路
`look(d, sign);`	`look(d, sign);`	返回 d=2，sign=false
`walk(1);`	`walk(1);`	选择编号为 1 的道路
`look(d, sign);`	`look(d, sign);`	返回 d=1，sign=false
`report(5, 5);`	`report(5, 5);`	返回洞穴数为 5，道路数为 5

注意，该例子只是对库函数的使用说明，并没有算法上的意义。

【评分方法】

如果你的程序有下列情况之一，该测试点 0 分：

- 访问了任何文件(包括临时文件)或者自行终止；
- 非法调用库函数；

- 让测试库异常退出。

否则你每个测试点的得分按这样来计算：

如果你所报告的洞穴数与通道数都不正确，得 0 分；如果只有其中一个正确，得 2 分；如果两个都正确，则根据 walk 函数的调用次数评分，公式如下：

$$your\_score = \begin{cases} 10 & your\_ans \le our\_ans \\ 5 + \left\lfloor \dfrac{our\_ans}{your\_ans} \times 5 + 0.5 \right\rfloor & your\_ans > our\_ans \end{cases}$$

其中 *your_ans* 表示你的程序调用 walk 函数的次数，*our_ans* 表示我们的程序的结果。