K. 流光解密 / Aura

这是一道通信题。

请务必在每次输出后刷新标准输出缓冲区。各编程语言的刷新方式如下：

- 对于 C/C++，请使用 `fflush(stdout)` 或 `cout.flush()`；
- 对于 Java，请使用 `System.out.flush()`；
- 对于 Python，请使用 `sys.stdout.flush()`。

【题目背景】

供电网络修复完毕，全息投影仪终于顺利启动。随着夜色渐深，半空中的全息投影愈发缤纷绚丽。一切就绪后，小 T 和小 S 正式拉开了夜间活动的帷幕，邀请大家结伴，共同参与这场精心筹备、考验双人合作默契的光影游戏” 流光解密”。

广场中央的光束交织在一起，缓缓汇聚成一棵全息光树。光树由若干个悬浮的光团结点与连接它们的流光线路组成，呈现出纯粹的树形结构。游戏开始时，所有线路均未点亮，挑战者无法观测到任何未点亮的线路。执行系统操作的 Karuha 会率先向驻守在控制台前的一名挑战者揭示一个神秘数字。随后，各条流光线路会逐一亮起，该挑战者必须在每条线路浮现的瞬间，决定其流动的方向；而站在舞台另一端的搭档，则需仅凭最终形成的有向树结构，准确推断出这个神秘数字。

作为庆典参与者，Neri 和 Noir 决定配合完成这项挑战。

【题目描述】

在这项挑战中，Neri 将驻守在控制台前。掌控系统的 Karuha 首先会向她给出两个正整数 n, s $(1 \le s \le 2^{n-1})$，分别表示全息光树包含的光团数量与神秘数字。

随后，Karuha 将依次点亮 $n-1$ 条流光线路，Neri 必须在每条线路亮起时，**立即**为其指定流动的方向。

对于站在主舞台另一端的 Noir，她将观察到整棵全息光树充满流光的最终形态。她需要根据这些线路的流动方向，推断出 Karuha 传递给 Neri 的神秘数字 s。

为了赢得这场光影挑战，Neri 和 Noir 需要提前制定一套策略，以确保在这个考验双人默契的游戏中顺利通关。

【实现细节】

你的程序将会**独立运行两次**，其中第一次将扮演 Neri 决定每条线路流动的方向，第二次将扮演 Noir 根据全息光树的最终形态推断出神秘数字。交互器将扮演 Karuha，向你的程序传递信息。

在第一次运行中，你的程序会首先收到全息光树包含的光团数量 n 与神秘数字 s，然后依次收到 $n-1$ 条点亮的流光线路。你的程序需要**立即**向交互库输出为其指定的流

动的方向，从而向第二次运行传递信息。

在第二次运行中，你的程序会从交互器收到来自第一次运行的信息，即全息光树上所有流光线路的流动方向，然后需要根据这些信息推断出 Karuha 传递给 Neri 的神秘数字 s。

【输入输出格式】

每个测试点中包含多组测试数据。输入的第一行包含两个正整数 T, r ($1 \le T \le 10^4$, $r \in \{1, 2\}$)，表示数据组数与阶段编号。对于每组测试数据：

- 若 $r = 1$，则
  - 第一行输入两个正整数 n, s ($3 \le n \le 30$, $1 \le s \le 2^{n-1}$)，分别表示全息光树包含的光团数量与神秘数字；
  - 接下来将会依次点亮 $n-1$ 条流光线路。每次点亮一条流光线路时，输入两个正整数 u, v ($1 \le u < v \le n$)，表示流光线路连接的两个光团的编号，你需要**立即**输出一行两个正整数 u, v 或 v, u，表示流动方向为从 u 到 v 或从 v 到 u。
  - **注意：**
    - 对于每条流光线路，你**必须**输出该条流光线路的流动方向并刷新标准输出缓冲区后，才能继续输入下一条点亮的流光线路。
- 若 $r = 2$，则
  - 第一行输入一个正整数 n ($3 \le n \le 30$)，表示全息光树包含的光团数量；
  - 接下来 $n-1$ 行，每行输入两个正整数 u, v ($1 \le u, v \le n$)，表示一条从光团 u 流向光团 v 的流光线路。
  - 你需要**立即**输出一行一个正整数，表示推断出的神秘数字 s。
  - **注意：**
    - 单个测试点内测试数据输入的先后顺序相较第一次运行时**可能不同**。
    - 对同一组测试数据，输入所有流光线路的顺序在两次运行中**可能不一致**，但所有光团的编号保持不变。
    - 对于每一组测试数据，你**必须**输出推断出的神秘数字 s 并刷新标准输出缓冲区后，才能继续输入下一组测试数据。

【样例】

第一次运行时，输入如下：

```
2 1
7 21
1 6
```

```
3 5
2 4
3 4
1 2
6 7
9 59
1 2
1 4
3 4
3 5
3 8
5 6
6 7
8 9
```

Neri 定向的结果如下：

```
6 1
3 5
4 2
3 4
1 2
7 6

1 2
1 4
3 4
3 5
8 3
5 6
6 7
9 8
```

注意：上述空行仅为方便理解两组测试数据之间的分隔，实际程序输出时无需打印多余的空行。

两组测试数据的所有流光线路定向后，全息光树的形态分别如下：

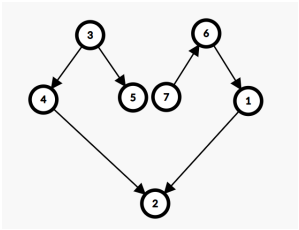

图 1: 第一组测试数据

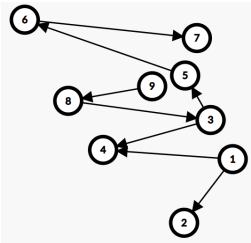

图 2: 第二组测试数据

根据上述输出，第二次运行时，一种可能的输入如下：

```
2 2
9
9 8
1 2
5 6
6 7
1 4
3 5
8 3
3 4
7
1 2
4 2
6 1
3 5
3 4
7 6
```

通过心灵感应，Noir 推断出第一组测试数据中 $s = 59$，第二组测试数据中 $s = 21$，因此输出：

```
59
21
```

【测试程序方式】

下发的脚本文件 `treedir_testing_tool.py` 可以协助你进行本地测试以验证程序的正确性，并打印详细的交互过程。测试时，请将该脚本与你编译生成的可执行文件置于同一目录下，随后在终端内运行以下命令：

```
python3 treedir_testing_tool.py [--quiet] <data_file>
  <program> <arguments>
```

其中：

- `-q, --quiet` 为可选参数，使用该参数后，测试脚本将不再打印详细的交互过程。
- `data_file` 为提供给测试脚本的输入数据文件路径。该文件需满足以下格式：
  - 第一行包含两个非负整数 $T, seed$，分别代表测试数据组数，与用于打乱测试数据顺序及树内连边顺序的随机数种子。若指定 $seed = 0$，则表示不进行打乱。
  - 每组数据的格式与前文描述的“第一次运行”的输入格式完全相同。
- `program` 为你编译生成的可执行文件路径。
- `arguments` 为传递给该可执行文件的额外命令行参数。

关于测试脚本的更多实现细节，请直接查阅其源代码。

注意：

1. 测试脚本与实际评测时的交互库实现**不完全相同**，本地测试结果不具有终断性，仅供调试阶段参考；
2. 测试脚本仅会对 `data_file` 中的数据进行基础的格式校验，**不会检查**其是否合法地满足了题目限制（例如，不会检查当 s 是否满足 $1 \le s \le 2^{n-1}$ 的限制，也不会判断所有流光线路能否正确构成一棵树）；
3. 测试脚本**不会监控**程序的时间与空间消耗，无法用于测试是否符合题目的时空限制。