第 37 届全国青少年信息学奥林匹克竞赛

CCF NOI 2020

第二试

时间：2020 **年** 8 **月** 19 **日** 08:00 ~ 13:00

题目名称	制作菜品	超现实树	翻修道路
题目类型	传统型	传统型	传统型
目录	dish	surreal	road
可执行文件名	dish	surreal	road
输入文件名	dish.in	surreal.in	road.in
输出文件名	dish.out	surreal.out	road.out
每个测试点时限	2.0 秒	1.0 秒	2.0 秒
内存限制	512 MB	512 MB	1 GB
子任务数目	20	25	20
测试点是否等分	是	是	是

提交源程序文件名

对于 C++ 语言	dish.cpp	surreal.cpp	road.cpp

编译选项

对于 C++ 语言	-lm -O2 -std=c++11

注意事项

1. 选手提交的源文件必须存放在已建立好的带有下发样例的文件夹中（该文件夹与试题同名）。
2. 文件名（包括程序名和输入输出文件名）必须使用英文小写。
3. C++ 中函数 main() 的返回值类型必须是 int，值必须为 0。
4. 对于因未遵守以上规则对成绩造成的影响，相关申诉不予受理。
5. 若无特殊说明，输入文件中同一行内的多个整数、浮点数、字符串等均使用一个空格进行分隔。
6. 若无特殊说明，结果比较方式为忽略行末空格、文末回车后的全文比较。
7. 程序可使用的栈空间大小与该题内存空间限制一致。
8. 在终端下可使用命令`ulimit -s unlimited`将栈空间限制放大，但你使用的栈空间大小不应超过题目限制。

制作菜品（dish）

【题目描述】

厨师准备给小朋友们制作 m 道菜，每道菜均使用 k 克原材料。为此，厨师购入了 n 种原材料，原材料从 1 到 n 编号，第 i 种原材料的质量为 d_i 克。n 种原材料的**质量之和恰好为** $m \times k$ **克**，其中 d_i 与 k 都是**正整数**。

制作菜品时，一种原材料可以被用于多道菜，但为了让菜品的味道更纯粹，厨师打算每道菜**至多使用 2 种**原材料。现在请你判断是否存在一种满足要求的制作方案。更具体地，方案应满足下列要求：

- 共做出 m 道菜。
- 每道菜至多使用 2 种原材料。
- 每道菜恰好使用 k 克原材料。
- 每道菜使用的每种原材料的质量都为正整数克。
- n 种原材料都被恰好用完。

若存在满足要求的制作方案，你还应该给出一种具体的制作方案。

【输入格式】

从文件 ***dish.in*** 中读入数据。

本题单个测试点包含多组测试数据。

第一行一个整数 T 表示数据组数。对于每组数据：

- 第一行三个正整数 n, m, k 分别表示原材料种数、需要制作的菜品道数、每道菜品需使用的原材料的质量。
- 第二行 n 个整数，第 i 个整数表示第 i 种原材料的质量 d_i。

【输出格式】

输出到文件 ***dish.out*** 中。

对于每组测试数据：

- **若不存在满足要求的制作方案，则输出一行一个整数** -1;
- 否则你需要输出 m 行，每行表示一道菜品的制作方案，根据使用的原材料种数，格式为下列两种之一：
  - 依次输出一行两个整数 i 和 x，表示该道菜使用 x 克第 i 种原材料制作。你应保证 $1 \le i \le n$，$x = k$。
  - 依次输出一行四个整数 i、x、j 和 y，表示该道菜使用 x 克第 i 种原材料与 y 克第 j 种原材料制作。你应保证 $1 \le i, j \le n$，$i \ne j$，$x + y = k$，$x, y > 0$。

本题使用**自定义校验器**检验你的答案是否正确，因此若有多种满足条件的方案，你只需要输出**任意一种**。

你应保证方案输出的格式正确，且同一行中相邻的两个数使用单个空格分隔，除此之外你的输出中**不应包含其他多余字符**。

【样例 1 输入】

```
4
1 1 10
10
4 3 100
80 30 90 100
5 3 1000
200 400 500 900 1000
6 4 100
25 30 50 80 95 120
```

【样例 1 输出】

```
1 10
1 80 2 20
2 10 3 90
4 100
-1
1 5 5 95
1 20 4 80
2 30 6 70
3 50 6 50
```

【样例 1 解释】

对于第二组数据，一种满足要求的制作方案为：

- 使用 80 克原材料 1 与 20 克原材料 2 做第一道菜。
- 使用 10 克原材料 2 与 90 克原材料 3 做第二道菜。
- 使用 100 克原材料 4 做第三道菜。

【样例 2】

见选手目录下的 ***dish/dish2.in*** 与 ***dish/dish2.ans***。

【样例 3】

见选手目录下的 ***dish/dish3.in*** 与 ***dish/dish3.ans***。

【测试点约束】

对于所有测试点：
$1 \le T \le 10$，$1 \le n \le 500$，$n-2 \le m \le 5000$，$m \ge 1$，
$1 \le k \le 5000$，$\sum_{i=1}^{n} d_i = m \times k$。
每个测试点的具体限制见下表：

<table>
<tr><th>测试点编号</th><th>n</th><th>m</th><th>k</th></tr>
<tr><td>$1 \sim 3$</td><td>≤ 4</td><td>≤ 4</td><td>≤ 50</td></tr>
<tr><td>$4 \sim 5$</td><td>≤ 10</td><td>≤ 10</td><td rowspan="4">≤ 5000</td></tr>
<tr><td>$6 \sim 7$</td><td rowspan="2">≤ 500</td><td>$= n-1$</td></tr>
<tr><td>$8 \sim 9$</td><td>$n-1 \le m \le 5000$</td></tr>
<tr><td>10</td><td rowspan="2">≤ 25</td><td rowspan="5">≤ 5000</td></tr>
<tr><td>$11 \sim 12$</td><td rowspan="2">≤ 500</td></tr>
<tr><td>$13 \sim 14$</td><td>≤ 50</td></tr>
<tr><td>$15 \sim 17$</td><td>≤ 100</td><td rowspan="2">≤ 5000</td></tr>
<tr><td>$18 \sim 20$</td><td>≤ 500</td></tr>
</table>

超现实树（surreal）

【题目背景】

下课铃声响起，机房里的两位女生从座位上站起来。（下面用 **X1,X2** 代指两人）

X2：省选前的集训真难熬啊……听课、考试、讲评、补题——对于现在的我来说，即使在梦里想到一道数据结构题，也会不由自主地开始思考吧。

X1：重复训练对我来说似乎并不是什么负担，但我确实感觉到解决题目带来的愉悦感在最近逐渐减弱了。也许我们需要一些精神上的“刺激”：一些不拘泥于繁复技术的智力游戏，来让我们找回对于数学和算法的兴趣。

X2：咦，我好像收到了一封用英文写的短信，似乎是……数学书上的一些片段。

【题目描述】

X1：我来翻译一下短信的内容。

> **定义**：本文所述的树是归纳定义的：单独的结点构成一棵树，以一棵树作为左（或右）孩子可以构成一棵树，以两棵树分别作为左、右孩子也可以构成一棵树。仅由以上规则用有限步生成的所有结构被称为树。

X2：也就是说，这里所说的树是指**非空、有根、区分左右孩子的二叉树**。
X1：的确如此。接下来书上定义了两棵树的同构。

> **定义**：称两棵树 T, T' 同构，记做 $T \equiv T'$，由以下四条规则定义：
>
> 1. 由单独结点构成的树是彼此同构的；
> 2. 如果两棵树的根结点均只有左子树，并且它们的左子树同构，那么这两棵树是同构的；
> 3. 如果两棵树的根结点均只有右子树，并且它们的右子树同构，那么这两棵树是同构的；
> 4. 如果两棵树的根结点均有左、右子树，并且它们的左、右子树分别对应同构，那么这两棵树是同构的。
>
> 很明显，同构关系构成了所有树上的一个等价关系。为了方便，我们将同构的树看作相同的树。

X2：将同构的树看成相同的树就是说树的结点是彼此相同的。简单地说，两棵树同构当且仅当**他们在结点无标号、区分左右孩子的意义下相同**；我们说两棵树不同，当且仅当它们不同构。

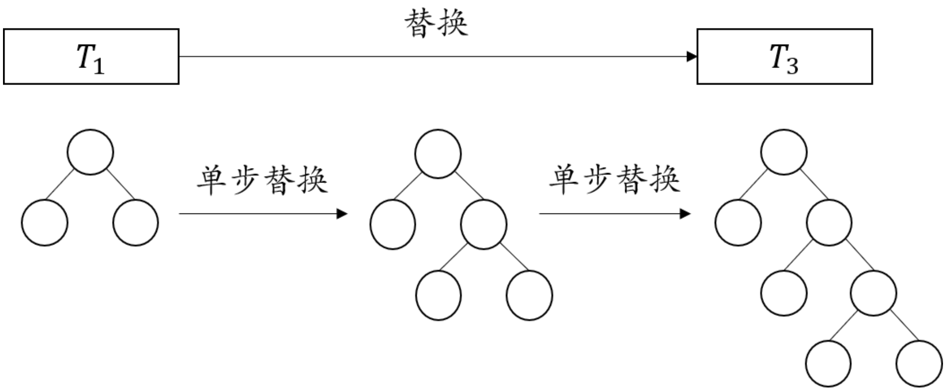
X1：书里还定义了树的**叶子**：和通常的定义一样，叶子指**没有任何孩子的结点**。

X2：这和我们熟悉的定义完全一致。嘛，数学家真是有点啰嗦……恐怕只有 X3 那种家伙会喜欢这种做派吧。

X1：我倒是对此不太反感——比起基于经验的“直觉”，准确的定义和严谨的证明还是更加让人安心。你看，下一个定义就没有那么直观了。

> **定义**：称一棵树 T **单步替换**成为 T'，如果将 T 的某一**叶子结点**替换为另一棵树 T'' 得到的树与 T' 同构，记做 $T \to T'$；称一棵树 T **替换**成为 T'，记做 $T \to^\star T'$，如果存在自然数 $n \geq 1$ 和树 $T_1, T_2, \ldots, T_n$，使得 $T \equiv T_1 \to T_2 \to \cdots \to T_n \equiv T'$。

X2：我来想想……所谓替换，就是删掉某个叶子结点并在对应的位置放入另一棵树，就像那个叶子结点“长出了”一个更大的子树一样；一棵树替换成为另一棵树，说明它可以经由**零次、一次或多次**单步替换得到那棵树。哦……我明白了！举例来说，任何一棵树都可以替换成它本身，换言之对于树 T，都有 $T \to^\star T$。下面这个图片可以帮助理解单步替换和替换的含义。



X1：你说得对。特别地，任何一棵树都可以替换得到无穷多棵不同的树，并且仅有一个结点构成的树可以替换得到任意其他的树。书上也有定义这样的东西。

> **定义**：对于一棵树 T，定义 $\mathrm{grow}(T)$ 表示 T 所能替换构成的树的集合，即 $\mathrm{grow}(T) = \{T' \mid T \to^\star T'\}$。更近一步，如果 $\mathscr{T} = \{T_1, T_2, \ldots, T_n\}$ 是一个树的有限集合，定义 $\mathrm{grow}(\mathscr{T})$ 为所有 $\mathrm{grow}(T_i)$ 的并集，其中 $i = 1, 2, \ldots, n$。即
>
> $$\mathrm{grow}(\mathscr{T}) = \bigcup_{T_i \in \mathscr{T}} \mathrm{grow}(T_i).$$

X2：我们把 $\mathrm{grow}(\mathscr{T})$ 称作树的集合 $\mathscr{T}$ **所生长得到的集合**吧——也就是说，树的集合 $\mathscr{T}$ 所生长得到的集合包含所有可以被某个 $T \in \mathscr{T}$ 替换得到的树。不妨把树的集合叫做**树林**。不太严谨地说，一个树林所生长得到的新树林就是其中所有树、以所有可能的方式生长得到的树林。显而易见，一个非空树林所生长得到的树林都是无穷树林。但这个无穷树林，或者说 $\mathrm{grow}(\mathscr{T})$，并不一定包含所有的树——更进一步，它甚至不一定包含“几乎所有”的树。

X1：让我来补充一下：我们称一个树林是**几乎完备**的（或称**几乎包含了所有的树**），如果仅有有限多的树不在其中。对于一个有限树林 $\mathscr{T}$，$\mathrm{grow}(\mathscr{T})$ 要么包含了所有的树，

要么包含了几乎所有的树，要么存在无穷多棵树不在其中。如果这是一道 OI 题，出题人一定会**在样例中给出三种情况的例子**吧。书上的关键定理也用了和我们相同的定义。

> **定理（几乎完备的可判定性）**：一个树的集合是**几乎完备**的，如果仅有有限棵树不在其中。那么，对于一个给定的树的有限集合 $\mathscr{T}$，存在高效的算法判定 $\mathrm{grow}(\mathscr{T})$ 是否是几乎完备的。

X2：这个问题变成一个纯粹的 OI 题目了！让我用我们的语言来重述一下题意：**给定一个有限大小的树林 $\mathscr{T}$，判定 $\mathrm{grow}(\mathscr{T})$ 是否是几乎完备的，即是否仅有有限棵树不能被树林中所包含的树生长得到**。

X1：也就是说，给定一个有限的树的集合 $\mathscr{T}$，判定是否仅有有限个树 T，满足 $T \notin \mathrm{grow}(\mathscr{T})$。所谓 $T \notin \mathrm{grow}(\mathscr{T})$，就是说不存在 $T' \in \mathscr{T}$，使得 $T' \to^{\star} T$。这和通常的 OI 题目的确非常不同：我甚至没有想到这个问题的一个算法。

X2：我也一样，不过我很久没有感受到这种解决未知问题的冲动了。

【输入格式】

从文件 ***surreal.in*** 中读入数据。

本题有多组测试数据，输入文件的第一行包含一个正整数 N，表示测试数据的组数。接下来包含恰好 N 组测试数据，每组测试数据具有以下的格式：

第一行是一个正整数 m，表示树的集合中树的个数。接下来按照以下格式输入 m 棵树：

- 首先是一个正整数 n，表示树中的结点个数，结点编号为 $1, 2, \ldots, n$；
- 接下来 n 行每行两个非负整数，其中第 i 行从左到右包含用空格隔开的 l_i 和 r_i，分别表示 i 号结点左、右孩子结点的编号。如果左（或右）孩子不存在，那么 l_i（或 r_i）为 0。当然，叶结点一定满足 $l_i = r_i = 0$。
- 输入数据保证构成一棵以 1 号结点作为根结点的树。**请注意**：结点的编号只是为了方便输入，任何同构的树都被视为是相同的。

所输入的 m 棵树中可能存在彼此同构的树；如果去除这些重复的树（即每种同构的树只留下一个），它们可以构成一个树的集合 $\mathscr{T}$。你需要判定这一树的集合所生长得到的集合 $\mathrm{grow}(\mathscr{T})$ 是否是几乎完备的。

【输出格式】

输出到文件 ***surreal.out*** 中。

输出包含 N 行，分别表示 N 组测试数据的答案。其中，第 i 行输出一个字符串：如果第 i 组测试数据所输入的树的集合所生长得到的集合是几乎完备的（换言之，仅有有限棵树不能被其生长得到），那么输出 `Almost Complete`；否则输出 `No`。**请注意输出字符串的拼写和大小写**。

【样例 1 输入】

```
1
1
1
0 0
```

【样例 1 输出】

```
Almost Complete
```

【样例 1 解释】

这一样例仅包含一组测试数据，其中树的集合 $\mathscr{T}$ 仅包含一棵由单个结点构成的树。由于单个结点可以删去唯一的叶子结点，一步替换得到任何树，$\mathrm{grow}(\mathscr{T})$ 包含了所有树，自然是几乎完备的。

【样例 2 输入】

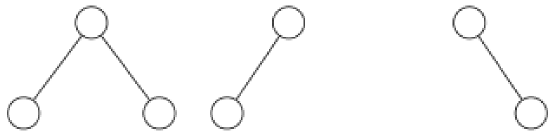
```
1
3
3
2 3
0 0
0 0
2
2 0
0 0
2
0 2
0 0
```

【样例 2 输出】

```
Almost Complete
```

【样例 2 解释】

这一样例仅包含一组测试数据，其中树的集合 $\mathscr{T}$ 包含三棵树，如下图所示。容易发现，仅有单个结点构成的树不在 $\mathrm{grow}(\mathscr{T})$ 中，其包含了几乎所有树，因而是几乎完备的。



【样例 3 输入】

```
1
2
3
2 3
0 0
0 0
2
2 0
0 0
```

【样例 3 输出】

```
No
```

【样例 3 解释】

这一样例仅包含一组测试数据，其中树的集合 $\mathscr{T}$ 包含两棵树。容易发现，对于所有的 $n \geq 2$，包含 n 个结点，每个非叶结点仅有右孩子的链状树都不在 $\mathrm{grow}(\mathscr{T})$ 中，因而存在无穷多棵树不在 $\mathrm{grow}(\mathscr{T})$ 中，$\mathscr{T}$ 不是几乎完备的。

【样例 4】

见选手目录下的 ***surreal/surreal4.in*** 与 ***surreal/surreal4.ans***。

【样例 5】

见选手目录下的 ***surreal/surreal5.in*** 与 ***surreal/surreal5.ans***。

【样例 6】

见选手目录下的 ***surreal/surreal6.in*** 与 ***surreal/surreal6.ans***。

【测试点约束】

全部数据满足：$\sum n \le 2 \times 10^6$，$\sum m \le 2 \times 10^6$，$\max h \le 2 \times 10^6$，$T \le 10^2$。其中，$\sum n$ 表示这一测试点所有测试数据中所出现的所有树的结点个数之和；$\sum m$ 表示这一测试点中所有测试数据中所出现的树的个数；$\max h$ 表示这一测试点中所出现的所有树的最高高度（仅包含一个结点的树高度为 1）。下表中的表项 $\sum n$，$\sum m$ 和 $\max h$ 含义与上面相同，描述了每一组测试点的数据范围。

特殊性质：下面是下表中会涉及的四种特殊性质的解释。

- 特殊性质 1：对于这一测试点中的每一组测试数据，都有 $m \le 4$，即树的集合中包括不超过 4 棵树；
- 特殊性质 2：对于这一测试点中的每一组测试数据，树的集合中所有的树具有相同的高度；
- 特殊性质 3：对于这一测试点中的每一组测试数据，树的集合仅包含链（换言之，每个非叶结点仅包含一个孩子）；
- 特殊性质 4：对于这一测试点中的每一组测试数据，树的集合仅包含满足以下两个条件之一的树：
  - 每个非叶结点仅包含一个孩子；
  - 恰好有两个叶结点，它们具有相同的父结点，并且除这三个结点外，其余结点均有且仅有一个孩子。

每个测试点的具体限制见下表：

<table>
<tr><th>测试点编号</th><th>N</th><th>$\sum n$</th><th>$\sum m$</th><th>$\max h$</th><th>特殊性质</th></tr>
<tr><td>1</td><td rowspan="9">100</td><td rowspan="3">≤ 1000</td><td rowspan="3">≤ 1000</td><td>≤ 1</td><td>无</td></tr>
<tr><td>2</td><td rowspan="2">≤ 2</td><td rowspan="2">性质 1</td></tr>
<tr><td>3</td></tr>
<tr><td>4</td><td rowspan="6">≤ 1000000</td><td rowspan="6">≤ 1000000</td><td>≤ 4</td><td>无</td></tr>
<tr><td>5</td><td>≤ 5</td><td>性质 2</td></tr>
<tr><td>6</td><td>≤ 8</td><td>无</td></tr>
<tr><td>7</td><td>≤ 9</td><td>性质 2</td></tr>
<tr><td>8</td><td>≤ 10</td><td>无</td></tr>
<tr><td>9</td><td>≤ 1000000</td><td>性质 3</td></tr>
<tr><td>10</td><td rowspan="7">20</td><td>≤ 1000</td><td>≤ 100</td><td>≤ 1000</td><td rowspan="4">性质 4</td></tr>
<tr><td>11</td><td>≤ 2000</td><td>≤ 2000</td><td>≤ 2000</td></tr>
<tr><td>12</td><td>≤ 100000</td><td>≤ 100000</td><td>≤ 100000</td></tr>
<tr><td>13</td><td>≤ 200000</td><td>≤ 200000</td><td>≤ 200000</td></tr>
<tr><td>14</td><td>≤ 800</td><td>≤ 200</td><td>≤ 800</td><td rowspan="12">无</td></tr>
<tr><td>15</td><td>≤ 1000</td><td>≤ 100</td><td>≤ 1000</td></tr>
<tr><td>16</td><td>≤ 2000</td><td>≤ 2000</td><td>≤ 2000</td></tr>
<tr><td>17</td><td rowspan="9">40</td><td>≤ 300000</td><td>≤ 300000</td><td>≤ 300000</td></tr>
<tr><td>18</td><td>≤ 600000</td><td>≤ 600000</td><td>≤ 600000</td></tr>
<tr><td>19</td><td>≤ 900000</td><td>≤ 900000</td><td>≤ 900000</td></tr>
<tr><td>20</td><td>≤ 1200000</td><td>≤ 1200000</td><td>≤ 1200000</td></tr>
<tr><td>21</td><td>≤ 1500000</td><td>≤ 1500000</td><td>≤ 1500000</td></tr>
<tr><td>22</td><td rowspan="4">≤ 2000000</td><td rowspan="4">≤ 2000000</td><td rowspan="4">≤ 2000000</td></tr>
<tr><td>23</td></tr>
<tr><td>24</td></tr>
<tr><td>25</td></tr>
</table>

翻修道路（road）

【题目描述】

C 国中包含 n 座城市，这些城市通过 m 条双向道路连接。城市从 1 到 n 编号，道路从 1 到 m 编号，i 号道路两端连接着城市 u_i 与城市 v_i，它的长度为 w_i 米。经由这些道路，从 C 国中任意一个城市出发，均能到达其他所有城市。

C 国人民喜欢环路旅程，但又不喜欢经过太多条道路，为此 C 国的道路被建造得非常特殊。更具体地，对于一条经过 l 条道路的简单环路（即除起点城市外**不经过重复城市**的环路），它可以表示为 $c_1 \to c_2 \to \cdots \to c_l \to c_1$（其中对于所有 $1 \le i < l$，城市 c_i 与城市 c_{i+1} 有道路相连；城市 c_l 与城市 c_1 有道路相连；对于所有 $1 \le i < j \le l$，有 $c_i \ne c_j$），若 $l > 3$，则 C 国的道路将满足下列条件：

- 存在两个在该环路上**不相邻**的城市 u, v，满足两个城市间有道路直接相连。即：存在 $1 \le u < v \le l$，使得 $v - u \ge 2$，u 和 v 不同时为 1 和 l，并且城市 c_u 与城市 c_v 间有道路直接相连。

现在 C 国有了新的翻修计划，需要在城市 s 与城市 t 间寻找一条路径进行翻修。翻修时路径中包含的所有道路将无法通行，为了保障人民的日常生活，C 国希望在翻修这条路径时，经由**剩余**的道路（即没被包含在翻修路径内的道路）依然能满足：**从 C 国中任意一个城市出发，均能到达其他所有城市**。

C 国找到了身为工程大师的你，请你帮助 C 国找出一条满足上述要求的翻修路径，并使得这条路径的总长**尽量小**。

【输入格式】

从文件 ***road.in*** 中读入数据。

第一行两个整数 n, m 分别表示城市个数与道路条数。

接下来 m 行每行三个整数 u_i, v_i, w_i，依次表示每条道路的两个端点与它的长度。

数据保证每条道路都一定连接两个不同城市，即 $u_i \ne v_i$。

最后一行两个整数 s, t，分别表示需要翻修的路径的两个端点。

【输出格式】

输出到文件 ***road.out*** 中。

仅一行一个整数，表示满足题目要求的情况下，翻修路径的总长的最小值。

如果不存在满足题目要求的路径，输出一行一个整数 -1。

【样例 1 输入】

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

【样例 1 输出】

```
6
```

【样例 1 解释】

路径 $(1,2,1),(2,3,1),(3,4,1)$ 是城市 1 和城市 4 间总长最小的路径，但不符合要求。
路径 $(1,3,5),(3,4,1)$ 符合要求，长度为 6。
路径 $(1,2,1),(2,4,6)$ 符合要求，长度为 7。
除上述两条路径外，没有其他满足要求的路径。

【样例 2 输入】

```
2 1
1 2 1
1 2
```

【样例 2 输出】

```
-1
```

【样例 3】

见选手目录下的 $\boldsymbol{road/road3.in}$ 与 $\boldsymbol{road/road3.ans}$。
该样例与测试点 $1 \sim 6$ 限制相同。

【样例 4】

见选手目录下的 ***road/road4.in*** 与 ***road/road4.ans***。
该样例与测试点 $7 \sim 10$ 限制相同。

【样例 5】

见选手目录下的 ***road/road5.in*** 与 ***road/road5.ans***。
该样例与测试点 $11 \sim 15$ 限制相同。

【样例 6】

见选手目录下的 ***road/road6.in*** 与 ***road/road6.ans***。
该样例与测试点 $16 \sim 20$ 限制相同。

【测试点约束】

对于所有测试点：$2 \le n \le 5 \times 10^5$，$2 \le m \le 10^6$，$s \ne t$。
$1 \le u_i, v_i \le n$，$u_i \ne v_i$，$1 \le w_i \le 10^9$，保证任意两条道路它们的端点不全相同。
保证给出的道路满足题面描述第二段中的性质。
每个测试点的具体限制见下表：

测试点编号	$n \le$	$m \le$	特殊限制
$1 \sim 6$	2000	4000	无
$7 \sim 10$	5×10^5	10^6	A
$11 \sim 15$			B
$16 \sim 20$			无

特殊限制 A：所有道路的长度均相等。

特殊限制 B：所有 $w_i = 1$ 的道路恰好构成 s 到 t 的一条路径，且其他 $w_i \ne 1$ 的道路的两条端点在这条路径上距离为 2。