亚太地区信息学奥林匹克竞赛

2008 年 5 月 10 日

题目名称	珠链交换器	免费道路	DNA
英文名	**beads**	**roads**	**dna**
输入方式	标准输入		
输出方式	交互式	标准输出	
时间限制	2 秒	1 秒	1 秒
空间限制	256MB	128MB	128MB
分值	100	100	100
	300		

语言	编译参数
C	gcc -o abc abc.c -std=c99 -O2 -DCONTEST -s -static -lm
C++	g++ -o abc abc.cpp -O2 -DCONTEST -s -static
Pascal	fpc -O1 -XS -dCONTEST abc.pas

比赛时间：3 道题共 5 个小时

珠链交换器

beads

X 教授展示了他的最新发明：珠链交换器(UBS)。顾名思义，就是一个能通过交换一些珠子使珠链更有趣的工具。

UBS 有 N 条平行于南北方向的传送带，已按照从左到右的顺序 1 至 N 编号。每条传送带的速度相同，且都是由北向南传送。有 M 个交换器，每个交换器都安放在某两个相邻的传送带上，任意两个交换器与传送带最北端的距离都是不同的(也就是说，所有的交换器可以按照它们与传送带最北端的距离严格排序)。所有的交换器按从北到南的顺序 1 到 M 编号。图 1 为一个 UBS 的俯视图：

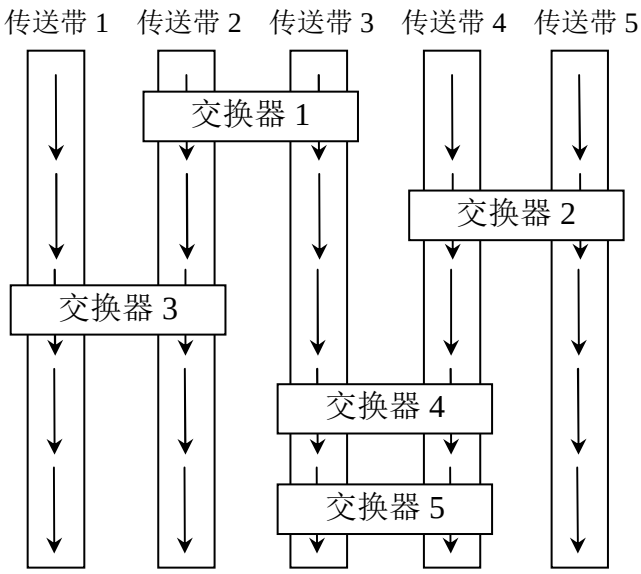
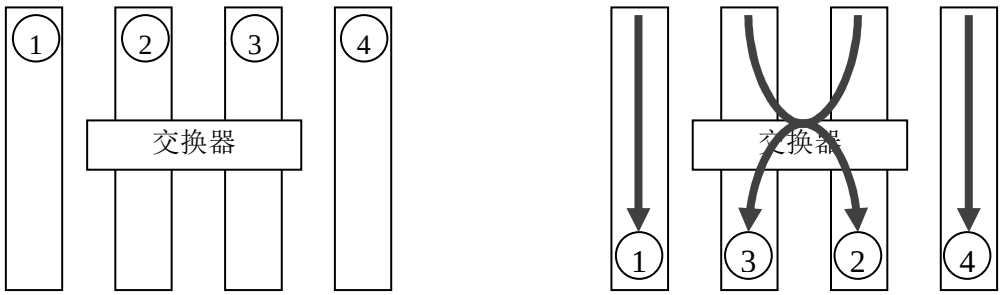


图 1 一个拥有 5 条传送带和 5 个交换器的 UBS

在使用 UBS 的时候，将 N 个珠子分别放在每条传送带的最北端，这样在传送带向南传送时，所有的珠子一直保持在同一条水平横线上。当两颗珠子同时到达一个交换器时，在右边传送带上的珠子被移动到左边的传送带上，同时左边传送带上的珠子被移动到右边的传送带上。经交换后，这两个珠子仍保持与其他珠子在一条水平线上。图 2 所示为一次交换的过程：



(a) (b)

图 2 (a)4 颗珠子被传送带传送 (b)在经过交换器后，2 号和 3 号珠子交换了顺序

任务

编写一个程序，对于给定的传送带条数 N，交换器个数 M，以及每个交换器的位置，回答以下形式的问题：

给出 K（传送带的编号）和 J（交换器的编号），当所有的珠子都刚好完全经过交换器 J（又称 J 号交换器）时，最开始时放在第 K 条传送带（又称 K 号传送带）最北端的珠子现在位于哪一条传送带上。

输入格式

从标准输入读入数据。第一行包含传送带的条数 $N(1 \le N \le 300{,}000)$和交换器的个数 $M(1 \le M \le 300{,}000)$。

接下来的 M 行，按照从北向南的顺序描述所有的交换器，每行包含一个整数 P（$1 \le P \le M-1$），表示在传送带 P 和 $P+1$ 之间有一个交换器。

交互方法

在读入上述数据之后，你的程序需要和表格 1 中所描述的库进行交互。库中的函数必须按下面的顺序调用：

1. 调用 getNumQuestions 以获取 $Q(1 \le Q \le 300{,}000)$，表示有多少个问题。
2. 以后 Q 次，每次

   (a) 调用函数 getQuestion 来获取一个问题。

   (b) 调用函数 answer 回答对应的问题。

特别注意，getNumQuestions 必须首先调用，且只能调用 1 次。getQuestion 和 answer 必须交替调用：当调用完 getQuestion 之后，你的程序不能马上再次调用 getQuestion，而必须先调用 answer，反之亦然。如果你的程序在某个测试点

违反了这一约定，则该测试点你的得分为 0。

函数原型	说明
Pascal function getNumQuestions(): integer C 和 C++ int getNumQuestions()	返回你的程序应该回答的问题个数
Pascal procedure getQuestion(var K: integer, var J: integer) C void getQuestion(int *K, int *J) C++ void getQuestion(int &K, int &J)	*K* 为珠子在最北端时的传送带编号 *J* 为交换器的编号
Pascal procedure answer(x:integer) C 和 C++ void answer(int x)	x 给出上一个 getQuestion 对应的答案

表 1　交互库

程序指令

如果你提交 Pascal 的程序，在你的代码中必须有如下的声明：

uses beadslib;

如果你提交 C 或 C++的程序，在你的代码中必须有如下的声明：

#include “beadslib.h”

伪函数库与样例程序

将为你提供一个包含样例库的源文件和样例的调用。这些文件在一个压缩包中，包含三个目录——pascal, c 和 cpp，分别是 Pascal,C 和 C++的源文件，每个目录中包含一个伪交互库的源文件和一个按正确顺序调用交互库的程序。

如果你使用 Pascal，伪交互库在一个名字为 beadslib 的单元中，其源文件为 beadslib.pas。文件 sample.pas 是使用这个库的样例。

如果你使用 C 语言，伪交互库的函数包含在 beadslib.h 中，函数在 beadslib.c 中，文件 sample.c 是使用这个库的样例。

如果你使用 C++语言，伪交互库的函数也包含在 beadslib.h 中(但这个文件和 C 语言对应的文件不一样)，函数实现在 beadslib.cpp 中，文件 sample.cpp 是使用这个库的样例。

伪交互库的执行方式如下：

- 当伪交互库的 getNumQuestions 被调用时，它打开文件 questions.txt，读出有多少个问题，并返回读到的问题个数。
- 当 getQuestion 被调用时，函数库文件中读出两个整数 *K* 和 *J*。
- 当 answer 被调用时，将参数 x 输出到标准输出中。
- 当调用函数的顺序不正确时，函数库输出一行错误信息到标准输出。

questions.txt 的格式如下：第一行包含问题的个数 *Q*，接下来 *Q* 行，每行两个整数 *K* 和 *J*，*K* 表示传送带的编号，*J* 表示交换器的编号。

样例输入

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

(此输入对应图 1)

questions.txt 样例

```
2
3 4
5 5
```

样例交互

函数调用	返回值和解释
getNumQuestions();	2 程序需要回答两个问题

Pascal
getQuestion(K, J);
C
getQuestion(&K, &J);
C++
getQuestion(K, J);
	K=3，*J*=4
	对于最开始时放在 3 号传送带最北端的珠子，当刚好经过 4 号交换器（的水平位置）后，它位于哪个传送带上？

answer(1)	当所有珠子经过 4 号交换器后，问题中所说的珠子位于 1 号传送带上

Pascal
getQuestion(K, J);
C
getQuestion(&K, &J);
C++
getQuestion(K, J);
	K=5，*J*=5
	对于最开始时放在 5 号传送带最北端的珠子，当刚好经过 5 号交换器（的水平位置）后，它位于哪个传送带上？

answer(4)	当所有珠子经过 5 号交换器后，问题中所说的珠子位于 4 号传送带上

时间和空间限制

你的程序必须在 2 秒钟内运行结束，至多使用 256M 内存。

评分

对于每个测试点，如果你的程序遵守调用约定且正确的回答了每一个问题，那么你将得到满分，否则得 0 分。

至少有 20 分的测试点中，M 和 Q 都不超过 10,000。

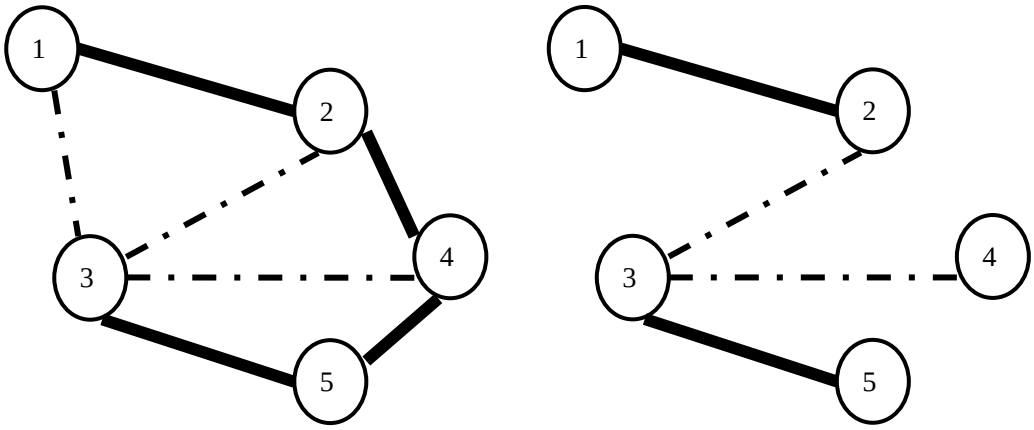
免费道路

roads

新亚（New Asia）王国有 *N* 个村庄，由 *M* 条道路连接。其中一些道路是鹅卵石路，而其它道路是水泥路。保持道路免费运行需要一大笔费用，并且看上去王国不可能保持所有道路免费。为此亟待制定一个新的道路维护计划。

国王已经决定保持尽可能少的道路免费，但是每两个不同的村庄之间都应该**由一条且仅由一条**免费道路的路径连接。同时，虽然水泥路更适合现代交通的需要，但国王也认为走在鹅卵石路上是一件有趣的事情。所以，国王决定保持刚好 *K* 条鹅卵石路免费。

举例来说，假定新亚王国的村庄和道路如图 3(a)所示。如果国王希望保持两条鹅卵石路免费，那么可以如图 3(b)中那样保持道路(1, 2)、(2, 3)、(3, 4)和(3, 5)免费。该方案满足了国王的要求，因为：(1)每两个村庄之间都有一条由免费道路组成的路径；(2)免费的道路已经尽可能少；(3)方案中刚好有两条鹅卵石道路(2, 3)和(3, 4)。



(a) (b)

图 3: (a)新亚王国中村庄和道路的一个示例。实线标注的是水泥路，虚线标注的是鹅卵石路。(b)一个保持两条鹅卵石路免费的维护方案。图中仅标出了免费道路。

任务

给定一个关于新亚王国村庄和道路的描述以及国王决定保持免费的鹅卵石道路数目，写一个程序确定是否存在一个道路维护计划以满足国王的要求，如果存在则任意输出一个方案。

输入格式

输入第一行包含三个由空格隔开的整数：

- N，村庄的数目($1 \le N \le 20,000$)；
- M，道路的数目($1 \le M \le 100,000$)；
- K，国王希望保持免费的鹅卵石道路数目($0 \le K \le N - 1$)。

此后 M 行描述了新亚王国的道路，编号分别为 1 到 M。第(i+1)行描述了第 i 条道路的情况。用 3 个由空格隔开的整数描述：

- u_i 和 v_i，为第 i 条道路连接的两个村庄的编号，村庄编号为 1 到 N；
- c_i，表示第 i 条道路的类型。$c_i = 0$ 表示第 i 条道路是鹅卵石路，$c_i = 1$ 表示第 i 条道路是水泥路。

输入数据保证一对村庄之间至多有一条道路连接。

输出格式

如果满足国王要求的道路维护方案不存在，你的程序应该在输出第一行打印 **no solution**。

否则，你的程序应该输出一个符合要求的道路维护方案，也就是保持免费的道路列表。按照输入中给定的那样输出免费的道路。如果有多种合法方案，你可以任意输出一种。

输入样例（见图 3(a)）

```
5 7 2
1 3 0
4 5 1
3 2 0
5 3 1
4 3 0
1 2 1
4 2 1
```

输出样例（见图 3(b)）

```
3 2 0
4 3 0
5 3 1
1 2 1
```

时间和空间限制

你的程序必须在 1 秒钟内结束运行并且使用不超过 128MB 内存。

评分

对于每组输入数据，如果你的程序输出正确便能得到 100%的分数，否则得 0 分。

在 20 分的测试数据中，K 的值最多为 10。

DNA

dna

分析如 DNA 序列这样的生命科学数据是计算机的一个有趣应用。从生物学的角度上说，DNA 是一种由腺嘌呤、胞嘧啶、鸟嘌呤和胸腺嘧啶这四种核苷酸组成的链式结构。这四种核苷酸分别用大写字母 **A**、**C**、**G**、**T** 表示。这样，一条 DNA 单链可以被表示为一个只含以上四种字符的字符串。我们将这样的字符串称作一个 *DNA 序列*。

有时生物学家可能无法确定一条 DNA 单链中的某些核苷酸。在这种情况下，字符 **N** 将被用来表示一个不确定的核苷酸。换句话说，**N** 可以用来表示 **A**、**C**、**G**、**T** 中的任何一个字符。我们称包含一个或者多个 **N** 的 DNA 序列为*未完成序列*；反之，就称作*完成序列*。如果一个完成序列可以通过将一个未完成序列中的每个 **N** 任意替换成 **A**、**C**、**G**、**T** 得到的话，就称完成序列*适合*这个未完成序列。举例来说，**ACCCT** 适合 **ACNNT**，但是 **AGGAT** 不适合。

研究者们经常按照如下方式排序四种核苷酸：**A** 优先于 **C**，**C** 优先于 **G**，**G** 优先于 **T**。如果一个 DNA 序列中的每个核苷酸都与其右边的相同或者优先，就将其归类为*范式*-1。举例来说，**AACCGT** 是范式-1，但是 **AACGTC** 不是。

一般来说，一个 DNA 序列属于*范式*-*j* ($j > 1$)，只要它属于范式-(j-1)或者是一个范式-(j-1)和一个范式-1 的连接。举例来说，**AACCC**、**ACACC** 和 **ACACA** 都是范式-3，但 **GCACAC** 和 **ACACACA** 不是。

同样，研究者们按照字典序对 DNA 序列进行排序。按照这个定义，最小的属于范式-3 的 DNA 序列是 **AAAAA**，最大的是 **TTTTT**。这里是另外一个例子，考虑未完成序列 **ACANNCNNG**。那么前 7 个适合这个未完成序列的 DNA 序列是：

ACAAACAAG
ACAAACACG
ACAAACAGG
ACAAACCAG
ACAAACCCG
ACAAACCGG
ACAAACCTG

任务

写一个程序，找到按字典序的第 R 个适合给定的长度为 M 的未完成序列的范式-K。

输入格式

输入第一行包含三个由空格隔开的整数：$M(1\le M\le 50{,}000)$，$K(1\le K\le 10)$和$R(1\le R\le 2\times 10^{12})$。第二行包含一个长度为 M 的字符串，表示未完成序列。保证适合该未完成序列的范式-K 的总数不超过 4×10^{18}，因此该数可以用 C 和 C++中的 long long 类型或者 Pascal 中的 Int64 类型表示。同时，R 不会超过适合给定未完成序列的范式-K 的总数。

输出格式

在第一行中输出第 R 个适合输入中的未完成序列的范式-K。

输入样例 1

```
9 3 5
ACANNCNNG
```

输出样例 1

```
ACAAACCCG
```

输入样例 2

```
5 4 10
ACANN
```

输出样例 2

```
ACAGC
```

编程提示

在 C 和 C++中，你应该使用 long long 数据类型。下面的程序片断给出了如何从标准输入中读写 long long：

```
long long a;
scanf("%lld",&a);
printf("%lld\n",a);
```

在 Pascal 中，你应该使用 Int64 类型。

时间和空间限制

你的程序必须在 1 秒钟内运行结束并且使用不超过 128MB 内存。

评分

对于每组输入数据，如果你的程序输出正确便得到 100%的分数，否则得 0 分。

在 20 分（20%）的测试数据中，M 的值最多为 10。