
少年圖靈計畫
Young Turing Program

1_打倒怪獸 - Defeat the monster

(10分)

問題敘述

為了擊敗想要毀滅世界的怪獸 - 「WA」，身為大魔導師「AC」的徒弟的你正在幫忙他尋找可以擊敗怪獸的咒語，幸運的是，AC 對這個咒語依稀有點印象，所以他希望你從魔導書裡面找出所有和他記憶裡的咒語相似的連續片段，相似指的是兩個字串裡最多只有一個字元不同，也就是對兩個字串 s_1, s_2 ，最多只有一個 i 滿足 $s_1[i] \neq s_2[i]$

輸入格式

輸入有兩行：
第一行輸入是大魔導師記憶裡的咒語，長度為 N。
第二行輸入是魔導書的內容，長度為 M。

輸出格式

請依照字典序輸出魔法書裡所有和咒語相似的連續片段，每個相似的連續片段之間以換行做分隔，如果沒有相似的片段則輸出「Not found」（不含引號）

資料範圍

- $1 \leq NM \leq 10^6$
- $1 \leq N \leq M \leq 10^6$
- 所有的輸入都是小寫英文字母

輸入範例1

```
key
kaykeykay
```

輸出範例1

```
kay
kay
key
```

輸入範例2

```
ytp
yyppabcdefgydp
```

輸出範例2

```
ydp
ypp
yyp
```

輸入範例3

```
acac
watlerecemle
```

輸出範例3

```
Not found
```

範例說明

在範例一中，"kay" 和咒語 "key" 只有第二個字元不同、"key" 和 "key" 完全相同， "kay" 和 "key" 也只有第二個字元不同，所以結果就是這三個字串按字典序排列後輸出。

在範例二中，"yyp" 和 "ytp" 只有第二個字元、 "ypp" 跟 "ydp" 也是只有第二個字元不同，所以把他們三個排序後輸出即可。

在範例三中，"watlerecemle" 中並不存在與 "acac" 相似的字串。

1_Defeat the monster

(10 points)

Description

To defeat the monster "WA", which wants to destroy the world. You, the student of the grand wizard "AC", are helping him to find the spell to defeat the monster. Luckily, "AC" still remembers the spell roughly, so he asks you to look for all contiguous fragment with the same length which is similar to the spell in his memory in a spellbook. "Similar" means that there is at most one character different in two strings. That is, for two string s_1, s_2, there is at most one i such that $s_1[i] \neq s_2[i]$.

Input Format

The input contains two lines.
The first line contains a string, which is the spell in AC's memory. The length of it is N.
The second line contains a string, which is the content of the spellbook. The length of it is M.

Output Format

Please print all contiguous fragment in the spellbook which is similar to the spell in AC's memory in lexicographical order. Two fragments are split by newline.
If there is no fragment similar to AC's memory, just print "Not found" (without double quotation mark).

Constraints

- $1 \leq NM \leq 10^6$
- $1 \leq N \leq M \leq 10^6$
- all characters are lower case alphabet

Input Example 1

```
key
kaykeykay
```

Output Example 1

```
kay
kay
key
```

Input Example 2

```
ytp
yyppabcdefgydp
```

Output Example 2

```
ydp
ypp
yyp
```

Input Example 3

```
acac
watlerecemle
```

Output Example 3

```
Not found
```

Example Explanation

In example 1, There is only one difference between "kay" and "key".
"key" and "key" are the same.
There is only one difference between "kay" and "key".
So, just print them in lexicographical order.

In example 2, There is only one difference between "yyp" and "ytp", so "yyp" is similar to "ytp". Similarly, "ypp" and "ydp" are similar to "ytp", too.

In example 3, There is no substring in "watlerecemle" that is similar to "acac".

2_隨機序列 - Random Sequence Generator

(10分)

問題敘述

小宏最喜歡隨機數列了！他使用的數字系統是 N-進位的系統，位數由十進位的 0 到 $N-1$ 表示。他總是喜歡自己在紙上自己嘗試生出隨即的數列。他生隨機數列的規則是：每一個數字都要出現一樣的次數。然而，他最近發現這個規則生出來的數列並不是非常的隨機。舉例來說，對於 $N=10$，有這些符合他的規則，但是不怎麼隨機的數列。

- `0123456789`
- `01234567890123456789`
- `00011122233344455566677788899`

所以呢，他制定了一個新的規則：對於每兩個**相鄰**的數字，他們的出現次數也要相同！譬如說，出現 `00`、`10`、`71`、`22` 的次數都要相同。請幫小宏生出一個符合他的新規定的隨機數列吧！他也不希望你生出來的序列太長，所以請限制輸出的序列長度不超過 10^6。請注意，他第一個規則已經作廢，所以新的序列的每一個數字的出現次數**不一定**要相同。

保證在給定的輸入範圍中，一定存在至少一個符合條件的解答。

輸入格式

輸入僅有一個數字 N（$2 \leq N \leq 500$），代表小宏使用 N-進位的數字系統。

輸出格式

第一行請輸出一個數字 K，代表你的序列的長度。第二行請輸出 K 個空白分開的數字 x_i，代表所生的隨機數列。

請滿足：

- $2 \leq K \leq 10^6$
- 對於所有的 $1 \leq i \leq K$，都有 $0 \leq x_i < N$
- 序列中所有相鄰元素的出現次數都相同

於眾多可能的答案中，輸出任何一解即可。

資料範圍

- $1 \leq N \leq 500$

輸入範例 1

```
2
```

輸出範例 1

```
9
0 0 0 1 1 0 1 1 0
```

輸入範例 2

```
3
```

輸出範例 2

```
19
0 0 0 1 0 2 1 0 1 1 1 2 2 0 2 1 2 2 0
```

輸入範例 3

```
4
```

輸出範例 3

```
33
0 0 0 1 0 2 0 3 1 0 1 1 1 2 1 3 2 0 2 1 2 2 2 3 3 0 3 1 3 2 3 3 0
```

範例說明

對於第一筆範例側資，00、01、10、11 都各出現了兩次。

對於第二筆範例側資，00、01、02、10、11、12、20、21、22 皆出現了兩次。請注意，這並不是最佳解：長度為 10 的 `0011220210` 也可以是答案。

2_Random Sequence Generator

(10 points)

Description

Ryan loves generating random number sequences! He uses a base-N numbering system where digits are marked with 0 to $N-1$ in decimal. His rule for generating random sequences is that all digits must appear with the same frequency. However, he's noticed that his rule doesn't hold much water: when $N=10$, non random-looking sequences which follow his rules like these appear:

- `0123456789`
- `01234567890123456789`
- `00011122233344455566677788899`

Therefore, he's come up with a new rule: that all two neighbouring digits must too appear with the same frequency! In this case, combinations such as `71`, `22`, `00`, `07` must appear with the same frequency. Can you write a program to generate such a sequence? Note that he has now completely ignored the first rule - the digits do **not** have to appear with the same frequency.

In addition, Ryan does not like his sequences to be too long - your generated sequence must not exceed 10^6 elements.

It can be proved that such a sequence always exists for the given input range.

Input Format

There is only one integer N ($2 \leq N \leq 500$) on the first line, which denotes the base system Ryan uses.

Output Format

On the first line, please output a number K denoting the length of your sequence. Then on the second line, output K space-separated numbers x_i denoting the elements of the sequence.

Your sequence must satisfy the following requirements:

- $2 \leq K \leq 10^6$;
- For all $1 \leq i \leq K$, $0 \leq x_i < N$;
- All consective two elements appear with the same frequency.

Any solution satisfying the above constraints will be judged as correct.

Constraints

- $1 \leq N \leq 500$

Input Example 1

```
2
```

Output Example 2

```
9
0 0 0 1 1 0 1 1 0
```

Input Example 2

```
3
```

Output Example 2

```
19
0 0 0 1 0 2 1 0 1 1 1 2 2 0 2 1 2 2 0
```

Input Example 3

```
4
```

Output Example 3

```
33
0 0 0 1 0 2 0 3 1 0 1 1 1 2 1 3 2 0 2 1 2 2 2 3 3 0 3 1 3 2 3 3 0
```

Example Explanation

For the first test, 00, 01, 10, 11 all appear twice.

For the second test, 00, 01, 02, 10, 11, 12, 20, 21, 22 also all appear twice. Note that this is not the shortest or the only answer: you might also find `0011220210`, which has length 10.

3_黑白舞蹈機 - Dance Dance Revolution

(15分)

(子任務1: 6分, 子任務2: 9分)

問題敘述

小 B 是一個喜歡跳舞的小孩，他時常隨著音樂進行有節奏的舞蹈。一天，他在遊樂場看到了一台「黑白舞蹈機」，對跳舞充滿興致的他，二話不說就開始挑戰它。「黑白舞蹈機」中一個關卡是由 N 個燈泡以及 M 個踏板組成，每個踏板都可以讓某兩個不同的燈泡改變亮暗，一開始有一些燈泡會是亮著的狀態，遊戲的目標是讓所有燈泡都變成暗的，除此之外，因為踩踏重複的踏板會讓小 B 顯得看起來沒有韻律，所以小 B 給自己設下的限制是不能踩踏已經踩過的踏板。小 B 想知道他必須依序踩踏哪些踏板才能達成目標，又或者是這個關卡一開始就心懷不軌讓人無法破關？

輸入

輸入第一行有兩個正整數 N, M ($1 \leq N \leq 10^5, 0 \leq M \leq 2 \times 10^5$)，分別代表燈泡數量以及踏板數量，燈泡的編號分別為 $1, 2, \ldots, N$，踏板的編號分別為 $1, 2, \ldots, M$。

輸入第二行有 N 個整數 c_i ($c_i \in \{0, 1\}$)，代表第 i 個燈泡的初始狀態，0 代表該燈泡為暗，1 代表該燈泡為亮。

接下來 M 行，第 i 行有兩個正整數 x, y ($1 \leq x, y \leq N,\ x \neq y$)，代表踩下編號為 i 的踏板會改變編號 x 與 y 的燈泡狀態。

輸出

若不存在一系列踩踏板操作使得所有燈泡都變成暗的，請在第一行輸出 `No`，否則輸出 `Yes`。

若第一行輸出 `Yes`，請在第二行輸出一個整數，代表踩的踏板數量，第三行請依先後順序輸出小 B 應該踩踏的踏板編號，請注意不能重複踩踏相同的踏板。若有多種踩踏板的方法讓所有燈泡變暗，請任意輸出一種方案即可。

子任務

子任務1: 保證以所有 (x, y) 為邊的圖會是一條鏈

範例輸入1

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

範例輸出1

```
Yes
1
5
```

範例輸入2

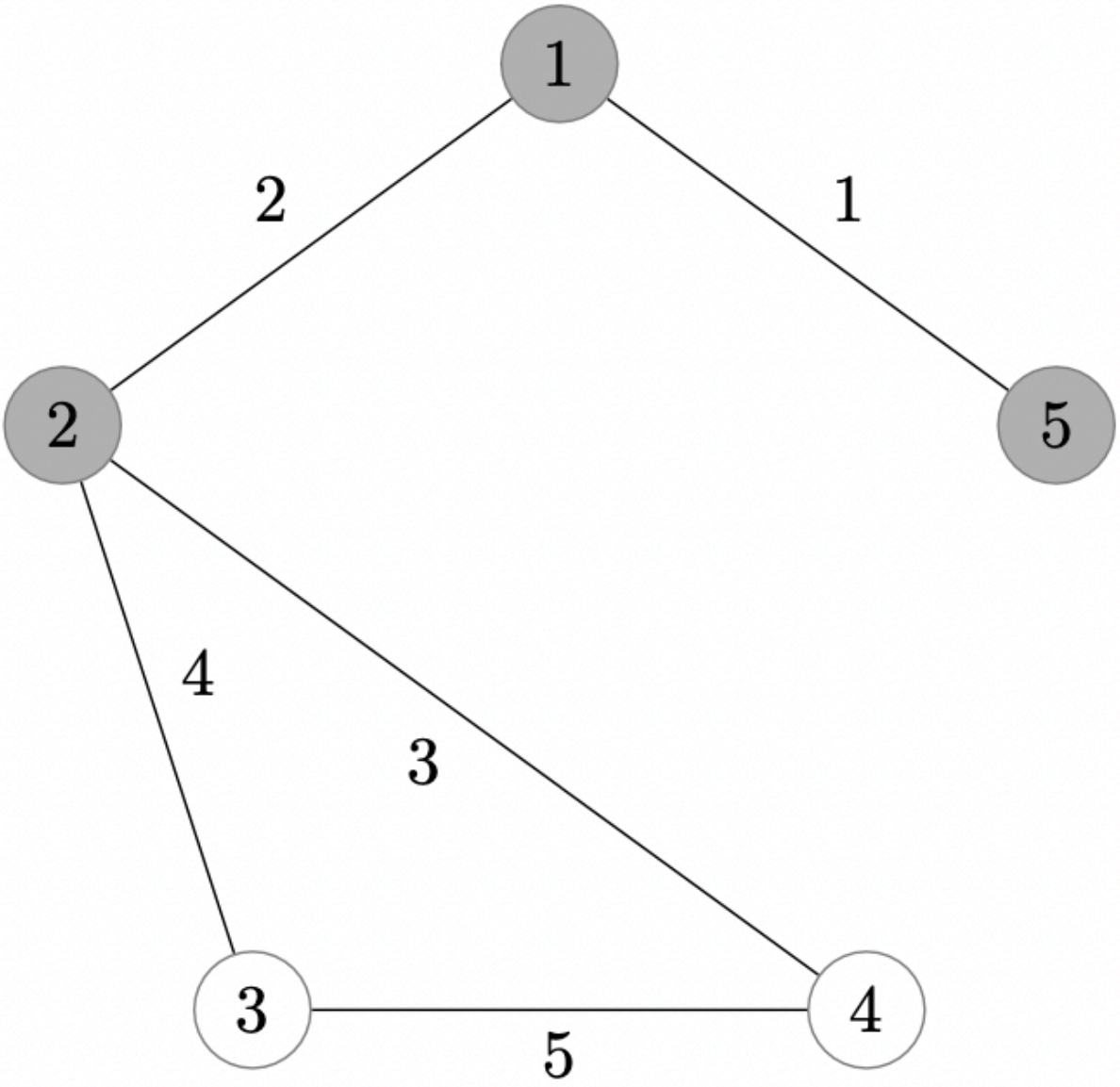
```
5 5
1 0 1 1 0
1 5
2 1
4 2
2 3
4 3
```

範例輸出2

```
No
```

範例說明

範例輸入1可參考下圖，踩下第 5 個踏板可改變第 3 及第 4 個燈泡狀態，即可讓所有燈泡都變成暗的。



3_Dance Dance Revolution

(15 points)

(Subtask 1: 6 points, Subtask 2: 9 points)

Description

Bob, a fanatic dancer, loves to dance rhythmically with the music. One day, he discovered a fantastic DDR, also known as "Light or Dark? Dance Dance Revolution". Enthusiastic about the dance as Bob is, he immediately started the challenge. In "Light or Dark? Dance Dance Revolution", each stage consists of N lamps and M pedals. Each pedal connects two different lamps so that it can change the state of these lamps. Initially, there are some lamps turning light, and other lamps turning dark. The goal of the game is to darken all lamps. Besides, Bob hopes his dance is metrical, so he cannot tap the duplicate pedals. Under these constraints, could Bob finish the target of the game? Please help him find a solution.

Input

The first line of input contains two integers N, M ($1 \leq N \leq 10^5, 0 \leq M \leq 2 \times 10^5$), representing the number of lamps and pedals, respectively. The lamps are indexed from 1 to N, while the pedals are indexed from 1 to M.

The second line of input contains N integers c_i ($c_i \in \{0, 1\}$), describing the initial state of i^{th} lamps. 0 means that the lamp is dark, and vice versa.

In the following M lines, the i^{th} line contains two different integers x, y ($1 \leq x, y \leq N,\ x \neq y$), meaning that the i^{th} pedals connects two lamps with index x, y.

Output

If there doesn't exist a series of pace to darken all lamps, output `No` in the first line, otherwise output `Yes`.

If the answer is `Yes` in the first line, please output an integer in the second line, describing the number of pedals needed to be tapped. In the third line, please output all indexes of lamps that Bob should tap in order. Notice that no duplicate pedals are allowed.

If there are multiple ways to darken all lamps, please output any of them.

Subtasks

Subtask 1: The input graph is a chain.

Sample Input 1

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

Sample Output 1

```
Yes
1
5
```

Sample Input 2

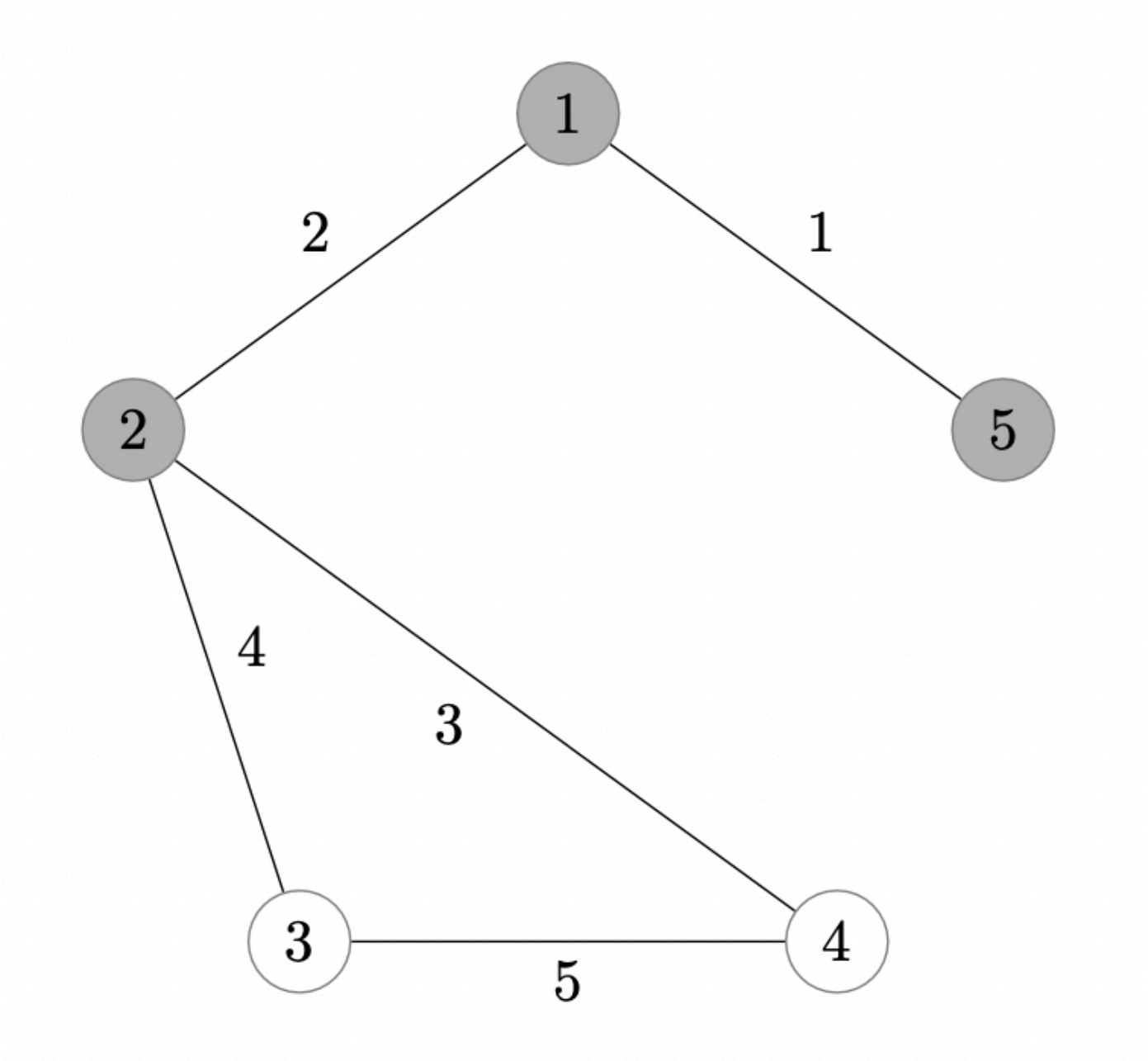
```
5 5
1 0 1 1 0
1 5
2 1
4 2
2 3
4 3
```

Sample Output 2

```
No
```

Hints

In Sample Input 1, tapping the 5^{th} pedals can darken the 3^{th} and the 4^{th} lamps, making all lamps dark.


1
2
1
2
5
4
3
3
4
5

4_跳格子 - Hopscotch

(15分)

(子任務1: 7分, 子任務2: 8 分)

問題敘述

蛋餅是一位拉麵愛好者，他每天都會吃至少一碗拉麵。但大量拉麵造成的卡路里使得蛋餅對自身的體重感到擔憂，因此他決定每天跳格子來消耗卡路里。

蛋餅花了一點時間找到了一個絕佳的跳格子場地，這個場地由 N 個格子組成，由左至右分別從第一個格子排成一直線到第 N 個格子，而每個格子上都寫著一個數字。

為了確實的消耗卡路里，蛋餅給自己定了一些跳格子的規則：

1. 蛋餅可以選擇任意一個格子起跳。
2. 每次要跳到下一個格子時，蛋餅只能往右跳。
3. 每次要跳到下一個格子前，假設蛋餅所在的格子上寫的數字為 x，並且他打算跳到寫著數字 y 的格子，那麼必須滿足 x 跟 y 的差不超過 2，也就是說 $|x-y| \leq 2$。
4. 蛋餅可以從任意一個格子結束跳格子的過程。

這樣一來，蛋餅就會有非常多種跳格子的方法可以選擇，於是他開始好奇，他到底有多少種方法可以從起跳到結束呢？而這裡他將兩種跳格子方法不一樣定義成「跳過的格子編號集合不一樣」。

但由於方法數可能很多，因此蛋餅只好奇方法數除以 10^9+7 的餘數，請你寫一支程式告訴蛋餅這個數字是多少吧！

註：蛋餅的跳躍力非常強，即使是從第一個格子直接跳到第 N 個格子，他也可以達成，因此不用擔心跳躍的距離會造成問題。

輸入格式

輸入首行有一個正整數 N，代表場地的格子數量。

接下來一行 N 個正整數 $a_1, \ldots, a_N$，代表第 i 個格子上的數字為 a_i。

輸出格式

輸出一個非負整數於一行，代表蛋餅跳格子的方法數除以 10^9+7 的餘數。

資料範圍

- $1 \leq N \leq 5 \times 10^5$
- $1 \leq a_i \leq N$

子任務

- 子任務 1 滿足 $N \leq 3000$
- 子任務 2 沒有特別限制

輸入範例 1

```
4
1 4 4 2
```

輸出範例 1

```
9
```

輸入範例 2

```
4
1 3 2 4
```

輸出範例 2

```
14
```

輸入範例 3

```
30
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

輸出範例 3

```
73741816
```

範例說明

在第一筆範例中，蛋餅可能的跳法為以下九種：

1. a_1
2. a_2
3. a_3

4. a_4
5. $a_1 \to a_4$
6. $a_2 \to a_3$
7. $a_2 \to a_4$
8. $a_3 \to a_4$
9. $a_2 \to a_3 \to a_4$

注意到像是 $a_1 \to a_2$ 是不符合規則的，因為 $|a_1 - a_2| = 3 > 2$，違背了第三條規則。

在第二筆範例中，只有 $a_1 \to a_4$ 是不符合規則的，因此方法數為 $2^4 - 2 = 14$ 種。

在第三筆範例中，蛋餅可以從任一個格子跳到更右邊的任何一個格子，因此實際上蛋餅可能的跳法有 $2^{30} - 1 = 1073741823$ 種。但由於蛋餅只好奇方法數除以 $10^9 + 7$ 的餘數，因此你必須輸出 73741816。

4_Hopscotch

(15 points)

(Subtask 1: 7 points, Subtask 2: 8 points)

Description

Omelet is a ramen lover. He eats at least one bowl of ramen every day. However, the calories from lots of ramen worry Omelet about his weight. Hence, he decides to play hopscotch every day to burn calories.

After spending some time, Omelet found an excellent hopscotch playground. The playground is formed by N cells, which line up from the first cell to the N-th cell from left to right. Each cell has a number written on it.

To burn calories, Omelet set himself some hopscotch rules,

1. Omelet can start the process of hopscotch at any of the cells.
2. Each time Omelet wants to jump to the next cell, he can only jump to the right.
3. Each time Omelet wants to jump to the next cell, assume that the cell he is standing on has a number x written on it, and he wants to jump to the cell with a number y written on it. Then the difference between x and y should not exceed 2. In other words, $|x-y| \leq 2$.
4. Omelet can stop the process of hopscotch at any of the cells.

Thus, Omelet will have many different choices in the processes of hopscotch. So he became curious about how many methods can he jump from the beginning to the end? Here, he defines two different jumping methods as "the sets of the index of two jumped cells are different".

Since the answer may be very large, Omelet is only curious about the remainder of the answer divided by 10^9+7. Please write a program to tell Omelet what the number is!

Note: Omelet's jumping power is extreme. Even if he jumps directly from the first cell to the N-th cell, he can still achieve it, so don't worry that the jumping distance will cause problems.

Input Format

The first line of the input contains a positive integer N, indicating the number of cells in the playground.

The second line of the input contains N positive integers $a_1, \ldots, a_N$, indicating that cell i has a number a_i written on it.

Output Format

Output a non-negative integer in one line, indicating that the remainder of the number of how many methods can Omelet jump from the beginning to the end, divided by 10^9+7.

Constraints

- $1 \leq N \leq 5 \times 10^5$
- $1 \leq a_i \leq N$

Subtasks

- Subtask 1 satisfies that $N \leq 3000$.
- Subtask 2 has no additional constraint.

Input Example 1

```
4
1 4 4 2
```

Output Example 1

```
9
```

Input Example 2

```
4
1 3 2 4
```

Output Example 2

```
14
```

Input Example 3

```
30
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Output Example 3

```
73741816
```

Example Explanation

In the first example, Omelet has the following nine methods,

1. a_1
2. a_2
3. a_3
4. a_4
5. $a_1 \to a_4$
6. $a_2 \to a_3$
7. $a_2 \to a_4$
8. $a_3 \to a_4$
9. $a_2 \to a_3 \to a_4$

Notice that something like $a_1 \to a_2$ does not follow the rules since $|a_1 - a_2| = 3 > 2$, which violates the third rule.

In the second example, only $a_1 \to a_4$ does not follow the rules. Hence, the answer is $2^4 - 2 = 14$.

In the third example, Omelet can jump to any cells to the right on every cell. Hence, Omelet has $2^{30} - 1 = 1073741823$ methods to jump. However, since Omelet is only curious about the remainder of the answer divided by $10^9 + 7$, you should output 73741816.

5_橋牌 - Bridge

(20分)

問題敘述

橋牌，一種風靡在世界上的紙牌遊戲，因為規則複雜而帶有競技性受到許多人歡迎。今天精誠企業辦了一場比賽，但比完賽後發現他們忘記統計成績而只留有紀錄因此希望由你幫忙復原比賽過程

具體而言，每場比賽的資料分成以下兩部分

叫牌:叫牌決定主打方及最後的合約。主打方的其中一人被稱作「莊家」，而會主打這一牌，另一位則會成為不做事的「夢家」。最後的合約也可能被（防守方）賭倍或（主打方）再賭倍。
在叫牌中，每個玩家在輪到自己時都可以做出以下其中一種「叫品」：

1. 叫價：宣告叫品的線位及花色，也可稱為「實質性叫品」。
2. 賭倍：當最後一個非派司的叫品是由敵方叫出。
3. 再賭倍：當最後一個非派司的叫品是敵方的賭倍。
4. 派司:不想做出以上三種叫品時，亦即棄權。

叫牌由發牌人開始，每個玩家依序順時鐘輪流叫出一個叫品。當出現連續三個派司後(或是開場連續四個PASS)，叫牌即結束。

「叫價」會指定線位和花色，所以指示了一個想打某合約的提議。想叫牌的玩家必須叫出比前一個叫品更高的叫品。所謂比較高的叫品，是指叫品的線位較高，或是在同一線位上，且花色的等級較高(NT>S>H>D>C)。因此在3H這個叫品之後，不能叫2S或3C，但可以叫3NT(無王)或4D 。
叫牌結束後，最後一個叫價（以及其後的賭倍和再賭倍）便成為合約，其線位決定達成合約所需的磴數，而花色則決定所使用的王牌，其中NT為無王合約，因此不存在王牌。
沒有贏得合約的搭檔叫「防家」。叫出最後叫價的一對還要細分：整個叫牌過程中首先叫出最後合約所屬花色的一方成為「莊家」，他的同伴則是「夢家」。
合約的線位是一個特定的目標。莊家必須想辦法拿到六加上他所喊的現為磴數才能「完成」合約。反之，我們就說防家「擊垮」了合約。

打牌:
打牌的步驟有十三磴，每一磴包含每位玩家手中的一張牌。A 在橋牌中最大，接著依序是 K、Q、J、T、9 等等，每個花色牌組中最小的牌是 2。一磴中最先出的一張牌稱作「引牌」，接著玩家順時鐘方向依序出牌。引牌可以是手上的任何一張牌，但其他的牌則必須打出和引牌同樣花色的牌，除非他們已經沒有該花色的牌。打出該磴最大牌張的玩家贏得這一磴，除非其他玩家的出牌中有王牌，在後者的狀況中則由打出最大王牌的玩家贏得此磴。贏得此磴的玩家在下一磴可以引牌，直到所有牌打完。
第一張引牌稱作「首引」（首攻），是由莊家左手方的防家出牌。

輸入

輸入第一行有一個整數 T，代表總共有T場牌局的資料還原
之後會有T局牌局得資料，每局牌局由若干行組成，其中第一行是單一一個字母，代表第一個喊出叫品的方位(順時針方向依序為 N、E、S、W)，之後會有若干行，分別代表依序的叫牌過程，每行一組，其中叫牌的表示方法如下

1. 叫價：XY。
2. 賭倍:`X`。
3. 再賭倍：`XX`。
4. 派司:`PASS`。
   並保證叫牌過程會以三或四個`PASS`結束。
   如果叫牌過程為四個`PASS`(亦即沒進行該複牌局)，則不會有打牌過程，否則打牌過程格式如下:
   在打牌過程會有 13 行，每行 4 個字串，依序代表那一輪首引玩家按照順時鐘方向的出牌，字串形式為 MN ，其中M為那張牌的花色，N為那張牌的數字。

輸出

對於每一副牌，如果該局為四個 `PASS` ，請輸出`ALL PASS`，
否則如果該局如果不合法，代表主辦單位紀錄出錯，請輸出`error`，具體而言不合法狀況如下:

1. 叫牌過程錯誤:
   1. 叫價並沒有比上一個叫價高。
   2. 對非對手的叫價賭倍。
   3. 對非對手的賭倍再賭倍。
2. 打牌過程錯誤如下:
   1. 有張牌在牌局出現超過一次。
   2. 有人在打牌時應跟首引花色而未跟。

否則請輸出該局最終合約、莊家方位、最終結果。
最終合約格式如同輸入，賭倍與再賭倍請直接在後面加上`X`或`XX`。
例如:`4HX`代表$4H$賭倍，`3NTXX`代表$3NT$再賭倍，莊家方位為單一字母，最終結果如果合約洽完成請輸出`=`，否則輸出 $+X$，如果莊家多吃X敦或是 $-X$，如果莊家少吃X敦。

資料範圍

- $1 \leq T \leq 100$。
- $1 \leq X \leq 7$。
- Y為`C`,`D`,`H`,`S`,`NT`其中之一。
- M為`C`,`D`,`H`,`S`其中之一。
- N為`A`,`K`,`Q`,`J`,`T`,`9`,`8`,`7`,`6`,`5`,`4`,`3`,`2`其中之一。

範例輸入 1

```
1
N
PASS
PASS
PASS
PASS
```

範例輸出 1

```
ALL PASS
```

範例1說明

四家都 PASS，因此輸出`ALL PASS`

範例輸入 2

```
3
E
1C
1NT
2C
PASS
PASS
PASS
C2 C3 C4 CA
CK C5 C6 C7
CQ C8 C9 CT
CJ S2 S3 S4
SA S5 S6 S7
SK S8 S9 ST
SQ SJ H2 H3
HA H4 H5 H6
H7 HK H8 H9
HQ HJ HT D2
DA D3 D4 D5
DK D6 D7 D8
DQ D9 DT DJ
S
1C
X
PASS
PASS
PASS
C2 C3 C4 CA
CK C5 C6 C7
CQ C8 C9 CT
CJ S2 S3 S4
SA S5 S6 S7
SK S8 S9 ST
SQ SJ H2 H3
HA H4 H5 H6
H7 HK H8 H9
```

```
HQ HJ HT D2
DA D3 D4 D5
DK D6 D7 D8
DQ D9 DT DJ
W
3C
X
XX
PASS
PASS
PASS
C2 C3 C4 CA
CK C5 C6 C7
CQ C8 C9 CT
CJ S2 S3 S4
SA S5 S6 S7
SK S8 S9 ST
SQ SJ H2 H3
HA H4 H5 H6
H7 HK H8 H9
HQ HJ HT D2
DA D3 D4 D5
DK D6 D7 D8
DQ D9 DT DJ
```

範例輸出 2

```
2CE=
1CXS+1
3CXXW-1
```

範例2說明

在第1局中，雖然最後訂約的 2C 是 W 家所喊，但因為第一個叫出 C 的是 E，因此莊家仍為 E 家。
第1到3局中皆為主打方拿到前面八墩。

範例輸入 3

```
1
E
1C
XX
PASS
PASS
PASS
C2 C3 C4 CA
```

```
CK C5 C6 C7
CQ C8 C9 CT
CJ S2 S3 S4
SA S5 S6 S7
SK S8 S9 ST
SQ SJ H2 H3
HA H4 H5 H6
H7 HK H8 H9
HQ HJ HT D2
DA D3 D4 D5
DK D6 D7 D8
DQ D9 DT DJ
```

範例輸出 3

```
error
```

範例3說明

因為 S 家對非對手的賭倍再賭倍，因此為 error。

5_Bridge

(20 points)

Description

Bridge, a kind of well-known poker game in the world, is popular for the complex rules and competitiveness. Today, Jinchen company held a Bridge game. When the game finished, they found they forget to count the score, so they only have the record and want you to recover the score.

Every game's record data has two part of the following.

Auction: Decide which pair to play and the final contract. One of these pairs will be the "declarer" who plays this game and the other will be the "dummy" who doesn't need to do anything. And the final contract would be (defender) double or (declarer or dummy) redouble.
In the Auction, every player can do one of the following "call" in his/her round.

1. bid: Specifying the level of their contract and either the trump suit or no trump (the denomination)
2. double: when the last not pass call is the opponent's bid
3. redouble: when the last not pass call is the opponent's double
4. pass: when you don't want to do the 3 calls below

The auction starts with one of the players, every player call one by one in the clockwise direction. Finish when three continuous pass (or four continuous pass in the start).

"Bid" will Specify the level of their contract and either the trump suit or no trump (the denomination), which suggests the wondering contract he/she wants to play. The player who wants to call a bid must be higher than the last bid last which means the level is higher or the level is equal but the suit is better (NT>S>H>D>C). So after 3C, you can't call 2S or 3C but you can call 3NT(no-trump).
When the auction finish, the last bid(and the double or redouble) will be the contract. The level will decide how many tricks they need to win, and the suit decides the trump, and NT is no-trump contract, so there doesn't have trump.
The partner who doesn't get a contract is called a defender, and the partner who gets a contract has to be a subdivision, the one who first calls the contract suit in the all Auction call declarer, and the other call dummy.
To finish the contract, The declarer needs to win 6+level or we say the defender beat down the contract.

Play:
There are 13 tricks in one game, every trick includes one of the cards for every four people. In the Bridge game the number is A>K>Q>J>T>9>8>7>6>5>4>3>2. The first card in one trick is called "lead", and then the player play cards in a clockwise direction. The lead card can be one of the cards in hand, but other players must play the card whose suit is the same as the lead card unless they don't have it. The one who plays the higher card with the lead suit wins this trick unless there are trump-suit cards in this trick, then the one who plays the higher card with the trump-suit win. The player wins this card will be the next trick's lead.
The first trick's lead is played from the declarer's left.

Input Format

There is one integer T in the first line, which means you have to count the T bridge game.
There are T bridge game's data follow. In every game, the first line has one letter, which means the first player to call(N, E, W, S in a clockwise direction). And then have a few lines, the call auction.

1. bid：XY。
2. double:`X`。
3. redouble：`XX`。
4. pass:`PASS`。

And call auction finishes with 3 or 4`PASS`.
If there are 4 `PASS`(they don't play that game), we don't have the play data. Or the play data show the following.
There are 13 lines, each line with 4 strings which means the card plays in a clockwise direction and begins with the lead player in that trick. The card is shown by MN, M is the suit,N is the number.

Output Format

For every game, if it begins with 4 `PASS` ， Please out put `ALL PASS`.
Or is that game illegal which means the data is wrong, please output `error`.The following case is illegal.

1. call auction is illegal:
    1. the bid doesn't higher than the last bit.
    2. call double but the last not pass call is not the opponent's bid
    3. call redouble but the last not pass call is not the opponent's double
2. play is illegal:
    1. some cards appear more than one time.
    2. you have the card with the lead suit but play the other suit's card.

Or output the final contract, who is declarer(show with position), the final result.
the final contract is shown the same as input, double, and redouble just add `X` or `XX` after the final contract.
EX: `4HX` means double to $4H$, `3NTXX` means redouble to $3NT$
The position of declarer is one letter.
If the declarer just finishes the contract(win 6+level tricks),please output `=`, or output $+X$ the number declarer win over 6+level, or output $-X$ the number declarer less than 6+level.

Constraints

- $1 \leq T \leq 100$.
- $1 \leq X \leq 7$.
- Y is one of `C`, `D`, `H`, `S`, `NT`.
- $M$is one of `C`, `D`, `H`, `S`.

- $N$is one of `A`,`K`,`Q`,`J`,`T`,`9`,`8`,`7`,`6`,`5`,`4`,`3`,`2`.

Sample Input 1

```
1
N
PASS
PASS
PASS
PASS
```

Sample Output 1

```
ALL PASS
```

Sample1 Explanation

four players all PASS at begin, so the output is `ALL PASS`

Sample Input 2

```
3
E
1C
1NT
2C
PASS
PASS
PASS
C2 C3 C4 CA
CK C5 C6 C7
CQ C8 C9 CT
CJ S2 S3 S4
SA S5 S6 S7
SK S8 S9 ST
SQ SJ H2 H3
HA H4 H5 H6
H7 HK H8 H9
HQ HJ HT D2
DA D3 D4 D5
DK D6 D7 D8
DQ D9 DT DJ
S
1C
```

```
X
PASS
PASS
PASS
C2 C3 C4 CA
CK C5 C6 C7
CQ C8 C9 CT
CJ S2 S3 S4
SA S5 S6 S7
SK S8 S9 ST
SQ SJ H2 H3
HA H4 H5 H6
H7 HK H8 H9
HQ HJ HT D2
DA D3 D4 D5
DK D6 D7 D8
DQ D9 DT DJ
W
3C
X
XX
PASS
PASS
PASS
C2 C3 C4 CA
CK C5 C6 C7
CQ C8 C9 CT
CJ S2 S3 S4
SA S5 S6 S7
SK S8 S9 ST
SQ SJ H2 H3
HA H4 H5 H6
H7 HK H8 H9
HQ HJ HT D2
DA D3 D4 D5
DK D6 D7 D8
DQ D9 DT DJ
```

Sample Output 2

```
2CE=
1CXS+1
3CXXW-1
```

Sample2 Explanation

In the first game, though 2C is called by the west player, the first one who called C in the E-W pair is the east player, so the declarer is the East player
In the first to the third game, the declarer pair get the first 8 tricks.

Sample Input 3

```
1
E
1C
XX
PASS
PASS
PASS
C2 C3 C4 CA
CK C5 C6 C7
CQ C8 C9 CT
CJ S2 S3 S4
SA S5 S6 S7
SK S8 S9 ST
SQ SJ H2 H3
HA H4 H5 H6
H7 HK H8 H9
HQ HJ HT D2
DA D3 D4 D5
DK D6 D7 D8
DQ D9 DT DJ
```

Sample Output 3

```
error
```

Sample3 Explanation

The South player redouble to not opponent's double, so must output `error`.

6_早上好YTP - Good Morning YTP

(25 分)
(子任務1: 9分, 子任務2: 16 分)

問題敘述

早上好YTP

現在我有一棵Treap

我很喜歡這棵Treap

但是

Link-Cut-Tree

比Treap

Link-Cut

Link-Cut-Tree

我最喜歡

所以...現在是刻資結時間

準備 1 2 3

兩個禮拜以後

Link-Cut-Tree ×3

不要忘記

不要錯過

記得去YTP刻Link-Cut-Tree

因為非常好資結

操作非常好

差不多一樣Treap

再見

經過上面的敘述，相信你已經相當了解這道題的內容。

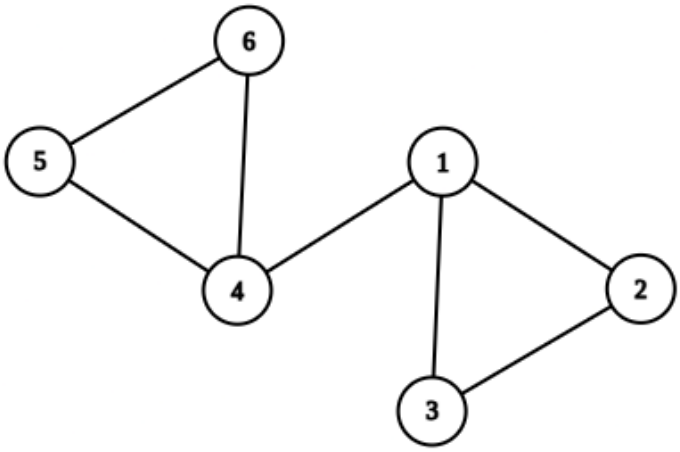
如果還沒也不用擔心，請閱讀下方的敘述。

有一張N個點的圖，一開始有0條邊。

現在請你依序加入M條邊，每加完一條邊請輸出當前圖中的橋的數量。

(保證不會有重複的邊，也不會有自環)

> 在圖論中，如果刪掉某條邊可以增加這張圖的連通塊數，這條邊就被稱作橋。



以這張圖為例(1, 4)是一座橋，因為刪掉這條邊後整張圖的連通塊數變多了。

輸入格式

第一行包含兩個整數N, M，代表有幾個節點跟共要加入幾條邊。

接下來的M行，每行有兩個整數x, y，代表加入一條連接點x跟點y的邊。

輸出格式

請輸出M行，第i行包含一個整數x，代表當前i條邊都加入圖中後，圖中有幾條邊是橋。

資料範圍

- $1 \leq N, M \leq 200000$
- $1 \leq M \leq N(N-1)/2$
- $1 \leq x, y \leq N$

子任務1 (9分) 的子測資中，保證前$N-1$條邊會是$(i, i+1)$的形式，但不保證出現的順序。

輸入範例 1

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

輸出範例 1

```
1
2
0
```

輸入範例 2

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

輸出範例 2

```
1
2
3
4
```

輸入範例 3

```
10 21
6 1
5 1
3 7
7 10
5 6
3 6
4 6
8 1
5 7
```

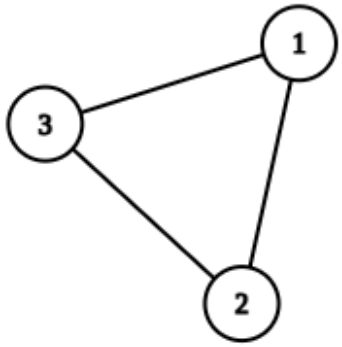
```
8 5
1 10
1 3
2 6
5 10
3 9
9 2
7 8
9 4
7 2
10 6
5 3
```

輸出範例 3

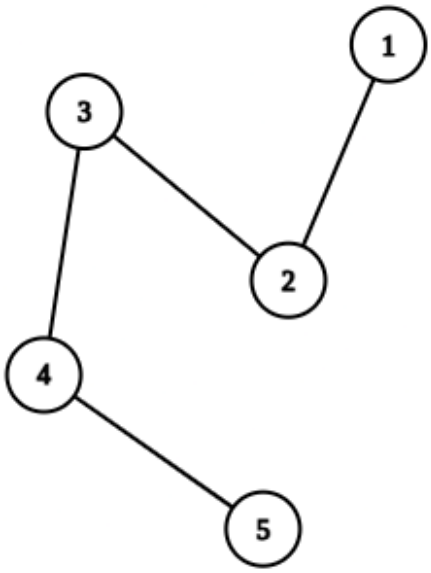
```
1
2
3
4
2
3
4
5
3
2
1
1
2
2
3
1
1
0
0
0
0
```

範例說明

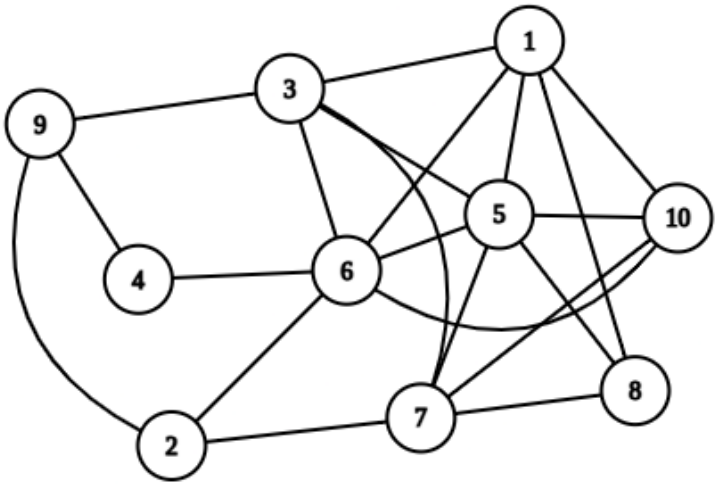
在輸入範例1中，加入第3條邊後所有邊都在環上，故沒有一條邊是橋。



在輸入範例2中，在任何時刻這張圖上都沒有環，故所有邊都是橋。



輸入範例3的圖：



6_Good Morning YTP

(25 points)
(Subtask 1: 9 points, Subtask 2: 16 points)

Description

Good morning YTP

Now I have a Treap

I really like this Treap

But

Link-Cut-Tree

Compare to Treap

Link-Cut

Link-Cut-Tree

I like the most

So...it's time to code data structure

Get ready 1 2 3

After two weeks

Link-Cut-Tree ×3

Do not forget this

Do not miss this

Remember to participate YTP to code Link-Cut-Tree

Because it's a very good data structure

Its operations are very good

Almost the same as Treap

Goodbye

After reading the description above, you should already know what's the problem's content.

Not worry if you don't. Please read the description below.

There's a graph with N vertices.

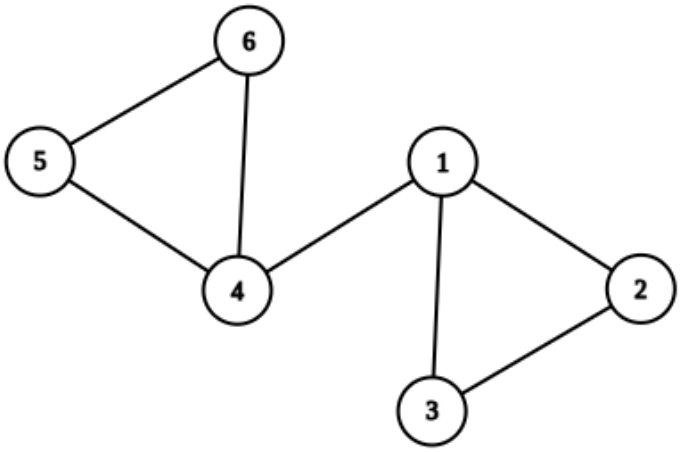
Initially, there's 0 edge in the graph.

Now you have to add M edges into the graph one by one.

After adding each edge, you should output the number of bridges in the graph.

(There's no duplicate edges or self-loop)

> In graph theory, a bridge is an edge of a graph whose deletion increases the graph's number of connected components.



Take this graph as example, (1, 4) is a bridge since after deleting this edge, the number of connected components in this graph have increased.

Input Format

First line contains two integers N, M, denote the number of vertices and the number of edges to be added respectively.

For the next M lines, each contains two integers x, y, denote the two endpoints of the edge to be added.

Output Format

Print M lines, the ith line contains one integer x, denote the number of bridges in the graph after first i edges are added to the graph.

Constraints

- $1 \leq N, M \leq 200000$
- $1 \leq M \leq N(N-1)/2$
- $1 \leq x, y \leq N$

For subtask 1 (9 points), it is guaranteed that the first $N-1$ edges would be in the form of $(i, i+1)$. (though its order may not be sorted)

Input Example 1

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

Output Example 1

```
1
2
0
```

Input Example 2

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

Output Example 2

```
1
2
3
4
```

Input Example 3

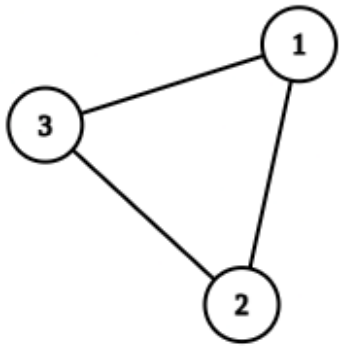
```
10 21
6 1
5 1
3 7
7 10
5 6
3 6
4 6
8 1
5 7
8 5
1 10
1 3
2 6
5 10
3 9
9 2
7 8
9 4
7 2
10 6
5 3
```

Output Example 3

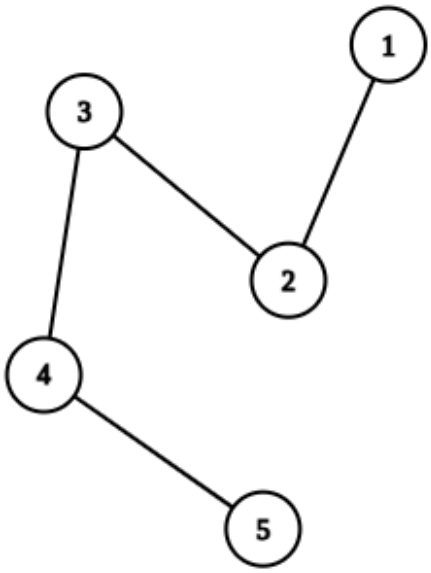
```
1
2
3
4
2
3
4
5
3
2
1
1
2
2
3
1
1
0
0
0
0
```

Example Explanation

In Input Example 1, after adding the third edge, every edge is on a cycle, therefore there's no bridge.



In Input Example 2, there's no cycle at any time, so every edge is a bridge.



The graph of Input Example 3: