Rujia Liu's Present 3

A Data Structure Contest

Celebrating the 100th Anniversary of Tsinghua University

April 23, 2011

UVa Online Judge

Problems

Almost Union-Find
Broken Keyboard (a.k.a. Beiju Text)
Cake Cutting
"Dynamic" Inversion
Easy Problem from Rujia Liu?
Fast Matrix Operations
Girls' Celebration
Happy Painting!
I Can Guess the Data Structure!
Jewel Magic
K Smallest Sums
Rujia Liu loves Wario Land!

I know that data structures can be very difficult to debug, especially when you're using online judges and don't have access to the test data. Thus, I decided to provide quite a lot of additional test data (apart from the sample I/O in the problem description) as a gift, for your reference. You can download them on the contest website.

Please make sure you pass all these additional test data before submitting. Real data are a lot more difficult, so you're still facing great challenges!

I'd like to thank Yiming Li, Yeji Shen, Dun Liang and Yao Li for writing alternative solutions for some of the problems, Erjin Zhou for problem L, and Xinhao Yuan for problem J. And finally, thanks to Prof. Miguel (again)!

Hello, everyone! My name is Rujia Liu. I used to do a lot of problem solving and problemsetting, but after graduated from Tsinghua University, I'm spending more and more time on my company ☹ Anyway, here I come again, with my 3rd present for UVa OJ: A Data Structure contest.

Recently, Tsinghua students are busy with its 100-th anniversary. For example, as a (graduated!) member of Tsinghua Chorus, I've just attended a performance in the Concert Hall of National Grand Theatre (NCPA) on April 4th. When you're reading this text, I'm most probably on the **luxury** stage of the final anniversary show in the campus 8-)

(A photo of me, taken in the dressing room in NCPA)

But why bother data structures? Well, I love data structures. I've enjoyed working as the TA for two data structure courses - I'd like to express my thanks to the teachers: Prof Junhui Deng and Prof Hong Wang, and also the students.

As a bonus for reading my story, the easiest problem is E, and K is a bit more difficult that it seems. Be careful!

ALMOST UNION-FIND

I hope you know the beautiful Union-Find structure. In this problem, you're to implement something similar, but not identical.

The data structure you need to write is also a collection of disjoint sets, supporting 3 operations:

1 p q	Union the sets containing p and q. If p and q are already in the same set, ignore this command.
2 p q	Move p to the set containing q. If p and q are already in the same set, ignore this command
3 p	Return the number of elements and the sum of elements in the set containing p.

Initially, the collection contains n sets: {1}, {2}, {3}, ..., {n}.

INPUT

There are several test cases. Each test case begins with a line containing two integers n and m (1<=n,m<=100,000), the number of integers, and the number of commands. Each of the next m lines contains a command. For every operation, 1<=p,q<=n. The input is terminated by end-of-file (EOF). The size of input file does not exceed 5MB.

OUTPUT

For each type-3 command, output 2 integers: the number of elements and the sum of elements.

SAMPLE INPUT

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

SAMPLE OUTPUT

```
3 12
3 7
2 8
```

Explanation

Initially: {1}, {2}, {3}, {4}, {5}
Collection after operation 1 1 2: {1,2}, {3}, {4}, {5}
Collection after operation 2 3 4: {1,2}, {3,4}, {5} (we omit the empty set that is produced when taking out 3 from {3})
Collection after operation 1 3 5: {1,2}, {3,4,5}
Collection after operation 2 4 1: {1,2,4}, {3,5}

Broken Keyboard (a.k.a. Beiju Text)

You're typing a long text with a broken keyboard. Well it's not so badly broken. The only problem with the keyboard is that sometimes the "home" key or the "end" key gets automatically pressed (internally).

You're not aware of this issue, since you're focusing on the text and did not even turn on the monitor! After you finished typing, you can see a text on the screen (if you turn on the monitor).

In Chinese, we can call it Beiju. Your task is to find the Beiju text.

Input

There are several test cases. Each test case is a single line containing at least one and at most 100,000 letters, underscores and two special characters '[' and ']'. '[' means the "Home" key is pressed internally, and ']' means the "End" key is pressed internally. The input is terminated by end-of-file (EOF). The size of input file does not exceed 5MB.

Output

For each case, print the Beiju text on the screen.

Sample Input

```
This_is_a_[Beiju]_text
[[]][][]Happy_Birthday_to_Tsinghua_University
```

Sample Output

```
BeijuThis_is_a__text
Happy_Birthday_to_Tsinghua_University
```

Cake Cutting

There is a very big rectangular (yes...) cake on the xy-plane, whose four corners are (0,0), (w,0), (w,h) and (0,h).

Each time you're hungry, you slice a piece from the cake and eat it. Your task is to output the area of the remaining cake, after each slice.

Input

There are several test cases. The first line contains threes integers n, w, h (1<=n<=200,000, 1<=w,h<=1000), the number of slices, the width and the height of the cake. Each of the following n lines contains four positive real numbers x1, y1, x2, y2 not greater than 1000. That means, you slice it along the straight line connecting (x1,y1) and (x2,y2), and eat the part on the right (if any), when looking from (x1,y1) to (x2,y2). The input is terminated by end-of-file (EOF). The size of input file does not exceed 10MB.

Output

For each slice, output the area of the cake after the slice, to at least three digits after the decimal point. We allow an absolute error of 10^{-3} for each value you output.

Sample Input

```
2 20 10
15.0 0.0 15.0 5.0
1.0 2.0 2.0 2.0
```

Sample Output

```
150.000
120.000
```

"Dynamic" Inversion

You are given a permutation {1,2,3,...,n}. Remove m of them one by one, and output the number of inversion pairs **before** each removal. The number of inversion pairs of an array A is the number of ordered pairs (i,j) such that i < j and A[i] > A[j].

Input

The input contains several test cases. The first line of each case contains two integers n and m (1<=n<=200,000, 1<=m<=100,000). After that, n lines follow, representing the initial permutation. Then m lines follow, representing the removed integers, in the order of the removals. No integer will be removed twice. The input is terminated by end-of-file (EOF). The size of input file does not exceed 5MB.

Output

For each removal, output the number of inversion pairs before it.

Sample Input

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

Sample Output

```
5
2
2
1
```

Explanation

(1,5,3,4,2)->(1,3,4,2)->(3,4,2)->(3,2)->(3)

Easy Problem from Rujia Liu?

Though Rujia Liu usually sets hard problems for contests (for example, regional contests like Xi'an 2006, Beijing 2007 and Wuhan 2009, or UVa OJ contests like Rujia Liu's Presents 1 and 2), he occasionally sets easy problem (for example, 'the Coco-Cola Store' in UVa OJ), to encourage more people to solve his problems :D

Given an array, your task is to find the k-th occurrence (from left to right) of an integer v. To make the problem more difficult (and interesting!), you'll have to answer m such queries.

Input

There are several test cases. The first line of each test case contains two integers n, m(1<=n,m<=100,000), the number of elements in the array, and the number of queries. The next line contains n positive integers not larger than 1,000,000. Each of the following m lines contains two integer k and v (1<=k<=n, 1<=v<=1,000,000). The input is terminated by end-of-file (EOF). The size of input file does not exceed 5MB.

Output

For each query, print the 1-based location of the occurrence. If there is no such element, output 0 instead.

Sample Input

```
8 4
1 3 2 2 4 3 2 1
1 3
2 4
3 2
4 2
```

Sample Output

```
2
0
7
0
```

Fast Matrix Operations

There is a matrix containing at most 10^6 elements divided into r rows and c columns. Each element has a location (x,y) where 1<=x<=r,1<=y<=c. Initially, all the elements are zero. You need to handle four kinds of operations:

1 x1 y1 x2 y2 v	Increment each element (x,y) in submatrix (x1,y1,x2,y2) by v (v>0)
2 x1 y1 x2 y2 v	Set each element (x,y) in submatrix (x1,y1,x2,y2) to v
3 x1 y1 x2 y2	Output the summation, min value and max value of submatrix (x1,y1,x2,y2)

In the above descriptions, submatrix (x1,y1,x2,y2) means all the elements (x,y) satisfying x1<=x<=x2 and y1<=x<=y2. It is guaranteed that 1<=x1<=x2<=r, 1<=y1<=y2<=c. After any operation, the sum of all the elements in the matrix will not exceed 10^9.

Input

There are several test cases. The first line of each case contains three positive integers r, c, m, where m (1<=m<=20,000) is the number of operations. Each of the next m lines contains a query. There will be at most twenty rows in the matrix. The input is terminated by end-of-file (EOF). The size of input file does not exceed 500KB.

Output

For each type-3 query, print the summation, min and max.

Sample Input

```
4 4 8
1 1 2 4 4 5
3 2 1 4 4
1 1 1 3 4 2
3 1 2 4 4
3 1 1 3 4
2 2 1 4 4 2
3 1 2 4 4
1 1 1 4 3 3
```

Sample Output

```
45 0 5
78 5 7
69 2 7
39 2 7
```

Girls' Celebration

In order to celebrate the 100-th anniversary of Tsinghua University, n girls are planning to hold a party. They're experts at singing and dancing, and they love to perform in groups. In their current design, there will be a stage and a row of seats. When a group of girls need to sing or dance on the stage, they stand up from their seats, and go to the stage. When they've finished performing, they return to their own seat (they don't exchange seats, because every girl has a lot of personal belongings on her seat).

They want this procedure look cool, so for each performance, the actresses' seats should be consecutive. For example, if there are 4 girls, and a performance is done by girl 1, 2 and 4, then they cannot seat in order of 1-2-3-4, since when girl 1, 2 and 4 stand up, it's strange to see a non-actress (girl 3) sitting between girl 2 and 4.

As I mentioned, they're too good at singing and dancing, so they managed to come up with a lot of combinations. Now they become a bit worried: is there a way to seat all the girls, such that the requirement above can be satisfied (i.e. for every combination, the actresses' seats are consecutive).

As a decent programmer, you decide (I know that actually you're being decided, but...) to write a program that can calculate the number of seat arrangements. Since the girls' are constantly thinking about new combinations, your program should be able to read new combinations and adjust the answer accordingly. When there are only few possible arrangement (i.e. at most k feasible solutions), your program should output all of them.

Input

There are several test cases. The first line contains three integers n, m, k (1<=n,m,k<=200), where n is the number of girls, m is the number of combinations, and k is the parameter described above. Each of the next m lines contains a set of integers, terminated by a zero. These integers are the IDs of the girls in the combination (girls are numbered 1 to n). The input is terminated by end-of-file (EOF). The size of input file does not exceed 1MB.

Output

For each new combination, output the number of seat arrangements, after considering this combination. If there is no way, print 0 and **ignore the combination**. If there are at most k ways, print them one in a line, in lexicographical order.

Sample Input

```
4 4 10
1 2 3 0
2 3 4 0
1 4 0
2 4 0
```

Sample Output

```
12
4
1 2 3 4
1 3 2 4
4 2 3 1
4 3 2 1
0
2
1 3 2 4
4 2 3 1
```

Happy Painting!

There is a forest of colorful rooted trees containing n nodes. You are given m operations. Execute them one by one, and output the results.

1 x y c	Change x's father to y. If x=y or x is a ancestor of y, simply ignore it. The edge between x and its old father is removed, and the new edge should be painted with color c.
2 x y c	Paint all the edges along the path x-y with color c. If there is no path between x and y, simply ignore it.
3 x y	Count the number of edges along the path x-y, and the total number of colors among these edges.

Input

The input contains several test cases. The first line of each test case contains two integers n and m (1<=n<=50,000, 1<=m<=200,000). Nodes are numbered from 1 to n. The second line contains n integers F[i] (0<=F[i]<=n), the father of each node (F[i] =0 means the node is the root of a tree). The next line contains n integers C[i] (1<=C[i]<=30), the colors of the edges between each node and its father (for root nodes, the corresponding color should be ignored). Each of the next m lines contains an operation. For all operations, 1<=x,y<=n, for each type-2 operation, 1<=c<=30. The input is terminated by end-of-file (EOF). The size of input file does not exceed 5MB.

Output

For each type-3 operation, output two integers: the number of edges and the number of colors among these edges.

Sample Input

```
6 6
0 1 1 3 3 0
1 2 1 1 1 1
3 2 3
1 3 2 3
3 2 3
3 5 6
1 6 1 1
3 4 6
```

Sample Output

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

I Can Guess the Data Structure!

There is a bag-like data structure, supporting two operations:

1 x	Throw an element x into the bag.
2	Take out an element from the bag.

Given a sequence of operations with return values, you're going to guess the data structure. It is a stack (Last-In, First-Out), a queue (First-In, First-Out), a priority-queue (Always take out larger elements first) or something else that you can hardly imagine!

Input

There are several test cases. Each test case begins with a line containing a single integer n (1<=n<=1000). Each of the next n lines is either a type-1 command, or an integer 2 followed by an integer x. That means after executing a type-2 command, we get an element x **without error**. The value of x is always a positive integer not larger than 100. The input is terminated by end-of-file (EOF). The size of input file does not exceed 1MB.

Output

For each test case, output one of the following:

stack	It's definitely a stack.
queue	It's definitely a queue.
priority queue	It's definitely a priority queue.
impossible	It can't be a stack, a queue or a priority queue.
not sure	It can be more than one of the three data structures mentioned above.

Sample Input

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

Sample Output

```
queue
not sure
impossible
stack
priority queue
```

Jewel Magic

I am a magician. I have a string of emeralds and pearls. I may insert new jewels in the string, or remove old ones. I may even reverse a consecutive part of the string. At anytime, if you point to two jewels and ask me, what is the length of the longest common prefix (LCP) of jewel strings starting from these two jewels, I can answer your question instantly. Can you do better than me?

Formally, you'll be given a string of 0 and 1. You're to deal with four kinds of operations (in the following descriptions, L denotes the current length of the string, and jewel positions are number 1 to L numbered from left to right):

1 p c	Insert a jewel c after position p (0<=p<=L. p=0 means insert before the whole string). c=0 means emerald, c=1 represents pearl.
2 p	Remove the jewel at position p (1<=p<=L).
3 p1 p2	Reverse the part starting from position p1, ending at position p2 (1<=p1<p2<=L)
4 p1 p2	Output the LCP length of jewel strings starting from p1 and p2 (1<=p1<p2<=L).

Input

There will be several test cases. The first line of each test case contains an integer n and m (1<=n,m<=200,000), where n is the number of pearls initially, m is the number of operations. The next line contains a 01 string of length n. Each of the next m lines contains an operation. The input is terminated by end-of-file (EOF). The size of input file does not exceed 5MB.

Output

For each type-4 operation, output the answer.

Sample Input

```
12 9
000100001100
1 0 1
4 2 6
3 7 10
4 1 7
2 9
4 3 11
4 1 9
4 1 7
4 2 3
```

Sample Output

3
6
2
0
3
2

Explanation

String after operation 1 0 1: 1000100001100
String after operation 3 7 10: 1000101000100
String after operation 2 9: 100010100100

K Smallest Sums

You're given k arrays, each array has k integers. There are k^k ways to pick exactly one element in each array and calculate the sum of the integers. Your task is to find the k smallest sums among them.

Input

There will be several test cases. The first line of each case contains an integer k (2<=k<=750). Each of the following k lines contains k positive integers in each array. Each of these integers does not exceed 1,000,000. The input is terminated by end-of-file (EOF). The size of input file does not exceed 5MB.

Output

For each test case, print the k smallest sums, in ascending order.

Sample Input

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

Sample Output

```
9 10 12
2 2
```

Rujia Liu Loves Wario Land!

I love a game series called "Wario Land", so I'd like to make a very difficult (indeed!!!) problem about it :) A big thank you goes to Erjin Zhou, for the idea and reference code. And a small thank you goes to Wenbin Tang, for reminding me that "Rujia Liu" also contains the letter L!

Suppose there are n places in the very beginning of Wario Land. The land was almost deprecated, so it does not have any roads at all! You'll be given m operations. Execute them one by one, and output the results.

1 x y	Wario wants to build a direct road between place x and y. If x and y are already connected (directly or indirectly), ignore this command (because Wario thinks it's a waste of time!).
2 x v	Change place x's treasure value to v. This is due to newly discovered treasures, or treasures that are stolen by someone else.
3 x y v	Among the places along the path between x and y (including x and y), how many of them have treasure value <= v? Wario also needs the product of these treasure values, modulo k (see below).

Input

The input contains several test cases. In each test case, the first line contains three integers n, m, k(1<=n<=50,000, 1<=m<=100,000, 2<=k<=33333). Places are numbered from 1 to n. The second line contains n integers V[i] (1<=V[i]<=k), the initial treasure values of each place. Each of the next m lines contains an operation. For each operation, 1<=x,y<=n, 1<=v<=k. The input is terminated by end-of-file (EOF). The size of input file does not exceed 10MB.

Output

For each type-3 operation, output the number of places and the product of their treasure values, modulo k. If there is no path between x and y, or every place along the path has treasure value > v, output a single 0 (rather than 0 0 or 0 1).

Sample Input

```
4 8 39
2 3 4 5
1 1 2
3 2 3 5
1 1 3
3 2 3 5
1 1 4
```

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

Sample Output

```
0
3 24
2 8
3 1
0
```

Obfuscation

In order to prevent you from preprocessing the operations, we adopt the following obfuscation scheme:

Each type-1 operation becomes 1 x+d y+d
Each type-2 operation becomes 2 x+d v+d
Each type-3 operation becomes 3 x+d y+d v+d

Where d is the last integer that you output, before processing this operation. If you haven't output anything yet, d=0.

After the obfuscation, the sample input would be:

```
4 8 39
2 3 4 5
1 1 2
3 2 3 5
1 1 3
3 2 3 5
1 25 28
3 27 28 28
3 11 12 13
3 4 5 2
```

This is the real input that your program will read.