Aufgaben
Get Connected to ICPC
Local Contest 2011

Zugang zum Contest:

Wiki: `http://icpc.ira.uka.de/public/wettbewerbe/getconnected2011`
Wettbewerbsserver: `http://domjudge.iti.uni-karlsruhe.de/team/`

Inhaltsverzeichnis

The 3n + 1 problem

Problems in Computer Science are often classified as belonging to a certain class of problems (e.g., NP, Unsolvable, Recursive). In this problem you will be analyzing a property of an algorithm whose classification is not known for all possible inputs.

1 The Problem

Consider the following algorithm:

1. input n
2. print n
3. if n = 1 then STOP
4. if n is odd then n := 3n+1
5. else n := n/2
6. goto 2

Given the input 22, the following sequence of numbers will be printed 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1. It is conjectured that the algorithm above will terminate (when a 1 is printed) for any integral input value. Despite the simplicity of the algorithm, it is unknown whether this conjecture is true. It has been verified, however, for all integers n such that $0 < n < 1000000$ (and, in fact, for many more numbers than this.) Given an input n, it is possible to determine the number of numbers printed (including the 1). For a given n this is called the cycle-length of n. In the example above, the cycle length of 22 is 16. For any two numbers i and j you are to determine the maximum cycle length over all numbers between i and j. (Both are inclusive.)

2 Input

The input will consist of a series of pairs of integers i and j, one pair of integers per line. All integers will be less than 1000 and greater than 0. You should process all pairs of integers and for each pair determine the maximum cycle length over all integers between and including i and j. You can assume that no operation overflows a 32-bit integer.

3 Output

For each pair of input integers i and j you should output i, j followed by the corresponding maximum cycle length. These three numbers should be written into a single line in that order seperated by a single space. The number i and j must appear in the same order as in the input. Each line must be terminated by a newline character.

4 Sample Input

```
1 10
100 200
201 210
1000 900
```

5 Sample Output

```
1 10 20
100 200 125
201 210 89
1000 900 174
```

Professor Snørbørden's Slides

1 Description

Marcus is a smart and diligent computer science student but the lectures of his Professor Snørbørden are not exactly the most thrilling moments of his studies. In other words: Professor Snørbørden is exceptionally boring. Marcus can hardly stay awake, when listening to the slow, monotonous words of the Professor and the only joy left for him, is that of pointing out the mistakes on the slides to Professor Snørbørden. Since the slides were designed by Professor Snørbørden himself, the errors are blatantly obvious and can easily be spotted by a clever student such as Marcus when flipping through them on the day before the lecture.

Though Marcus wants to point out as many mistakes as possible to the professor for his own amusement, the sedative power of Snørbørden's voice is so overwhelming, that Marcus cannot help but get some sleep during the lecture. Having seen all the slides in advance he can predict the boredom and therefore the amount of sleep that he needs. The amount of sleep is measured in slides which is an accurate way to measure time in Professor Snørbørden's lecture. Marcus has control over when he starts sleeping but being an experienced sleeper, Marcus also knows that he has to sleep in one long block rather than fragmenting his sleep throughout the lecture in order for the sleep to be revitalising. Sadly this restriction may cause Marcus to miss some of the flawed slides and therefore deny the auditorium the fun of yet another instance of a confused Professor Snørbørden which is the only entertainment there is.

Though Marcus is - as mentioned above - pretty clever, his problem solving skills deteriorated recently (probably due to being exposed to Professor Snørbørden). Marcus therefore needs your help to determine when to start his sleep in order to miss as few flawed slides as possible.

2 Input

The input file specifies several test cases. For every test case, there are two lines in the input file, the first containing the numbers s and n fulfilling $1 \leq s \leq n \leq 100000$ where s is the amount of sleep that Marcus needs (measured in slides) and n the number of slides in the lecture. The second line contains n numbers $b_0, b_1, ...b_{n-1}$, each of which is

either 0 or 1. b_i is 0 iff the ith slide is correct, consequently b_i is 1 iff the ith slide is flawed.

The input ends with one line containing the twice the number 0.

3 Output

For every test case output the number of the slide before which Marcus should start sleeping in order to miss as few flawed slides as possible. If there are several possibilities output the earliest one. The numbering of Professor Snørbørden's slides starts with zero (slide zero being the title slide and even this one being occasionally flawed).

4 Sample Input

```
3 5
1 1 0 0 0
5 7
1 0 0 1 0 0 1
5 15
0 1 1 1 0 0 1 0 0 0 1 1 0 1 0
0 0
```

5 Sample Output

```
2
1
4
```

The Dean's Daughters

1 Description

The Dean of the Faculty of Computer Science, the generous man he is, invites all his students to a banquet in his mansion at the end of every wintersemester. All the students show up, not so much for the anecdotes the Dean shares with the attendants but for the free beer and, above all, to see the beautiful daughters of the Dean. Since the number of female computer science students is negligible as is the number of male ones with girlfriends (we're talking about a nerdy subject after all), it is fair to say that every student attending the banquet has a crush on (exactly) one of the Deans daughters. Standing in a queue in front of the Deans house waiting for the Dean to let them in, every students is chatting with the student directly in front and directly behind him. Though computer science students tend to be peaceful in general, in case two students with the crush on the same daughter stand next to each other, they see no choice, but to settle the conflict like real men: Go home and meet on a neutral server to fight Counter Strike one on one, until both of them realize that gaming is better than girls anyway. Whenever two students wander off this way, the adjacent students close the gap and continue to chat.

The Dean wants to set the table and therefore needs to know how many students will be attending his banquet, that is, how many people will be waiting in front of the house when he opens the door.

2 Input

The input file consists of several test cases. There are two lines for each test case, the first containing a number $1 \leq s \leq 100000$ which denotes the number of students and a line with s numbers $d_1, d_2, \ldots, d_s$ specifying that the student standing at position i has a crush on the daughter d_i (the daughters being numbered from oldest to youngest). It can be assumed that the Dean has no more than 100000 daughters.

A seperate line containing the number 0 marks the end of the input and should not be processed.

3 Output

For each testcase output on a seperate line the number of students that will be left in the queue once it has stabilised.

4 Sample Input

```
3
1 1 1
5
2 2 3 1 1
7
2 1 3 1 3 2 3
10
1 2 3 4 5 5 4 3 2 1
0
```

5 Sample Output

```
1
1
7
0
```

Bicoloring

1 Description

In 1976 the "Four Color Map Theorem" was proven with the assistance of a computer. This theorem states that every map can be colored using only four colors, in such a way that no region is colored using the same color as a neighbor region. Here you are asked to solve a simpler similar problem. You have to decide whether a given arbitrary connected graph can be bicolored. That is, if one can assign colors (from a palette of two) to the nodes in such a way that no two adjacent nodes have the same color. To simplify the problem you can assume:

no node will have an edge to itself. the graph is nondirected. That is, if a node a is said to be connected to a node b, then you must assume that b is connected to a. the graph will be strongly connected. That is, there will be at least one path from any node to any other node.

2 Input

The input consists of several test cases. Each test case starts with a line containing the number n ($1 < n \leq 200$) of different nodes. The second line contains the number of edges l. After this, l lines will follow, each containing two numbers that specify an edge between the two nodes that they represent. A node in the graph will be labeled using a number a ($0 \leq a < n$). An input with n = 0 will mark the end of the input and is not to be processed.

3 Output

You have to decide whether the input graph can be bicolored or not, and print it as shown below.

4 Sample Input

```
3
3
0 1
1 2
2 0
9
8
0 1
0 2
0 3
0 4
0 5
0 6
0 7
0 8
0
```

5 Sample Output

```
NOT BICOLORABLE.
BICOLORABLE.
```

ASCII Trees

Description

John has been slacking off lately, and barely managed to sent in some buggy code before the deadline of the obligatory annual Christmas exercise sheet of the freshman's programming course, where the task is to generate ASCII Christmas trees of variable size. Having got back the results today (yeah, his tutor isn't the quickest one), he realizes that he's most likely failed to get the exercise credits required to take part in the exam. Not to take any chances for next year, he's resolved to do this properly, right now, knowing that the exact same task will turn up on next year's Christmas exercise sheet again. John has already conceived how his tree should look like, set up of triangles stacked on top of another. A triangle of size s contains s lines of lowercase os, with the kth line (counting from 1) containing $2k - 1$ os, like this for $s = 4$:

```
   o
  ooo
 ooooo
ooooooo
```

Simple, yet elegant. For aesthetical reasons however, if a triangle is on top of another tringle in a tree, the upper triangle has to have a smaller size than the one below it. To impress his tutor, John wants to print out a tree as high as possible (the height being the amount of lines), however there's one problem with that: his printer is low on ink, and is only capable of printing a certain amount of os before it runs out of ink. Now John wants to know from you the minimal amount n of os that is required to print a valid tree of height h, to see whether his printer will still manage to fully print this tree.

For instance, to get a tree of height $h = 8$, John will have to print out at least 26 os, because the most efficient tree of this height takes 3 triangles, one of size 1 on top of one of size 3, with one of size 4 at the bottom, which is valid because $1 < 3 < 4$:

```
   o
   o
  ooo
 ooooo
   o
  ooo
 ooooo
ooooooo
```

Input

Each line contains one value h, the desired height of the tree, $1 \leq h \leq 999999$. The input is terminated by a line containing the value 0, which should not be processed.

Output

For each desired tree height h in the input, print out the corresponding n, i.e. the minimal amount of os needed for a tree of height h, followed by a linebreak.

Sample Input

```
1
8
999999
22934
0
```

Sample Output

```
1
26
943217079
3284674
```

Fish

1 Description

In a small coastal country, all towns are situated on a long coastline (which we will model as a straight line). A long straight road runs along the coast, connecting the towns. The position of each town can be described by a single non-negative integer – the distance (in kilometers) from the start of the road.

Most of the citizens are fishermen, and they catch great amounts of fish. After the fishing season is over and before the tourist season starts, the fish can be transported between different towns. A town can accommodate x tourists if it has x tons of fish available. The goal is to accommodate the largest possible number of tourists while distributing them evenly between towns. In other words, we want to find the largest integer y for which it is possible to distribute fish so that each town can accommodate at least y tourists.

In one shipment, an integral number of tons of fish is sent from one town to another. During transportation, one ton of fish per kilometer traveled is lost to hungry pillagers descending from the mountains. More formally, if a town ships f tons of fish to another town that is d kilometers away, then $f - d$ tons will arrive at the destination; if f is less than d, then the entire shipment is lost.

It is possible to arbitrarily repackage and combine shipments in intermediate towns. For example, we can send shipments from towns a and b to town c, combine half of the remaining fish from both shipments with the fish originating in c and send it in a single large shipment from town c to town d.

2 Task

Write a program that, given the positions of all towns and the amount of fish each town produces, determines the largest number of tourists that can be accommodated by each city after the fish has been distributed.

3 Input

The first line contains a single integer n, $1 \leq n \leq 10^5$, the number of towns. Each of the following n lines contains two integers p and f, $0 \leq p$, $f \leq 10^{12}$, the position of a town

(in kilometers) and the amount of fish it produces (in tons). The towns will be sorted in ascending order of position. The positions of all towns will be distinct.

4 Output

Output the largest number of tourists y from the task description followed by a newline character.

5 Sample Input

```
3
1 0
2 21
4 0

3
5 70
15 100
1200 20

4
20 300
40 400
340 700
360 600
```

6 Sample Output

```
6

20

415
```