POJ 3349 Snowflake Snow Snowflakes

Time Limit:4000MS Memory Limit:65536K

Description

You may have heard that no two snowflakes are alike. Your task is to write a program to determine whether this is really true. Your program will read information about a collection of snowflakes, and search for a pair that may be identical. Each snowflake has six arms. For each snowflake, your program will be provided with a measurement of the length of each of the six arms. Any pair of snowflakes which have the same lengths of corresponding arms should be flagged by your program as possibly identical.

Input

The first line of input will contain a single integer n, $0 < n \leq 100000$, the number of snowflakes to follow. This will be followed by n lines, each describing a snowflake. Each snowflake will be described by a line containing six integers (each integer is at least 0 and less than 10000000), the lengths of the arms of the snow ake. The lengths of the arms will be given in order around the snowflake (either clockwise or counterclockwise), but they may begin with any of the six arms. For example, the same snowflake could be described as 1 2 3 4 5 6 or 4 3 2 1 6 5.

Output

If all of the snowflakes are distinct, your program should print the message:
No two snowflakes are alike.
If there is a pair of possibly identical snow akes, your program should print the message:
Twin snowflakes found.

Sample Input

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

Sample Output

```
Twin snowflakes found.
```

ZJ2 d062: [2005] C - Computer Connectivity

內容：

Consider a set of N computers numbered from 1 to N, and a set S of M computer pairs, where each pair (i,j) in S indicates that computers i and j are connected. The connectivity rule says that if computers i and j are connected, and computers j and k are connected, then computers i and k are connected, too, no matter whether (i,k) or (k,i) is in S or not. Based on S and the connectivity rule, the set of N computers can be divided into a number of groups such that for any two computers, they are in the same group if and only if they are connected. Note that if a computer is not connected to any other one, itself forms a group. A group is said to be largest if the number of computers in it is maximum among all groups. The problem asks to count how many computers there are in a largest group.

輸入說明：

The first line of the input file contains the number of test cases. For each test case, the first line consists of N ($1 \le N \le 30000$) and M ($1 \le M \le 100000$), where N is the number of computers and M is the number of computer pairs in S. Each of the following M lines consists of two integers i and j ($1 \le i \le N$, $1 \le j \le N$, $i = j$) indicating that (i,j) is in S. Note that there could be repetitions among the pairs in S.

輸出說明：

For each test case, the output should contain one number on a line, denoting the amount of computers in a largest group.

範例輸入：

```
2
3 4
1 2
3 2
2 3
1 2
10 12
```

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

範例輸出 ：

```
3
6
```

POJ 3273 Monthly Expense

Time Limit:2000MS Memory Limit:65536K

Description

Farmer John is an astounding accounting wizard and has realized he might run out of money to run the farm. He has already calculated and recorded the exact amount of money ($1 \le money_i \le 10,000$) that he will need to spend each day over the next N ($1 \le N \le 100,000$) days.

FJ wants to create a budget for a sequential set of exactly M ($1 \le M \le N$) fiscal periods called "fajomonths". Each of these fajomonths contains a set of 1 or more consecutive days. Every day is contained in exactly one fajomonth.

FJ's goal is to arrange the fajomonths so as to minimize the expenses of the fajomonth with the highest spending and thus determine his monthly spending limit.

Input

Line 1: Two space-separated integers: N and M
Lines 2..N+1: Line i+1 contains the number of dollars Farmer John spends on the ith day

Output

Line 1: The smallest possible monthly limit Farmer John can afford to live with.

Sample Input

```
7 5
100
400
```

```
300
100
500
101
400
```

Sample Output

```
500
```

POJ 3258 River Hopscotch

Time Limit:2000MS Memory Limit:65536K

Description

Every year the cows hold an event featuring a peculiar version of hopscotch that involves carefully jumping from rock to rock in a river. The excitement takes place on a long, straight river with a rock at the start and another rock at the end, L units away from the start ($1 \le L \le 1,000,000,000$). Along the river between the starting and ending rocks, N ($0 \le N \le 50,000$) more rocks appear, each at an integral distance D_i from the start ($0 < D_i < L$).

To play the game, each cow in turn starts at the starting rock and tries to reach the finish at the ending rock, jumping only from rock to rock. Of course, less agile cows never make it to the final rock, ending up instead in the river.

Farmer John is proud of his cows and watches this event each year. But as time goes by, he tires of watching the timid cows of the other farmers limp across the short distances between rocks placed too closely together. He plans to remove several rocks in order to increase the shortest distance a cow will have to jump to reach the end. He knows he cannot remove the starting and ending rocks, but he calculates that he has enough resources to remove up to M rocks ($0 \le M \le N$).

FJ wants to know exactly how much he can increase the shortest distance **before** he starts removing the rocks. Help Farmer John determine the greatest possible shortest distance a cow has to jump after removing the optimal set of M rocks.

Input

Line 1: Three space-separated integers: L, N, and M
Lines 2..N+1: Each line contains a single integer indicating how far some rock is away from the starting rock. No two rocks share the same position.

Output

Line 1: A single integer that is the maximum of the shortest distance a cow has to jump after removing *M* rocks

Sample Input

```
25 5 2
2
14
11
21
17
```

Sample Output

```
4
```