Task: BIT
Three Bit Computer

day 3, source file `bit.*`, available memory 32 MB *CPSPC 2004, June 10*

The scientists in the Kingdom of Byteland want to develop a new type of computer, namely the Three Bit Computer (TBC). They predict that the new machine will have the power to solve many hard and still unsolved problems. However, there are some technical difficulties that have to be resolved first. You have been asked to assist the scientists in solving one of them.

The scientists are now working on the initialization procedure for the computer memory. The current version of TBC has n bits of memory numbered from 1 to n. Each bit can have one of three values a, b, c or might be uninitialized. The following memory initialization operations are supported by the computer:

- two consecutive uninitialized bits, i.e., the bits numbered i and $i+1$ for $1 \leq i < n$, can be set to two different values,
- two consecutive bits, one unitialized and one initialized to value x, can be set to two different values not equal to x.

For example, the following initialization procedure is possible for $n = 4$, $uuuu \to uuab \to ucbb \to babb$, where u denotes an uninitialized bit of memory.

Task

Write a program that:

- reads the values to which the the memory has to be initialized,
- checks if the initialization is possible,
- writes the answer to the output file.

Input

The input file `bit.in` can contain several initialization patterns. The first line of the input file contains a single positive integer N, ($1 \leq N \leq 10$), the number of patterns. The description of the patterns follows. Description of a single pattern consists of two consecutive lines. The first one contains a positive integer l_i, ($1 \leq l_i \leq 100\,000$), the length of the i-th pattern (i.e., the size of computer's memory). The second line contains a sequence of length l_i consisting of letters a, b, c — the pattern itself.

Output

The output file `bit.out` should have N lines, one for each pattern. The i-th line should consist of a single word `YES`, if the initialization is possible, or `NO` otherwise.

Example

For the input file `bit.in`:

```
2
4
aabb
4
aaab
```

the correct result is the output file `bit.out`:

```
NO
YES
```

Task: BIT2
Three Bit Computer strikes back

day 4, source file `bit2.*`, available memory 32 MB *CPSPC 2004, June 11*

The Three Bit Computer is a huge failure in spite of continuing efforts of Byteland's leading scientists, but now they have a new and powerful idea: the Quantum Three Bit Computer (QTBC). TBC can now be regarded a warm-up before the real thing — the QTBC.

They predict that the new machine will have the power, blah, blah., many silly problems still to solve, blah, blah. Now, let us get to the point.

The initialization procedure is again a major issue, but with quantum computers it is a completely different story. The trouble now is that everything you do has side effects, i.e., it influences everything else. Taking care of this is very time consuming and initializing the memory bit by bit is simply impossible. But there is another approach. The scientists have developed large scale controlled impulses (LSCI), which influence all the memory bits at the same time, and moreover, it is known exactly what their influence is. They are also very fast, so emitting even a large number of impulses is better then initializing the memory bit by bit. The first question to consider is whether there exist a sequence of impulses that will zero the whole memory. Your task is to write a computer program to answer this question.

More formally, each memory bit can be in one of n states numbered $0, \ldots, n-1$. The LSCI impulse f changes all the bits in exactly the same way, i.e. it can be viewed as a function $f : \{0, \ldots, n-1\} \to \{0, \ldots, n-1\}$. For example, $f(3) = 5$ means that if the impulse f is emitted, then every bit in state 3 will change its state to 5. The scientists know how to emit several impulses $f_1, \ldots, f_k$. You have to find out if there exists a sequence of impulses that brings all the bits to state 0 regardless of their initial state.

Task

Write a program that:

- reads the descriptions of the impulses available,
- checks if zeroing the memory is possible,
- writes the answer to the output file.

Input

The input file `bit2.in` can contain several test cases. The first line of the input file contains a single positive integer T, ($1 \leq T \leq 10$), the number of test cases. The description of the test cases follows. Description of a single test case starts with a line containing two positive integers n, k, ($1 \leq n \leq 200$, $1 \leq k \leq 5$), where n is the number of different states of memory bits and k is the number of different impulses that can be emitted. The next k lines contain the description of the impulses, the i-th line contains the description of the i-th impulse. The description of an impulse f is a sequence of integers $f(0) \ldots f(n-1)$ describing the influence f has on the state of any memory bit. These integers are separated by single spaces.

Output

The output file `bit2.out` should have T lines, one for each test case. The i-th line should consist of a single word `YES` if zeroing the memory in the i-th test case if possible, or `NO` otherwise.

Example

For the input file `bit2.in`:

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

the correct result is the output file `bit2.out`:

```
NO
YES
```

Task: EXP
Express delivery

Stage CSPC 2006. Day Fourth. Source file `exp.*` *17.06.2006*
Available memory: 32 MB.

Your company, Byteland Express, has to deliver some parcels to clients located somewhere in the city. The city consists of $2 \cdot 10^9 + 1$ south–north streets and $\cdot 10^9 + 1$ west–east streets (both numbered from 0 to $2 \cdot 10^9$). A postman drives from one junction to a neighbouring one in one minute. The company is located next to the junction of streets number x_c and y_c.

All parcels have to be delivered as fast as possible, so driving to a client located next to the junction of the x-th and y-th street, can take at most (=exactly) $|x_c - x| + |y_c - y|$ minutes. It takes a very short time to give a parcel to a client, so while driving to one client, a postman can give a parcel to another client (but can not choose longer route to do that). Your task is to determine how many postmen are needed to perform the delivery.

Task

Write a program, that

- reads using SVIO the position of the company and positions of all the clients,
- finds the minimal number of postmen needed to deliver all the parcels,
- writes the result using SVIO.

Input

First line of input contains one integer N ($1 \leq N \leq 1\,000\,000$). Second line contains location of the company, the following N lines contain locations of the clients. A description of the location consists of two integers: coordinates x and y ($0 \leq x, y \leq 2 \cdot 10^9$). Every two points (from among company or clients) *differ on both coordinates* (every x is different and every y is different).

Output

In the only line of the standard output write one integer: the number of postmen needed to deliver parcels to all clients.

Example

For the input data:

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

the correct result is:

```
3
```

Die Young (Ne21)

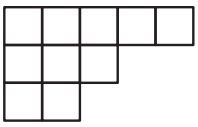
Input File: stdin
Output File: stdout
Source Code: young.pas/.c/.cpp

35 points
Time limit: 1 s
Memory limit: 64 MB

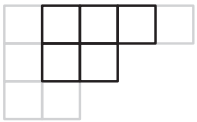
Young diagram is a well known way to describe a partition of a positive integer number. A partition of a number n is a representation as a sum of one or several integer numbers $n = m_1 + m_2 + \ldots + m_k$ where $m_1 \geq m_2 \geq \ldots \geq m_k$.

A diagram consists of n boxes arranged in k rows, where k is the number of terms in the partition. A row representing the number m_i contains m_i boxes. All rows are left-aligned, and sorted from longest to shortest.

The diagram on the picture below corresponds to the partition $10 = 5 + 3 + 2$.



Sometimes it is possible to *inscribe* one Young diagram into the other. Diagram X can be inscribed into the diagram Y if it is possible to delete some boxes from diagram Y so that it turns to diagram X. Note that it is only allowed to remove some boxes, it is not allowed to rotate or flip the diagram. For example, the picture below shows that the diagram for $5 = 3 + 2$ can be inscribed into the diagram for $10 = 5 + 3 + 2$.



On the other hand, for example, it is impossible to inscribe the diagram for $8 = 4+4$ into the diagram for $10 = 5 + 3 + 2$.

Given n, your task to find such partition of n that the corresponding Young diagram has the greatest possible number of diagrams that can be inscribed into it.

For example, there are 36 Young diagrams that can be inscribed into the diagram for $10 = 5 + 3 + 2$. However, it is not the maximal possible value. The diagram for $10 = 4 + 2 + 2 + 1 + 1$ has the better value, there are 41 diagrams that can be inscribed into it.

Input

Input file contains n ($1 \leq n \leq 100$).

Output

At the first line of the output file print the maximal number of Young diagrams that can be inscibed into some Young diagram for the partition of n.

At the second line print one or more integer numbers — the number of boxes in each row of the optimal diagram.

Example

Input	Output
10	41 4 2 2 1 1