Codeforces Round #599 (Div. 2)

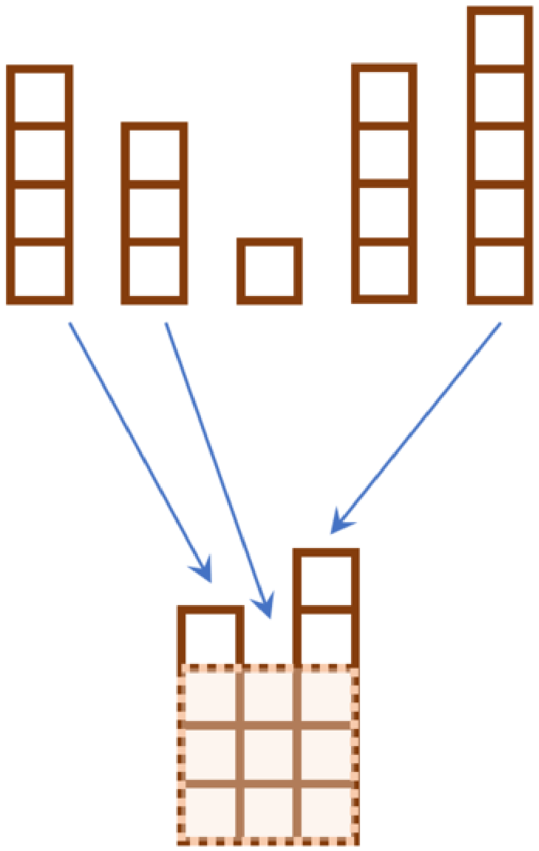
A. Maximum Square

1 second, 256 megabytes

Ujan decided to make a new wooden roof for the house. He has n rectangular planks numbered from 1 to n. The i-th plank has size $a_i \times 1$ (that is, the width is 1 and the height is a_i).

Now, Ujan wants to make a square roof. He will first choose some of the planks and place them side by side in some order. Then he will glue together all of these planks by their vertical sides. Finally, he will cut out a square from the resulting shape in such a way that the sides of the square are horizontal and vertical.

For example, if Ujan had planks with lengths 4, 3, 1, 4 and 5, he could choose planks with lengths 4, 3 and 5. Then he can cut out a 3×3 square, which is the maximum possible. Note that this is not the only way he can obtain a 3×3 square.



What is the maximum side length of the square Ujan can get?

Input

The first line of input contains a single integer k $(1 \leq k \leq 10)$, the number of test cases in the input.

For each test case, the first line contains a single integer n ($1 \leq n \leq 1\,000$), the number of planks Ujan has in store. The next line contains n integers $a_1, \ldots, a_n$ $(1 \leq a_i \leq n)$, the lengths of the planks.

Output

For each of the test cases, output a single integer, the maximum possible side length of the square.

input

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

output

```
3
4
1
3
```

The first sample corresponds to the example in the statement.

In the second sample, gluing all 4 planks will result in a 4×4 square.

In the third sample, the maximum possible square is 1×1 and can be taken simply as any of the planks.

B1. Character Swap (Easy Version)

1 second, 256 megabytes

This problem is different from the hard version. In this version Ujan makes ***exactly*** *one exchange. You can hack this problem only if you solve both problems.*

After struggling and failing many times, Ujan decided to try to clean up his house again. He decided to get his strings in order first.

Ujan has two **distinct** strings s and t of length n consisting of only of lowercase English characters. He wants to make them equal. Since Ujan is lazy, he will perform the following operation exactly once: he takes two positions i and j ($1 \leq i, j \leq n$, the values i and j can be equal or different), and swaps the characters s_i and t_j. Can he succeed?

Note that he has to perform this operation **exactly** once. He **has** to perform this operation.

Input

The first line contains a single integer k $(1 \leq k \leq 10)$, the number of test cases.

For each of the test cases, the first line contains a single integer n ($2 \leq n \leq 10^4$), the length of the strings s and t.

Each of the next two lines contains the strings s and t, each having length exactly n. The strings consist only of lowercase English letters. It is guaranteed that strings are different.

Output

For each test case, output "`Yes`" if Ujan can make the two strings equal and "`No`" otherwise.

You can print each letter in any case (upper or lower).

input

```
4
5
souse
houhe
3
cat
dog
2
aa
az
3
abc
bca
```

output

```
Yes
No
No
No
```

In the first test case, Ujan can swap characters s_1 and t_4, obtaining the word "house".

In the second test case, it is not possible to make the strings equal using exactly one swap of s_i and t_j.

B2. Character Swap (Hard Version)

1 second, 256 megabytes

This problem is different from the easy version. In this version Ujan makes at most $2n$ swaps. In addition, $k \le 1000, n \le 50$ and it is necessary to print swaps themselves. You can hack this problem if you solve it. But you can hack the previous problem only if you solve both problems.

After struggling and failing many times, Ujan decided to try to clean up his house again. He decided to get his strings in order first.

Ujan has two **distinct** strings s and t of length n consisting of only of lowercase English characters. He wants to make them equal. Since Ujan is lazy, he will perform the following operation at most $2n$ times: he takes two positions i and j ($1 \le i, j \le n$, the values i and j can be equal or different), and swaps the characters s_i and t_j.

Ujan's goal is to make the strings s and t equal. He does not need to minimize the number of performed operations: **any** sequence of operations of length $2n$ or shorter is suitable.

Input

The first line contains a single integer k ($1 \le k \le 1000$), the number of test cases.

For each of the test cases, the first line contains a single integer n ($2 \le n \le 50$), the length of the strings s and t.

Each of the next two lines contains the strings s and t, each having length exactly n. The strings consist only of lowercase English letters. It is guaranteed that strings are different.

Output

For each test case, output "Yes" if Ujan can make the two strings equal with at most $2n$ operations and "No" otherwise. You can print each letter in any case (upper or lower).

In the case of "Yes" print m ($1 \le m \le 2n$) on the next line, where m is the number of swap operations to make the strings equal. Then print m lines, each line should contain two integers i, j ($1 \le i, j \le n$) meaning that Ujan swaps s_i and t_j during the corresponding operation. You do not need to minimize the number of operations. Any sequence of length not more than $2n$ is suitable.

input

```
4
5
souse
houhe
3
cat
dog
2
aa
az
3
abc
bca
```

output

```
Yes
1
1 4
No
No
Yes
3
1 2
3 1
2 3
```

C. Tile Painting

1 second, 256 megabytes

Ujan has been lazy lately, but now has decided to bring his yard to good shape. First, he decided to paint the path from his house to the gate.

The path consists of n consecutive tiles, numbered from 1 to n. Ujan will paint each tile in some color. He will consider the path *aesthetic* if for any two **different** tiles with numbers i and j, such that $|j - i|$ is a divisor of n greater than 1, they have the same color. Formally, the colors of two tiles with numbers i and j should be the same if $|i - j| > 1$ and $n \bmod |i - j| = 0$ (where $x \bmod y$ is the remainder when dividing x by y).

Ujan wants to brighten up space. What is the maximum number of different colors that Ujan can use, so that the path is aesthetic?

Input

The first line of input contains a single integer n ($1 \le n \le 10^{12}$), the length of the path.

Output

Output a single integer, the maximum possible number of colors that the path can be painted in.

input

```
4
```

output

```
2
```

input

```
5
```

output

```
5
```

In the first sample, two colors is the maximum number. Tiles 1 and 3 should have the same color since $4 \bmod |3 - 1| = 0$. Also, tiles 2 and 4 should have the same color since $4 \bmod |4 - 2| = 0$.

In the second sample, all five colors can be used.

D. 0-1 MST

1 second, 256 megabytes

Ujan has a lot of useless stuff in his drawers, a considerable part of which are his math notebooks: it is time to sort them out. This time he found an old dusty graph theory notebook with a description of a graph.

It is an undirected weighted graph on n vertices. It is a complete graph: each pair of vertices is connected by an edge. The weight of each edge is either 0 or 1; exactly m edges have weight 1, and all others have weight 0.

Since Ujan doesn't really want to organize his notes, he decided to find the weight of the minimum spanning tree of the graph. (The weight of a spanning tree is the sum of all its edges.) Can you find the answer for Ujan so he stops procrastinating?

Input

The first line of the input contains two integers n and m ($1 \leq n \leq 10^5$, $0 \leq m \leq \min(\frac{n(n-1)}{2}, 10^5)$), the number of vertices and the number of edges of weight 1 in the graph.

The i-th of the next m lines contains two integers a_i and b_i ($1 \leq a_i, b_i \leq n$, $a_i \neq b_i$), the endpoints of the i-th edge of weight 1.

It is guaranteed that no edge appears twice in the input.

Output

Output a single integer, the weight of the minimum spanning tree of the graph.

input

```
6 11
1 3
1 4
1 5
1 6
2 3
2 4
2 5
2 6
3 4
3 5
3 6
```

output

```
2
```

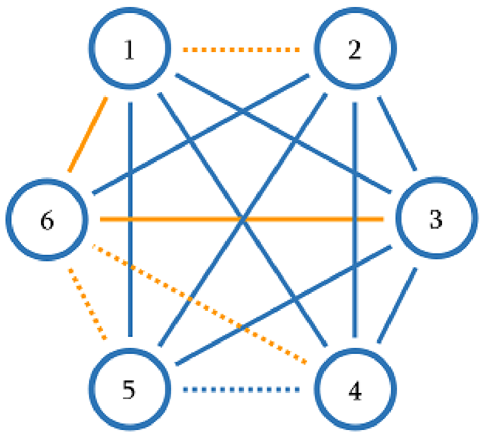
input

```
3 0
```

output

```
0
```

The graph from the first sample is shown below. Dashed edges have weight 0, other edges have weight 1. One of the minimum spanning trees is highlighted in orange and has total weight 2.



In the second sample, all edges have weight 0 so any spanning tree has total weight 0.

E. Sum Balance

1 second, 256 megabytes

Ujan has a lot of numbers in his boxes. He likes order and balance, so he decided to reorder the numbers.

There are k boxes numbered from 1 to k. The i-th box contains n_i integer numbers. The integers can be negative. **All of the integers are distinct.**

Ujan is lazy, so he will do the following reordering of the numbers **exactly once**. He will pick a single integer from each of the boxes, k integers in total. Then he will insert the chosen numbers — one integer in each of the boxes, so that the number of integers in each box is the same as in the beginning. Note that he may also insert an integer he picked from a box back into the same box.

Ujan will be happy if the sum of the integers in each box is the same. Can he achieve this and make the boxes perfectly balanced, like all things should be?

Input

The first line contains a single integer k ($1 \leq k \leq 15$), the number of boxes.

The i-th of the next k lines first contains a single integer n_i ($1 \leq n_i \leq 5\,000$), the number of integers in box i. Then the same line contains n_i integers $a_{i,1}, \ldots, a_{i,n_i}$ ($|a_{i,j}| \leq 10^9$), the integers in the i-th box.

It is guaranteed that all $a_{i,j}$ are distinct.

Output

If Ujan cannot achieve his goal, output "No" in a single line. Otherwise in the first line output "Yes", and then output k lines. The i-th of these lines should contain two integers c_i and p_i. This means that Ujan should pick the integer c_i from the i-th box and place it in the p_i-th box afterwards.

If there are multiple solutions, output any of those.

You can print each letter in any case (upper or lower).

input

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

output

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

input

```
2
2 3 -2
2 -1 5
```

output

```
No
```

input

```
2
2 -10 10
2 0 -20
```

output

```
Yes
-10 2
-20 1
```

In the first sample, Ujan can put the number 7 in the 2nd box, the number 2 in the 3rd box, the number 5 in the 1st box and keep the number 10 in the same 4th box. Then the boxes will contain numbers $\{1, 5, 4\}$, $\{3, 7\}$, $\{8, 2\}$ and $\{10\}$. The sum in each box then is equal to 10.

In the second sample, it is not possible to pick and redistribute the numbers in the required way.

In the third sample, one can swap the numbers -20 and -10, making the sum in each box equal to -10.