Problem A. Adjacent Product Sum

Time limit: 1 second
Memory limit: 256 megabytes

You have n numbers $a_1, a_2, \ldots, a_n$. You want to arrange them in a circle so as to maximize the sum of products of pairs of adjacent numbers.

Formally, you want to find a permutation $b_1, b_2, \ldots, b_n$ of $a_1, a_2, \ldots, a_n$ such that $b_1 b_2 + b_2 b_3 + \ldots + b_{n-1} b_n + b_n b_1$ is maximal.

Find this maximum value.

Input

The first line contains a single integer t $(1 \le t \le 10^4)$ — the number of test cases. The description of test cases follows.

The first line of each test case contains a single integer n $(3 \le n \le 2 \cdot 10^5)$ — the number of numbers.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(-10^6 \le a_i \le 10^6)$.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer — the maximum possible value of the expression $b_1 b_2 + b_2 b_3 + \ldots + b_{n-1} b_n + b_n b_1$ for all permutations $b_1, b_2, \ldots, b_n$ of $a_1, a_2, \ldots, a_n$.

Example

standard input	standard output

4
3
1 2 3
6
1 1 1 1 0 0
5
100000 100000 100000 100000 -100000
5
1 2 3 4 5
	11
	3
	10000000000
	48

Note

In the first test case, there is only one way to arrange the numbers in a circle (not counting rotations and symmetries) — $(1, 2, 3)$. The sum of products of pairs of adjacent numbers is $1 \cdot 2 + 2 \cdot 3 + 3 \cdot 1 = 11$.

In the second test case, one of the optimal arrangements is $(1, 1, 1, 1, 0, 0)$. For it this sum is equal to $1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 0 + 0 \cdot 0 + 0 \cdot 1 = 3$.

In the third test case, there is a unique (up to rotations and symmetries) way to arrange the numbers in a circle: $(100000, 100000, 100000, 100000, -100000)$, the answer is $100000^2 = 10^{10}$. Note that the answer may not fit into int32.

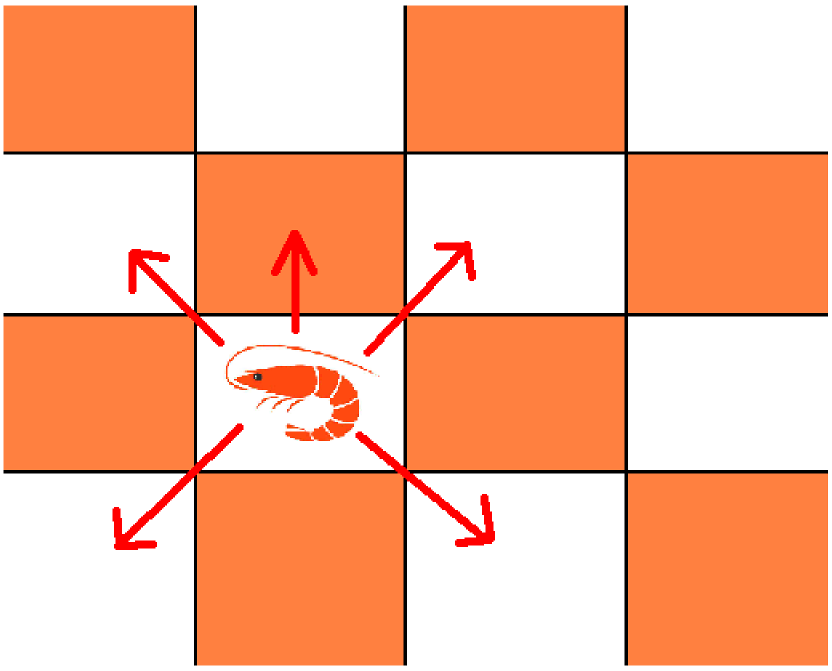
In the fourth test case, one of the optimal permutations is $(1, 2, 4, 5, 3)$, the answer for which is $1 \cdot 2 + 2 \cdot 4 + 4 \cdot 5 + 5 \cdot 3 + 3 \cdot 1 = 2 + 8 + 20 + 15 + 3 = 48$.

Problem B. Best Chess Piece

Time limit: 1 second
Memory limit: 256 megabytes

You decided that chess is boring, and it's time to create a new piece — a prawn. In one move, it can move forward by 1, or to any diagonally adjacent cell.

Formally, if the prawn is currently at the cell (x, y), then in one move, it can move to one of the cells $(x, y+1), (x-1, y-1), (x-1, y+1), (x+1, y-1), (x+1, y+1)$.



Currently, the prawn is at cell $(0, 0)$ of the infinite board. At least how many moves does it need to make to get to the cell (a, b)?

Input

The first line contains a single integer t $(1 \le t \le 10^4)$ — the number of test cases. The description of test cases follows.

The only line of each test case contains two integers a, b $(-10^9 \le a, b \le 10^9)$ — indicating that prawn needs to get to the cell (a, b).

Output

For each test case, output a single integer — the smallest number of moves that the prawn has to make to get to the cell (a, b).

Example

standard input	standard output

4
0 0
1000 1000
0 -1
3 -1
	0
	1000
	3
	3

Note

In the first test case, the prawn is already at its destination, so the answer is 0.

In the second test case, one of the possible routes for the prawn is $(0, 0) \to (1, 1) \to (2, 2) \to \ldots \to (1000, 1000)$.

In the third test case, one of the possible routes for the prawn is $(0, 0) \to (0, 1) \to (1, 0) \to (0, -1)$.

Problem C. Cyclic Segment Sums

Time limit: 1 second
Memory limit: 256 megabytes

You are given two integers n, k $(2 \le k < n)$.

For an array $a_1, a_2, \ldots, a_n$, which **consists only of zeros and ones**, we define the array $f(a)$ as follows:

- $f(a)_i = a_i + a_{i+1} + \ldots + a_{i+k-2} + a_{i+k-1}$ (here we assume that $a_{n+i} = a_i$, i.e. numbers are arranged in a circle).

  For example, for $n = 4, k = 2$, $f([0,1,1,0]) = [1,2,1,0]$.

Consider all 2^n possible arrays a, and for each of them, find $f(a)$. How many different arrays are there among them? Since the answer may be very large, print the number of these arrays modulo 998244353.

Two arrays are considered different if they differ in at least one position.

Input

The first line contains a single integer t $(1 \le t \le 10^5)$ — the number of test cases. The description of test cases follows.

The first line of each test case contains two integers n, k $(2 \le k < n \le 10^6)$.

Output

For each test case print the number of different arrays among $f(a)$, modulo 998244353.

Example

standard input	standard output

4
3 2
4 2
42 3
123123 123
	8
	15
	780086989
	126500246

Note

For $n = 3, k = 2$, there are 8 different arrays a of ones and zeros. The corresponding arrays $f(a)$ are pairwise distinct for them.

For $n = 4, k = 2$, there are 16 distinct arrays a of ones and zeros. The only pair of matching arrays $f(a)$ is $[0,1,0,1]$ and $[1,0,1,0]$: for both arrays $f(a)$ is $[1,1,1,1]$.

Problem D. Different XOR

Time limit: 1 second
Memory limit: 256 megabytes

For an array $[b_1, b_2, \ldots, b_k]$ of integers, let's define its **weight** as the XOR of all its elements.

Here XOR denotes the **bitwise exclusive OR** operation. For example, $13\ XOR\ 6 = 11$, because in binary, $13 = \texttt{1101}$ and $6 = \texttt{0110}$, so their XOR must be equal to $\texttt{1011} = 11$. The weight of array $[13, 1, 4]$, for example, is 8.

You are given an array $[a_1, a_2, \ldots, a_n]$ of integers. We want to divide it into several (**more than one**) consecutive subarrays whose weights are pairwise different. Determine if this is possible. If it is possible, find one of such partitions.

Input

The first line contains a single integer t $(1 \le t \le 10^4)$ — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n $(2 \le n \le 2 \cdot 10^5)$ — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i < 2^{30})$ — the elements of the array.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each set of input data, if no such partitioning exists, print `NO`.

Otherwise, print `YES`. On the following line, print a single integer k $(2 \le k \le n)$ — the number of subarrays into which you are splitting a.

On the i-th of the next k lines print two numbers l_i, r_i $(1 \le l_i \le r_i \le n)$, denoting that the i-th of your arrays is $[a_{l_i}, a_{l_i+1}, \ldots, a_{r_i}]$. You can print these subarrays in any order, but each number from 1 to n must appear in **exactly one** of the segments $[l_i, r_i]$.

Example

standard input	standard output
4	NO
2	YES
0 0	3
3	1 1
1 2 3	2 2
5	3 3
16 8 4 2 1	YES
6	2
42 42 42 42 42 42	1 1
	2 5
	NO

Note

In the first test case, there is no way to split $[0, 0]$ into at least two subarrays with distinct XORs.

In the second test case, you can split array $[1, 2, 3]$ into 3 subarrays $[1], [2], [3]$ correspondingly, with XORs $1, 2, 3$ correspondingly.

Problem E. Equalize

Time limit: 1 second
Memory limit: 256 megabytes

Let's call array $[b_1, b_2, \dots, b_m]$ **good**, if $b_1 \cdot b_2 \cdot \dots \cdot b_m = b_1 + b_2 + \dots + b_m$.

You are given an array $a_1, a_2, \dots, a_n$. In one operation, you can append any **integer** element to this array. What's the smallest number of operations you have to make to make it **good**?

Input

The first line contains a single integer t $(1 \le t \le 5 \cdot 10^4)$ — the number of test cases. The description of test cases follows.

The first line of each test case contains a single integer n $(1 \le n \le 2 \cdot 10^5)$ — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \dots, a_n$ $(-10^6 \le a_i \le 10^6)$ — the elements of the array.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the smallest number of operations you have to make to make the array good.

Example

standard input	standard output

5
1
5
2
3 2
6
1 1 -1 2 1 1
3
-1 0 1
6
1 -1 2 -1 1 1
	0
	1
	2
	0
	1

Note

In the first test case, the array $[5]$ is already good, as both product and sum are 5.

In the second test case, initially, the product is 6, and the sum is 5. We can append 1 to this array, it will become $[3, 2, 1]$, with both product and sum equal to 6.

In the third test case, we can append -2 and 1 to our array, it will become $[1, 1, -1, 2, 1, 1, -2, 1]$, with both sum and product equal to 4.

In the fourth test case, the array $[-1, 0, 1]$ is already good, with both product and sum equal to 0.

In the fifth test case, you can append 3 to our array, it will become $[1, -1, 2, -1, 1, 1, 3]$, with both sum and product equal to 6.

Problem F. Form ABC

Time limit: 1 second
Memory limit: 256 megabytes

Let's call a string t, consisting of characters `A`, `B`, `C` **alphabetic**, if it contains `ABC` as a subsequence.

You are given a string s consisting of characters `A`, `B`, `C`. It's guaranteed that it contains at least one `A`, at least one `B`, and at least one `C`.

In one operation, you can swap any two adjacent characters of the string. What's the smallest number of operations you have to make to make the string **alphabetic**?

A string a is a subsequence of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters.

Input

The first line of the input contains a single integer t ($1 \le t \le 5 \cdot 10^4$). The description of the test cases follows.

The only line of each test case contains a single string s (length of s doesn't exceed $2 \cdot 10^5$). It's guaranteed that s consists of characters `A`, `B`, `C` and that it contains at least one `A`, at least one `B`, and at least one `C`.

It's guaranteed that the sum of lengths of s over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, output the smallest number of operations you have to make to make the string **alphabetic**.

Example

standard input	standard output
6 ABC BCA CBA CBCBABA BAAACCCB CCCBBBAAA	0 2 3 3 1 5

Note

In the first test case, string `ABC` already contains `ABC` as a subsequence.

In the second test case, it's enough to do two operations: `BCA` $\to$ `BAC` $\to$ `ABC`.

In the third test case, it's enough to do three operations: `CBA` $\to$ `BCA` $\to$ `BAC` $\to$ `ABC`.

In the fourth test case, it's enough to do three operations: `CBCBABA` $\to$ `CBCABBA` $\to$ `CBACBBA` $\to$ `CBABCBA`.

In the fifth test case, it's enough to do one operation: `BAAACCCB` $\to$ `BAAACCBC`. Note that the resulting string doesn't have to contain `ABC` as a substring.

Problem G. Graph and Hamiltonian Cycles

Time limit: 1 second
Memory limit: 256 megabytes

You are given a string s of length $2n$, containing n characters `W` and n characters `B`.

Let's build a graph on $2n$ nodes. If $s_i \neq s_j$ for some $1 \leq i < j \leq 2n$, then there is an edge of weight $|i - j|$ between nodes i and j in this graph. There are no other edges.

Find the number of shortest Hamiltonian cycles in this graph. As this number can be very large, output it modulo 998244353.

As a reminder, a Hamiltonian cycle is a cycle that visits each node exactly once. The length of the cycle is equal to the sum of the weights of its edges. Two cycles are called different if there is an edge that one contains and the other doesn't.

Input

The first line contains a single integer t $(1 \leq t \leq 10^4)$ — the number of test cases. The description of test cases follows.

The first line of each test case contains a single integer n $(2 \leq n \leq 10^6)$.

The second line of each test case contains a string s of length $2n$, containing n characters `W` and n characters `B`.

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

Output

For each test case, output the number of shortest Hamiltonian cycles in this graph, modulo 998244353.

Example

standard input	standard output
3 2 WWBB 3 WBWBWB 7 WWWWBWBBWWBBBB	1 2 62208

Note

In the first test case, the graph has 4 edges: $(1, 3)$ with weight 2, $(1, 4)$ with weight 3, $(2, 3)$ with weight 1, and $(2, 4)$ with weight 2.

There is only one Hamiltonian cycle here: $1 \to 3 \to 2 \to 4 \to 1$ (Note that, for example, cycle $1 \to 4 \to 2 \to 3$ contains the same set of edges, so we have already counted it).

Problem H. Hating 3-Colorability

Time limit: 1 second
Memory limit: 256 megabytes

A graph is called k-colorable if it's possible to color each its node in one of k colors so that for any two nodes u and v, which are connected by an edge, their colors will be different.

You are given a **connected bipartite** graph, where one part has n nodes, numbered from 1 to n, and the second part has n nodes, numbered from $n+1$ to $2n$. There is no edge between any two nodes from the first part, and there is no edge between any two nodes from the second part.

(A graph is called bipartite if its nodes can be split into two nonempty parts, such that each edge connects nodes from different parts).

This graph, of course, is 2-colorable: you can color all nodes from the first part in color 1 and from the second in color 2. But you don't like that. You don't want this graph to be 2-colorable. You don't even want it to be 3-colorable!

You want to add some edges to this graph so that it stops being 3-colorable (in particular, you can edges between two nodes from the same part). What's the smallest number of edges you have to add?

Input

The first line contains two integers n, m ($2 \le n \le 10^4, 2n-1 \le m \le min(n^2, 2 \cdot 10^5)$) — the size of each part and the number of edges correspondingly.

The i-th of the next m lines contains two integers u_i, v_i ($1 \le u_i \le n$, $n+1 \le v_i \le 2n$), denoting the edge (u_i, v_i).

It's guaranteed that no edge will appear more than once. It's guaranteed that the graph on these edges is connected.

Output

Output a single integer — the smallest number of edges that you have to add to this graph so that it becomes not 3-colorable.

Examples

standard input	standard output

2 4
1 3
1 4
2 3
2 4
	2

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

Note

In the first sample, you can add edges $(1, 2)$ and $(3, 4)$ to the graph. You will get a complete graph on 4 nodes, which is not 3-colorable.

In the second sample, you can't add less than 3 edges to make graph not 3-colorable.

Problem I. Increasing Split Swapper

Time limit: 2 seconds
Memory limit: 256 megabytes

Let's call permutation $(p_1, p_2, \ldots, p_n)$ **splittable** if it's possible to split it into two increasing subsequences so that every element goes to exactly one subsequence. For example, permutation $(3, 1, 5, 2, 4)$ is splittable, as we can split it into two increasing subsequences $(3, 5)$ and $(1, 2, 4)$, while permutation $(3, 2, 1)$ isn't splittable.

You are given a permutation $(p_1, p_2, \ldots, p_n)$, and you must process q queries. In the i-th query, you have to swap elements p_{u_i}, p_{v_i}, and tell whether permutation is **splittable**.

Note that queries are not independent (each next query is applied to the result of applying previous queries, not to the initial permutation).

A sequence a is a subsequence of a sequence b if a can be obtained from b by deletion of several (possibly, zero or all) elements. For examples of what "can be split into two increasing subsequences"means, refer to the notes.

Input

The first line of the input contains two integers n, q $(2 \le n \le 2 \cdot 10^5, 1 \le q \le 2 \cdot 10^5)$ — the length of the permutation and the number of queries correspondingly.

The i-th of the next q lines contains two integers u_i, v_i $(1 \le u_i, v_i \le n,\ u_i \ne v_i)$, denoting that in the i-th query, you have to swap elements p_{u_i}, p_{v_i}.

Output

After each query, output `YES`, if the permutation is **splittable**, and `NO` if it's not.

You can print `YES` and `NO` in any case (e.g. the strings `yEs`, `yes`, `Yes` will be taken as a positive answer).

Example

standard input	standard output
`5 5`	`YES`
`2 1 3 4 5`	`NO`
`4 5`	`YES`
`2 4`	`NO`
`4 3`	`YES`
`1 5`	
`5 4`	

Note

After the first query, permutation becomes $(2, 1, 3, 5, 4)$, which can be split into increasing subsequences $(2, 3, 5)$ and $(1, 4)$.

After the second query, permutation becomes $(2, 5, 3, 1, 4)$, which can't be split into two increasing subsequences.

After the third query, permutation becomes $(2, 5, 1, 3, 4)$, which can be split into two increasing subsequences $(2, 5)$ and $(1, 3, 4)$.

After the fourth query, permutation becomes $(4, 5, 1, 3, 2)$, which can't be split into two increasing subsequences.

After the fifth query, permutation becomes $(4, 5, 1, 2, 3)$, which can be split into two increasing subsequences $(4, 5)$ and $(1, 2, 3)$.

Problem J. Joke

Time limit: 1 second
Memory limit: 256 megabytes

You are given an array $[a_1, a_2, \ldots, a_n]$ of positive integers. You want to find some integer x, for which the following conditions are satisfied:

- x appears in the array a
- There are no i, j (with $1 \le i < j \le n$), for which $|a_i - a_j| = x$

It can be proved that at least one such x always exists under these conditions. If there are multiple such x, you can output any of them.

Input

The first line contains a single integer t $(1 \le t \le 10^4)$ — the number of test cases. The description of test cases follows.

The first line of each test case contains a single integer n $(2 \le n \le 2 \cdot 10^5)$ — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$ — the elements of the array.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output any integer x, for which the conditions from the statement are satisfied. If there are multiple such x, you can output any of them.

Example

standard input	standard output

2
4
2022 2022 2022 2022
5
1 3 5 7 9
	2022
	1

Note

In the first test case, 2022 is the only number present in the array, so we have to output it.

In the second test case, we can output any number among $1, 3, 5, 7, 9$, as the difference of odd integers is always even.

Problem K. Knowledgeable Andrii

Time limit: 1 second
Memory limit: 256 megabytes

Andrii had a connected graph with n vertices. For every two different vertices i and j of this graph, he calculated the length of the shortest path between them — $d_{i,j}$. Unfortunately, then Andrii lost the graph and forgot the numbers $d_{i,j}$. But he remembered the parity of all numbers $d_{i,j}$.

So for every two different vertices i, j Andrii told you $a_{i,j} = d_{i,j} \bmod 2$. Construct an example of a graph that Andrii could have, or determine that such a graph does not exist and Andrii is lying to you.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. Then follows a description of the test cases.

The first line of each test case contains one integer n ($2 \le n \le 500$) — the number of vertices. n binary lines follow.

The i-th of the next n lines contains a binary string s_i of length n. The j-th character of this string is `0` if $a_{i,j} = 0$, and `1` if $a_{i,j} = 1$.

It is guaranteed that $a_{i,i} = 0$ for all $1 \le i \le n$, and $a_{i,j} = a_{j,i}$ for all $1 \le i < j \le n$.

It is guaranteed that the sum of n^2 over all test cases does not exceed 250000.

Output

For each test case, if such a graph does not exist, print `NO`.

Otherwise, print `YES`. On the next line print a single integer m ($n - 1 \le m \le \frac{n(n-1)}{2}$) — the number of edges. In the i-th of the next m lines print two numbers u_i, v_i ($1 \le u_i, v_i \le n, u_i \ne v_i$), denoting the edge between the vertices u_i and v_i.

All edges must be pairwise distinct. The graph must be connected.

You can print `YES` and `NO` in any case (e.g. the strings `yEs`, `yes`, `Yes` will be taken as a positive answer).

Example

standard input	standard output
3	YES
3	3
011	1 2
101	1 3
110	2 3
4	NO
0100	YES
1000	4
0001	1 2
0010	2 3
5	3 4
01010	4 5
10101	
01010	
10101	
01010	

Note

In the first test case, such a graph on three vertices exists — you can just take a triangle. All pairwise distances are equal to 1 and hence odd.

It can be shown that in the second test case, such a graph does not exist.

In the third test case, we have a chain with edges $(1,2),(2,3),(3,4),(4,5)$. In it, the distance between vertices i, j is odd if and only if i and j have different parity.

Problem L. Least Annoying Constructive Problem

Time limit: 1 second
Memory limit: 256 megabytes

Consider a complete graph on n nodes. You have to arrange all its $\frac{n(n-1)}{2}$ edges on the circle in such a way that every $n-1$ consecutive edges on this circle form a tree.

It can be proved that such an arrangement is possible for every n. If there are many such arrangements, you can find any of them.

As a reminder, a tree on n nodes is a connected graph with $n-1$ edges.

Input

The only line of the input contains a single integer n ($3 \le n \le 500$).

Output

Output $\frac{n(n-1)}{2}$ lines. The i-th line should contain two integers u_i, v_i ($1 \le u_i < v_i \le n$). All pairs (u_i, v_i) have to be distinct, and for every i from 1 to $\frac{n(n-1)}{2}$, edges $(u_i, v_i), (u_{i+1}, v_{i+1}), \ldots, (u_{i+n-2}, v_{i+n-2})$ have to form a tree.

Here $u_{\frac{n(n-1)}{2}+i} = u_i, v_{\frac{n(n-1)}{2}+i} = v_i$ for every i.

Examples

standard input	standard output

3	1 2
	2 3
	1 3

4	1 2
	3 4
	2 3
	1 4
	1 3
	2 4