SINGLEOP1

We want x to be maximised after the xor.

The comparison is from high to low, so we should make the first occurrence of 0 a 1.

			1	1	1	0	1	0	0	1	0	1	1			
						1	1	1	0	1	0	0	1	0	1	1
BIT-XOR			1	1	1	1	0	1	0	0	0	1	0			

If 0 does not exist in the string, output n.

SINGLEOP2

Same as the previous question, just with minimal hope.

The second occurrence of 1 will be converted to 0, which will ensure that the answer is minimal.

			1	1	0	1	1	0	0	1	0	1	1	
				1	1	0	1	1	0	0	1	0	1	1
BIT-XOR			1	0	1	1	0	1	0	1	1	1	0	

BINARYSUB

Consider the strings `["ab", "ba"]`, which can be replaced by `[["0110", "010"], ["1001", "101"]]`.

For a string, if the `ab` and `ba` that are packed and merged are different, the resulting result will be different.

a	b	b	a	b	b	a
01	10	101		10	10	01

a	b	b	a	b	b	a
010		10	010		101	

So just dp , `dp[i][j]` means: is the current calculation up to position i and has the previous one been packed by the previous one.

ARRCONS

For each bit, calculate the answer separately and add it up at the end.

If we want the i-th bit to be included in the final sum, then the bits of the corresponding array must all be 1.

	6		1	1	0
	2		0	1	0
	3		0	1	1
BIT-AND			0	1	0

So for the i-th bit, if we know that there are x possible numbers that satisfy this bit as 1, we simply calculate $2^i \times x^n$ as the contribution of this bit to the answer.

The formula for calculating x is obvious.

TWOZERO

The fraction of some balls is equal to the sum of the fractions of those balls minus the number of balls, which is difficult to calculate.

So just subtract one from the fraction of all the balls, then that the fraction of some balls is simply the sum of the fractions of these balls.

The problem then becomes one of having n 1 point balls and m -1 point balls, such that the sum of the weights is divisible by 3.

You can find the answer on oeis.

DRAG

The problem is divided into many layers.

1

Obviously, each line segment is independent of each other, so if the sum of the beauty of all the line segments is greatest, then the beauty of all the segments is greatest separately.

Next calculate $W(B)$

According to the above, the number of solutions with the greatest sum of beauty is equal to the product of the number of solutions with the greatest beauty in each line segment.

The beauty of a line segment is $\gcd(x,y)$ if, for that segment, it occurs by my choice of (x,y).

So for a b_i the maximum number of choices of beauty is b_i's minimum prime factor -1.

2

Next calculate S

S is equivalent, for a set, to querying the sum of the products of all the subelements in all its subsets.

It is equal to $\prod\limits_{s_i}(s_i+1)-1$.

Since it is different, this -1 can be left out. Also, the $+1$ in the above is offset by the -1.

3

The problem then becomes that given n primes, there are q queries, each given an interval, and the product of different subsequences in this interval is asked.

For this problem, we can use Mo to solve it.