Solution

Outline Originally, the problem wants to find the parity of distinct sets of disjoint cycles on the given directed graph that can be arranged in such a way that there be a path between any two consecutive cycles of them. The problem subtasks give us a good vision for solving the original problem. So we try to solve the problem subtasks to achieve the main solution:

1 Problem on Cactus graphs

First of all, we have to find the answer for all of the SCCs (strongly connected components) on the graph. At the first, we find the SCCs from the `Topological sort` of the graph, and then we do a simple `DFS` on each of these components and try to count the parity of valid sets of cycles in each of them with `DP`:

$dp_{u,0}$ = `The number of distinct sets of cycles that are in the subtree of vertex` u.

$dp_{u,1}$ = `The number of distinct sets of cycles that are in the subtree of vertex` u`, except exactly one of them that has a back-edge from subtree of vertex` u `to a vertex out of it.`

These states can be easily filled from each other in $O(n+m)$ while `DFS`, so we leave this part to the reader. Now we have the answer of the problem for each component in $dp_{root,0}$ (why?) and we should try to find the original answer to the problem from them. To do this, we define another variable and try to calculate it with `DP` again:

$answer_i$ = `The parity of valid sets of cycles such that the first picked cycle is in the` i`th component and the last one could be anywhere (The initial value of` $answer_i$`:` $dp_{root_i,0}$`).`

We do the following algorithm for each component from the last one to the first one and find the real value of $answer_i$ in $O(n \times n)$:

`choose the next component that has a picked cycle and is reachable from the current component and then do the following operation on it:`

$$answer_i \mathrel{+}= (dp_{root_i,0} \times answer_j)$$

Don't forget that we just need the parity of this values, so we do all the operations modulo 2 (+ operator is like $\oplus$ and $\times$ operator is like $\wedge$, so we can behave to all of the variables like a boolean).

2 Problem on strongly connected graphs

In this section, we know that any set of disjoint cycles are valid. So we have to find the parity of distinct sets of disjoint cycles on the graph.

A cycle $\langle c_1, c_2, ..., c_k \rangle$ represents edges $\langle c_1c_2, c_2c_3, ..., c_kc_1 \rangle$. Consider the $n \times n$ adjacency matrix M of the graph and for each valid set, select the cells of the edges that are in cycles of the set and manually select cell (u, u) for each u that is not in any cycle of the set. The selected cells form a diagonal in matrix M. So we can correspond any valid set of cycles in the graph to a diagonal in the adjacency matrix. We can select a set if and only if the multiplication of values of the corresponding diagonal cells to it be equal to 1, otherwise there is an edge that the original graph doesn't have. So the parity of the valid sets is equal to the parity of determinant of the matrix M (why?). In the general case, we can find the determinant of an $n \times n$ matrix in $O(n^3)$. But in this case, we should try to do it better...

The exact value of the determinant is not important for us, because we just need to calculate it modulo 2. So we can correspond all of the operations to the basic logical operations again (like the previous part) and now we can implement `Gaussian elimination` algorithm to find the parity of the determinant with bitset! Now we can calculate the answer in $O(\frac{n^3}{32})$ which is fast enough to pass the testcases [or even $O(\frac{n^3}{64})$ if we implement the bitset with blocks of size `64 (long long)` instead of size `32 (int)`].

3 Problem on any kind of graphs

In this part, we should just calculate the answer for each connected component with the **Section 2** method and then try to find the answer with the method that presents in the last part of **Section 1**.