Div.2 B

First of all, instead of cycles, imagine we have bamboos (paths). A valid move in the game is now taking a path and deleting an edge from it (to form two new paths). So, every player in his move can delete an edge in the graph (with components equal to paths). So, no matter how they play, winner is always determined by the parity of number of edges (because it decreases by 1 each time). Second player wins if and only if the number of edges is even. At first it's even (0). In a query that adds a cycle (bamboo) with an odd number of vertices, parity and so winner won't change. When a bamboo with even number of vertices (and so odd number of edges) is added, parity and so the winner will change.

Time Complexity: $\mathcal{O}(n)$

B

Reduction to TSP is easy. We need the shortest Hamiltonian path from s to e. Consider the optimal answer. Its graph is a directed path. Consider the induced graph on first i chairs. In this subgraph, there are some components. Each components forms a directed path. Among these paths, there are 3 types of paths:

1. In the future (in chairs in right side of i), we can add vertex to both its beginning and its end.
2. In the future (in chairs in right side of i), we can add vertex to its beginning but not its end (because its end is vertex e).
3. In the future (in chairs in right side of i), we cannot add vertex to its beginning (because its beginning is vertex s) but we can add to its end.

There are at most 1 paths of types 2 and 3 (note that a path with beginning s and ending e can only exist when all chairs are in the subgraph. i.e. induced subgraph on all vertices).

This gives us a dp approach: $dp[i][j][k][l]$ is the answer for when in induced subgraph on the first i vertices there are j components of type 1, k of type 2 and l of type 3. Please note that it contains some informations more than just the answer. For example we count $d[i]$ or $-x[i]$ when we add i to the dp, not j (in the problem statement, when $i < j$). Updating it requires considering all four ways of incoming and outgoing edges to the last vertex i (4 ways, because each edge has 2 ways, left or right). You may think its code will be hard, but definitely easier than code of B.

Time Complexity: $\mathcal{O}(n^2)$