Question 1

1. **Algorithm**
   Node with ID 0 changes all the values of its edges to 1 and stop. All other nodes when see the first time that some of XORs of edges are changing, change the value for the edge with the minimal local number (we will call it "the edge from which propagation came", supposing that all nodes can enumerate their incident edges) and also change value for all other edges not changed from the other side before, and then stop.
   Proof
   At the beginning all edges have XOR equals 1 and therefore they all are included to the tree. When node changes value of one of its incident edges, then this edge remains only if the corresponding neighbor changes its value too. It happens only when the propagation comes to the node the first time and only with one such edge (therefore there wouldn't be loops in the final graph, and it would be a tree). This will be a spanning tree, because in each round propagation comes to at least one node until all nodes were affected. Therefore, it will need less than n rounds and it's $O(n)$ (in case of line graph with 0 at one of the ends it will take exactly n rounds).

2. **Algorithm**
   All the same as in previous, but as soon as propagation comes to the any node, it also changes edge from which it came. And when node observes that edge which it already changed, changes another time, it changes it and the edge, from which propagation came to it. Node with ID 0 will receive these "backpropagation waves" starting from the second round continuously once every two rounds (need one for propagation to come to the new node, and one for the previous node to backpropagate it to the earlier nodes) until all nodes are achieved by propagation. So 0–node can say that everything stopped when it doesn't receive any backpropagation signal for 2 rounds.
   Proof
   Time is now no more than $2n$, as after finishing in not more than n rounds backpropagation should come to the 0–node, and it will take not more than n rounds too (as it propagates one node/round)

3. **Algorithm**
   The same as in Algorithm from 1. In the first round only zero-node does something. Then every other node each time it's active does the same as in 1.
   Proof
   Due to propagation through not used before edges, all nodes will be affected and included to the final graph. Constructed graph will be still a tree, because any time maximum one propagation–coming edge per node–round (i.e. one round is n node–rounds) is added to the final graph, so there will not be loops (with loop there should a node in it which added

two propagation–incoming edges, which is impossible with our algorithm).

Best case
If we have line tree graph (path) with numeration 0-n from left to right and sequence of nodes is from 0 to $n-1$ too, then this won't be different from usual algorithm (propagation order is the same as sequential ordering) and will be 1 sequential round (can't be less as we need to process all n nodes sequentially). So it's 1 i.e. $O(1)$ in sequential rounds (or n i.e. $O(n)$ in node–rounds).
Worst case
Effectively during one sequential round at least one node will proceed with the algorithm (otherwise if not, algorithm will stop forever, as nothing in the graph changes). Therefore if we achieve that it's only one node is added, it would maximally be n sequential rounds (which can't be less) or equally n^2 node–rounds.

Let's construct such a graph.
We can take the same line tree graph (path) as before, but now sequentially go from $n-2$ to 0 and then to $n-1$, therefore at the ith round (numeration from 1) only node $i-1$ does something (as first we need activated 0, then only 1 is connected to it and can propagate, then 2 and etc.), therefore till the end we'll need n sequential rounds and n^2 node–rounds.