Question: https://leetcode.com/problems/critical-connections-in-a-network/

Now this is a problem where I got to learn a lot.

While going through the solutions and youtube videos after being unable to solve it.

I came across the famous Tarjan’s algorithm and learned about it’s different use cases so what I feel is if you are reading this post, give yourself some time to go through the algorithm and it’s different use cases, that will be a asset to not only your DSA preparation but also an approach which you can use in real life projects.

Code:

class Solution {

int[] ids; // denotes ids - we start from a random node and assign an ever increasing integer as id as we do a dfs through this node

int[] low; // denotes low link values - lowest id value reachable from this node

boolean[] visited;

private static int i=-1;

public List<List<Integer>> criticalConnections(int n, List<List<Integer>> connections) {

low = new int[n];

ids = new int[n];

visited = new boolean[n];

List<List<Integer>> bridges = new ArrayList<>();

List<Integer>[] graph = new ArrayList[n];

for (int i=0; i<n; i++) {

graph[i] = new ArrayList<>();

}

for (List<Integer> connection: connections) {

int from = connection.get(0);

int to = connection.get(1);

graph[from].add(to);

graph[to].add(from);

}

for (int i=0;i<n;i++) {

if (!visited[i]) {

dfs(graph, i, -1, bridges);

}

}

return bridges;

}

void dfs(List<Integer>[] graph, int at, int parent, List<List<Integer>> bridges) {

visited[at] = true;

low[at] = ids[at] = ++i;

for (Integer to: graph[at]) {

if (to == parent) continue;

if (!visited[to]) {

dfs(graph, to, at, bridges);

low[at] = Math.min(low[at], low[to]);

if (ids[at] < low[to]) {

List<Integer> bridge = new ArrayList<>(List.of(at, to));

bridges.add(bridge);

}

} else

low[at] = Math.min(low[at], ids[to]);

}

}

}

Github Link :<https://lnkd.in/ecwtJeaz>