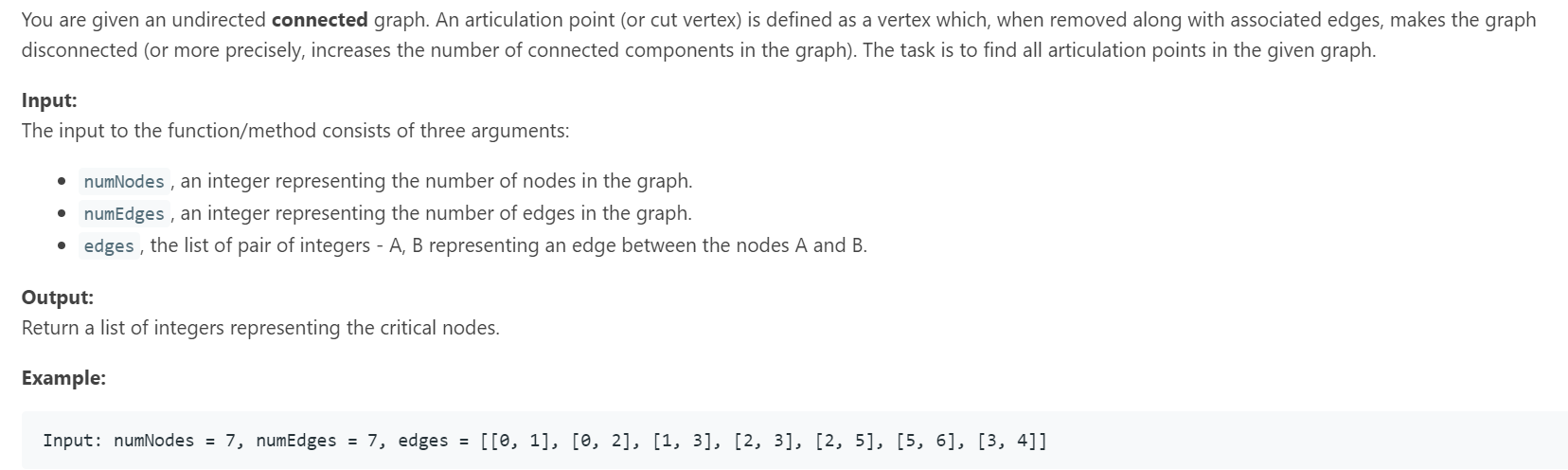
Amazon | OA 2019 | Critical Routers

<https://leetcode.com/discuss/interview-question/436073/>



static int time = 0;

public static void main(String[] args) {

int numRouters1 = 7;

int numLinks1 = 7;

int[][] links1 = {{0, 1}, {0, 2}, {1, 3}, {2, 3}, {2, 5}, {5, 6}, {3, 4}};

System.out.println(getCriticalNodes(links1, numLinks1, numRouters1));

}

private static List<Integer> getCriticalNodes(int[][] links, int numLinks, int numRouters) {

time = 0;

Map<Integer, Set<Integer>> map = new HashMap<>();

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

map.put(i, new HashSet<>());

}

for(int[] link : links) {

map.get(link[0]).add(link[1]);

map.get(link[1]).add(link[0]);

}

Set<Integer> set = new HashSet<>();

int[] low = new int[numRouters];

int[] ids = new int[numRouters];

int parent[] = new int[numRouters];

Arrays.fill(ids, -1);

Arrays.fill(parent, -1);

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

if(ids[i] == -1)

dfs(map, low, ids, parent, i, set);

}

return new ArrayList<>(set);

}

private static void dfs(Map<Integer, Set<Integer>> map, int[] low, int[] ids, int[] parent, int cur, Set<Integer> res) {

int children = 0;

ids[cur] = low[cur]= ++time;

for(int nei : map.get(cur)) {

if(ids[nei] == -1) {

children++;

parent[nei] = cur;

dfs(map, low, ids, parent,nei, res);

low[cur] = Math.min(low[cur], low[nei]);

if((parent[cur] == -1 && children > 1) || (parent[cur] != -1 && low[nei] >= ids[cur]))

res.add(cur);

}

else if(nei != parent[cur])

low[cur] = Math.min(low[cur], ids[nei]);

}

}