Question: https://leetcode.com/problems/restore-the-array-from-adjacent-pairs/

So the idea here is to create a adjacency list(mem) using HashMap.

Then created a vidited hash Table(vis) using Hashmap, where value 1 means visited and 0 means not visited.

Now since its an array the starting node or vertex will have only one edge, so in adjacency list search for a element who has 1 adjacent node, now initialize that as starting element of array.

Then perform BFS.

Code:

class Solution {

public int[] restoreArray(int[][] adjacentPairs) {

Map<Integer, List<Integer>> mem = new HashMap<>();

List<Integer> l1,l2;

for(int pair[] : adjacentPairs){

l1=mem.getOrDefault(pair[0], new ArrayList<Integer>());

l2=mem.getOrDefault(pair[1], new ArrayList<Integer>());

l1.add(pair[1]);

l2.add(pair[0]);

mem.put(pair[0], l1);

mem.put(pair[1], l2);

}

int res[] = new int[adjacentPairs.length+1];

Map<Integer, Integer> vis = new HashMap<>();

for(Map.Entry<Integer, List<Integer>> el:mem.entrySet()){

vis.put(el.getKey(),0);

if(el.getValue().size()==1 && res[0]==0){

res[0]=el.getKey();

vis.put(el.getKey(),1);

}

}

bfs(mem, vis, res);

return res;

}

public void bfs(Map<Integer, List<Integer>> mem, Map<Integer, Integer> vis, int res[]){

Queue<Integer> q = new LinkedList<>();

int i=0;

q.add(res[0]);

while(!q.isEmpty()&&i<vis.size()){

int node=q.remove();

for(int adjnode:mem.get(node)){

if(vis.get(adjnode)!=1){

q.add(adjnode);

}

}

res[i++]=node;

vis.put(node,1);

}

return;

}

}

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