Thursday, December 16, 2021 2:17 PM

310. Minimum Height Trees

Medium

4277173Add to ListShare

QUESTION

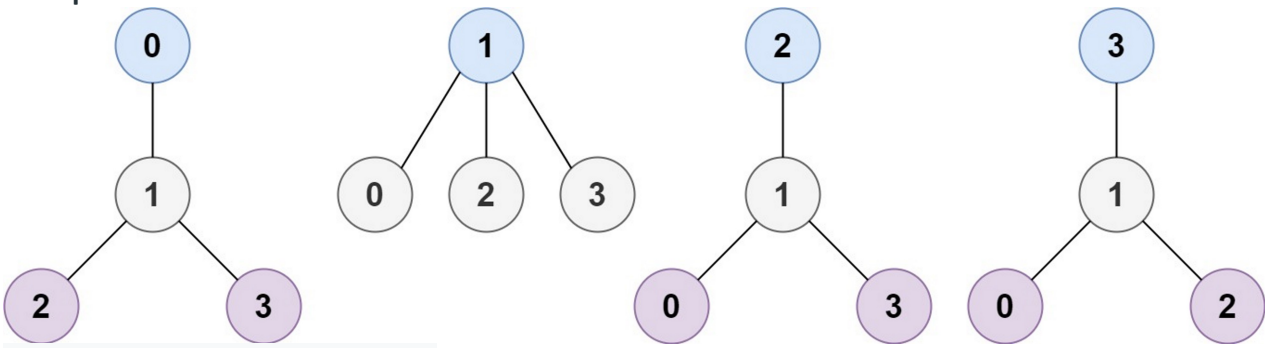
A tree is an undirected graph in which any two vertices are connected by *exactly* one path. In other words, any connected graph without simple cycles is a tree.

Given a tree **o**f `n` nodes labelled from `0` to **`n - 1,`** and an array of **`n - 1`** `edges` where **`edges[i] = [ai, bi]`** indicates that there is an undirected edge between the two nodes **`ai`** and **`bi`** in the tree, you can choose any node of the tree as the root. When you select a node `x` as the root, the result tree has height **`h`**. Among all possible rooted trees, those with minimum height (i.e. **`min(h)`**) are called **minimum height trees** (MHTs).

Return *a list of all **MHTs'** root labels*. You can return the answer in **any order**.

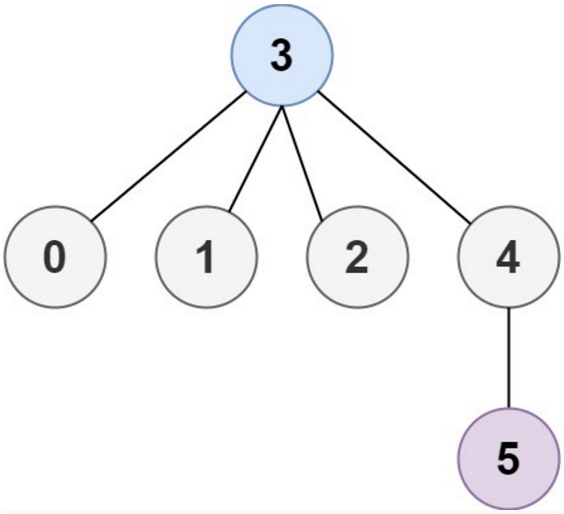
The **height** of a rooted tree is the number of edges on the longest downward path between the root and a leaf.

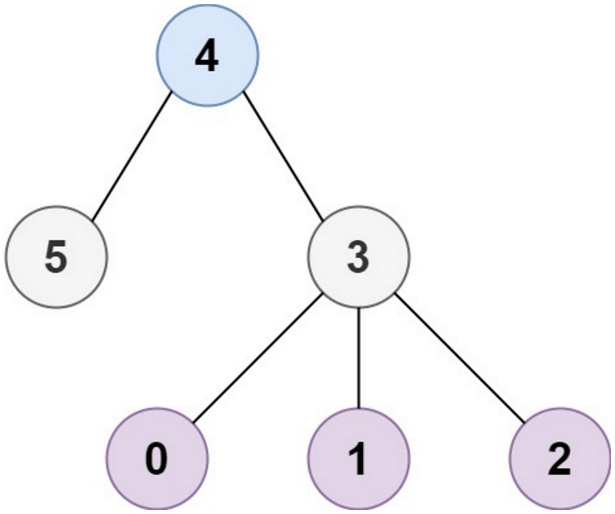
Example 1:



Input: n = 4, edges = [[1,0],[1,2],[1,3]]
Output: [1]
Explanation: As shown, the height of the tree is 1 when the root is the node with label 1 which is the only MHT.

Example 2:





Input: n = 6, edges = [[3,0],[3,1],[3,2],[3,4],[5,4]]
Output: [3,4]
Example 3:
Input: n = 1, edges = []
Output: [0]
Example 4:
Input: n = 2, edges = [[0,1]]
Output: [0,1]

Constraints:

- 1 <= n <= 2 * 104
- edges.length == n - 1
- 0 <= ai, bi < n
- ai != bi
- All the pairs (ai, bi) are distinct.
- The given input is **guaranteed** to be a tree and there will be **no repeated** edges.

From <https://leetcode.com/problems/minimum-height-trees/>

```
packagedsaProblems;
importjava.util.*;


publicclassMinimumHeightTree{
publicstatic List<Integer>findMinHeightTrees(intn,int[][]edges){
List<Integer>list=newArrayList<>();
ArrayList<ArrayList<Integer>>adjList=newArrayList<>();
for(inti=0;i<=n;i++){
adjList.add(newArrayList<>());


}
for(inti=0;i<n;i++){
intu=edges[i][0];
intv=edges[i][1];
adjList.get(u).add(v);
```

```
adjList.get(v).add(u);


}

intm=adjList.size();
intindegree[]=newint[m];

for(inti=0;i<m;i++){
indegree[i]=adjList.get(i).size();

}

Queue<Integer>queue=newLinkedList<>();

for(inti=0;i<m;i++){
if(indegree[i]==1)
queue.add(i);

}

while(!queue.isEmpty()){
ints=queue.size();
list.clear();

for(inti=0;i<s;i++){
intpop=queue.peek();
queue.poll();
list.add(pop);



ArrayList<Integer>smallList=adjList.get(pop);

for(intele:smallList)
{
indegree[ele]--;
if(indegree[ele]==1)
queue.add(ele);

}


}

}
```

```
System.out.println("List:"+list);


if(n==1)list.add(0);
returnlist;


}

publicstaticvoidmain(String[]args){
intarr[][]={{3,0},{3,1},{3,2},{3,4},{5,4}};

System.out.println("output:"+findMinHeightTrees(arr.length,arr));

}
}
```