310. Minimum Height Trees

link

最基本的BFS暴力破解 但是超时了.

```
public List<Integer> findMinHeightTrees(int n, int[][] edges) {
    List<Integer> res = new ArrayList<>();
    if(n == 1){
        res.add(0);
        return res;
    }
    Set<Integer>[] sets = new HashSet[n];
    for(int[] edge : edges){
        if(sets[edge[0]] == null)
            sets[edge[0]] = new HashSet<Integer>();
        if(sets[edge[1]] == null)
            sets[edge[1]] = new HashSet<Integer>();
        sets[edge[0]].add(edge[1]);
        sets[edge[1]].add(edge[0]);
    }
    int min = Integer.MAX_VALUE;
    int[] heights = new int[n];
    for(int i = 0; i < n; ++i){
        heights[i] = getHeight(i, sets);
        min = Math.min(min, heights[i]);
    }
    for(int i = 0; i < n; ++i){
        if(heights[i] == min)
            res.add(i);
    }
    return res;
}
public int getHeight(int v, Set<Integer>[] sets){
    Queue<Integer> queue = new LinkedList<>();
    int len = sets.length;
    int[] visited = new int[len];
    int[] level = new int[len];
    visited[v] = 1;
    level[v] = 1;
    queue.offer(v);
    int max = 0;
    while(!queue.isEmpty()){
        int tmp = queue.poll();
        for(int i : sets[tmp]){
            if(visited[i] == 0){
                visited[i] = 1;
                queue.offer(i);
                if(level[i] == 0){
                    level[i] = level[tmp] + 1;
                    max = Math.max(max, level[i]);
                }
            }
        }
```

```
        }
        return max;
    }
```

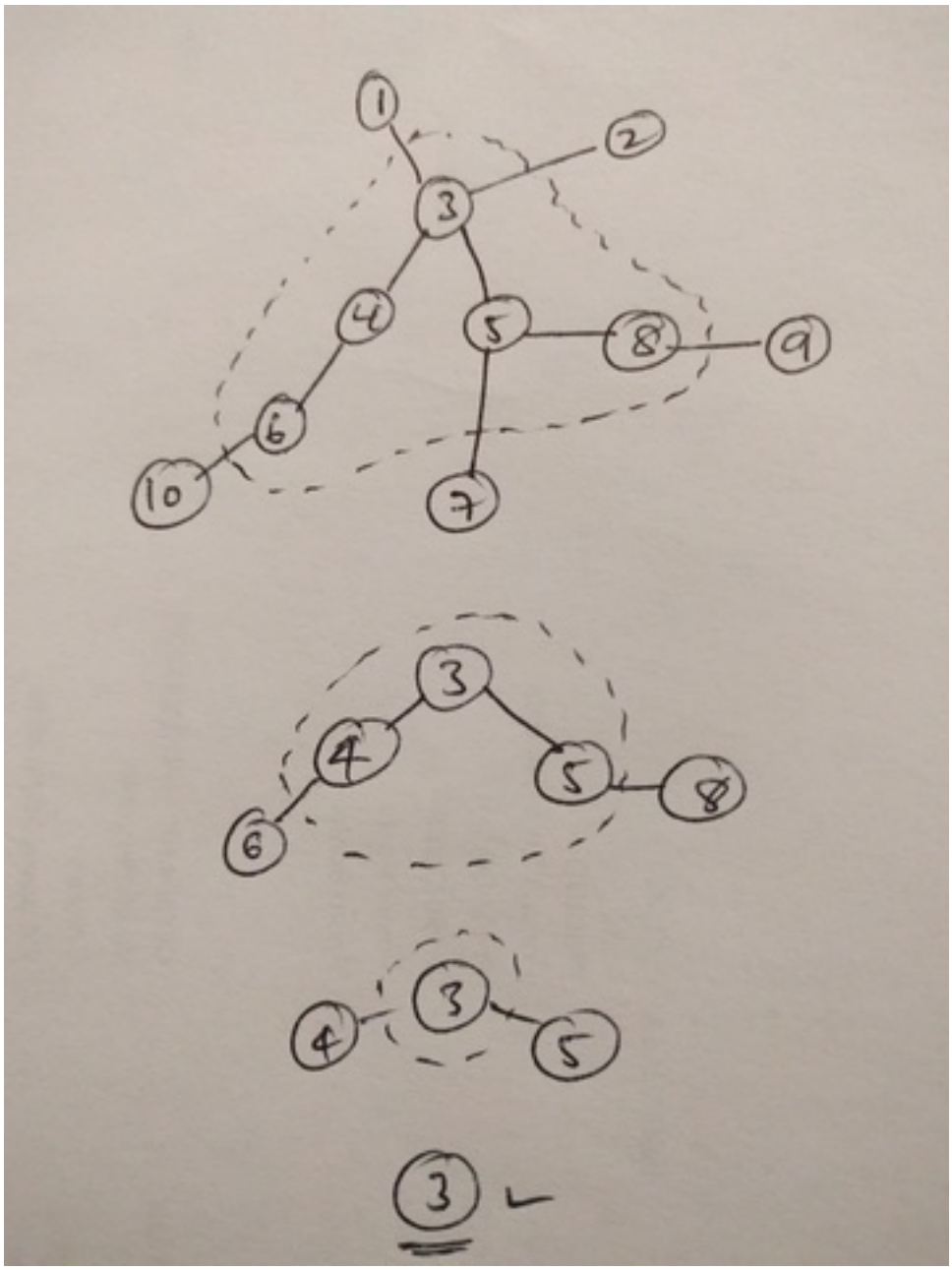
拓扑排序

link

找一个入度为0的节点(没有父节点), 然后删除这个节点和这个节点为起点的边的顺序.

算法成立的原因, 每次我们relax(v -> w)此时v已经relax过我们只更新了w, 因为v是已经relax的, 并且因为拓扑排序, 没有节点指向v, 所以v是最小的.

每一次都删除图的最外层的叶子结点，直到图有两个或一个或着两个节点



link

```
public List<Integer> findMinHeightTrees(int n, int[][] edges) {
    if (n == 1) return Collections.singletonList(0);

    List<Set<Integer>> adj = new ArrayList<>(n);
    for (int i = 0; i < n; ++i) adj.add(new HashSet<>());
    for (int[] edge : edges) {
        adj.get(edge[0]).add(edge[1]);
        adj.get(edge[1]).add(edge[0]);
```

```
        }

        List<Integer> leaves = new ArrayList<>();
        for (int i = 0; i < n; ++i)
            if (adj.get(i).size() == 1) leaves.add(i);

        while (n > 2) {
            n -= leaves.size();
            List<Integer> newLeaves = new ArrayList<>();
            for (int i : leaves) {
                int j = adj.get(i).iterator().next();
                adj.get(j).remove(i);
                if (adj.get(j).size() == 1) newLeaves.add(j);
            }
            leaves = newLeaves;
        }
        return leaves;
    }
```

这个题应该还可以用bfs和topological sort入度做