LeetCode 133

Description

Clone an undirected graph. Each node in the graph contains a label and a list of its neighbors.

OJ's undirected graph serialization:

Nodes are labeled uniquely.

We use # as a separator for each node, and , as a separator for node label and each neighbor of the node.

As an example, consider the serialized graph {0,1,2#1,2#2,2}.

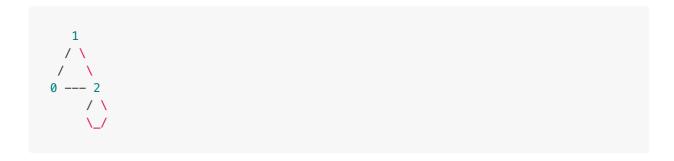
The graph has a total of three nodes, and therefore contains three parts as separated by #.

First node is labeled as 0. Connect node 0 to both nodes 1 and 2.

Second node is labeled as 1. Connect node 1 to node 2.

Third node is labeled as 2. Connect node 2 to node 2 (itself), thus forming a self-cycle.

Visually, the graph looks like the following:



Solution

```
public UndirectedGraphNode cloneGraph(UndirectedGraphNode root) {
   if (root == null) return null;

   // use a queue to help BFS
   Queue<UndirectedGraphNode> queue = new LinkedList<UndirectedGraphNode>();
   queue.add(root);

   // use a map to save cloned nodes
   Map<UndirectedGraphNode, UndirectedGraphNode> map = new HashMap<UndirectedGraphNo
   // clone the root
   map.put(root, new UndirectedGraphNode(root.label));

while (!queue.isEmpty()) {</pre>
```

```
UndirectedGraphNode node = queue.poll();

// handle the neighbors
for (UndirectedGraphNode neighbor : node.neighbors) {
    if (!map.containsKey(neighbor)) {
        // clone the neighbor
        map.put(neighbor, new UndirectedGraphNode(neighbor.label));
        // add it to the next level
        queue.add(neighbor);
    }

    // copy the neighbor
    map.get(node).neighbors.add(map.get(neighbor));
}

return map.get(root);
}
```