**[Clone Graph](https://leetcode.com/problems/clone-graph/)**

DFS:

class Solution {

public Node cloneGraph(Node node) {

if(node == null) {

return null;

}

HashMap<Node , Node> map = new HashMap<>();

return helper(node , map);

}

public Node helper(Node node , HashMap<Node , Node> map) {

if(node == null) {

return null;

}

if(map.containsKey(node)) {

return map.get(node);

}

Node newNode = new Node(node.val , new ArrayList<>());

map.put(node , newNode);

for(Node child: node.neighbors) {

newNode.neighbors.add(helper(child , map));

}

return newNode;

}

}

Time complexity : O(|N| + |E|). Number of nodes and edges in graph

Space Complexity : O(|N| + |E|). Number of nodes and edges in graph

BFS:

class Solution {

public Node cloneGraph(Node node) {

if(node == null) {

return null;

}

HashMap<Integer , Node> map = new HashMap<>();

map.put(node.val , new Node(node.val , new ArrayList<>()));

Queue<Node> queue = new LinkedList<>();

queue.add(node);

while(!queue.isEmpty()) {

Node temp = queue.poll();

if(temp.neighbors == null || temp.neighbors.size() == 0) {

continue;

}

for(Node n : temp.neighbors) {

if(!map.containsKey(n.val)) {

map.put(n.val , new Node(n.val, new ArrayList<>()));

queue.add(n);

}

map.get(temp.val).neighbors.add(map.get(n.val));

}

}

return map.get(node.val);

}

}

Time complexity : O(|N| + |E|). Number of nodes and edges in graph

Space Complexity : O(|N| + |E|). Number of nodes and edges in graph