133. Clone Graph

作者：qianrong wu

思路

遍历一个图有两种方式：bfs和dfs。所以clone一个图也可以采用这两种方法。不管使用dfs还是bfs都需要一个哈希表map来存储原图中的节点和新图中的节点的一一映射。map的作用在于替代bfs和dfs中的visit数组，一旦map中出现了映射关系，就说明已经复制完成，也就是已经访问过了。

dfs解法

```
# Definition for a undirected graph node
# class UndirectedGraphNode:
#     def __init__(self, x):
#         self.label = x
#         self.neighbors = []

class Solution:
    # @param node, a undirected graph node
    # @return a undirected graph node
    # @BFS
    def cloneGraph(self, node):
        def dfs(input, map):
            if input in map:
                return map[input]
            output = UndirectedGraphNode(input.label)
            map[input] = output
            for neighbor in input.neighbors:
                output.neighbors.append(dfs(neighbor, map))
            return output
        if node == None: return None
        return dfs(node, {})
```

bfs解法

```
# Definition for a undirected graph node
# class UndirectedGraphNode:
#     def __init__(self, x):
#         self.label = x
#         self.neighbors = []

class Solution:
    # @param node, a undirected graph node
    # @return a undirected graph node
    # @BFS
    def cloneGraph(self, node):
        if node == None: return None
        queue = []; map = {}
        newhead = UndirectedGraphNode(node.label)
        queue.append(node)
        map[node] = newhead
        while queue:
            curr = queue.pop()
            for neighbor in curr.neighbors:
                if neighbor not in map:
                    copy = UndirectedGraphNode(neighbor.label)
                    map[curr].neighbors.append(copy)
                    map[neighbor] = copy
                    queue.append(neighbor)
                else:
                    # turn directed graph to undirected graph
                    map[curr].neighbors.append(map[neighbor])
        return newhead
```

总结

LeetCode中关于图的题很少，有向图的仅此一道，还有一道关于无向图的题是 Clone Graph 无向图的复制。图这种数据结构相比于树，链表要更为复杂一些，尤其是有向图，很麻烦。