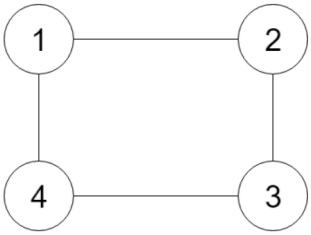
LC 133. Clone Graph

Question

Given a reference of a node in a **connected** undirected graph, return a **deep copy** (clone) of the graph. Each node in the graph contains a val (`int`) and a list (`List[Node]`) of its neighbors.

Example:




```
Input:
{"$id":"1","neighbors":[{"$id":"2","neighbors":[{"$ref":"1"},{"$id":"3","neighbors":[{"$ref":"2"},{"$id":"4

Explanation:
Node 1's value is 1, and it has two neighbors: Node 2 and 4.
Node 2's value is 2, and it has two neighbors: Node 1 and 3.
Node 3's value is 3, and it has two neighbors: Node 2 and 4.
Node 4's value is 4, and it has two neighbors: Node 1 and 3.
```

Solution

```python
# 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
    def cloneGraph(self, node):
        #Solution 2
        #bfs代码:
        if node == None: return None
        queue = []
        dic = {}
        newhead = UndirectedGraphNode(node.label)
        queue.append(node)
```

```
        dic[node] = newhead
        while queue:
            curr = queue.pop()
            for neighbor in curr.neighbors:
                if neighbor not in dic:
                    copy = UndirectedGraphNode(neighbor.label)
                    dic[curr].neighbors.append(copy)
                    dic[neighbor] = copy
                    queue.append(neighbor)
                else:
                    # turn directed graph to undirected graph
                    dic[curr].neighbors.append(dic[neighbor])
        return newhead


        #Solution
        #题意：实现对一个图的深拷贝。
        #解题思路：由于遍历一个图有两种方式：bfs和dfs。
        #所以深拷贝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代码：
        def dfs(i, dic):
            if i in dic:
                return dic[i]
            output = UndirectedGraphNode(i.label)
            dic[i] = output
            for neighbor in i.neighbors:
                output.neighbors.append(dfs(neighbor, dic))
            return output
        if node == None: return None
        return dfs(node, {})
```