#include <unordered\_map>

#include <queue>

#include <vector>

Class Node {

Public:

Int val;

Std::vector<Node\*> neighbors;

Node() : val(0), neighbors(std::vector<Node\*>()) {}

Node(int \_val) : val(\_val), neighbors(std::vector<Node\*>()) {}

Node(int \_val, std::vector<Node\*> \_neighbors) : val(\_val), neighbors(\_neighbors) {}

};

Class Solution {

Public:

Node\* cloneGraph(Node\* node) {

If (!node) return nullptr;

Std::unordered\_map<Node\*, Node\*>queue;

Std::queue<Node\*> queue;

Queue.push(node);

Clones[node] = new Node(node->val);

While (!queue.empty()) {

Node\* current = queue.front();

Queue.pop();

For (Node\* neighbor : current->neighbors) {

If (clones.find(neighbor) == clones.end()) {

Clones[neighbor] = new Node(neighbor->val);

Queue.push(neighbor);

}

clone’s neighbors

Clones[current]->neighbors.push\_back(clones[neighbor]);

}

}

Return clones[node];

}

};