create same copy + traverse -

08 April 2023 11:21

```
/*
// Definition for a Node.
class Node {
public:
int val;
vector<Node*> neighbors;
Node() {
val = 0;
neighbors = vector<Node*>();
}
Node(int _val) {
val = _val;
neighbors = vector<Node*>();
}
Node(int _val, vector<Node*> _neighbors) {
val = _val;
neighbors = _neighbors;
}
};
*/


class Solution {
public:


void traverseG(Node* node){
vector<Node*> tbt,t,td;
tbt.push_back(node);
Node* tmp;
while(tbt.size()>0){
tmp = tbt[0];


tbt.erase(tbt.begin()+0);
if(find(t.begin(),t.end(),tmp)!=t.end()){
continue;
}
t.push_back(tmp);
cout<<tmp->val<<" - ";
for(auto it:tmp->neighbors){
cout<<it->val<<",";
tbt.push_back(it);
}
cout<<endl;
}


}
bool checkNode(vector<Node*>* v,int val){
for(auto it:*v){
if(it->val == val){
return true;
}
}
return false;
}
Node* getNode(vector<Node*> *v,int val){
Node* it;
for(auto it:*v){
if(it->val == val){
return it;
}
}
return it;
}
Node* cloneGraph(Node* node) {
if(node == NULL)
return node;
vector<Node*> created,trav,tbtrav;
Node *tmp,*n, *op=new Node(node->val);
created.push_back(op);


tbtrav.push_back(node);


while(tbtrav.size()>0){
// cout<<"tbtrav size - "<<tbtrav.size()<<endl;
tmp = tbtrav[0];
tbtrav.erase(tbtrav.begin()+0);
if(find(trav.begin(),trav.end(),tmp)!=trav.end()){
continue;
}
trav.push_back(tmp);
if(!checkNode(&created,tmp->val)){
created.push_back(new Node(tmp->val));
}
n = getNode(&created,tmp->val);
// cout<<"n -"<<n->val<<endl;
for(auto it:tmp->neighbors){
// cout<<"for neighbor - "<<it->val<<endl;
if(checkNode(&created,it->val)){
// cout<<"node already preset"<<endl;
n->neighbors.push_back(getNode(&created,it->val));
}else{
// cout<<"node not preset - "<<it->val<<endl;
created.push_back(new Node(it->val));
tbtrav.push_back(it);
n->neighbors.push_back(getNode(&created,it->val));
}
}
}


// traverseG(op);
return op;
}
};
```