拓扑排序的模板

typedef struct node {

int input\_de;//存储当前节点的入度

int idx;//存储当前节点的位置

vector<int> lie\_node;

node() {

input\_de = 0;

idx = 0;

}

}node\_s;

class topology {

public:

vector<int> re\_ans;

map<int, string> recall;

vector<node\_s> near\_grap;

queue<int> ans;//存储要移除的节点

void map\_s() {

recall[0] = "高等数学1";

recall[1] = "高等数学2";

recall[2] = "线性代数";

recall[3] = "离散数学";

recall[4] = "概率论";

recall[5] = "数学建模";

recall[6] = "毕业设计";

}

topology(vector<vector<int>> &grap,int num) {

map\_s();

near\_grap.resize(num);

for (int i = 0; i < num; i++) {

for (int j = 0; j < num; j++) {

if (grap[i][j] == 1) {

near\_grap[i].idx = i;

near\_grap[j].input\_de++;

near\_grap[i].lie\_node.push\_back(j);

}

}

}

for (auto key : near\_grap) {

if (key.input\_de == 0) {

ans.push(key.idx);

}

}

}

void ans\_topo() {

while (!ans.empty()) {

int mem = ans.front();//移除节点

re\_ans.push\_back(mem);//记录序列

ans.pop();//从队列中将访问的节点移除

for (auto idxs : near\_grap[mem].lie\_node) {//与节点相链接的节点的入度-1；

--near\_grap[idxs].input\_de;

if (near\_grap[idxs].input\_de == 0) {//如果节点的入度变为0，则将当前节点放到要移除节点队列中

ans.push(idxs);

}

}

}

//输出映射后的topology 排序

cout << recall[re\_ans[0]] << "-->";

for (int i = 1; i < re\_ans.size()-1; i++) {

cout << recall[re\_ans[i]] << "-->";

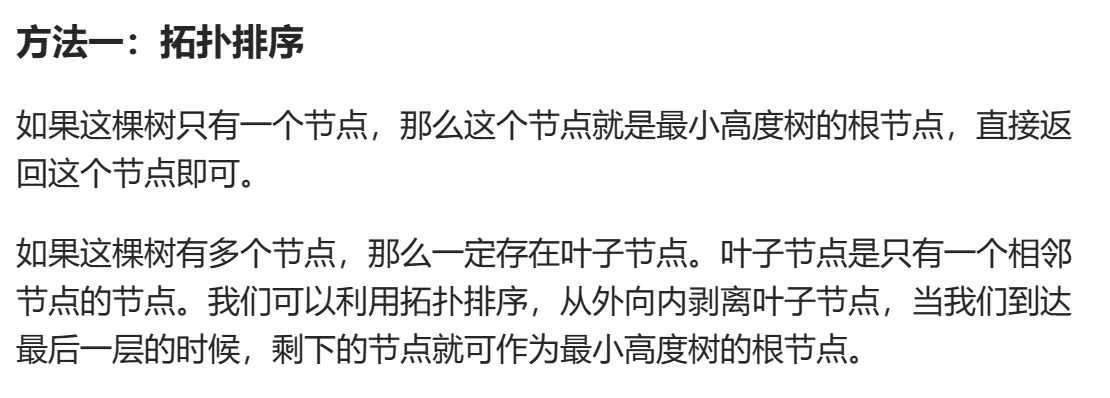
}

cout << recall[re\_ans[re\_ans.size() - 1]]<<"\n";

}

};





根据思路画图分析。

vector<int> findMinHeightTrees(int n, vector<vector<int>>& edges) {    if(n==1){

        return {0} ;

    }

     vector<vector<int>> recall(n);

     vector<int> degree(n,0);

     vector<int> ans ;

     for(auto e\_v : edges){

        int x = e\_v[0];//创建一个邻接表

        int y = e\_v[1];

        recall[x].push\_back(y);

        recall[y].push\_back(x);

         degree[x]++;

         degree[y]++;

     }

     queue<int> q;

     for(int i = 0  ; i  < n ;i++){

         if(degree[i]==1){

            q.push(i);

         }

     }

     while(!q.empty()){

        ans.clear() ;

        for(int i = q.size();i >0 ; i--){

            int a = q.front();

            q.pop();

            ans.push\_back(a);

            for( auto v : recall[a]){

                if(--degree[v]==1){

                    q.push(v);

                }

            }

        }

     }

     return ans ;

    }