山东大学 软件 学院

众智科学与网络化产业课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202000300125 | 姓名：贾星宇 | | 班级：2020级5班 |
| 实验题目：**极化关系下网络结构的稳定性** | | | |
| 实验学时：4 | | 实验日期：2022年4月21日 | |
| 实验目的：  进一步加深对极化关系、网络结构稳定性的理解  学会用无向图及代码来模拟实际的社会网络关系并判断其稳定性。 | | | |
| 硬件环境：  Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz | | | |
| 软件环境：  CLion 2021.2.3 | | | |
| 实验步骤与内容：  【题目要求】  /\*  \*  输入：任意极化关系下图的邻接矩阵（注意边有正负）  输出：是否含有奇数度负向边圈  所谓“极化关系”，指的是网络中的关系分为“友好”和“敌对”两种，这在人际关系和国际关系的一些特定时期都是显著的。在这样的模型中，结构的稳定性是关注的重点，即一个网络结构中的各个关系性质是趋向于不变，还是趋向于改变（从友好变为敌对，或者反过来）？落实到计算问题上，就是要检测图中是否存在包含奇数个敌对边（负向边）的圈。  \*/  【知识储备】  在一个社会中，两个人的关系可能是友善的，也可能是互相抱有敌意的。为了模拟这种现象，我们可以把社会网络中某两个节点之间的边，标注为正关系（友）或负关系（敌）。对于网络中的一个三角结构，我们就可以讨论它是否平衡。   * 平衡结构有以下两种。这在直观上也很好理解，如果三个人互为好友（左图），或者两个关系很好的人同时看不上第三个人（右图），是没有旁的因素打破这种关系的，这就可以长时间维持。   balanced triangles   * 不平衡结构也是以下两种。如果两个人关系不好但有一个共同朋友（左图），这个共同朋友就很难做人，要么倒向其中一边，要么努力让他俩冰释前嫌。如果三个人互有敌意（右图），敌意稍轻的两个人就有动力联合起来，共同对付第三个人。   unbalanced triangles  一网络要么全部是正关系，要么可以表示为如下的结构：  结构平衡定理  策略：只认为矩阵中值为‘-1’的边为边，随后对此无向图进行广度优先算法（BFS）并对节点分层。如果在同一层内存在两个节点之间有边，则表示存在用-边构成的奇圈，则说明此网络结构不稳定。  【代码编写】  利用邻接矩阵存储图，矩阵（i，j）=1表示i与j的关系是友好，=-1表示i与j的关系是敌对。  #include <iostream>  #include "queue"  using namespace std;  void begDFS(int \*\*graph,int len,int begNode,int \* flag,int \* states){  queue<int> theQueue;  theQueue.push(begNode);  flag[begNode] = 1;  states[begNode] = 0;  while (theQueue.size()!=0) {  int targetNode = theQueue.front();  theQueue.pop();  for (int i = 0; i < len; ++i)  if (graph[targetNode][i] == 1 && flag[i] != 1){  theQueue.push(i);  flag[i] = 1;  states[i] = states[targetNode]+1;  }  }  }  void dfs(int \*\*graph,int len,int \* states){  int \* flag = new int[len];  for (int i = 0; i < len; ++i) flag[i] = 0;  for (int i = 0; i < len; ++i) {  for (int j = 0; j < len; ++j) {  cout<<graph[i][j]<<" ";  }  cout << endl;  }  for (int i = 0; i < len; ++i) {  if (flag[i]==0) begDFS(graph,len,i,flag,states);  }  }  //检查同级顶点有无相连情况  bool checkIsMeet(int \*\* graph,int levelLen,int \* levelNode){  for (int i = 0; i < levelLen-1; ++i) {  for (int j = i+1; j < levelLen; ++j) {  if (graph[levelNode[i]][levelNode[j]]==1)return false;  }  }  return true;  }  int main() {  //5  //0 -1 -1 0 0 -1 0 0 0 0 -1 0 0 -1 -1 0 0 -1 0 -1 0 0 -1 -1 0  //4  //0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0  int length;  cin >> length;  int \*\* graphMartix = new int\*[length];  int \* states = new int[length];  for (int i = 0; i < length; ++i) {  graphMartix[i] = new int[length];  }  //创建图  for (int i = 0; i < length; ++i) {  for (int j = 0; j < length; ++j) {  cin >> graphMartix[i][j];  }  }  //初始化图(把-1权值的边看作边，其余的皆不看做)  for (int i = 0; i < length; ++i) {  for (int j = 0; j < length; ++j) {  if (graphMartix[i][j] == -1)graphMartix[i][j] = 1;  else graphMartix[i][j] = 0;  }  }  int result = false;  //判断每一级  for (int i = 0; i < length; ++i) {  int \* sameLevel = new int[length];  for (int j = 0; j < length; ++j) {  sameLevel[j] = -1;  }  int count = 0;  for (int j = 0; j < length; ++j) {  if (states[j] == i)sameLevel[count++] = j;  }  if (!checkIsMeet(graphMartix, count, sameLevel))result = true;  }  if (result) cout << "Contains odd-degree negative edge loops, the structure is unstable!";  else cout << "It does not contain odd-degree negative edge circles, and this structure is stable!";  return 0;  }  【结果测试】  输入第一个图：  //5  //0 -1 -1 0 0 -1 0 0 0 0 -1 0 0 -1 -1 0 0 -1 0 -1 0 0 -1 -1 0    输出：    输入第二个图：  //4  //0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0    输出： | | | |
| 结论与分析：  1. 所谓“极化关系”，指的是网络中的关系分为“友好”和“敌对”两种，这在人际关系和国际关系的一些特定时期都是显著的。在这样的模型中，结构的稳定性是关注的重点，即一个网络结构中的各个关系性质是趋向于不变，还是趋向于改变（从友好变为敌对，或者反过来）？落实到计算问题上，就是要检测图中是否存在包含奇数个敌对边（负向边）的圈。  2. 在一个社会中，两个人的关系可能是友善的，也可能是互相抱有敌意的。为了模拟这种现象，我们可以把社会网络中某两个节点之间的边，标注为正关系（友）或负关系（敌）。对于网络中的一个三角结构，我们就可以讨论它是否平衡。  3.证明为什么用奇数来判断 | | | |