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：**计算聚集系数和邻里重叠度** | | | |
| 实验学时：4 | | 实验日期：2022年4月1日 | |
| 实验目的：  加深对聚集系数及邻里重叠度的理解与应用  学习聚集系数和邻里重叠度是如何影响社会网络的  学习用高级程序设计语言来计算聚集系数、邻里重叠度，并且尝试提高计算性能 | | | |
| 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：任意的无向图  输出：  1）每个节点的聚集系数  2）每个节点对的邻里重叠度  相关定义：  聚集系数：节点A的聚集系数 = A的任意两个朋友之间也是朋友的概率（即邻居间朋友对的个数除以总对数）  邻里重叠度：与A、B均为邻居的节点数/ 与节点A、B中至少一个为邻居的节点数  \*/  【代码编写】  #include <iostream>  using namespace std;  int\* findFriends(int \*\* graph,int mine,int len){  int \*result = new int[len];  int count = 0;  for (int i = 0; i < len; ++i)  if (graph[i][mine]==1)  result[count++] = i;  for (int i = count; i < len; ++i) {  result[i] = -1;  }  return result;  }  //计算组合，从n个数中挑m个  int C(int m,int n){  int a = 1;  for (int i = 0; i < m; ++i) {  a \*= (n-i);  }  return a/m;  }  //得到它有几个朋友  int getTheLength(int \* arr,int length){  int result = 0;  for (int i = 0; i < length; ++i) {  if (arr[i]!=-1)result++;  }  return result;  }  //得到他的朋友中互为朋友的个数  int getBeFriends(int \*friends,int len,int \*\*graph){  int result = 0;  for (int i = 0; i < len; ++i) {  for (int j = 0; j < len; ++j) {  if (graph[friends[i]][friends[j]]==1)result++;  }  }  return result/2;  }  //计算共同朋友  int\* getSameFriends(int a,int b,int \*\* graph,int length){  int \* sameFriends = new int[length];  int count = 0;  for (int i = 0; i < length; ++i) {  if (graph[a][i]==1&&graph[b][i]==1) sameFriends[count++] = i;  }  for (int i = count; i < length; ++i) {  sameFriends[i] = -1;  }  return sameFriends;  }  //计算总共朋友  int\* getBothFriends(int a,int b,int \*\* graph,int length){  int \* sameFriends = new int[length];  int count = 0;  for (int i = 0; i < length; ++i) {  if (graph[a][i]==1||graph[b][i]==1)  if (i!=a && i!= b)sameFriends[count++] = i;  }  for (int i = count; i < length; ++i) {  sameFriends[i] = -1;  }  return sameFriends;  }  int main() {  //利用邻接矩阵存储有向图  //4  //0 1 1 1 1 0 0 1 1 0 0 0 1 1 0 0  //5  //0 1 1 1 0 1 0 1 0 0 1 1 0 0 1 1 0 0 0 1 0 0 1 1 0  int length;  cin >> length;  int c2l = C(2,length);  int \*\* graphMartix = new int\*[length];  int \*\* friends = new int\*[length];  int \*\* sameFriends = new int \*[2];  int \*\* bothFriends = new int \*[2];  double \* clusteringCoefficient = new double[length];//每个节点的聚集系数  double \* neighborhoodOverlap = new double[C(2,length)];//节点对的邻里重叠度  for (int i = 0; i < length; ++i) {  graphMartix[i] = new int[length];  friends[i] = new int[length];  sameFriends[i] = new int[c2l];  bothFriends[i] = new int[c2l];  }  //创建图  for (int i = 0; i < length; ++i) {  for (int j = 0; j < length; ++j) {  cin >> graphMartix[i][j];  }  }  //————————————————————————————————  //得到朋友是谁  for (int i = 0; i < length; ++i)  friends[i] = findFriends(graphMartix,i,length);  // for (int i = 0; i < length; ++i) {  // for (int j = 0; j < length; ++j) {  // cout << friends[i][j]<<',';  // }  // cout << endl;  // }  //计算节点聚集系数  int tlen = length;  for (int i = 0; i < tlen; ++i) {  if (getTheLength(friends[i],length)<2)  clusteringCoefficient[i] = 0;  else clusteringCoefficient[i] =  ((double)getBeFriends(friends[i], getTheLength(friends[i],length),graphMartix)  /  (double)C(2,getTheLength(friends[i],length)));  }  //打印：  for (int i = 0; i < length; ++i)  cout <<"node "<<i<<"'s clustering coefficient is "<<clusteringCoefficient[i]<<endl;  //每对节点的邻里重叠度  int countB = 0;  int countS = 0;  int countN = 0;  for (int i = 0; i < length; ++i) {  for (int j = i+1; j < length; ++j) {  sameFriends[countS++] = getSameFriends(i,j,graphMartix,c2l);  bothFriends[countB++] = getBothFriends(i,j,graphMartix,c2l);  cout << "the friend are both their friends:";  for (int k = 0; k < c2l; ++k) {  cout << sameFriends[countS-1][k]<<",";  }  cout << " the friends are their friends:";  for (int k = 0; k < c2l; ++k) {  cout << bothFriends[countB-1][k]<<",";  }  cout<<endl;  neighborhoodOverlap[countN++] =  (double)getTheLength(sameFriends[countS-1],c2l)/(double) getTheLength(bothFriends[countB-1],c2l);  cout << "node "<<i<<" and node "<<j<<"'s neighborhood overlap is "<<neighborhoodOverlap[countN-1]<<endl;  cout << endl;  }  }    return 0;  }  【结果测试】  测试两个无向图：  第一个：    输入及输出：    第二个：    输入及输出： | | | |
| 结论分析与体会：  1.A节点的聚集系数 = A的任意两个朋友之间也是朋友的概率  聚集系数为0表示A的朋友之间互不认识  聚集系数为1表示A的所有朋友都互相认识  2.节点A、B的邻里重叠度指的是他们的共同朋友数在总朋友数中的占比  邻里重叠度为0表示两人的朋友都互不认识  邻里重叠度为1表示两人的朋友都相互认识  3.社会网络结构的基本意向：  用桥（捷径、邻里重叠度低的边或弱关系）连接起来的相对比较密集的节点群。 | | | |