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：**表决问题** | | | |
| 实验学时：4 | | 实验日期：2022年5月1日 | |
| 实验目的： | | | |
| 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：  【实验题目】  输入：给定m个人对n个项目按排序的投票。  输出：  1）确定其中是否隐含有孔多塞悖论（涉及到在有向图上尝试节点的拓扑排序）。  2）如果没有，就直接给出群体序，如果有，就按照一个特定的属性序，指出哪些投票是不满足单峰性质的，认为它们是“废票”，剔除后按照中位项定理给出群体排序。  【先习条件】  1.若所有选举人的排序都满足“单峰偏好”，则按照少数服从多数规则两两比较侯选项产生的群体偏好是完备且传递的。  2. 逐次找出“最大的”（群体意义）  记L1, L2, …, LM为个体排序表，Li(1)为对应个体表中第一个（最大的）元素  将Li(1), i=1,2,…,M按照X1, X2, …… XN的特征序排列（一共M个，有的X可能有多次出现）  从如此排列的M个元素中取中间项为群体排序的第一项（最大的元素）  从L1, L2, …, LM中删除该元素，留下的依然是单峰排序表，接着可以取出第二个，等等  3. 要说明，相继取出的那些“中间项”，在少数服从多数原则下，比其他所有还剩下的候选项都要大  只需说明，当Li(1), i=1,2,…,M按照X1, X2, …… XN 的序排列后，其中位项与其他M-1项中的不同元素在两两比较中均能基于M个个体排序中的情形，以少数服从多数原则胜出  从一个例子看，若排列情况如下：  X1, X1, X2, X2, X3 (注:它们是5个个体排序的头)  为什么说5人中至少3人认为X2>X1?  中间那位如何认为？第4位如何认为？第5位如何认为？  对称地，我们看到中间和左边的人都会认为X2>X3。    【设计步骤】   1. 根据人们的喜好计算出物品之间的优先级，结果矩阵i，j=1表示物品i优先级高于物品j   函数：  *//喜好矩阵L:Lij表示第i个人对于选项j的优先级（1~j） //根据输入的n个人（行）对m个选项（列）的nxm喜好矩阵L，构造出mxm矩阵P：Pij=1表示选项i优先级>选项j* void getPriority(int \*\* L,int \*\* P,int n,int m){  *//初始化P* for (int i = 0; i < m; ++i) {  for (int j = 0; j < m; ++j) {  P[i][j] = 0;  }  }   for (int i = 0; i < m; ++i) {  for (int j = i+1; j < m; ++j) {  *//比较物品i和j* int votes = 0;  for (int k = 0; k < n; ++k) {  *//如果i优先于j,votes++;j优先于i,votes--;  //最终votes>0,i win;votes<0,j win* if (L[k][i]<L[k][j])votes++;  else if(L[k][i]>L[k][j])votes--;  }  if (votes>0){  P[i][j] = 1;  P[j][i] = -1;  }else if (votes<0){  P[i][j] = -1;  P[j][i] = 1;  }  }  } }  结果测试：       1. 判断得到的优先级序列是否满足孔多塞悖论：   基本思想：判断得到的优先级有向图是不是无回路图  *//判断是否含有孔多塞悖论：图中是否可以构造出拓扑序列---有向图是否五环 //每次找一个入度为零的点，将所有和它相连的点的入度减一（删除相连的边），重复此步骤（有n个点，每次选择一个入度为0的点，进行n次）。 // 如果这时还有入度不为零的点，证明有环，输出NO，反之输出YES。 //true表示含有悖论，需要删掉不满足单峰性质的* bool ifCondorcet(int \*\*P,int item){  *//计算每个顶点的度数* int \* nodeDegree = new int[item];  for (int i = 0; i < item; ++i) {  nodeDegree[i] = 0;  for (int j = 0; j < item; ++j) {  if (P[j][i]==1)  nodeDegree[i]++;  }  }   stack<int> zeroDegreeNode;  for (int i = 0; i < item; ++i) {  if (nodeDegree[i]==0)zeroDegreeNode.push(i);  }  while (!zeroDegreeNode.empty()){  int theNode = zeroDegreeNode.top();  nodeDegree[theNode]=-1;*//避免此点被再次入栈* zeroDegreeNode.pop();  for (int i = 0; i < item; ++i) {  if (P[theNode][i]==1)  nodeDegree[i]--;  }  for (int i = 0; i < item; ++i) {  if (nodeDegree[i]==0)  zeroDegreeNode.push(i);  }  }   for (int i = 0; i < item; ++i) {  if (nodeDegree[i]>=0)return true;  }  return false; }  0表示没有孔多塞悖论：    1表示有孔多塞悖论，需要删掉不满足单峰性质的人     1. 删除不满足单峰性质的人：   核心思想：不能有离最高排名的人远但是还排名高的人：  *//根据表L找出不满足单峰性质的人并将其内容全改为-1 //假设items的某性质权值按照从高到低的顺序排列（矩阵中编号小的items权值大）* void deleteNoSingle(int \*\*L,int people,int item){  *//对每个人分析* for (int i = 0; i < people; ++i) {  for (int j = 0; j < item; ++j) {  for (int k = j+1; k < item; ++k) {  if (L[i][j]==1||L[i][k]==1)  if ((abs(k-j)>abs(L[i][k]-L[i][j]))){  for (int l = 0; l < item; ++l) {  L[i][l] = -1;  }  }  }  }  } }  如果满足，则保持不变：    如果不满足，则将此人选项置为-1:     1. 给出群体排序（按照输入的优先级挨个比对）   *//给出群体排序* int \* sortRes(int \*\* L,int item,int people){  int \* result = new int[item];  for (int i = 0; i < item; ++i) {  int theNum = 0;  for (int j = 0; j < people; ++j) {  if (L[j][i]>0)theNum+=L[j][i];  }  result[i] = theNum;  }   return result; }  正确的无悖论：    有悖论的先删除违反单峰原则的：    其中按第一个选项，第二个……排序，数值小的优先级高 | | | |
| 结论分析与体会：  实现思路：  将每个人对于n个选项的优先级进行从高到低排序，并且由k个人构成k×n的矩阵A；将每个顶点看做一个选项，如果顶点a优先级高于点b，则存在一条边a→b  ·确定是否含孔多塞悖论：首先根据群众表（矩阵）收集到的每两对顶点的优先度，以少数服从多数的原则判断边的方向，并以此构造有向网状结构B  ·如果含有孔多塞悖论，则需要去除掉不满足单峰性质的人：若Xi被他排在了第一位，则对于k<j<i，Xj要排在Xk的前面；且对于i<j<k，Xj要排在Xk的前面  即选择了一个“最爱”后，对其他选项的偏爱程度应参照所关注的性质在两边随与这个最爱的距离下降（左右两边相互之间没有要求）；  如果有则不满足需要踢掉；如果没有则继续判断由最大顶点分开的两组队列中有无。随后给出群体序  ·如果不含有孔多塞悖论，则直接构造群体序：利用上述矩阵A，依次按照【先习条件】【2】的方法对矩阵进行搜索，得到一个数组，内容为各选项的群体序。  附：核心代码  */\*  \* 输入：给定m个人对n个项目按排序的投票。 输出： 1）确定其中是否隐含有孔多塞悖论（涉及到在有向图上尝试节点的拓扑排序）。 2）如果没有，就直接给出群体序，如果有，就按照一个特定的属性序，指出哪些投票是不满足单峰性质的，认为它们是“废票”，剔除后按照中位项定理给出群体排序。   \*/* #include <iostream> #include "stack" using namespace std;   *//喜好矩阵L:Lij表示第i个人对于选项j的优先级（1~j） //根据输入的n个人（行）对m个选项（列）的nxm喜好矩阵L，构造出mxm矩阵P：Pij=1表示选项i优先级>选项j* void getPriority(int \*\* L,int \*\* P,int n,int m){  *//初始化P* for (int i = 0; i < m; ++i) {  for (int j = 0; j < m; ++j) {  P[i][j] = 0;  }  }   for (int i = 0; i < m; ++i) {  for (int j = i+1; j < m; ++j) {  *//比较物品i和j* int votes = 0;  for (int k = 0; k < n; ++k) {  *//如果i优先于j,votes++;j优先于i,votes--;  //最终votes>0,i win;votes<0,j win* if (L[k][i]<L[k][j])votes++;  else if(L[k][i]>L[k][j])votes--;  }  if (votes>0){  P[i][j] = 1;  P[j][i] = -1;  }else if (votes<0){  P[i][j] = -1;  P[j][i] = 1;  }  }  } }   *//判断是否含有孔多塞悖论：图中是否可以构造出拓扑序列---有向图是否五环 //每次找一个入度为零的点，将所有和它相连的点的入度减一（删除相连的边），重复此步骤（有n个点，每次选择一个入度为0的点，进行n次）。 // 如果这时还有入度不为零的点，证明有环，输出NO，反之输出YES。 //true表示含有悖论，需要删掉不满足单峰性质的* bool ifCondorcet(int \*\*P,int item){  *//计算每个顶点的度数* int \* nodeDegree = new int[item];  for (int i = 0; i < item; ++i) {  nodeDegree[i] = 0;  for (int j = 0; j < item; ++j) {  if (P[j][i]==1)  nodeDegree[i]++;  }  }   stack<int> zeroDegreeNode;  for (int i = 0; i < item; ++i) {  if (nodeDegree[i]==0)zeroDegreeNode.push(i);  }  while (!zeroDegreeNode.empty()){  int theNode = zeroDegreeNode.top();  nodeDegree[theNode]=-1;*//避免此点被再次入栈* zeroDegreeNode.pop();  for (int i = 0; i < item; ++i) {  if (P[theNode][i]==1)  nodeDegree[i]--;  }  for (int i = 0; i < item; ++i) {  if (nodeDegree[i]==0)  zeroDegreeNode.push(i);  }  }   for (int i = 0; i < item; ++i) {  if (nodeDegree[i]>=0)return true;  }  return false; }  *//根据表L找出不满足单峰性质的人并将其内容全改为-1 //假设items的某性质权值按照从高到低的顺序排列（矩阵中编号小的items权值大）* void deleteNoSingle(int \*\*L,int people,int item){  *//对每个人分析* for (int i = 0; i < people; ++i) {  for (int j = 0; j < item; ++j) {  for (int k = j+1; k < item; ++k) {  if (L[i][j]==1||L[i][k]==1)  if ((abs(k-j)>abs(L[i][k]-L[i][j]))){  for (int l = 0; l < item; ++l) {  L[i][l] = -1;  }  }  }  }  } }   *//给出群体排序* int \* sortRes(int \*\* L,int item,int people){  int \* result = new int[item];  for (int i = 0; i < item; ++i) {  int theNum = 0;  for (int j = 0; j < people; ++j) {  if (L[j][i]>0)theNum+=L[j][i];  }  result[i] = theNum;  }   return result; }       int main() {   *//初始化L和P* int people,item;  cout << "people nums and item nums:" << endl;  cin >> people >> item;  int \*\* L = new int \*[people];  for (int i = 0; i < people; ++i) {  L[i] = new int[item];  }  int \*\* P = new int \*[item];  for (int i = 0; i < item; ++i) {  P[i] = new int[item];  }  cout << "people like:"<<endl;  for (int i = 0; i < people; ++i) {  for (int j = 0; j < item; ++j) {  cin >> L[i][j];  }  }     getPriority(L,P,people,item);  cout << ifCondorcet(P,item)<<endl;  if (ifCondorcet(P,item))deleteNoSingle(L,people,item);  for (int i = 0; i < people; ++i) {  for (int j = 0; j < item; ++j) {  cout << L[i][j]<<",";  }  cout << endl;  }    cout << "group result:"<<endl;  int \* result = new int[item];  result = sortRes(L,item,people);  for (int i = 0; i < item; ++i) {  cout << result[i]<<",";  }   return 0; } | | | |