

**实验报告**

**（2019 / 2020学年第一学期）**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | 离散数学 | | | | | |
| 实验名称 | 图的生成及欧拉（回）路的确定 | | | | | |
| 实验时间 | 2020 | 年 | 12 | 月 | 9 | 日 |
| 指导单位 | 计算机学院计算机科学与技术系 | | | | | |
| 指导教师 |  | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 学生姓名 |  | 班级学号 |  |
| 学院(系) | 计算机学院 | 专业 | 信息安全 |

**实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验名称** | **图的生成及欧拉（回）路的确定** | | | **指导教师** | 柯昌博 |
| **实验类型** | **综合** | **实验学时** | **4** | **实验时间** | 2020.12.9 |
| 1. **实验目的和要求**   对具有*n*个结点的无向图，判断其能否被一笔画。  即对给定*n*个结点的无向图，进行欧拉图和半欧拉图的判定，若是欧拉图或半欧拉图，则输出所有的欧拉（回）路。 | | | | | |
| 二、**实验环境(实验设备)**  硬件：微型计算机  软件：Windows 操作系统、Microsoft Visual C++6.0、C++ | | | | | |
| **三、实验原理及内容**  **主要数据结构和算法**：Fleury算法（找出欧拉（半）图的一条欧拉（回）路）  深度优先搜索（找出全部欧拉（回）路）  并查集（用于判断图是否连通）  深度优先搜索判断连通性（备选） | | | | | |

**实验报告**

|  |
| --- |
| **程序流程**：1.读入结点数n和边数m，生成邻接矩阵  2.判断图的连通性  3.判断各个结点度数。当奇数度结点为0个时，从图中任意结点出发，都能找到欧拉回路；当奇数度结点为2个时，从其中一点出发，能找到一条欧拉路，到另一个结点。  4.用DFS找到全部欧拉回路，或用Fluery算法找到一条欧拉（回）路。  **定义：** 欧拉通路(回路)：通过图（无向图或有向图）中所有边一次且仅一次行遍图中所有顶点的     通路(回路)称为欧拉通路(回路)。 欧拉图与半欧拉图：具有欧拉回路的图称为欧拉图，具有欧拉通路而无欧拉回路的     图称为半欧拉图。 桥：设无向图G=，若存在边集E的一个非空子集E1，使得p(G-E1)>p(G)，而对     于E1的任意真子集E2，均有p(G-E2)=p(G)，则称E1是G的边割集，或简称割集；     若E1是单元集，即E1={e}，则称e为割边或桥。[p(G)表示图G的连通分支数.]     **Fleury算法原理**：    设G为欧拉图，一般说来G中存在若干条欧拉回路，下面是求欧拉回路的Fleury算法： Fleury算法： （1）任取v0∈V(G)，令P0=v0； （2）设Pi=v0e1v1e2...eivi已经行遍，按下面方法来从E(G)-{e1,e2,...,ei}中选      取ei+1：     （a）ei+1与vi想关联；     （b）除非无别的边可供行遍，否则ei+1不应该为Gi=G-{e1,e2,...,ei}中的桥. （3）当（2）不能再进行时，算法停止。 可以证明，当算法停止时所得简单回路Pm=v0e1v1e2...emvm（vm=v0）为G中的一条欧拉回路。  **完整代码+注释**  #include <iostream>  #include <stack>  #include<cstring>  #define INF 100000  using namespace std;  int G[100][100];  int m;int n;//m为边数,n为点数，  int path[50], k = 0; //记录欧拉路的路径  int visitEdge[100][100];  bool visted[50]; //标记点是否被访问  int begin1; //判是否为连通图，搜索的起点  stack<int> s;//fluery  int fa[INF]={0};//并查集  void allDfs(int x,int edgeNumber=0)//dfs输出全部欧拉路  {if(edgeNumber==0){k=0;} //每次从一个新的起点开始时，初始化  path[k++]=x;  if (edgeNumber==m)  {  for(int i=0;i<m+1;i++)  {cout<<path[i]<<" ";  }  cout<<endl;  return;  }  else  {for(int i=0;i<n;i++)  {  if(G[x][i])    { G[x][i]--;G[i][x]--;    allDfs(i,edgeNumber+1);  G[x][i]++;G[i][x]++;  k--;  }  }  }  }  void DFS(int x)  {  visted[x] = true;  for(int i = 0; i < n; i++)  if(!visted[i] && G[x][i])  DFS(i);  }    //从所设定的起点深度优先遍历图，若有一个点没被访问，则为非连通图  bool Judge()//dfs判断图连通性  {  DFS(0);  for(int i = 0; i < n; i++)  if(!visted[i])  return false;  return true;  }  void init()//并查集初始化，每个结点各自为一个集合  {  for (int i = 0; i < n; ++i)  fa[i] = i;  }  int find(int x)//寻找x的祖先  {  if(fa[x]==x) return fa[x];  else  {fa[x]=find(fa[x]);//路径压缩  return fa[x];}    }  int unionn(int x,int y)  {  int fx=find(x);  int fy=find(y);  if(fx==fy) return 0;  fa[fx]=fy;  return 1;  }  void dfsfleury(int x)  {  s.push(x);  for(int i = 0; i < n; i++)  {  if(G[i][x] > 0)  {  G[x][i] --;  G[i][x] --;  dfsfleury(i);  break;  }  }  }  void fleury(int x)  {  int b;  s.push(x);  while(!s.empty())  {  b = 0;  for(int i =0 ; i < n; i++)  {  if(G[s.top()][i] > 0)  {  b = 1;  break;  }  }  if(b == 0)//栈顶元素没有边与i相连 ，则输顶元素出栈  {  cout << s.top() << " ";  s.pop();  }  else//栈顶元素与i相连，则以该元素为起点，遍历一条路径直到无路，该路径入栈，路径上的所有边删除  {  int t = s.top();  s.pop();  dfsfleury(t);  }  }  }  int main()  { int num = 0;//奇数结点个数  int start[50]= {0};int j=0;//欧拉路开始结点 和数组指针j  cout<<"请输入点数和边数"<<endl;    while(cin >> n >> m,n)  {  int l=0;//用于连通性判断  while(!s.empty())//初始化栈s  s.pop();  memset(G,0,sizeof(G));  init();//初始化并查集  k=0;j=0;num = 0;//初始化k ,j,num  cout<<"请输入各边（结点编号从0开始）"<<endl;  for(int i = 0; i < m; i++)  {  int a,b;  cin >> a >> b;  G[a][b] ++;  G[b][a] ++;  l+=unionn(a,b);  }  if(l!=n-1)//如果一个图是连通图，其各结点的最小生成树的边数为点数-1 即n-1  {cout << "Non-connected graph" << endl;  cout<<"请输入点数和边数"<<endl;  continue;  }  /\*if(!Judge())// dfs判断连通性  {  cout << "Non-connected graph" << endl;  cout<<"请输入点数和边数"<<endl;  continue;  }\*/    for(int i = 0; i <= n-1; i++)  {  int d = 0;//每个点度数  for(int j = 0; j <= n-1; j++)  d += G[i][j];  if(d % 2 == 1)//如果度数为奇数  { num++;  if(num>2){break;}  start[j++] = i;    }  }    if(num == 0 || num == 2)  {  if(num == 0)  {cout << "欧拉回路是" << endl;  for(int i=0;i<n;i++)  {allDfs(i,0);  //fleury(i);  }  }  else  {cout << "欧拉路是（无回路）" << endl;  for(int i=0;i<2;i++)  {allDfs(start[i],0);  //fleury(start[i]);  }  }        cout<<endl;}  else  cout << "No Euler path" << endl;  cout<<"请输入点数和边数"<<endl;  }  return 0;  }  **运行截图** |

**实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **四、实验小结**（包括问题和解决方法、心得体会、意见与建议等）  **主要问题**  1.Fleury算法只能在欧拉图中找到一条欧拉路，关于如何在欧拉图中找到所有欧拉路只会暴力搜索，暂时还没有发现一个更好的方法。  2.Fleury算法不便于理解，可以在欧拉图上，用手推的方法，走一遍fleury流程，以加深理解。  **实验心得**  **1.**同一个问题，可以尝试使用不同方法解决。比如判断图的连通性，最常规的做法是dfs，但也可以用并查集的知识。  2.本实验是在无向图上找欧拉路。注意不能把无向图默认当成简单图，还要考虑到有平行边的情况。  3.并查集的father数组每次使用前都要初始化。在main函数中每一轮输入前都要初始化各相关变量。  4.复杂度分析：构建邻接矩阵 O（n^2）  DFS找出全部欧拉（回）路 O（m^2）  总复杂度O（m^2）  **意见与建议**  无 | | | | | |
| 1. **支撑毕业要求指标点**   支撑毕业要求的指标点为：  √   * 1-4掌握计算机科学与技术领域的专业知识，能将专业知识用于分析和解决计算机领域复杂工程问题。 * 2-1能够应用数学、自然科学和工程科学的基本知识，识别和分析计算机领域复杂工程问题的特征。 | | | | | |
| **六、指导教师评语 (含学生能力达成度的评价)** | | | | | |
| **成绩** |  | **批阅人** |  | **日期** |  |

如果不太想写太多字，“指导教师评语”也可以设计为如下的各选择项用打勾形式（仅仅作为一个简单示例，请各课程负责人根据课程和实验情况以及支撑的指标点来自行设定选择项，同一门课程的不同实验评分细则项允许存在不同）：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **评分细则** | **评分项** | **优秀** | **良好** | **中等** | **合格** | **不合格** |
| **遵守实验室规章制度** |  |  |  |  |  |
| **学习态度** |  |  |  |  |  |
| **算法思想准备情况** |  |  |  |  |  |
| **程序设计能力** |  |  |  |  |  |
| **解决问题能力** |  |  |  |  |  |
| **课题功能实现情况** |  |  |  |  |  |
| **算法设计合理性** |  |  |  |  |  |
| **算法效能评价** |  |  |  |  |  |
| **回答问题准确度** |  |  |  |  |  |
| **报告书写认真程度** |  |  |  |  |  |
| **内容详实程度** |  |  |  |  |  |
| **文字表达熟练程度** |  |  |  |  |  |
| **其它评价意见** |  | | | | |
| **本次实验能力达成评价（总成绩）** |  |  |  |  |  |