# 实验九 图的应用

## 一、实验目的

1. 学习图的基本定义方式
2. 熟练掌握图的矩阵存储和邻接表存储。
3. 掌握图的两种遍历方法。
4. 能利用图的遍历解决实际问题。

## 二、实验要求

【项目1】---假设不带权有向图采用邻接矩阵 g 存储，设计实现以下功能的算法：

（1）求出图中每个顶点的入度。

（2）求出图中每个顶点的出度。

（3）求出图中出度为0的顶点数。

|  |
| --- |
| 请将程序代码附录于此 |
| int A[6][6]=  {  {0,5,INF,7,INF,INF},  {INF,0,4,INF,INF,INF},  {8,INF,0,INF,INF,9},  {INF,INF,5,0,INF,6},  {INF,INF,INF,5,0,INF},  {3,INF,INF,INF,1,0}  };  int count=0,count1=0,count2=0,count3=0,count4=0,count5=0;  for (int i = 0; i < 6; i++)  {  if(A[0][i]!=INF)  count++;  if (A[1][i]!=INF)  count1++;  if (A[2][i]!=INF)  count2++;  if (A[3][i]!=INF)  count3++;  if (A[4][i]!=INF)  count4++;  if (A[5][i]!=INF)  count5++;  }  printf("0的入度：%d\n",count);  printf("1的入度：%d\n",count1);  printf("2的入度：%d\n",count2);  printf("3的入度：%d\n",count3);  printf("4的入度：%d\n",count4);  printf("5的入度：%d\n",count5);  printf("\n");  count=0;count1=0;count2=0;count3=0;count4=0;count5=0;  for (int i = 0; i < 6; i++)  {  if(A[i][0]!=INF)  count++;  if (A[i][1]!=INF)  count1++;  if (A[i][2]!=INF)  count2++;  if (A[i][3]!=INF)  count3++;  if (A[i][4]!=INF)  count4++;  if (A[i][5]!=INF)  count5++;  }  int out[6];  out[0]=count;  out[1]=count1;  out[2]=count2;  out[3]=count3;  out[4]=count4;  out[5]=count5;  printf("0的出度：%d\n",count);  printf("1的出度：%d\n",count1);  printf("2的出度：%d\n",count2);  printf("3的出度：%d\n",count3);  printf("4的出度：%d\n",count4);  printf("5的出度：%d\n",count5);  printf("\n");  count=0;  for (int i = 0; i < 6; i++)  {  if(out[i]==0)  {  count++;  printf("出度为0的点是：%d\n",i);  }  }  if (count==0)  printf("没有出度为0的点\n\n"); |
| 请将运行结果截图附录于此 |
|  |

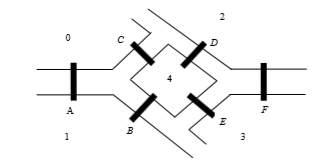
【项目2】---假设不带权有向图采用邻接表 G 存储，设计实现以下功能的算法： （1）求出图中每个顶点的入度。

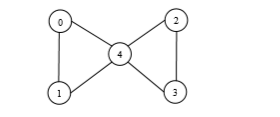
（2）求出图中每个顶点的出度。

（3）求出图中出度为0的顶点数。

|  |
| --- |
| 请将程序代码附录于此 |
| Main.cpp  #include <stdio.h>  #include <stdlib.h>  #include"malloc.h"  #include"graph.h"  int main()  {  int adj\_mat[MAX\_V][MAX\_V]=  {  {0,1,0,1,0},  {0,0,1,1,0},  {0,0,0,1,1},  {0,0,0,0,0},  {1,0,0,1,0}  };  int x=5,y=8;  AdjGraph\* AG=NULL;  CreateAdj(AG,adj\_mat,x,y);  DispAdj(AG);  printf("\n");  AdjInDegree(AG);  printf("\n");  AdjOutDegree(AG);  printf("\n");  AdjZeroOutDegree(AG);  return 0;  }  Alog.cpp  #include <stdio.h>  #include <stdlib.h>  #include"malloc.h"  #include"graph.h"  void CreateAdj(AdjGraph \*&G,int A[MAX\_V][MAX\_V],int v,int e)  {  int i,j;  ENode\* p=NULL;  G=(AdjGraph\*)malloc(sizeof(AdjGraph));  for(i=0;i<v;i++)  {  G->Ver[i].first=NULL;  G->Ver[i].info='A'+i;  }  for(i=0;i<v;i++)  for(j=v-1;j>=0;j--)  if(A[i][j]!=0 && A[i][j]!=INF)  {  p=(ENode\*)malloc(sizeof(ENode));  p->adjVNo=j;  p->next=G->Ver[i].first;  G->Ver[i].first=p;  }  G->e=e;  G->v=v;  }  void DispAdj(AdjGraph \*G)  {  int i;  ENode \*p;  printf("图的邻接表存储\n");  for(i=0;i < G->v;i++)  {  p=G->Ver[i].first;  printf("%d的顶点%c:",i,G->Ver[i].info);  while(p!=NULL)  {  printf("%d→",p->adjVNo);  p=p->next;  }  printf("NULL\n");  }  }  void AdjInDegree(AdjGraph \*G)  {  int i,j,count;  ENode \*p;  for(i=0;i < G->v;i++)  {  count=0;  printf("%d的顶点%c入度为:",i,G->Ver[i].info);  for(j=0;j < G->v;j++)  {  p=G->Ver[j].first;  while(p!=NULL)  {  if(p->adjVNo==i)  count++;  p=p->next;  }  }  printf("%d\n",count);  }  }  void AdjOutDegree(AdjGraph \*G)  {  int i,count;  ENode \*p;  for(i=0;i < G->v;i++)  {  p=G->Ver[i].first;  count=0;  printf("%d的顶点%c出度为:",i,G->Ver[i].info);  while(p!=NULL)  {  count++;  p=p->next;  }  printf("%d\n",count);  }  }  void AdjZeroOutDegree(AdjGraph \*G)  {  int i,count=0;  ENode \*p;  for(i=0;i < G->v;i++)  {  if(G->Ver[i].first==NULL)  count++;  }  printf("出度为0的顶点个数为%d\n",count);  }  Graph.cpp  #ifndef HEAD\_H\_INCLUDED  #define HEAD\_H\_INCLUDED  #include <stdio.h>  #include <stdlib.h>  #define MAX\_V 5  #define INF 2147483647  typedef struct  {  int No;  char info;  } Ver;  typedef struct  {  int adjMat[MAX\_V][MAX\_V];  int v;  int e;  Ver ver[MAX\_V];  } MatGraph;  typedef struct edgeNode  {  int adjVNo;  struct edgeNode \*next;  } ENode;  typedef struct  {  char info;  ENode \* first;  } VNode;  typedef struct  {  VNode Ver[MAX\_V];  int v;  int e;  } AdjGraph;  void CreateAdj(AdjGraph \*&G,int A[MAX\_V][MAX\_V],int v,int e);  void DispAdj(AdjGraph \*G);  void AdjInDegree(AdjGraph \*G);  void AdjOutDegree(AdjGraph \*G);  void AdjZeroOutDegree(AdjGraph \*G);  #endif // HEAD\_H\_INCLUDED |
| 请将运行结果截图附录于此 |
|  |

【项目3】----设有 5 地（0～4）之间架设有 6 座桥（A～F），如下图所示，设计一个算法， 从某一地出发，经过每座桥恰巧一次，最后仍回到原地。

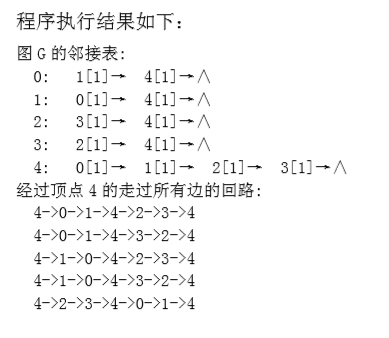


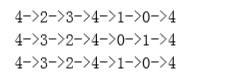


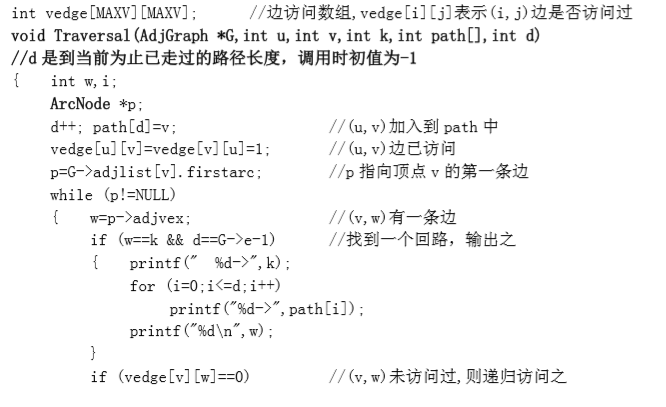
【解题思路】：

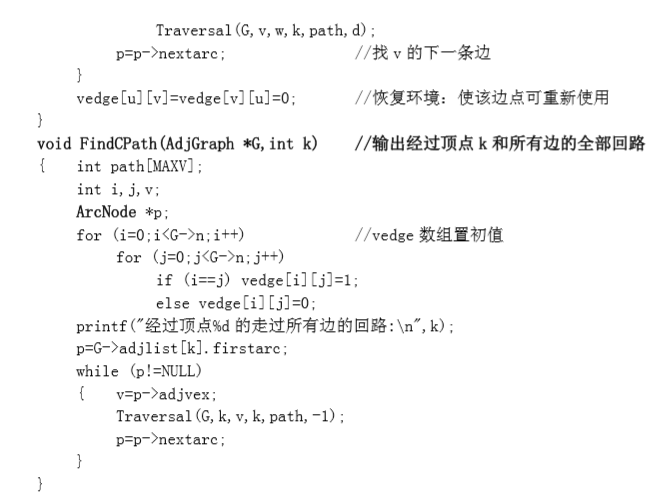
该地图对应的一个无向图 G，本题变为从指定点 k 出发找经过所 有 6 条边回到 k 顶点的路径，由于所有顶点的度均为偶数，可以找到这样的路径。

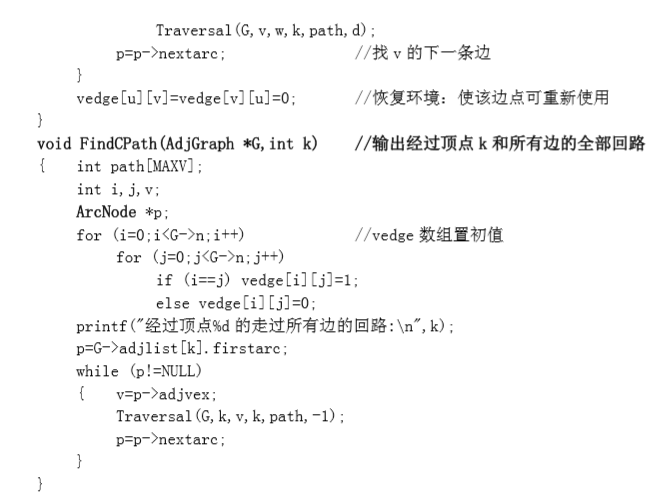
运行结果如下：











|  |
| --- |
| 过程函数代码已经给出，请写出主调函数的代码，附录于此： |
| #include <iostream>  #include<malloc.h>  #include"h.h"  using namespace std;  int vedge[MAXV][MAXV];  int main()  {  /\*int A[MAXV][MAXV]={{0,1,INF,INF,2},{1,0,INF,INF,3},  {INF,INF,0,6,4},{INF,INF,7,0,5},{2,3,4,5,0}};\*/  int A[MAXV][MAXV]={{0,1,INF,INF,1},{1,0,INF,INF,1},  {INF,INF,0,1,1},{INF,INF,1,0,1},{1,1,1,1,0}};  AdjGraph \*g;  int x=5;  int y=6;  CreateAdj(g,A,x,y);//创建邻接表G  DispAdj(g);  FindCPath(g,4);  } |
|  |

## 实验上传方法：

将源程序文件和本word文档（添加了相应的源程序和截图）上传。多个文件请将其放入一个文件夹压缩后上传。请于指定时间前上交，过时不补！