第8次作业

## 第6章习题

**6.6已知有向图G用邻接矩阵存储，设计算法以分别求解顶点vi的入度、出度和度。**

【算法描述】

void getDegrees(Graph G, int i, int &ind, int &outd)

{

//ind, outd 分别返回有向图中顶点 i 的入度和出度

int j;

ind=0;

outd=0;

//求顶点 i 的入度

for(j=1;j<=G.VerNum;j++)

{

if(G.AdjMatrix[i][j]>=1 && G.AdjMatrix[i][j] //兼顾有向图和网

outd++;

}

}

顶点的度 deg=ind+outd。

**6.7已知图G用邻接矩阵存储，设计算法以分别实现函数firstadj（G，v）和nextadj(G,v,w)。**

int firstAdj(Graph &G,int v)

{

int w;

for(w=1;w<=G.VNum;w++)

{

if((G.AdjMatrix[v][w]>=1) && (G.AdjMatrix[v][w] <INF)

return w;

}

return 0;

}

int nextAdj(Graph &G,int v,int w)

{

int k;

for(k=w+1;k<=G.VNum;k++)

{

if((G.AdjMatrix[v][k]>=1) && (G.AdjMatrix[v][k] <INF)

return k; //返回 v 的位于 w 之后的下一个邻接点 k

}

return 0; //不存在下一个邻接点，返回 0

}

**6.8设图G用邻接矩阵A[n+1,n+1]表示，设计算法以判断G是否是无向图。**

【算法描述】

bool isUdg(Graph G) {

int i,j;

for(i=1;i<=G.VerNum;i++)

for(j=1;j<=G.VerNum;j++)

{

if(G.AdjMatrix[i][j]!=G.AdjMatrix[j][i])

return false; //出现一点不对称，不是无向图，返回 false

}

//循环结束，邻接矩阵对称，再判定边数是否小于等于 n(n-1)/2 if(G.ArcNum<=G.VerNum\*(G.VerNum-1)/2)

return true; //边数小于等于 n(n-1)/2

else

return false;

}

**6.9已知图G用邻接表存储，设计算法输出其所有边或弧。（假设各表头指针在数组A[n+1]中）**

【算法描述】

void getEdge(Graph G,int E[MaxVerNum+1][MaxVerNum+1])

{

//A[][]为保存边信息的邻接矩阵，0 单元不用

EdgeNode \*p;

int i;

for(i=1;i<=G.VerNum;i++) //对图的顶点表进行循环

{

p=G.VerList[i].firstEdge; //获取顶点 i 的边链表头指针

while(p)

{

if(G.gKind==DG || G.gKind==DN || E[p->adjVer][i]==0)

{ //排除无向图网的重复边

E[i][p->adjVer]=1; //保存边数据

}

p=p->next;

}

}

}

**6.10对下列图，分别执行dfs(1)和dfs(5)，写出遍历序列，并构造出相应的dfs生成树。**

图6-1 题6.10图

(a)

(b)

**无向图**

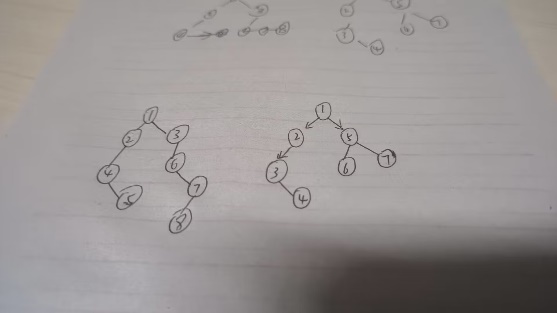
dfs(1)序列：1、2、4、5、3、6、7、8

dfs(5)序列：5、2、1、3、6、7、8、4

**有向图**

dfs(1)序列：1、2、3、4、5、6、7

dfs(5)序列：5、2、3、4、6、7



**6.11对下图的邻接表，不用还原出原图，请执行dfs(1)，写出遍历序列，并构造出相应的dfs生成树。**

图6-27 例6-6的邻接表图

info ptr

4

1

2

3

4

5

6

7

8

**∧**

3

2

**∧**

5

4

3

**∧**

7

**∧**

4

**∧**

8

**∧**

5

5

**∧**

8

4

**∧**

6

DFS(1)遍历序列：12348567。

DFS(1)生成树：

