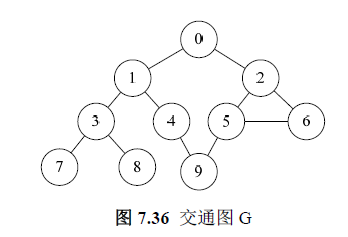
7.4 答：有n个顶点的无向连通图最多有 n(n-1)/2 条边，最少有 n-1 条边；

有n个顶点的有向强连通图最多有 n(n-1)条边，最少有 n 条边；

7.5:

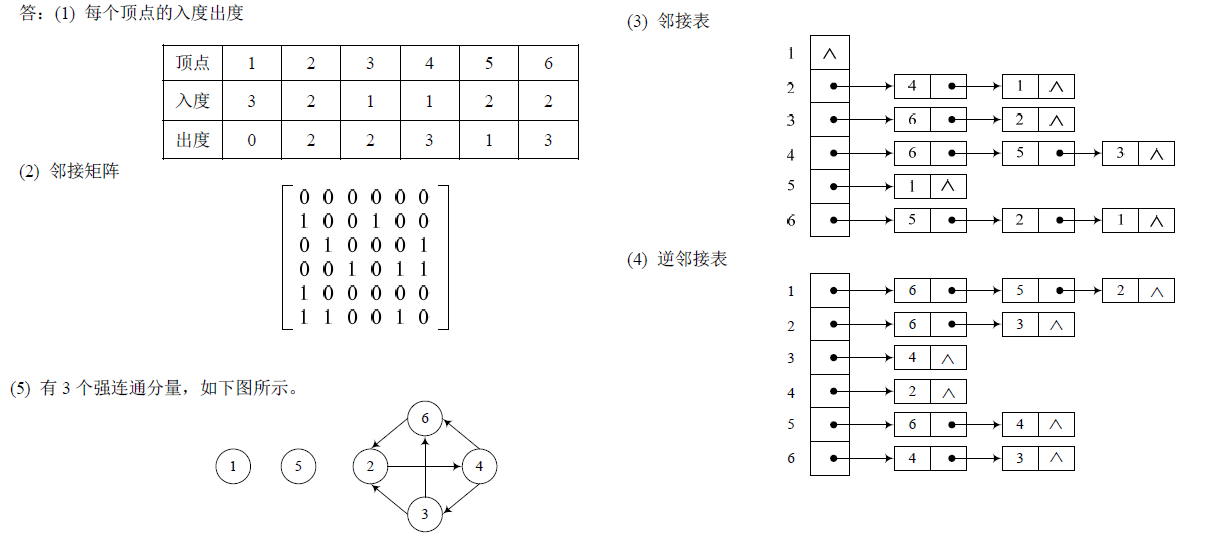


答：深度优先搜索得到的顶点访问序列：0、1、3、7、8、4、9、5、6、2；

广度优先搜索得到的顶点访问序列：0、1、2、3、4、5、6、7、8、9。

7.6 答：该有向图的拓扑排序序列为 3、1、4、5、2、6

7.8



**补充题：**

写出从图的邻接表表示转换成邻接矩阵表示的算法。

Void AdjlistToArray ( int A[][])

{

for ( i=1; i<=n; i++ )

for ( j=1; j<=n; j++ )

A[i,j]=0;

for ( i=1; i<=n; i++ ) {

p=adjlist[i].firstarc;

while ( p!=NULL) {

A[i,p->adjvex]=1;

P=p->nextarc;

}

}

}

**算法 7.11**

Status TopologicalSort(ALGraPhG) {

//有向图G采用邻接表存储结构。

//若G无回路，则输出G的顶点的一个拓扑序列并返回OK，否则ERROR。

FindInDegree(G,indegree); //对个顶点求入度indegree[0..vernum-1]

InitStack(s);

for(i=0;i<G.vexnum;++i)//建零入度顶点栈S

if (!indegree[i]) Push(S,i);//入度为0者进栈

count=0;//对输出顶点计数

while(!StackEmpty(S)) {

Pop(S,i); printf(i,G.vertices[i].data); ++count; //输出i号顶点并计数

for(p=G.vertices[i].firstarc; p; p=p->nextarc) {

k=p->adjvex; //对i号顶点的edge 领结点的入度减1

if (!(--indegree[k])) Push(S,k); //对入度减为0,则入栈

}//for

}//while

if (count<G.vexnum) return ERROR;//该有向图有回路

}//TopologicalSort { V6, V1, V4, V3, V5, V2 }

Void FindInDegree(Adjlist G, int indegree[max-vertex-num]);

{ int i; ArcNode \*p;

for (i=0; i<G.Vexnum; i++)

indegree [i]=0;

for (i=0; i<G.Vexnum; i++) {

p=G.Vertices[i].fistarc;

while (p!=NULL) {

indegree [p->adjvex]++;

p=p->nextarc;

} //while

}//for

}// FindInDegree