7.15

```
int InsertVex ( GraphG, elemtype v)
{ if( G. vexnum > Max_VNum-1)
     return error;
   G. vexs[ G.vexnum] = V;
   G.vexnum ++;
   return OK;
   }
```

```
int InsertArc (Graph G, elemtype V , elemtype w)
{      if(! G.arcs[i][j]. adjvex)
        { G.arcs[i][j]. adjvex =1;
         G. arcnumer = G.arcnumer +1;
     }   return OK; }
```

```
int  Delete (Graph , elemtype v)
  { number = G. vexnumber;
    if (i = LocateVex(G, v) <0)
      return  error;
        temp= G.vexs[i];
       G. vexs[i] = G. vexs(n];
       G. vexs[n] = temp;
   for(j=0; j<n; j++)
     { G.arcs[j][i] = G. arcs[j][n];
       G.arcs[ i ][j] = G.arcs[n][j];
       }
```

```
G.vexnumber --;
return OK;
}
```

```
int DeleteArc( Graph,G, elemtype V,
                               elemtype w)
{ if( G.arcs[i][j]. adjvex)
   { G.arcs[i][j]. adjvex=0;
      G. arcnumber --;
     }
   return OK; }
```

7.27.
```
int  V[MaX]
int  exist (Graph G , int i, int j, int k)
  { if ( i==j  && k==0)
        return  Ok ;
    else if ( k>0)
      { V[i]=1;
        for ( p=G.vertice[i].first ; p ; p=p->next)
          { flat = p -> adjvex;
            if(!v[flat] && exist (flat, j, k-1))
               return  Ok;
          }
      }
    V [i]=0;
    return  error; }
```

```
7.31.   void DFSfirst(Graph G, int i)
        { int j;
          for (p = G.vertexes[i] first; p; p=p-> next)
            { j = p->adjvex;
              if (!v[j])
               DFSfirst(G, j);
             }
              first[~~flat~~] = i;
              flat ++;
        }

     void DFSsecond(G, int i)
        int j;
        v[i]=1;
        for(p = G. vertexes[i].first; p; p->p-> next)
         { j= p->adjvex;
           if (!v[j]) DFSsecond(G,j);
         }
        }
```

for（V 中的每个 x）

```
void function (Graph G)
  flat =0;
  int i,j. number;
  for (i=0; i<number ;i++)
     v[i]=0;
  for(i=0;i<n;i++)
   { if (!v[i])
      DFSFirst(G,i);
for(i=n-1; i>=0 ;i--)
   {     j=finish[i];
      if (!v[j])
        { DFSSecond (G,i);
     }
  }
```

```
int  function (ALGraph G)
{
   Indegree ( G)   //求入度
   Initstack (s)
   for (i=0; i < G.vex ; i++)
     { if (! indegree [i])
          Push( s.i)      // 入度为0的进栈.
   temp = 0;
   while (! StackEmpty(s))   // 栈不空
     { pop(S,i);
       for (p= G.vertice[i].first ; p; p=p->next)
         { j = p -> adjnex;
           if (! indegree[k])
             { indegree[k] --;
               Push(S,k); }
         }
     }
}
```