四、给定一个用邻接表保存的无向图G，设计算法int IsConnected(MGraph G)判断该图是否连通，若连通则返回1，否则返回0。邻接表定义如下：

```
typedef struct ArcNode{
  int adjvex;                                  //该弧所指向的顶点
  struct ArcNode *nextarc;                     //指向下一条弧的指针
}ArcNode;                                      //边结点
typedef struct{
  int data;                                    //顶点信息
  ArcNode *firstarc;                           //指向第一条依附该顶点的弧的指针
}VNode;                                        //顶点结点
typedef struct{
  VNode AdjList[MAXV];                         //顶点数组
  int vexnum, arcnum;                          //图的当前顶点数和弧数
}MGraph;
```

(1) 给出算法的基本设计思想。

(2) 采用C或C++语言描述算法，关键之处给出注释。

(3) 说明所设计算法的时间复杂度。

判断图是否连通只需要从任意结点出发遍历即可，遍历完成后如果存在结点未访问则说明该图不连通，否则说明该图连通。遍历可以选择广度优先遍历BFS或深度优先遍历DFS中的一种，下面使用BFS。

【参考答案】

(1) 使用队列q保存待访问结点，首先将0号元素入队，每次选择队头元素u出队访问并记录，然后将u相邻的顶点依次入队，循环执行直到队列访问完，BFS完成。接下来循环扫描每个顶点是否都已被访问，如果存在顶点未被访问，则说明不连通，返回0；否则连通，返回1。

(2) 代码如下：

```
int isConnected(MGraph G){
 bool visited[MAXV]={false};       //记录顶点是否被访问过
 int q[MAXV];                      //队列q存储待访问顶点
 int front=0, rear=0;              //队头front和队尾rear
 int u=0;                          //选择0号顶点作为起始顶点
 visited[u]=true;
 q[rear++]=u;
 while (front<rear){
     int u=q[front++];
     ArcNode* p=G.AdjList[u].firstarc;
     while (p!=NULL){
         int v=p->adjvex;          //v是u的相邻顶点
         if (!visited[v]){         //如果v未被访问则入队
             visited[v]=true;
             q[rear++]=v;
         }
         p=p->nextarc;
     }
 }
 for (int i=0; i<G.vexnum; i++)    //检查所有顶点是否访问过
     if (!visited[i])
         return 0;                 //图不连通
 return 1;                         //图连通
```