**阅读下列说明和C代码，回答问题1至问题3，将解答写在答题纸的对应栏内。**

**【说明】**

**对有向图进行拓扑排序的方法是：**

**（1）初始时拓扑序列为空；**

**（2）任意选择一个入度为0的顶点，将其放入拓扑序列中，同时从图中删除该顶点以及从该顶点出发的弧；**

**（3）重复（2），直到不存在入度为0的顶点为止（若所有顶点都进入拓扑序列则完成拓扑排序，否则由于有向图中存在回路无法完成拓扑排序）。**

**函数int\* TopSort(LinkedDigraph G)的功能是对有向图G中的顶点进行拓扑排序，返回拓扑序列中的顶点编号序列，若不能完成拓扑排序，则返回空指针。其中，图G中的顶点从1开始依次编号，顶点序列为v1，v2，„，vn，图G采用邻接表示，其数据类型定义如下：**

**#define MAXVNUM 50             /\*最大顶点数\*/**

**typedef struct ArcNode{          /\*表结点类型\*/**

**int adjvex;              /\*邻接顶点编号\*/**

**struct ArcNode \*nextarc;     /\*指示下一个邻接顶点\*/**

**}ArcNode;**

**typedef struct AdjList{      /\*头结点类型\*/**

**char vdata;                /\*顶点的数据信息\*/**

**ArcNode \*firstarc;         /\*指向邻接表的第一个表结点\*/**

**}AdjList;**

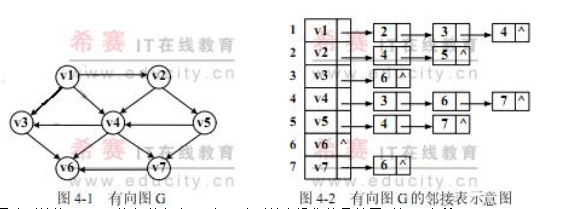
**typedef struct LinkedDigraph{       /\*图的类型\*/**

**int n;             /\*图中顶点个数\*/**

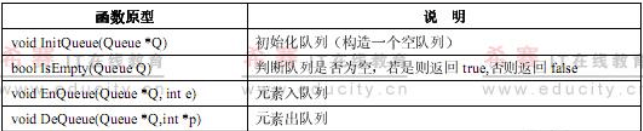
**AdjList Vhead[MAXVNUM];     /\*所有顶点的头结点数组\*/**

**}LinkedDigraph;**

**例如，某有向图G如图4-1所示，其邻接表如图4-2所示。**

****

**函数TopSort中用到了队列结构(Queue的定义省略)，实现队列基本操作的函数原型如下表所示：**

****

**【C代码】**

**int \*TopSort(LinkedDigraph G) {**

**ArcNode \*p;          /\*临时指针，指示表结点\*/**

**Queue Q; /\*临时队列，保存入度为0的顶点编号\*/**

**int k = 0;     /\*临时变量，用作数组元素的下标\*/**

**int j = 0, w = 0;    /\*临时变量，用作顶点编号\*/**

**int \*topOrder, \*inDegree;**

**topOrder = (int \*)malloc((G.n+1) \* sizeof(int));**

**/\*存储拓扑序列中的顶点编号\*/**

**inDegree = (int \*)malloc((G.n+1) \* sizeof(int));**

**/\*存储图G中各顶点的入度\*/**

**if (!inDegree || !topOrder) return NULL;**

**（1）                /\*构造一个空队列\*/**

**for ( j = 1; j <= G.n; j++ ) {     /\*初始化\*/**

**topOrder[j] = 0;      inDegree[j] = 0;**

**}**

**for (j = 1; j <= G.n; j++)     /\*求图G中各顶点的入度\*/**

**for( p = G.Vhead[j].firstarc; p; p = p->nextarc )**

**inDegree[p-> adjvex] += 1;**

**for (j = 1;j <= G.n; j++)/\*将图G中入度为0的顶点保存在队列中\*/**

**if ( 0 == inDegree[j] ) EnQueue(&Q,j);**

**while (!IsEmpty(Q)) {**

**（2）          /\*队头顶点出队列并用w保存该顶点的编号\*/**

**topOrder[k++] = w;**

**/\*将顶点w的所有邻接顶点的入度减1（模拟删除顶点w及从该顶点出发的弧的操作）\*/**

**for(p = G.Vhead[w].firstarc; p; p = p->nextarc) {**

**（3）-= 1;**

**if (0 ==（4）**

**) EnQueue(&Q, p->adjvex);**

**}/\* for \*/**

**}/\* while \*/**

**free(inDegree);**

**if (  （5）  )**

**return NULL;**

**return topOrder;**

**} /\*TopSort\*/**

**【问题1】（9分）**

**根据以上说明和C代码，填充C代码中的空（1）～（5）。**

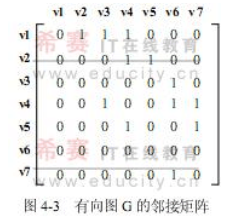
**【问题2】（2分）**

**对于图4-1所示的有向图G，写出函数TopSort执行后得到的拓扑序列。若将函数TopSort中的队列改为栈，写出函数TopSort执行后得到的拓扑序列。**

**【问题3】（4分）**

**设某有向无环图的顶点个数为n、弧数为e，那么用邻接表存储该图时，实现上述拓扑排序算法的函数TopSort的时间复杂度是（6）。**

**若有向图采用邻接矩阵表示（例如，图4-1所示有向图的邻接矩阵如图4-3所示），且将函数TopSort中有关邻接表的操作修改为针对邻接矩阵的操作，那么对于有n个顶点、e条弧的有向无环图，实现上述拓扑排序算法的时间复杂度是（7）。**

****

**答案：**

**问题1：（1）InitQueue（&Q）（2）DeQueue（&Q,&w）（3）inDegree[p->adjvex]**

**(4)inDegree[p->adjvex] (5)k<G.n 或 k!=G.n**

**问题2：队列方式：V1 V2 V5 V4 V3 V7 V6或1 2 5 4 3 7 6**

**栈方式：V1 V2 V5 V4 V7 V3 V6或1 2 5 4 7 3 6**

**问题3：（6） O(n+e) （7）O(n2)**