一、单选题（）

（ C ）1. 在一个图中，所有顶点的度数之和等于图的边数的_____倍。

A．1/2　　B. 1　　C. 2　　D. 4

（ B ）2. 在一个有向图中，所有顶点的入度之和等于所有顶点的出度之和的_____倍。

A．1/2　　B. 1　　C. 2　　D. 4

（ B ）3. 有 8 个结点的无向图最多有_____条边。

A．14　　B. 28　　C. 56　　D. 112

（ C ）4. 有 8 个结点的无向连通图最少有_____条边。

A．5　　B. 6　　C. 7　　D. 8

（ C ）5. 有 8 个结点的有向完全图有_____条边。

A．14　　B. 28　　C. 56　　D. 112

（ B ）6. 用邻接表表示图进行广度优先遍历时，通常是采用_____来实现算法的。

A．栈　　B. 队列　　C. 树　　D. 图

（ A ）7. 用邻接表表示图进行深度优先遍历时，通常是采用_____来实现算法的。

A．栈　　B. 队列　　C. 树　　D. 图

（ C ）8. 已知图的邻接矩阵，根据算法思想，则从顶点 0 出发按深度优先遍历的结点序列是

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

A．0 2 4 3 1 5 6
B. 0 1 3 6 5 4 2
C. 0 4 2 3 1 6 5
D. 0 3 6 1 5 4 2

建议：0 1 3 4 2 5 6

（ D ）9. 已知图的邻接矩阵同上题 8，根据算法，则从顶点 0 出发，按深度优先遍历的结点序列是

A. 0 2 4 3 1 5 6　　B. 0 1 3 5 6 4 2　　C. 0 4 2 3 1 6 5　　D. 0 1 3 4 2 5 6

（ B ）10. 已知图的邻接矩阵同上题 8，根据算法，则从顶点 0 出发，按广度优先遍历的结点序列是

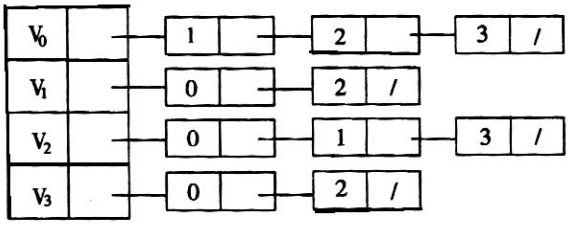
A. 0 2 4 3 6 5 1　　B. 0 1 3 6 4 2 5　　C. 0 4 2 3 1 5 6　　D. 0 1 3 4 2 5 6

（建议：0 1 2 3 4 5 6）

（ C ）11. 已知图的邻接矩阵同上题 8，根据算法，则从顶点 0 出发，按广度优先遍历的结点序列是

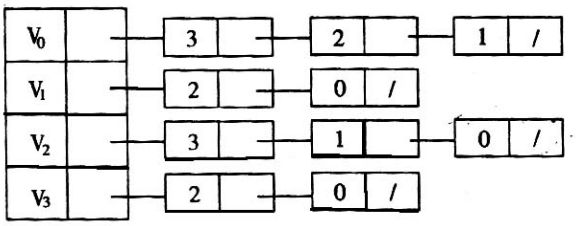
A. 0 2 4 3 1 6 5　　B. 0 1 3 5 6 4 2　　C. 0 1 2 3 4 6 5　　D. 0 1 2 3 4 5 6

（ D ）12. 已知图的邻接表如下所示，根据算法，则从顶点 0 出发按深度优先遍历的结点序列是



A. 0 1 3 2　　B. 0 2 3 1
C. 0 3 2 1　　D. 0 1 2 3

（ A ）13. 已知图的邻接表如下所示，根据算法，则从顶点 0 出发按广度优先遍历的结点序列是



A. 0 3 2 1　　B. 0 1 2 3
C. 0 1 3 2　　D. 0 3 1 2

（ A ）14. 深度优先遍历类似于二叉树的
A. 先序遍历　　B. 中序遍历　　C. 后序遍历　　D. 层次遍历

（ D ）15. 广度优先遍历类似于二叉树的
A. 先序遍历　　B. 中序遍历　　C. 后序遍历　　D. 层次遍历

（ A ）16. 任何一个无向连通图的最小生成树
A. 只有一棵　　B. 一棵或多棵　　C. 一定有多棵　　D. 可能不存在
（注，生成树不唯一，但最小生成树唯一，即边权之和或树权最小的情况唯一）

二、填空题（）

1. 图有__邻接矩阵__、__邻接表__等存储结构，遍历图有__深度优先遍历__、__广度优先遍历__等方法。

2. 有向图 G 用邻接表矩阵存储，其第 i 行的所有元素之和等于顶点 i 的__出度__。

3. 如果 n 个顶点的图是一个环，则它有__n__棵生成树。　（以任意一顶点为起点，得到 n-1 条边）

4. n 个顶点 e 条边的图，若采用邻接矩阵存储，则空间复杂度为__$O(n^2)$__。

5. n 个顶点 e 条边的图，若采用邻接表存储，则空间复杂度为__O(n+e)__。

6. 设有一稀疏图 G，则 G 采用__邻接表__存储较省空间。

7. 设有一稠密图 G，则 G 采用__邻接矩阵__存储较省空间。

8. 图的逆邻接表存储结构只适用于__有向__图。

9. 已知一个图的邻接矩阵表示，删除所有从第 i 个顶点出发的方法是__将邻接矩阵的第 i 行全部置 0__。

10. 图的深度优先遍历序列__不是__惟一的。

11. n 个顶点 e 条边的图采用邻接矩阵存储，深度优先遍历算法的时间复杂度为 $O(n^2)$；若采用邻接表存储时，该算法的时间复杂度为 O(n+e)。

12. n 个顶点 e 条边的图采用邻接矩阵存储，广度优先遍历算法的时间复杂度为 $O(n^2)$；若采用邻接表存储，该算法的时间复杂度为 O(n+e)。

13. 图的 BFS 生成树的树高比 DFS 生成树的树高 小或相等。

14. 用普里姆(Prim)算法求具有 n 个顶点 e 条边的图的最小生成树的时间复杂度为 $O(n^2)$；用克鲁斯卡尔(Kruskal)算法的时间复杂度是 $O(elog_2e)$。

15. 若要求一个稀疏图 G 的最小生成树，最好用 克鲁斯卡尔(Kruskal) 算法来求解。

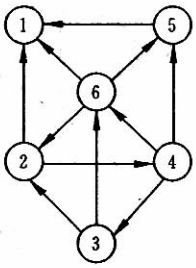
16. 若要求一个稠密图 G 的最小生成树，最好用 普里姆(Prim) 算法来求解。

17. 用 Dijkstra 算法求某一顶点到其余各顶点间的最短路径是按路径长度 递增 的次序来得到最短路径的。

18. 拓扑排序算法是通过重复选择具有 0 个前驱顶点的过程来完成的。

三、简答题（）

1.已知如图所示的有向图，请给出该图的:



（1） 每个顶点的入/出度；
（2） 邻接矩阵；
（3） 邻接表；
（4） 逆邻接表。

顶点	1	2	3	4	5	6
入度						
出度						

答案：

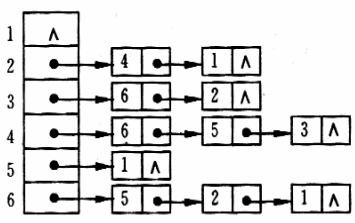
7.1 （1）

顶点	1	2	3	4	5	6
入度	3	2	1	1	2	2
出度	0	2	2	3	1	3

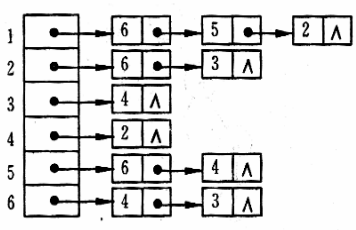
（2）邻接矩阵

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

（3）邻接表



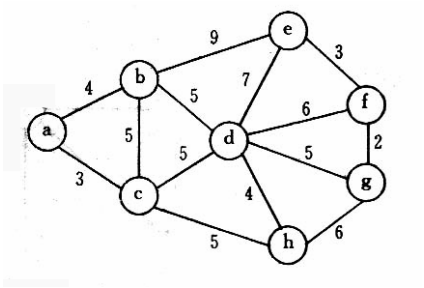
（4）逆邻接表



2.请对下图的无向带权图：

（1） 写出它的邻接矩阵，并按普里姆算法求其最小生成树；

（2） 写出它的邻接表，并按克鲁斯卡尔算法求其最小生成树。

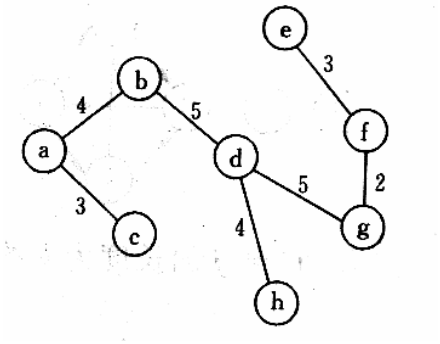


解：设起点为 a。可以直接由原始图画出最小生成树，而且最小生成树只有一种（类）！

邻接矩阵为：

$$\begin{bmatrix} 0 & 4 & 3 & \infty & \infty & \infty & \infty & \infty \\ 4 & 0 & 5 & 5 & 9 & \infty & \infty & \infty \\ 3 & 5 & 0 & 5 & \infty & \infty & \infty & 5 \\ \infty & 5 & 5 & 0 & 7 & 6 & 5 & 4 \\ \infty & 9 & \infty & 7 & 0 & 3 & \infty & \infty \\ \infty & \infty & \infty & 6 & 3 & 0 & 2 & \infty \\ \infty & \infty & \infty & 5 & \infty & 2 & 0 & 6 \\ \infty & \infty & 5 & 4 & \infty & \infty & 6 & 0 \end{bmatrix}$$

最小生成树→



PRIM 算法（横向变化）：

V	b	c	d	e	f	g	h	U	V-U
Vex lowcost	a 4	a 3	a ∞	a ∞	a ∞	a ∞	a ∞	{a}	{b,c,d,e,f,g,h}
Vex lowcost	a 4	0	c 5	a ∞	a ∞	a ∞	c 5	{a,c}	{b, d,e,f,g,h}
Vex lowcost	0	0	c 5	b 9	a ∞	a ∞	c 5	{a,c,b}	{d,e,f,g,h}
Vex lowcost	0	0	0	d 7	d 6	d 5	d 4	{a,c,b,d }	{e,f,g,h}
Vex lowcost	0	0	0	d 7	d 6	d 5	0	{a,c,b,d ,h }	{e,f,g }
Vex lowcost	0	0	0	d 7	g 2	0	0	{a,c,b,d ,h , g}	{ f,e }
Vex lowcost	0	0	0	f 3	0	0	0	{a,c,b,d ,h , g, f }	{e }
Vex lowcost	0	0	0	0	0	0	0	{a,c,b,d ,h , g, f, e }	{ }

邻接表为：

a	→	b	4	→	c	3										
b	→	a	4	→	c	5	→	d	5	→	e	9	^			
c	→	a	3	→	b	5	→	d	5	→	h	5	^			
d	→	b	5	→	c	5	→	e	7	→	f	6	→	g	5	→	h	4^
e	→	b	9	→	d	7	→	f	3	^								
f	→	d	6	→	e	3	→	g	2	^								

g	→	d	5	→	f	2	→	h	6	^
h	→	c	5	→	d	4	→	g	6	^

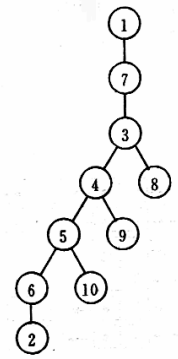
克鲁斯卡尔算法步骤
(按边归并,堆排序)：

先罗列：f---2---g　　a—3--c　　f—3—e　　a—4---b　　d—4—h
(a,b,c)　(e,f,g)　(d,h)　　取 b—5—d,　g—5--d　就把三个连通分量连接起来了。

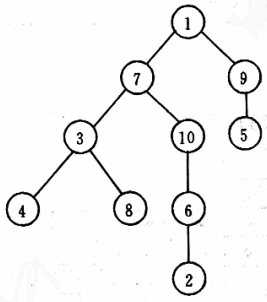
3.已知二维数组表示的图的邻接矩阵如下图所示。试分别画出自顶点 1 出发进行遍历所得的深度优先生成树和广度优先生成树。

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	1	0	1	0
2	0	0	1	0	0	0	1	0	0	0
3	0	0	0	1	0	0	0	1	0	0
4	0	0	0	0	1	0	0	0	1	0
5	0	0	0	0	0	1	0	0	0	1
6	1	1	0	0	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0	1
8	1	0	0	1	0	0	0	0	1	0
9	0	0	0	0	1	0	1	0	0	1
10	1	0	0	0	0	1	0	0	0	0

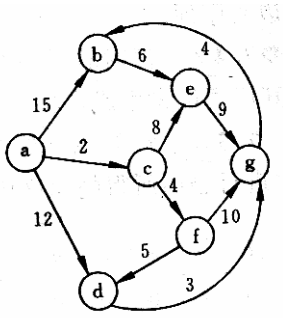
深度优先生成树



广度优先生成树



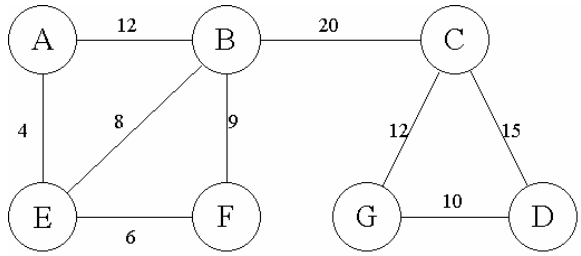
4.试利用 Dijkstra 算法求图中从顶点 a 到其他各顶点间的最短路径，写出执行算法过程中各步的状态。



解：最短路径为：（a,c,f,e,d,g,b）

Dist \ 终点	b	c	d	e	f	g	S (终点集)
K=1	15 (a,b)	**2 (a,c)**	12 (a,d)				{a,c}
K=2	15 (a,b)		12 (a,d)	10 (a,c,e)	**6 (a,c,f)**		{a,c,f}
K=3	15 (a,b)		11 (a,c,f,d)	**10 (a,c,e)**		16 (a,c,f,g)	{a,c,f,e}
K=4	15 (a,b)		**11 (a,c,f,d)**			16 (a,c,f,g)	{a,c,f,e,d }
K=5	15 (a,b)					**14 (a,c,f,d,g)**	{a,c,f,e,d,g}
K=6	**15 (a,b)**						{a,c,f,e,d,g,b}

四、给定下列网 G:

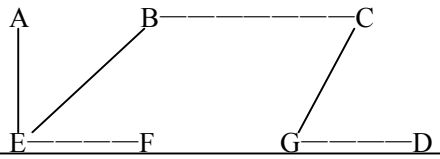


1 试着找出网 G 的最小生成树，画出其逻辑结构图；

2 用两种不同的表示法画出网 G 的存储结构图；

3 用 C 语言（或其他算法语言）定义其中一种表示法（存储结构）的数据类型。

解：1. 最小生成树可直接画出，如右图所示。



2. 可用邻接矩阵和邻接表来描述：

$$\begin{bmatrix} \infty & 12 & \infty & \infty & 4 & \infty & \infty \\ 12 & \infty & 20 & \infty & 8 & 9 & \infty \\ \infty & 20 & \infty & 15 & \infty & \infty & 12 \\ \infty & \infty & 15 & \infty & \infty & \infty & 10 \\ 4 & 8 & \infty & \infty & \infty & 6 & \infty \\ \infty & 9 & \infty & \infty & 6 & \infty & \infty \\ \infty & \infty & 12 & 10 & \infty & \infty & \infty \end{bmatrix}$$

描述存储结构的数据类型可参见教材或电子教案：

注：用两个数组分别存储顶点表和邻接矩阵

```
#define INFINITY   INT_MAX                    //最大值∞
#define MAX_VERTEX_NUM        20   //假设的最大顶点数（可取为 7）
Typedef   enum  {DG,  DN,  AG,AN  }    GraphKind;  //有向/无向图，有向/无向网
Typedef   struct   ArcCell{              //弧（边）结点的定义
VRType       adj;            //顶点间关系，无权图取 1 或 0；有权图取权值类型
  InfoType    *info;     //该弧相关信息的指针
}ArcCell, AdjMatrix [ MAX_VERTEX_NUM ] [MAX_VERTEX_NUM ];
Typedef struct{                                  //图的定义
VertexType vexs [MAX_VERTEX_NUM ] ;    //顶点表，用一维向量即可
AdiMatrix arcs:                        //邻接矩阵
```

邻接表为：

a	→	b	12	→	e	4	^					
b	→	a	12	→	c	20	→	e	8	→	f	9	^
c	→	b	20	→	d	15	→	g	12	^		
d	→	c	15	→	g	10	^					
e	→	a	4	→	b	8	→	f	6	^		
f	→	b	9	→	e	6	^					
g	→	c	12	→	d	10						

五、算法设计题（分）

1.编写算法，由依次输入的顶点数目、弧的数目、各顶点的信息和各条弧的信息建立有向图的邻接表。

解：Status Build_AdjList(ALGraph &G) //输入有向图的顶点数,边数,顶点信息和边的信息建立邻接表

```
{
  InitALGraph(G);
  scanf("%d",&v);
  if(v<0) return ERROR; //顶点数不能为负
  G.vexnum=v;
```

```
    scanf("%d",&a);
    if(a<0) return ERROR; //边数不能为负
    G.arcnum=a;
    for(m=0;m<v;m++)
        G.vertices[m].data=getchar(); //输入各顶点的符号
    for(m=1;m<=a;m++)
    {
        t=getchar();h=getchar(); //t 为弧尾,h 为弧头
        if((i=LocateVex(G,t))<0) return ERROR;
        if((j=LocateVex(G,h))<0) return ERROR; //顶点未找到
        p=(ArcNode*)malloc(sizeof(ArcNode));

        if(!G.vertices.[i].firstarc) G.vertices[i].firstarc=p;
        else
        {
            for(q=G.vertices[i].firstarc;q->nextarc;q=q->nextarc);
            q->nextarc=p;
        }
        p->adjvex=j;p->nextarc=NULL;
    }//while
    return OK;
}//Build_AdjList
```

2.试在邻接矩阵存储结构上实现图的基本操作：DeleteArc(G,v,w) ，即删除一条边的操作。（如果要删除所有从第 i 个顶点出发的边呢？ 提示： 将邻接矩阵的第 i 行全部置 0 ）

解：//本题中的图 G 均为有向无权图。

```
Status Delete_Arc(MGraph &G,char v,char w)//在邻接矩阵表示的图 G 上删除边(v,w)
    {
    if((i=LocateVex(G,v))<0) return ERROR;
    if((j=LocateVex(G,w))<0) return ERROR;
    if(G.arcs[i][j].adj)
    {
        G.arcs[i][j].adj=0;
        G.arcnum--;
    }
    return OK;
}//Delete_Arc
```

3.试基于图的深度优先搜索策略写一算法，判别以邻接表方式存储的有向图中是否存在由顶点 v_i 到顶点 v_j 的路径（i≠j）。注意：算法中涉及的图的基本操作必须在此存储结构上实现。

```
int visited[MAXSIZE]; //指示顶点是否在当前路径上
int exist_path_DFS(ALGraph G,int i,int j)//深度优先判断有向图 G 中顶点 i 到顶点 j
是否有路径,是则返回 1,否则返回 0
{
    if(i==j) return 1; //i 就是 j
```

```
  else
  {
    visited[i]=1;
    for(p=G.vertices[i].firstarc;p;p=p->nextarc)
    {
      k=p->adjvex;
      if(!visited[k]&&exist_path(k,j)) return 1;//i 下游的顶点到 j 有路径
    }//for
  }//else
}//exist_path_DFS
```

解 2：（以上算法似乎有问题：如果不存在路径，则原程序不能返回 0。我的解决方式是在原程序的中引入一变量 level 来控制递归进行的层数。具体的方法我在程序中用红色标记出来了。）

```
int visited[MAXSIZE]; //指示顶点是否在当前路径上
int level＝1;//递归进行的层数
int exist_path_DFS(ALGraph G,int i,int j)//深度优先判断有向图 G 中顶点 i 到顶点 j
是否有路径,是则返回 1,否则返回 0
{
  if(i==j) return 1; //i 就是 j
  else
  {
    visited[i]=1;
    for(p=G.vertices[i].firstarc;p;p=p->nextarc，level--)
    { level++;
      k=p->adjvex;
      if(!visited[k]&&exist_path(k,j)) return 1;//i 下游的顶点到 j 有路径
    }//for
  }//else
if (level==1)   return 0;
}//exist_path_DFS
```

附加题：采用邻接表存储结构，编写一个判别无向图中任意给定的两个顶点之间是否存在一条长度为 k 的简单路径的算法。

（注 1：一条路径为简单路径指的是其顶点序列中不含有重现的顶点。

注 2：此题可参见严题集 P207-208 中有关按“路径”遍历的算法基本框架。）

```
int visited[MAXSIZE];
int exist_path_len(ALGraph G,int i,int j,int k)//判断邻接表方式存储的有向图 G
的顶点 i 到 j 是否存在长度为 k 的简单路径
{
{
  if(i==j&&k==0) return 1; //找到了一条路径,且长度符合要求
  else if(k>0)
  {
```

```
        visited[i]=1;
        for(p=G.vertices[i].firstarc;p;p=p->nextarc)
        {
           l=p->adjvex;
           if(!visited[l])
              if(exist_path_len(G,l,j,k-1)) return 1; //剩余路径长度减一
        }//for
        visited[i]=0; //本题允许曾经被访问过的结点出现在另一条路径中
    }//else
    return 0; //没找到
}//exist_path_len
```