1996年试题参考答案

实做题

1、

先序遍历：A B F C D E

中序遍历：F B A D C E

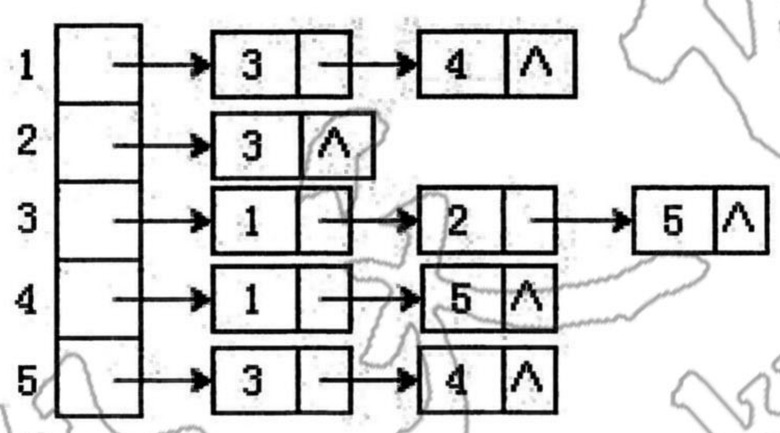
后序遍历：F B D E C A

2、

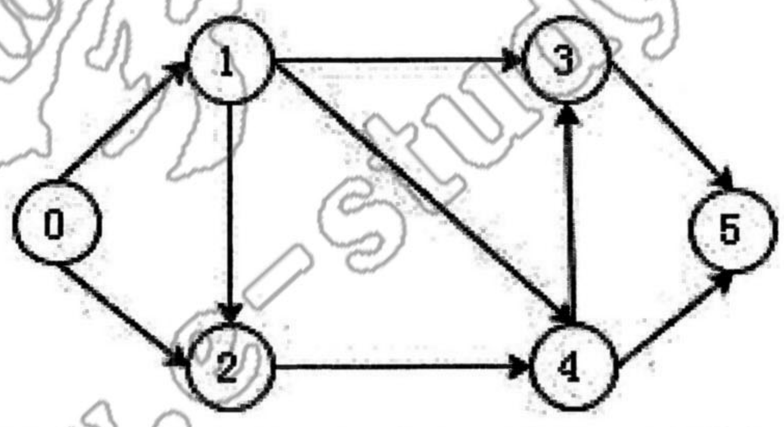
①邻接矩阵存储结构：

序号	1	2	3	4	5
1	0	0	1	1	0
2	0	0	1	0	0
3	1	1	0	0	1
4	1	0	0	0	1
5	0	0	1	1	0

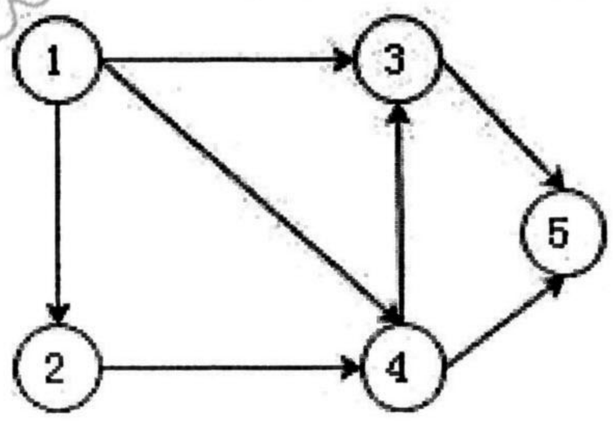
②邻接表存储结构



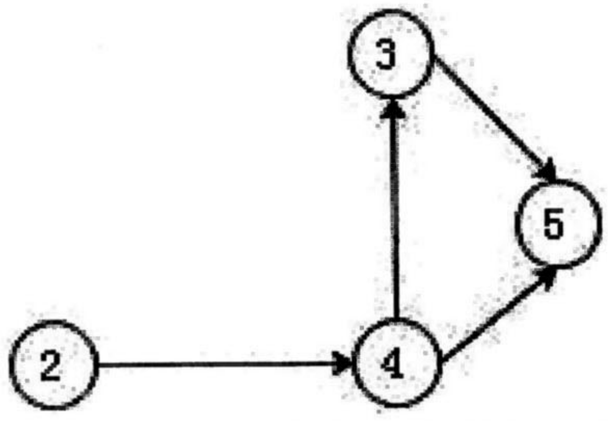
3、



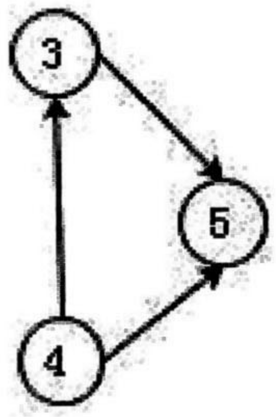
1）如图，开始时，寻找入度为0的的顶点：只有0，输出0并删去与0相关边，得到的图形如下：



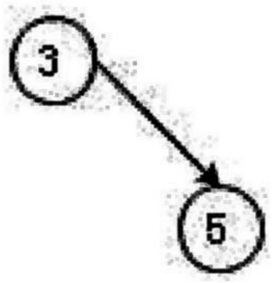
2）此时，继续寻找入度为0的顶点：只有1，输出1并删去与1相关边，得到的图形如下：



3）此时，入度为 0 的顶点只有 2，输出 2 并删去与 2 相关边，得到的图形如下：



4）此时，入度为 0 的顶点只有 4，输出 4 并删去与 4 相关边，得到的图形如下：



5）此时，入度为 0 的顶点只有 3，输出 3 并删去与 3 相关边，得到的图形如下：

6）输出顶点 5。此时全部结点都已经输出，拓扑完毕。

综上所述，图的拓扑序列为 0,1,2,4,3,5。

4、

链栈的结点定义：

```
typedef struct LNode
{
        int data; //不妨假设其数据类型为整型变量 int
        struct LNode *next;
}LNode;
```

（1）初始化

```
void InitStack(LNode *&l)//l 要发生变化，因此用引用型&
{
        l=(LNode*)malloc(sizeof(LNode));//头结点
        l->next=Null;//收尾操作
}
```

（2）进栈

```
void push(LNode *&l,int x)
{
```

```
    LNode *p;
    p=(LNode*)malloc(sizeof(LNode));
    p->next=NULL;
    //头插法
    p->data=x;
    p->next=l->next;
    l->next=p;
}
```

（3）退栈

```
int pop(LNode *&l,int &x)
{
    LNode *p;
    if(l->next==NULL)
        return 0;//出栈判空
    p=l->next;
    x=p->data;
    l->next=p->next;
    free(p);//释放存储空间
    return 1;
}
```

5、参考答案参见课本即可。

9、

结点定义：

```
typedef struct BSTNode
{
    KeyType key;
    struct BSTNode *Left;
    struct BSTNode *Right;
}BSTNode;
```

算法：

```
BSTNode * DeleteBST(BSTNode *b, KeyType x)
{
    if(b)
    {
        if(b->key==x)
            b = DelNode(b);
        else
            if(b->key>x)
                b->Left=DeleteBST(b->Left,x);
```

```
                else
                        b->Right=DeleteBST(b->Right,x);
        }
        return b;
}
BSTNode * DelNode(BSTNode *p)
{
        if(p->Left)
        {
                BSTNode *r=p->Left;//r 指向其左子树;
                BSTNode *prer=p->Left;//prer 指向其左子树;
                while(r->Right!=NULL)//搜索左子树的最右边的叶子结点 r
                {
                        prer=r;
                        r=r->Right;
                }
                p->key=r->key;
                if(prer!=r)//若 r 不是 p 的左孩子，把 r 的左孩子作为 r 的父亲的右孩子
                        prer->Right=r->Left;
                else
                        p->Left=r->Left; //否则结点 p 的左子树指向 r 的左子树
                free(r);
                return p;
}
else
{
                BSTNode *q=p->Right;//q 指向其右子树;
                free(p);
                return q;
        }
}
```