一、选择题

1. B

【详解】本题考查完全二叉树定义及性质，高度为 6 的完全二叉树最少结点数，应该是高度为 5 的满二叉树的结点数加 1，即 2^5-1+1=32，所以选 B。

2.C

【详解】本题考查的是图的定义，树的定义。假设有 m 棵树，即

顶点数：v1+v2+…+vm=n

边数： (v1-1)+(v2-1)+…(vm-1)=k 得出 m=n-k, 所以选 C.

3.D

【详解】本题考查的是图的邻接矩阵存储及对称矩阵的压缩存储。简单无向图的邻接矩阵是对称的，且对角线元素均是 0，故压缩存储只须存储下三角或上三角（均不包括对角线）即可。下标从 1 开始，B[k]=1+2+3+…+n-1=n(n-1)/2, 所以选 D

4.D

【详解】本题考查各种排序的算法思想，堆排序、冒泡排序、选择排序第一次元素即在相应的位置上，只有插入排序有可能如问题所述，所以选 D 。

5.C

【详解】本题考查 K 路归并的概念，通过 2 路归并可推导出答案，所以选 C 。

6.B

【详解】考查前缀码的概念，任意一个编码都不是其它任何一个编码的前缀，B 不满足。所以选 B 。

7.B

【详解】本题考查二叉树性质，二叉树上只有度为 0 和度为 2 的结点，这样要求最小结点的二叉树每层只能出现叶结点（h = 1 时）或每层只有两个结点，可导出答案，所以选 B。

8.D

【详解】本题考查树与二叉树的转换方法，根据转换原则，兄弟相连，所以选 D 。

9.C

【详解】本题考查的是二叉判定树的概念。在二叉判定树中失败结点均为叶结点，而内结点即给出的结点数 n 且度数均为 2 ，根据二叉树性质知 n0=n2+1，所以选 C。

10.D

【详解】本题考查的是堆得概念。根据题意知道，小根堆的形态是完全二叉树，根据小根堆的特性可知，最大的结点应在叶结点中，不可能在非叶结点上，只有 D 满足条件，所以选 D 。

二、填空题

11-1. O(n) 【详解】插入在第一位置之前，所有元素都要向后移动 1 次，共 n 次，故填 O(n)。

11-2. O(n) 【详解】若插入概率相同，则元素移动次数分别为 1,2,3，…，n 次，平均次数为（n+1）/2,故填 O(n)。

12-1. 三元组; 12-2. 十字链表【详解】稀疏矩阵的存储方案。

13-1. 0; 【详解】顺序扫描表达式遇到数字进栈，遇到运算符则取出两个数字计算，结果进栈，继续重复过程，可以得出结果。

13-2. 34X*+2Y*3/-【详解】对中缀表达式从左至右依次扫描，由于操作数的顺序保持不变，当遇到操作数时直接输出，设立一个栈用以保存操作符，扫描到操作符时，将操作符压入栈中，进栈的原则是保持栈顶操作符的优先级要高于栈中其他操作符的优先级；否则，将栈顶操作符依次退栈并输出，直到满足要求为止；遇到“（”进栈，当遇到“）”时，退栈输出直

到“)”为止。

14-1. 1； 14-2.n-1【详解】根据完全二叉树的定义知前 n-1 个结点有两个子树，只有 n 有一个左子树为 2n。故度数为 2 的结点数是 n-1，度为 1 的结点数为 1。

15-1. h+1； 15-2. h【详解】根据 B-树的定义知叶结点的位置，插入时要找到插入位置在树的最下层即树的高度。

三、简答题

16.①见图 1

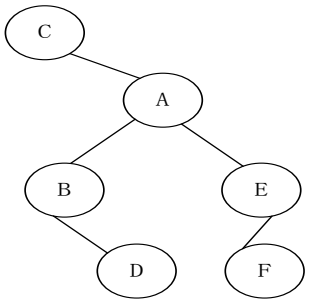

图 1 构造的二叉树

②方法是：由前序先确定 root，由中序可确定 root 的左、右子树。然后由其左子树的元素集合和右子树的集合对应前序遍历序列中的元素集合，可继续确定 root 的左右孩子。将他们分别作为新的 root，不断递归，则所有元素都将被唯一确定，问题得解。

③见图 2

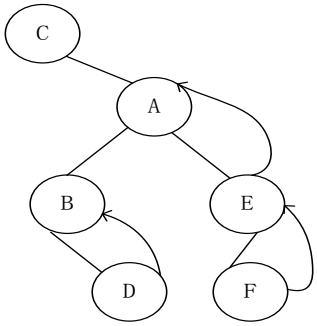

图 2 图 1 的后序线索树

17. ①见图 3

	a	b	c	d
a	0	1	3	5
b	∞	0	2	4
c	∞	6	0	2
d	∞	4	3	0

图 3 每对顶点的最短距离矩阵

② 偏心度为每一列的最大值，即 a:无穷大； b：6； c：3； d：5。

③ 最小的偏心度为 I 中心点即为 C。

四、算法设计

18.

（1）思路:初始时，找到数组的第一个数字和最后一个数字的位置 i 和 j。当两个数字的和大于输入的数字时，把 j 向右移动一个位置；当两个数字的和小于数字时，把 i 向左移动一个位置；当相等时，则找到。这样扫描的顺序是从数组的两端向数组的中间扫描。

（2）算法如下

```
bool FindSum（）
(
    int data[],
    unsigned int length,
    int sum,
    int& num1,
    int& num2
    bool found = false;
    if(length < 1)
        return found;
    int ahead = length - 1;
    int behind = 0;
    while(ahead > behind)
    {
        long long curSum = data[ahead] + data[behind];
        if(curSum == sum)
        {
            num1 = data[behind];
            num2 = data[ahead];
            found = true;
            break;
        }
        else if(curSum > sum)
            ahead --;
        else
            behind ++;
    }
    return found;
}
```

（3）从算法的基本设计思想，可知算法只对数组从两侧到中间进行一次扫描，因此算法的时间复杂度为 O（n）；另外，算法不需要额外的存储空间，所以空间复杂度是 O（1）。

19.

（1）思路：当森林（树）采用孩子—兄弟表示法存储时，若结点没有孩子，则他必是叶子结点，总的叶子结点个数是孩子子树上的叶子树和兄弟子树上叶结点个数之和。

（2）存储结构定义如下：

```
typedef struct node {
    datatype data；
    struct node* firstchild，rightsib；
```

} * Forest；
算法如下：

```
int LeavesCounter(Forest f )
{
    if (t==NULL) return 0;
else if (f->firstchild==NULL)
    return (1+ LeavesCounter（f->rightsib）);
else
    return eavesCounter (f->firstchild)+LeavesCounter(f->rightsib))；
}
```

（3）采用树（二叉树）的递归遍历算法，所以，时间和空间复杂度均与二叉树的遍历算法相同，分别为 O（n）和 O（$\log_2 n$）。