2012-2013 山大软件数据结构期末试题(真题)回顾

一、简答题。

1.插入排序、选择排序、冒泡排序、基数排序、堆排序的算法中其比较次数与初始数据集顺序无关的是？请说明理由。

2.已知待散列的线性表为（1,8,16,27,25,28 等数据），散列用的一维地址空间为 11，假定选用的散列函数是 H（K）= K mod 11，将其存入线性开型寻址散列和链表结构。

3.给一个树的层序遍历，中序遍历，写出其后序遍历。

4.给出二叉搜索树的层序遍历，问这个二叉搜索树是否是完全二叉树。

画出来看看

5.请说明广度优先搜索和深度优先搜索算法中所使用的堆栈、队列的作用。

因为深度优先需要无路可走时按照来路往回退，正好是后进先出

广度优先则需要保证先访问顶点的未访问邻接点先访问，恰好就是先进先出

二、应用题。

1.有学号 1-36 名学生，如果 i , j 两个学生住在同一个宿舍用（i,j）表示，集合 S={（1,2），(4,19）.....}如何求集合 S 中包含多少宿舍。

并查集(可用链栈)
可用图连通分支.

将(a,b)转换为点 a,b 的连接，然后判断最后形成的图有几个连通分支

2.构建霍夫曼树，求 ABCDEF 的霍夫曼代码

3.有 20 门课程，如果 i , j 两门课的学习顺序为先学 i，再学 j 那么用（ i , j ）表示，集合 S={(2,3),(4,6)....}，求至少要安排多少学期.

(i,j)i 为父亲，j 而儿子，将关系集合化成森林，看有几棵树

4.给出 ABCDE 消耗邻接矩阵，求 A 到个点的最短路径 dijkstra

三、算法程序题。

1.一个递增的链表，编写一个算法去除链表中的重复元素。例如，将（7,12,12,14,23）变为(7,12,14,23)，请写出算法思想和算法实现并分析算法的复杂性。

当前的节点和下一个节点，如果相同则删除下一个节点，这就成了链表删除节点的问题。

2.编写一个算法如何判断一个用二叉树链表存储的二叉树是否是最大堆，写出算法思想和算法实现。

最后一个题要注意的点：1，二叉树要么没有孩子，要么一定有左孩子。

2，注意最后一层，叶子一定都靠在左边，可以开一个树高的数组，来存储这一行前面是否有空出的，因为完全二叉树最后一行的叶子是要靠左。3，起码要符合基本的堆的条件：孩子比父亲要小或者大

1.插入排序、选择排序、冒泡排序、基数排序、堆排序的算法中其比较次数与初始数据集顺序无关的是？请说明理由。

无关：选择排序：每次都必须将余下的全部选出最大的或最小的

基数排序 不论初始集是否有序都需将每个数的每一位都余一遍

堆排序：堆排序每次取出根节点，然后合并两棵子树

比较次数为树高与顺序无关。

有关：插入排序，如果每次要插入的元素都比现有的最后一个元素大

那么比较一次就行，

冒泡排序 如果一次冒泡过程中元素没有一次交换则说明已有序

5.请说明广度优先搜索和深度优先搜索算法中所使用的堆栈、队列的作用。

广度优先 将一点相连所有点插入队列，一次弹出一个

可以按先进先出的顺序再次遍历下一轮的顶点

深度优先：每经过一个点将其推入堆栈当一个点无路可走

时可利用堆栈找到上个点继续深度搜索

3.有 20 门课程，如果 i，j 两门课的学习顺序为先学 i，再学 j 那么用（i，j）表示，集合 S={(2,3),(4,6)....}，求至少要安排多少学期.

将所有的(i,j)关系构建入一棵树

看这个树有几层.

1.一个递增的链表，编写一个算法去除链表中的重复元素。例如，将（7,12,12,14,23）变为（7,12,14,23），请写出算法思想和算法实现并分析算法的复杂性。

递增!!!看题!!!

```
typedef struct node{
        int element;
        node* next;
};
void delete(node* a){
    node *b=a;
    while(b!=null){
        if(b->element==b->next->element){
            b->next=b->next->next;
        }
        b=b->next;
    }
}
```

复杂度O(n)

2.编写一个算法如何判断一个用二叉树链表存储的二叉树是否是最大堆，写出算法思想和算法实现。

```
int count=0;    int a[1000];
```
将堆放入数组.

```
bool IsMaxHeap(node* root, int i){
    if(root!=null){
        a[i]=root->element;   count++;
    }
    if(root->lchild!=null){
        if(root->element<root->l->element){
            return false
        }else{
            IsMaxHeap(root->lchild, 2*i);
        }
    }
```

检查断点元素存在孩子

```
    if(root->rchild!=null){
        if(root->element<root->r->element){
            return false
        }else{
            IsMaxHeap(root->rchild, 2*i+1);
        }
    }
    for(int i=1; i<=count; i++){
        if(a[i]==null){
            return false;
        }
    }
    return true;
}
```

检查填放后数组有没有空位,如果有说明不是完全二叉树.