1~5 BDACA
6~10 DADDC

41

(1) BST 的最大高度为 n-1，当输入的 n 个结点值是以不递减(或者不递增)的方式输入时。

(2) BST 的中序遍历输出的序列为不递减序列。(证明参照 BST 树的特点，根节点值大于等于左子树的值，且小于等于右子树的值)

(3)

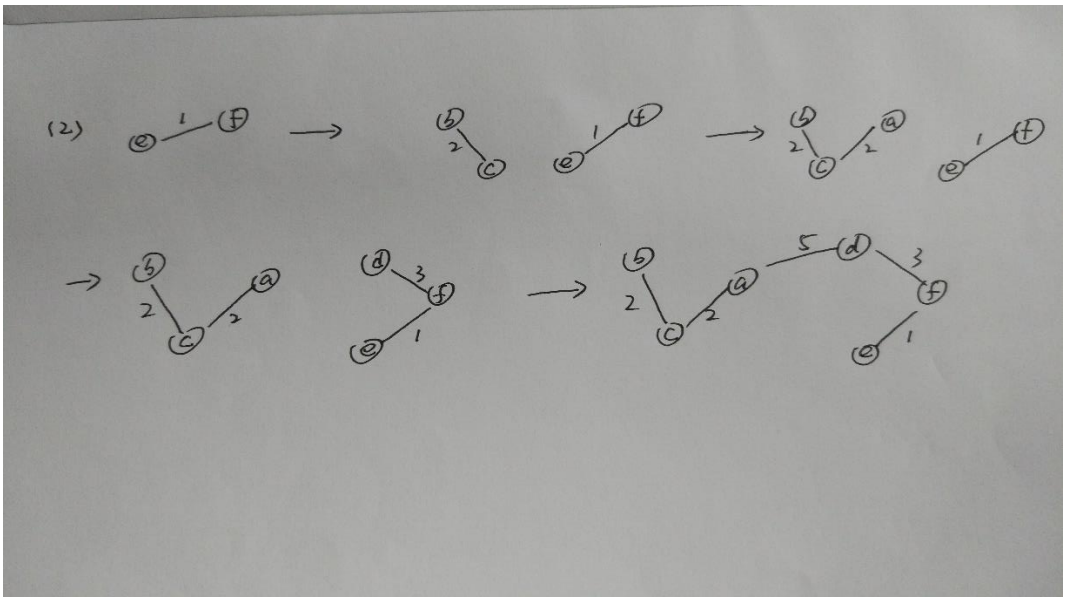
```
struct BitNode
{
        int data;
        BitNode* lchild;
        BitNode* rchild;
};

void inorder(BitNode* T)
{
        if(T)
        {
                inorder(T->lchild);
                if(T->data>=L && T->data<=R)
                    cout<<T->data<<" ";
                inorder(T->rchild);
        }
}
```

42

(1) 先构造一个只含 n 个顶点，而边集为空的子图，把子图中各个顶点看成各棵树上的根结点。之后，从网的边集 E 中选取一条权值最小的边，若该条边的两个顶点分属不同的树，则将其加入子图，即把两棵树合成一棵树；反之，若该条边的两个顶点已落在同一棵树上，则不可取，而应该取下一条权值最小的边再试之。依次类推，直到森林中只有一棵树，也即子图中含有 n-1 条边为止。时间复杂度为 O(elog^e)。

(2)

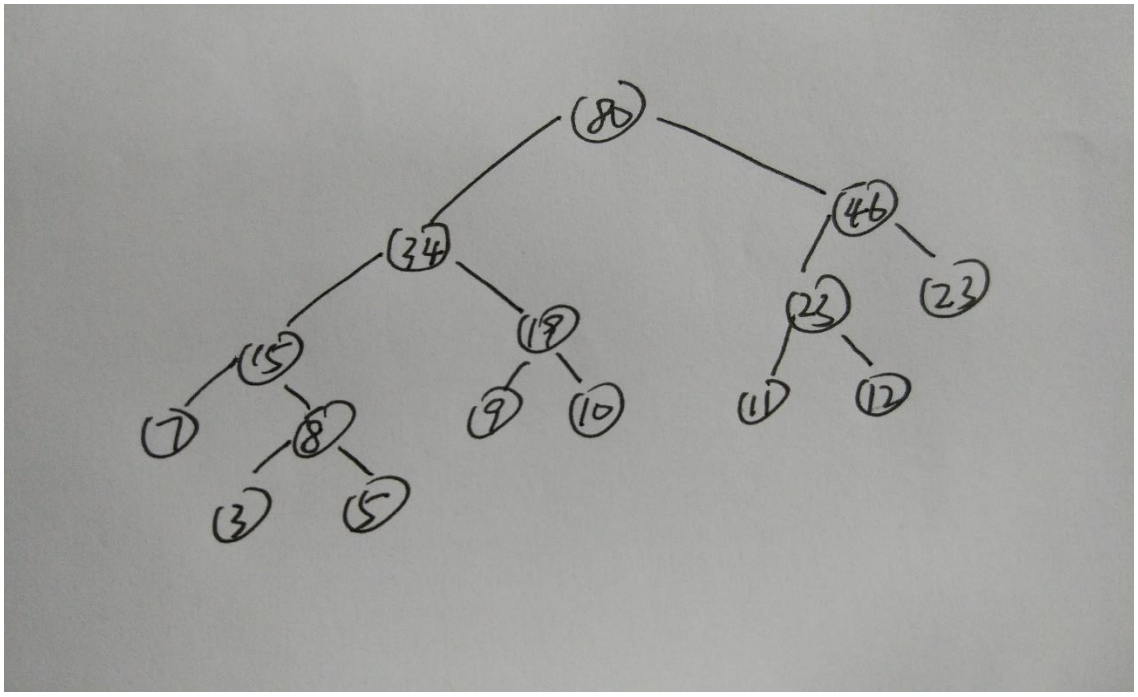


(3) Kruskal 算法每次选择边最小的两个结点。如果每条边的权重不一样，那么选择结点的时候的顺序就是唯一的，因此最后得到的最小支撑树也是唯一的。

43

(1) 该合并算法的主要思想是采用构建哈夫曼树的思想。每次取出整数序列中最小的两个数，然后合并为一个大的数，将合并出的大的整数放入原序列，同时删除序列中刚刚比较的两个整数。反复进行上述操作直到序列只剩一个数结束。

(2) 合并过程：首先 3 和 5 合并得到 8；然后将得到的 8 与 7 合并得到 15；之后将 9 和 10 合并得到 19；然后将 11 和 12 合并得到 23；之后将 15 和 19 合并得到 34；然后将 23 和 23 合并得到 46；最后将 34 和 46 合并得到 80。如下图。



(3) 合并的二叉树如上图。由于每次合并都是选择整数序列中最小的两个数，以此为叶节点合并出他们的根节点，然后将这个合并出来的根节点继续和其它整数合并，知道所有的整数合并完成，因此最小和次小的两个整数一定在最下层。