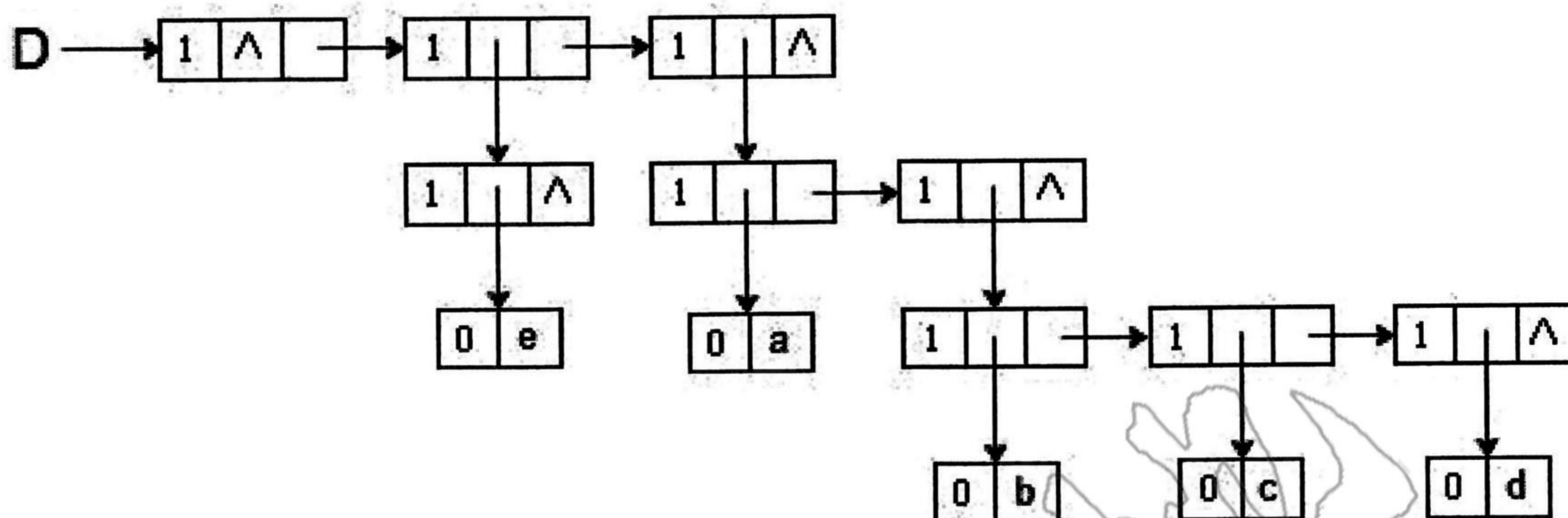
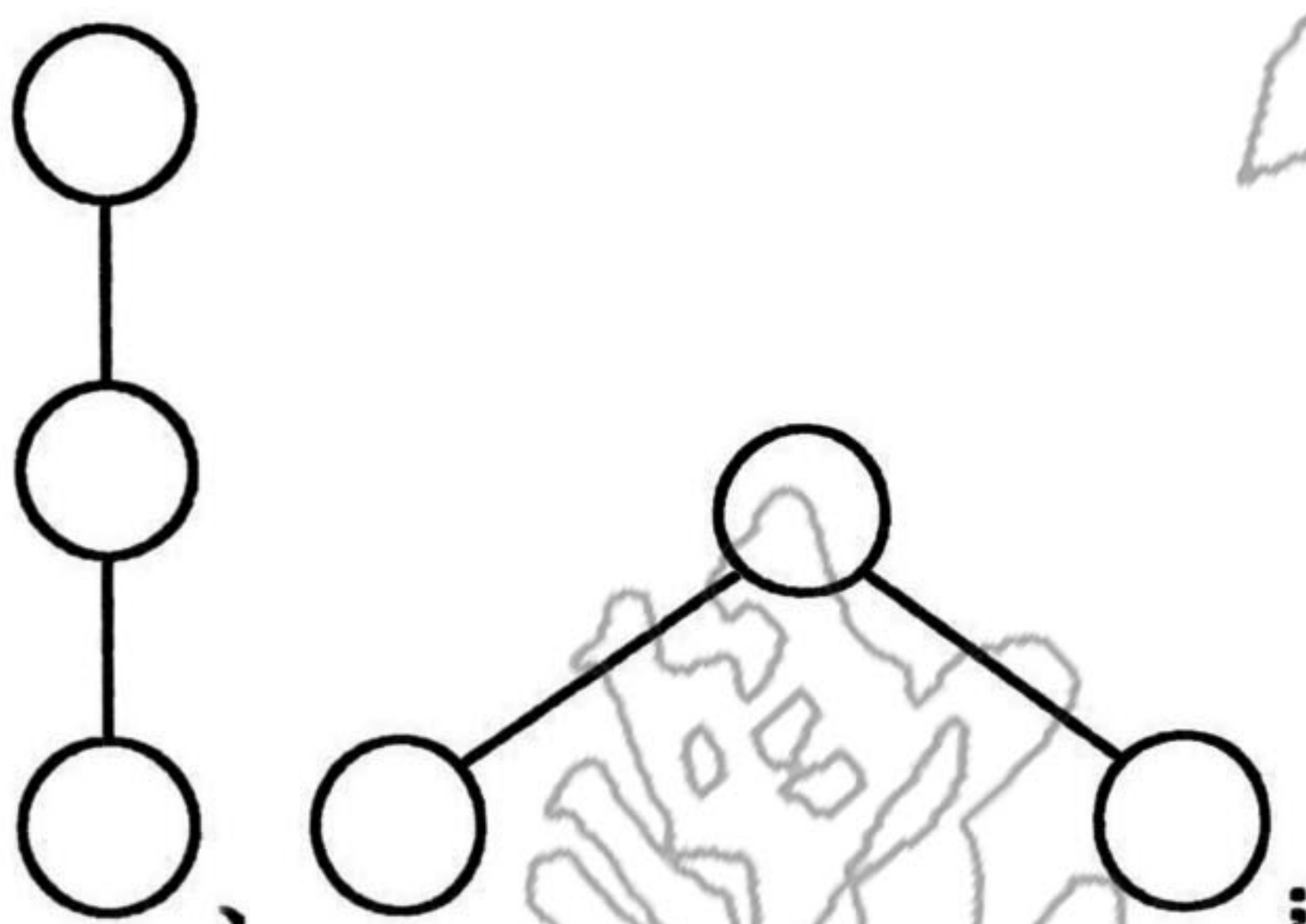


2002 年试题参考答案

1、








2、



(1) 三个结点的树:

三个结点的二叉树:

	1	2	3	4	5
树图形					

(2)

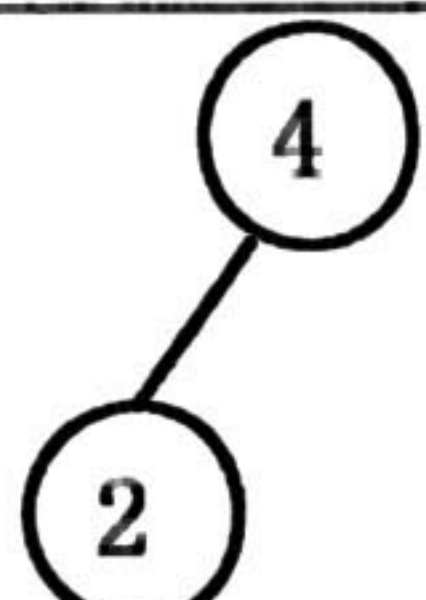
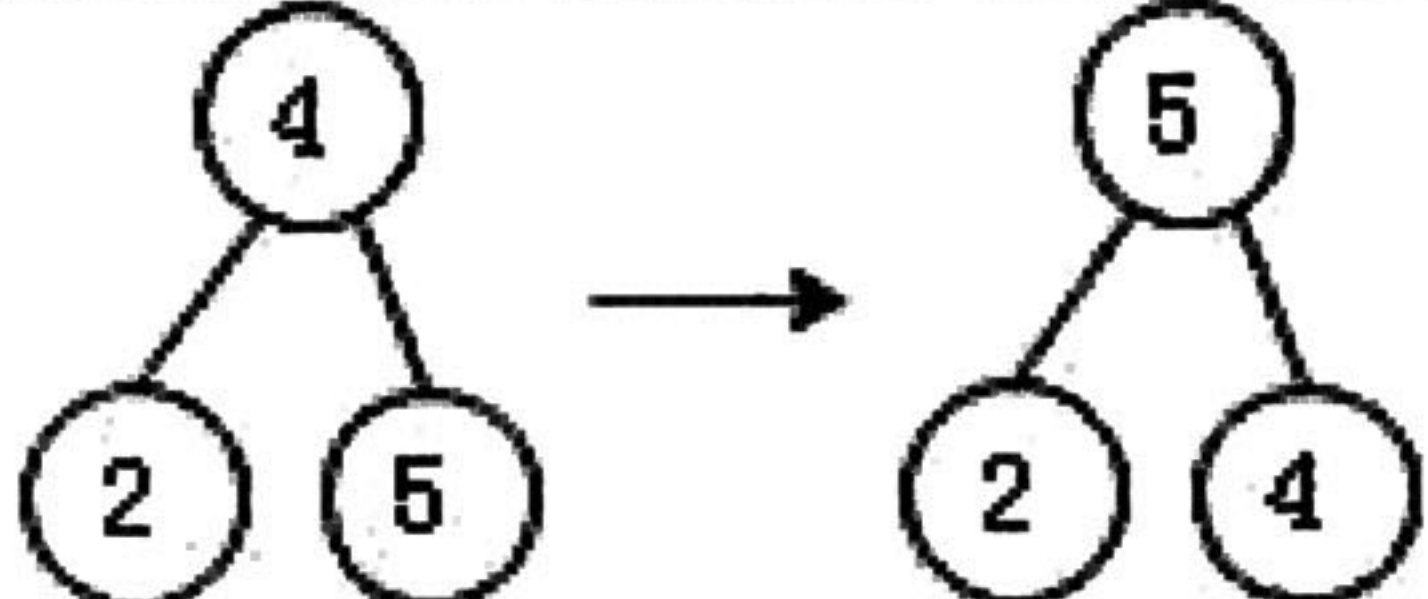
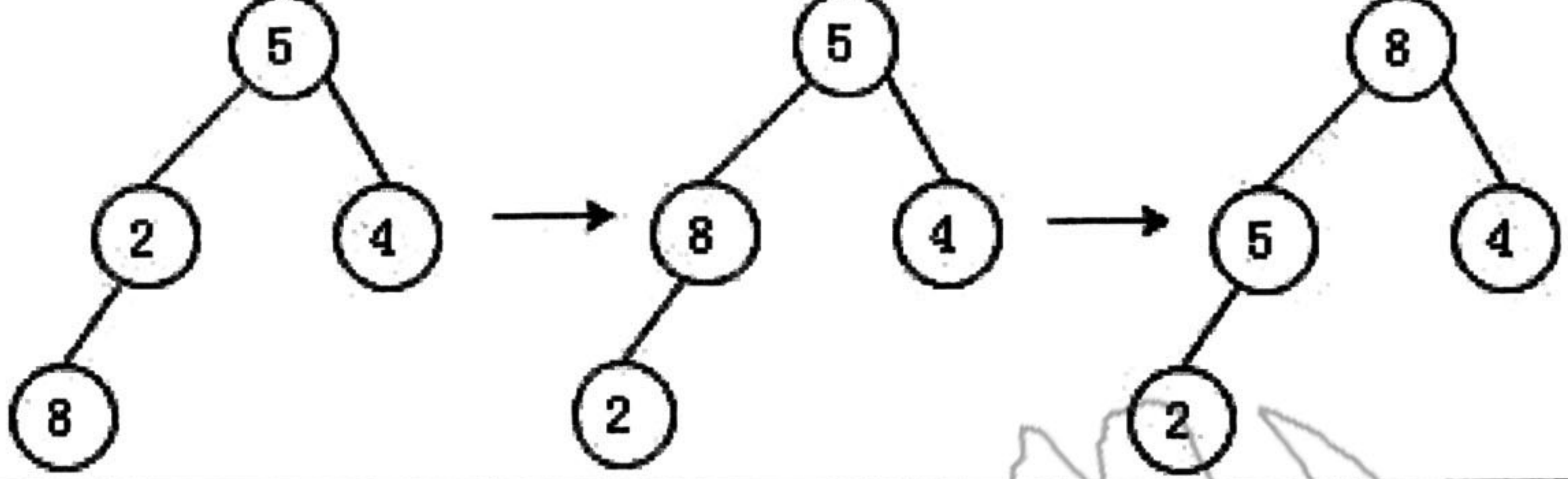
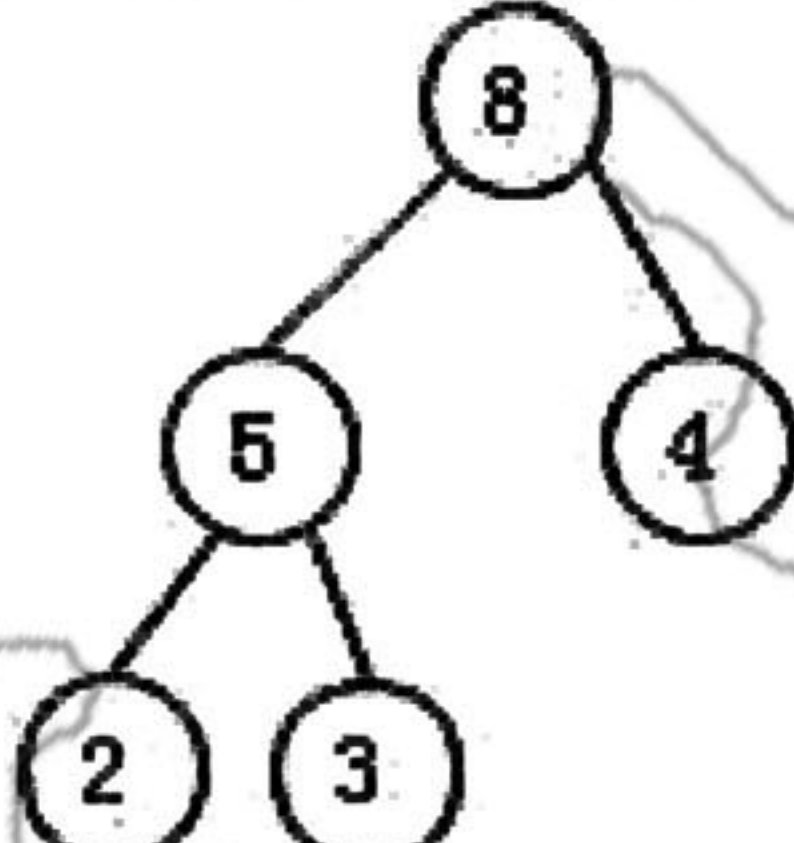
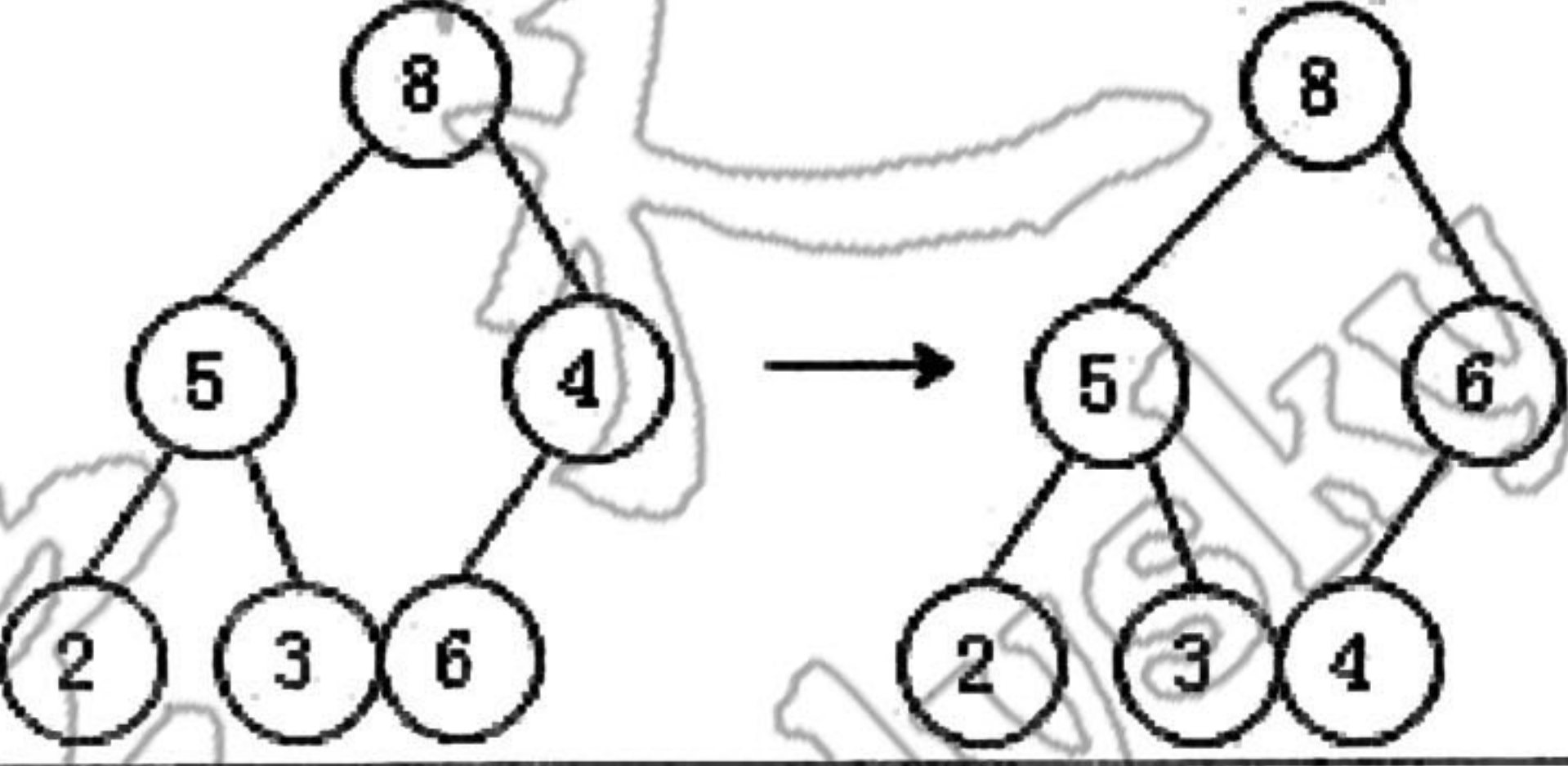
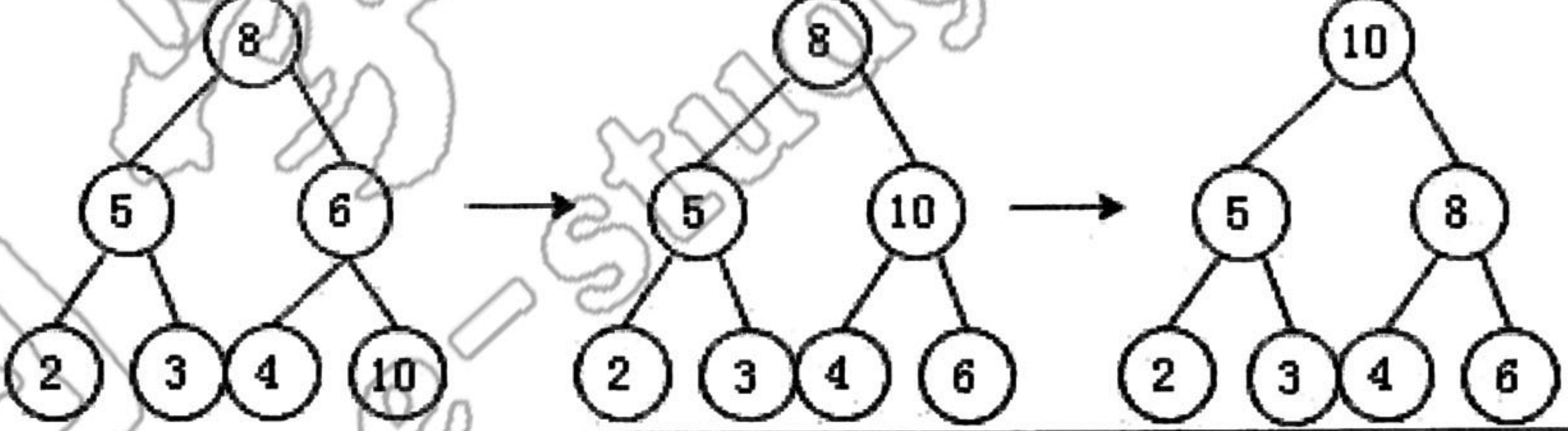
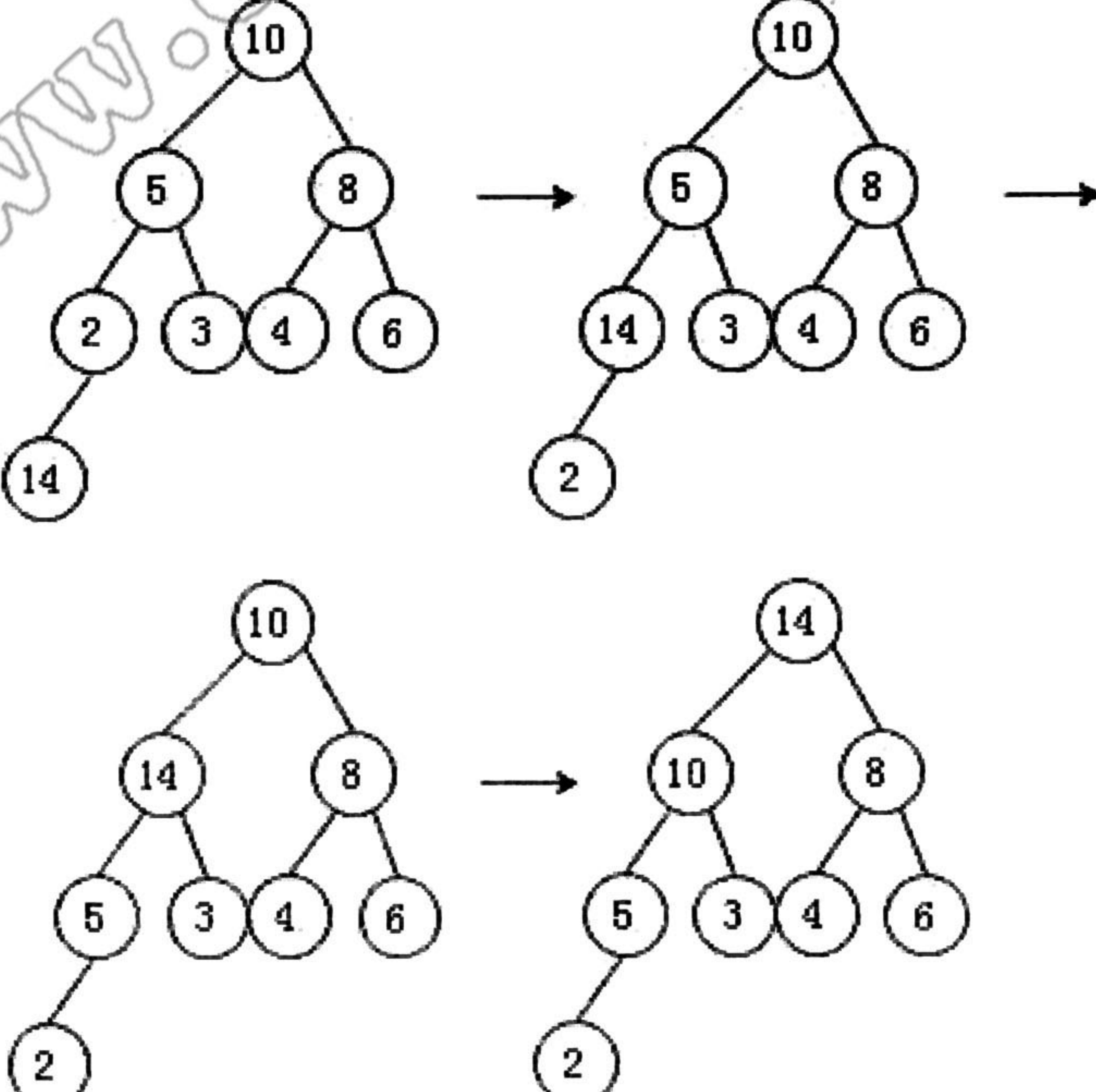
树编号	1	2	3	4	5
先序遍历	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3
中序遍历	3,2,1	2,3,1	2,1,3	1,2,3	1,3,2
后序遍历	3,2,1	3,2,1	2,3,1	3,2,1	3,2,1

3.

(1) 算法思想: 根据待排序序列调整成完全二叉树满足堆的格式, 那么堆顶元素就是相应的最大或者最小值, 除去堆顶元素将堆底元素放置在堆顶继续调整成堆, 这时堆顶元素就是第二小元素, 反复执行上述操作, 每执行一次, 有序序列元素多一个、无序序列元素少一个, 直至堆中没有元素, 即所成完全二叉树为空。

(2)

插入数据	堆变化
4	4

2	
5	
8	
3	
6	
10	
14	

本答案由学长友情提供，我们未核实其全部正确性。免费提供，仅供参考！
更多专业课视频和资料，请见：www.e-studysky.com；咨询QQ：3505993547

4、

(1)

```
#define Maxnum 100//顶点数目的最大值
typedef int VertexType;
typedef struct VNode
{
    VertexType data;
    int count;//统计顶点入度
    ArcNode *first;
}VNode;
int TopSort(ALGraph *G)
{
    int i,j,num=0;
    int stack[Maxnum],top=-1;//辅助栈初始化
    ArcNode *p;
    for(i=1;i<G->vexnum;i++)
        if(G->vertices[i].count==0)
            stack[++top]=i;//统计入度为 0 的点
    while(top!=-1)
    {
        i=stack[top--];
        num++;//记录拓扑排序出点的个数
        cout<<i<<" ";
        p=G->vertices[i].first;
        while(p)
        {
            //与刚出栈顶点的边被删除后修改相关顶点入度
            //同时找寻入度为 0 的点入栈
            j=p->adjvex;
            --(G->vertices[j].count);
            if(G->vertices[j].count==0)
                stack[++top]=j;
            p=p->next;
        }
    }
    if(num==G->vexnum)
        return 1;//所有顶点均输出，拓扑成功
    else
        return 0;
}
```

(2) 参照习题举例即可，尽量举一些简单的例子，不要难为自己。

5、

```
#include<iostream.h>
template <class Type>
class Queue
```


本答案由学长友情提供，我们未核实其全部正确性。免费提供，仅供参考！
更多专业课视频和资料，请见：www.e-studysky.com；咨询QQ：3505993547

{//循环队列的类定义

public:

Queue(int sz=10){ Initqueue(sz);}

void Initqueue(int sz);

~Queue(){ delete []Q;}

void enqueue(Type & item);

Type dlqueue();

void MakeEmpty() { length=0;}

//置空队列

int IsEmpty() const{ return length==0;}

//判队列空

int IsFull() const{ return length==m;}

//判队列满

private:

int rear,length; //队尾指针和队列长度

Type *Q; //存放队列元素的数组

int m; //队列最大可容纳元素个数

}

template <class Type>

Queue<Type>::Initqueue():m(sz),rear(m-1),length (0)

{

Q = new Type[m]; //创建队列空间

}

template<class Type>

void Queue<Type>::enqueue(Type &item)

{

if(IsFull())

return;

length++;

//长度加 1

rear=(rear+1)%m;

//队尾位置进 1

Q[rear]=item;

//进队列

}

template<class Type>

Type Queue<Type>::dlqueue()

{

if(IsEmpty())

//判断队列是否不空，空则出错处理

return;

length--;

//队列长度减 1

return Q[(rear-length+m)%m];//返回原队头元素值

}

6、

二叉链表存储结构:

typedef struct BTNode

{

ElemType data;

struct BTNode *lchild;

本答案由学长友情提供，我们未核实其全部正确性。免费提供，仅供参考！
更多专业课视频和资料，请见：www.e-studysky.com；咨询QQ：3505993547

```
    struct BTNode *rchild;  
} BTNode;
```

算法如下：

(1) 详见 98 真题->程序设计题->第 7 题

(2)

```
void Swap(BTNode *bt)  
{//采用后序遍历思想  
    if(bt==NULL)  
        return;  
    Swap(bt->lchild);  
    Swap(bt->rchild);  
    BTNode *temp;//从本句开始 4 行代码为对根结点的访问操作  
    temp=bt->lchild;  
    bt->lchild=bt->rchild;  
    bt->rchild=temp;  
}
```

二、程序阅读题

1、

运行结果：

```
son1::set(-1)called  
son2::set(0)called  
another one!  
son1::set(0)called  
son2::set(1)called  
Redraw the objects  
son1::set(-2)called  
son2::set(-1)called
```

2、

运行结果：

```
A's default constructor called.  
A's default constructor called.  
B's default constructor called.  
A's constructor called.  
A's constructor called.  
B's constructor called.  
B's destructor called.  
A's destructor called.  
A's destructor called.  
1,5,2  
B's destructor called.  
A's destructor called.  
A's destructor called.
```

3、

运行结果：

(1,2)

(6,9)

5,6

(6,9)

4、

运行结果：

Constructor called for ABCDABCDI

Before calling fun

Constructor called for ABABCDEFI

In fun()

Destructor called for xxxxxxxxx

After calling fun

Destructor called for xxxxxxxxx