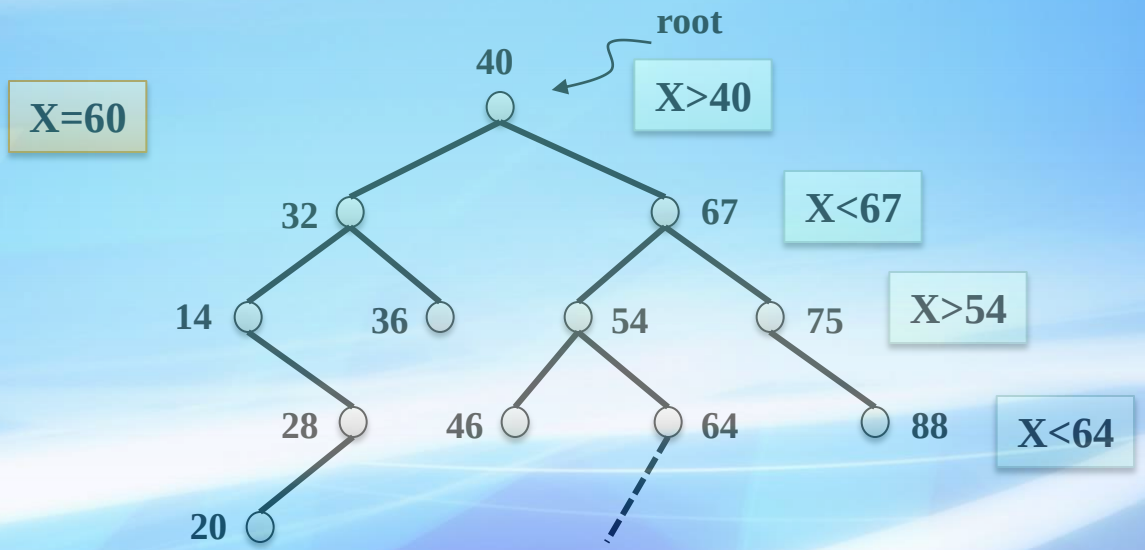
检索树（上）

《数据结构》

主讲人：李清



能否更快？

如何查找二叉树中的元素？

遍历 T(n)=O(n)

回顾

- 有序顺序表
- 二分查找

什么样的二叉树能用类似二分查找方法实现查找呢？

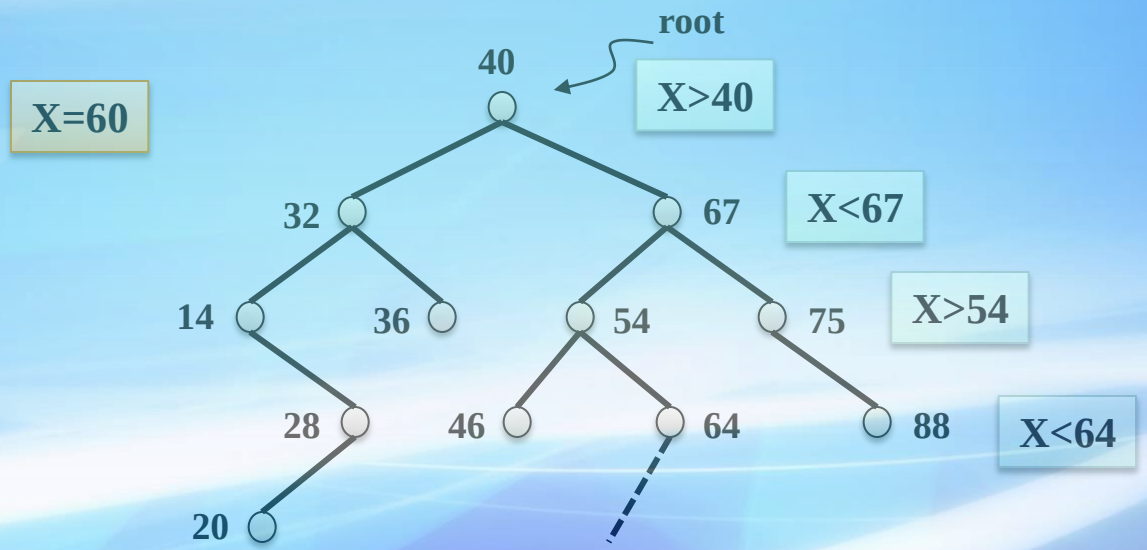
教学目标和要求

1. 能够准确描述检索树的基本特征；
2. 能够编程实现检索树的查找、插入、构造、删除算法（递归、非递归）。

检索树的基本概念和查找

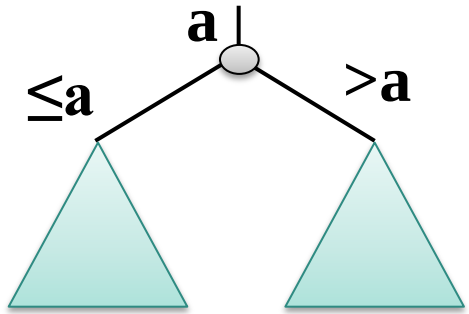
《数据结构》

主讲人：李清



检索树

也称排序树，二叉树中任何一个值为a的结点，其左子树上结点值均小于或等于a，其右子树上的结点值均大于a。



检索树“左小右大”，中序序列是“从小到大”

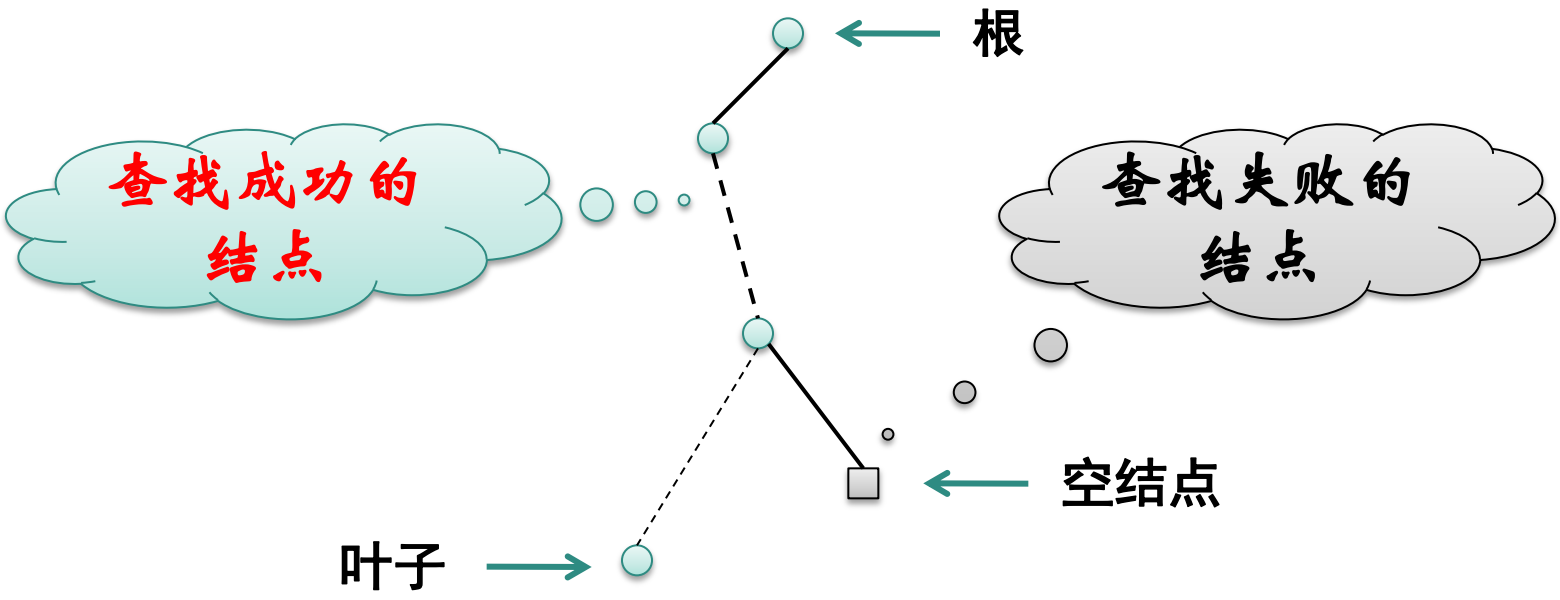
检索树如何进行查找呢？

检索树的查找----基本思想

只需沿着根到某个叶结点的一条路径搜索，使x与当前结点值比较：

- 遇到空树，结点x不在树中，查找失败；
- 如果相等，找到x，查找成功；
- 若x小于当前结点值，递归地查找左子树；
- 若x大于当前结点值，递归地查找右子树；

检索树的查找----基本思想



一条道走到底，不走回头路

算法：检索树的查找（递归）

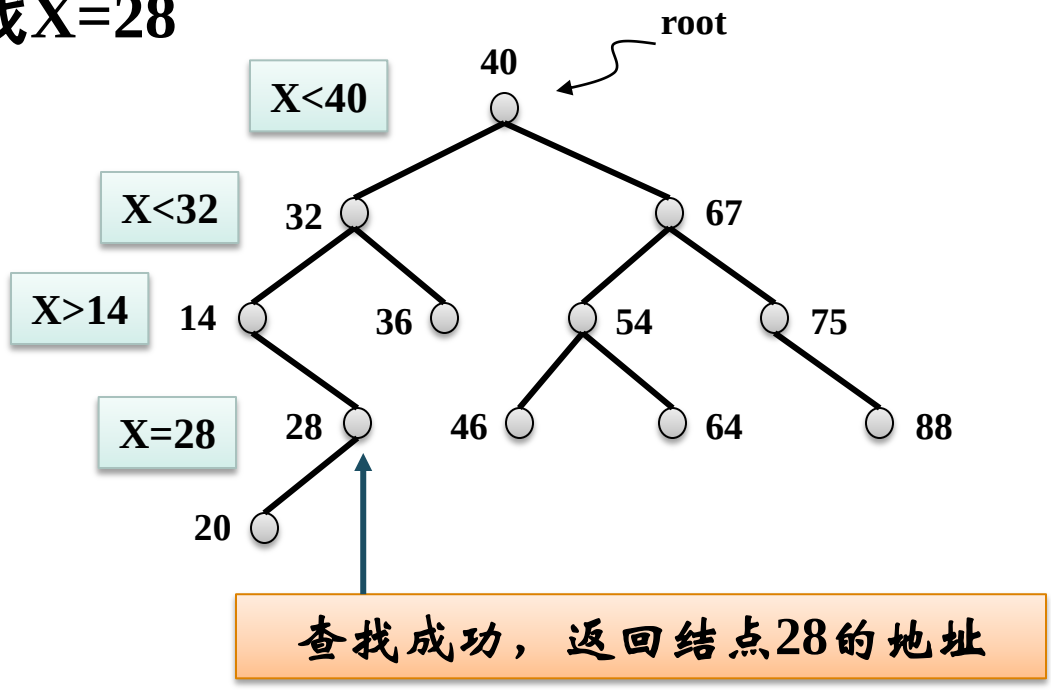
```
Bptr search(element_type x,Bptr p) //递归
{
   if(!p) return NULL;//查找失败
   if(x==p->data) return p ;//查找成功
   if(x<p->data) return search(x,p->Lson) ;//递归查找左子树
   else return search(x,p->Rson) ;//递归查找右子树
}
main()
{  …
   p=search(x,root);
   …
}
```

算法：检索树的查找（非递归）

```
Bptr search(element_type x,Bptr p) //非递归
{
   while(p){
       if(x==p->data) return p ;//查找成功
       if(x<p->data)p=p->Lson ;//查找左子树
       else p=p->Rson;//查找右子树
    }
    return NULL; ;//查找失败
}
main()
{   …
    p=search(x,root);
    …
}
```

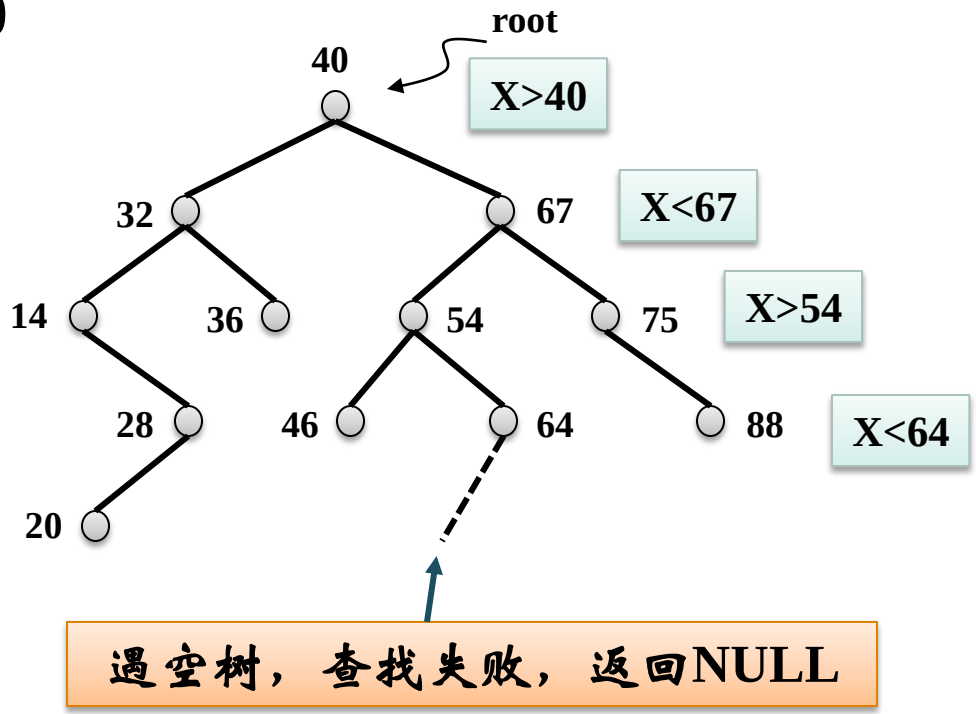
实例：检索树的查找

查找X=28



实例：检索树的查找

查找X=60



检索树的查找算法分析

用**查找长度**度量算法的时间复杂性

查找过程中与结点元素值进行比较的次数

检索树的查找算法分析

- 查找成功，查找长度等于结点x在树中的层数。
- 查找不成功，查找将终止于某个结点y，这个结点y必然至少缺少一个儿子，那么查找长度等于y的层数。

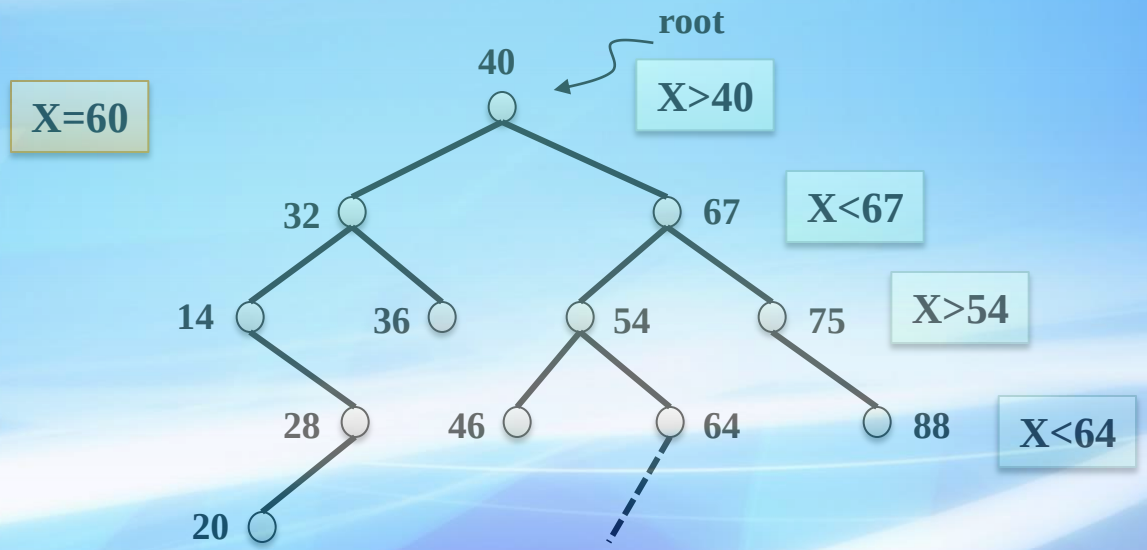
检索树查找长度不超过树的高度h

（$\log n \le h \le n$）

检索树的插入和构造

《数据结构》

主讲人：李清



问题

检索树是如何插入的？

检索树是如何构造的？

检索树的插入

找到插入结点的有序位置进行插入

基本原则：简单、保中序

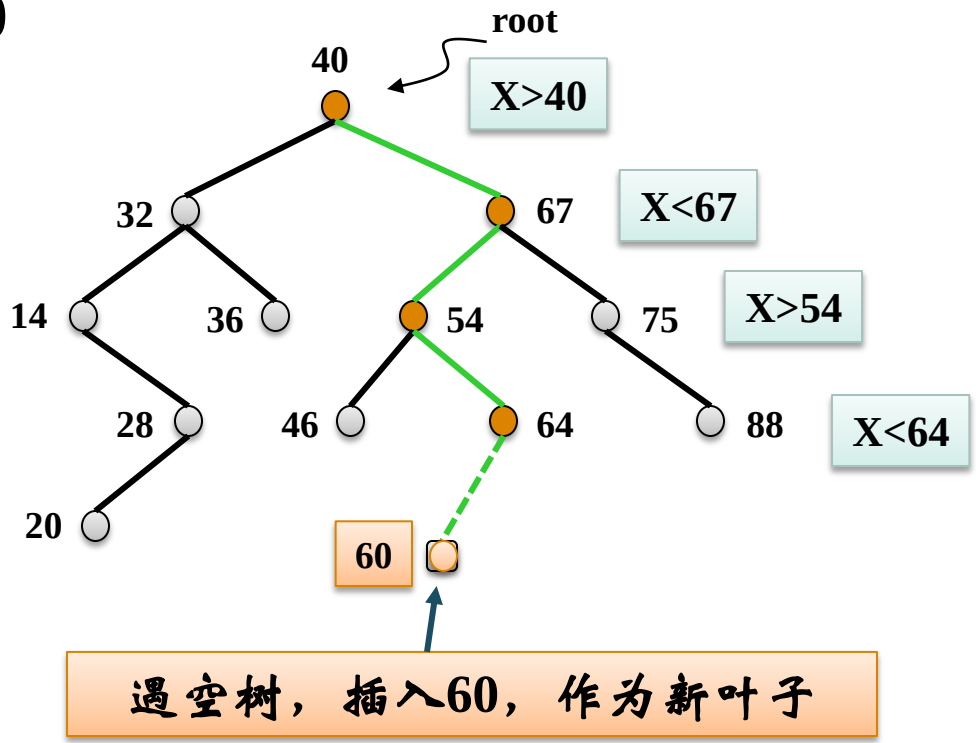
检索树的插入

从根结点起，将要插入的元素x与当前的结点比较，如果x小于或等于当前结点值，就向左搜索；如果x大于当前结点值，就向右搜索；

直至遇到空结点，就将x作为一片新叶插在这个空位置上。

实例：检索树的插入

插入X=60



算法：检索树的插入和构造（递归）

```
void insert(element_type x,Bptr &p)
            //新插入的结点总是作为新叶子
{
   if(!p)
   { p=newBnode;
     p->data=x;
     p->Lson=p->Rson=NULL;
     return;
   }
  if(x<=p->data) insert(x,p->Lson);
  else insert(x,p->Rson);
}//T(n)=O(h)
```

传引用？

递归查找插入位置

```
Bptr create()
{  Bptr root;
   root=NULL;
   scanf(“%d”,&x);
   while(x!=ZERO)
     //边输入边插入
     scanf(“%d”,&x);
   }
   return root;
}
```

算法：检索树的插入和构造（非递归）

```
void insert(element_type x,Bptr p)
{ Bptr q;
  while(p)
  { q=p;
        //q：插入结点的父结点
    if(x<=p->data)p=p->Lson;
    else p=p->Rson;
  }//查找插入位置
  p=newBnode;
  p->data=x;
  p->Lson=p->Rson=NULL;
  if(x<=q->data)q->Lson=p;
  else q->Rson=p;
}//T(n)=O(h)
```

记住插入点的父亲

```
Bptr create()
{  Bptr root=NULL;
   scanf("%d",&x);
   if(x!=ZERO)
   { root=newBnode;
     root->data=x;
     root->Lson=NULL;
     root->Rson=NULL;}
   else return NULL;
   scanf("%d",&x);
   while(x!=ZERO)
    { insert(x,root);
      scanf("%d",&x);
    }
   return root;
}//
```

首结点单独处理

实例：检索树的插入和构造

输入序列（0是输入结束标记）：