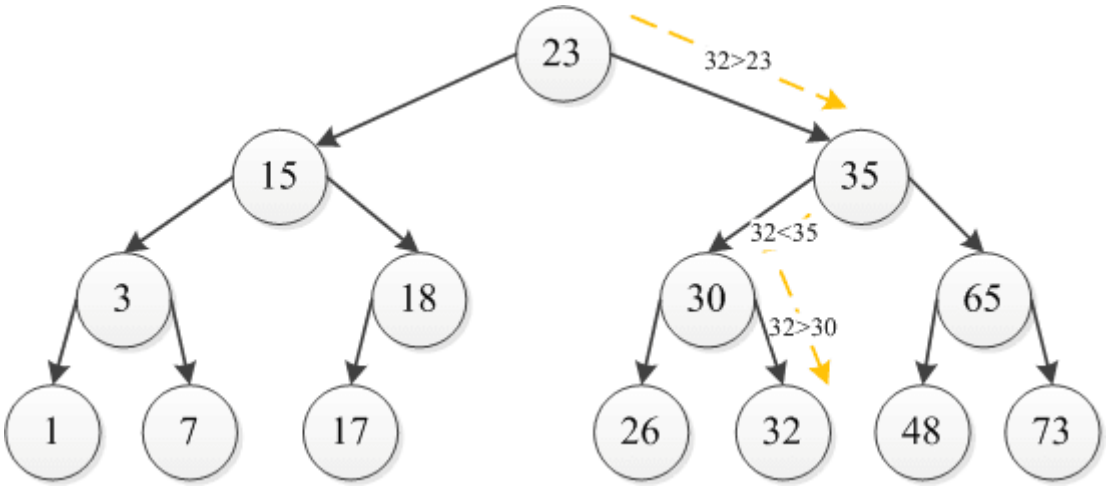
树形与线性表在搜索操作中的对比

【例】查找过程从根节点开始，比较待查找节点的值与根节点值的大小，如果小于，就递归查找左子树；如果大于，就递归查找右子树；如果等于，则查找过程结束。现在如图所示的二叉树中搜索 32。

1、若在二叉搜索树中搜索



具体操作如下：

32 > 23，查找右子树；

32 < 35，查找左子树；

32 > 30，查找右子树；

32 == 32，查找过程结束。

2、若在线性表中搜索

假设数据随机排列，现从第一个元素开始，依次对每个元素遍历，若遇到值相等，就返回该数据对应的位置。

【操作分析】

对于查找二叉树：

根据二叉树的特点：

1、若任意节点的左子树不空，则左子树上所有节点的值均小于它的根节点的值；

2、若任意节点的右子树不空，则右子树上所有节点的值均大于它的根节点的值；

3、任意节点的左、右子树也分别为二叉搜索树；

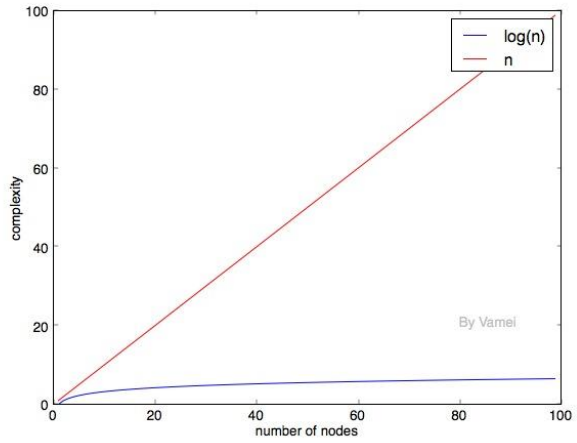
4、没有键值相等的节点。

因此，当总数据量为 n 时，二叉树深度 log2(n)，而在每个中深度深度查找中，查找均为顺序查找，对每次深度查找需查找 log2(n)次，因此总的时间复杂度 O(log(n))；

对于线性表：

每次查找均为顺序遍历，因此，最优情况为 1 次，最坏情况为 n 次，平均查找次数为 n/2 次，时间复杂度 O(n)。

【数据结构的作用】



如上图，数据量越大，两者效率差距越大，当 n=10000 时，两者运行时间能够相差近 1000 倍，且数据量越大，运行效率相差越大。

而搜索是很常用的操作，因此选择合适的数据结构去完成相应的工作，可以大大提高程序运行的效率。数据结构对程序的运行速度有着至关重要的作用。