|  |  |  |
| --- | --- | --- |
| 16.4.21 | 顺序查找 | 平均查找次数：(n+1)/2 |
| seqSearch1 直接从前向后比较  seqSearch2 优化，将a[0]设置为哨兵，从后往前比较，输出0为没找到 | | |

|  |  |  |
| --- | --- | --- |
| 16.4.22 | 二分查找 | 平均查找次数：log2(n+1)-1(n较大时) O(log2n) |
| binSearch 非递归  binSearchRecursion递归  主要就是判断low<=high时，进行key和a[mid]的比较，然后确定新的low和high | | |

|  |  |  |
| --- | --- | --- |
| 16.4.22 | 二叉排序树 | 平均查找次数：log2(n+1)-1(最好性能，等同于二分查找；最差为单支顺序查找(n-1)/2)。最好是平衡二叉树 |
| 1. 注意在进行树的构造和删除时需要使用\*&对节点直接修改，否则只用指针不会影响； 2. 删除比较麻烦，主要是根据其特性，用左孩子或左孩子的右孩子代替自己/右孩子或右孩子的左孩子代替自己。 | | |

|  |  |  |
| --- | --- | --- |
| 16.4.25 | 是否为二叉查找树后序遍历 | 递归 |
| 最后一位为根节点，寻找左子树和右子树，并对其分别判别，使用递归 | | |