查找技术：

1. 最简单的顺序查找。
2. 对顺序查找的优化，在[0]号设置一个哨兵，这样逆序查找，这样就每次都不用检查数组的下表是否发生了越界，因为不可能发生越界。这个所谓的哨兵就是带查找的值。

实践证明设置了哨兵的顺序查找会节省一半的时间开销。

案例

**int[0] = k;**

**i= n;**

**while(int[i]!=k){**

**i--;**

**}**

**return i;**

二分法查找

（1）非递归算法

**public** **static** **int** search(**int**[] sz,**int** k) {

**int** low = 0;

**int** high = sz.length - 1;

**int** middle;

**while** (low <= high) {

middle = (low + high) / 2;

**if** (k < sz[middle]) {

high = middle - 1;

} **else** **if** (k > sz[middle]) {

low = middle + 1;

} **else** {

**return** middle;

}

}

**return** -1;

}

**二叉链表的层序遍历**

**实现使用队列实现**

**将根节点入队**

**循环{**

**取出队首元素**

**访问数据**

**若该节点具有左孩子，则将其入队**

**若该节点具有右孩子，则将其入队**

**}**

**二叉链表的前序 中序 后序遍历**

将字符串中的运算符和数字分别读出的算法

**public** **static** ArrayList<String> praseString(String input) {

ArrayList<String> list = **new** ArrayList<String>();

**char**[] nb = input.toCharArray();

StringBuilder str = **new** StringBuilder();

**for** (**int** i = 0; i < nb.length; i++) {

**char** c = nb[i];

**if** ((c >= '0' && c <= '9') || c == '.') {

str.append(c);

} **else** {

**if** (str.length()>0) {

list.add(str.toString());

str.delete(0, str.length());

}

list.add(c + "");

}

}

**return** list;

}