KMP算法用于查找一个字符串是否包含一个子串

用两个指针分别指字符串和子串一个个index比较，符合的话直接跳到 下一个不符合的地方，不符合平移到下一个前缀相同的位置

1. 排序

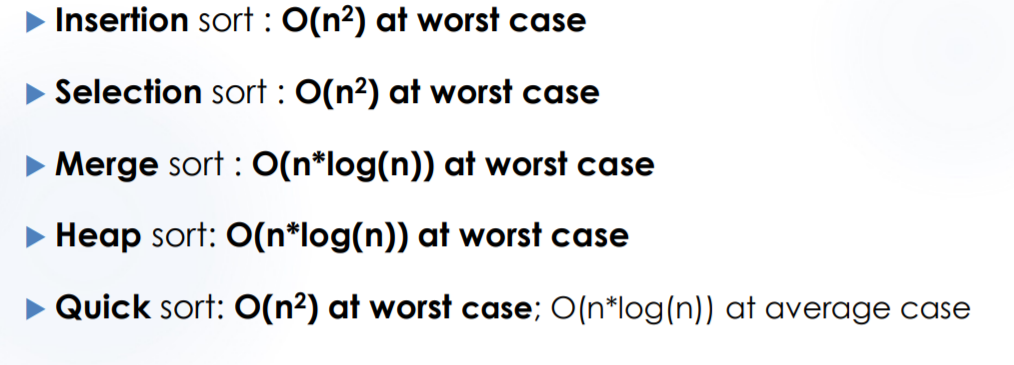


Comparison of sorting algorithm

1.Merge sort比Quick sort需要extra memory，因为每次call merge都会create新的array去放

2.selection sort的swap次数最少

3.对于mostly sorted array，insertion sort最好

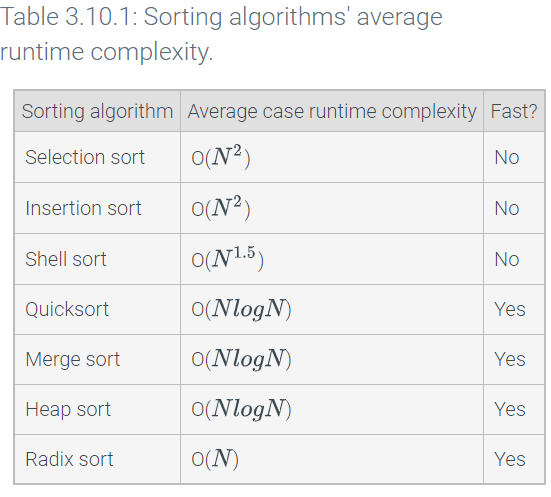


3.归并排序，计数排序，桶排序，基数排序不是in place，因为要extra memory

4.归并排序，计数排序，桶排序，基数排序，冒泡排序，插入排序是stable

Overview：

1.fast sorting algorithm是average runtime complexity为O(N\*logN)



Fast: Radix, Quick, Merge, Heap

1. 快速排序 O(nlogn) worst O(n^2):pivot选最大/最小

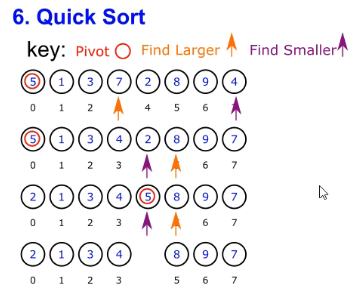
选quickSort因为不容易是O(n^2)

best/average O(n\*logn)--选pivot时候正好是中位数median

unstable

用了divide and conquer--分治法

先选择一个数作为pivot开始partition，把比pivot小的element放pivot左边，大的放右边，再进行recursive call来排序



1. 任意选择一个数作为pivot--5
2. 从左到右开始找比5大的数--7设置成Find Larger
3. 在7的右边从右往左找有没有比5小的数(如果没有则partition完成),把比5小的数设置成Find Smaller
4. 把Find Larger和Find Smaller的数互换位置
5. 之后重复上面从左往右找比5大的第一个数作为Find Larger--8，从8的右边从右往左找比5小的数--这次没有了--也就是Find Smaller cross over Find Larger
6. partition完成，准备开始recursive
7. 在recursive之前再做一次交换，把pivot和Find Smaller的element互换
8. 开始recursive call on 2 1 3 4，另一个recursive call on 8 9 7--对2134设置pivot重复上面步骤



1. 归并(merge)排序 O(nlogn) worst O(nlogn) 稳定

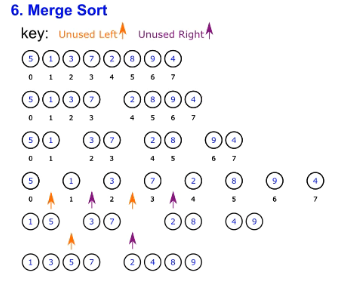
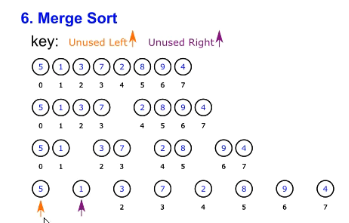
要额外空间

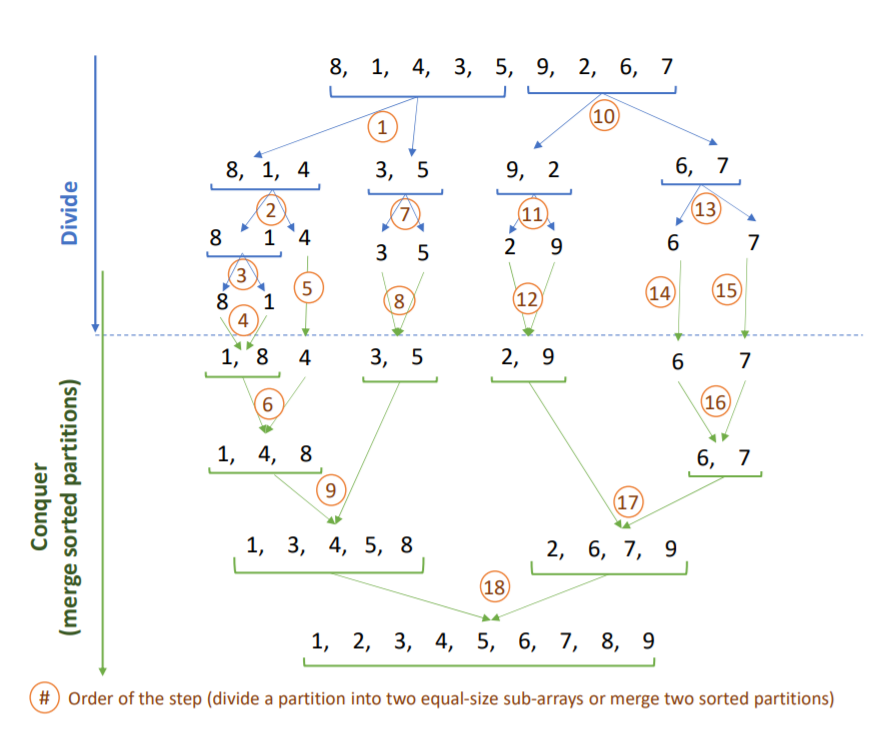
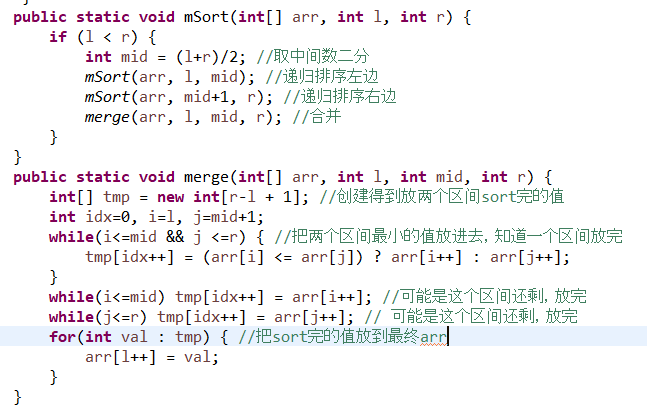
outer recursive call O(logn) inner loop--O(n), 要走logn次O(n)的操作

如果是两个sorted array则是O(n)

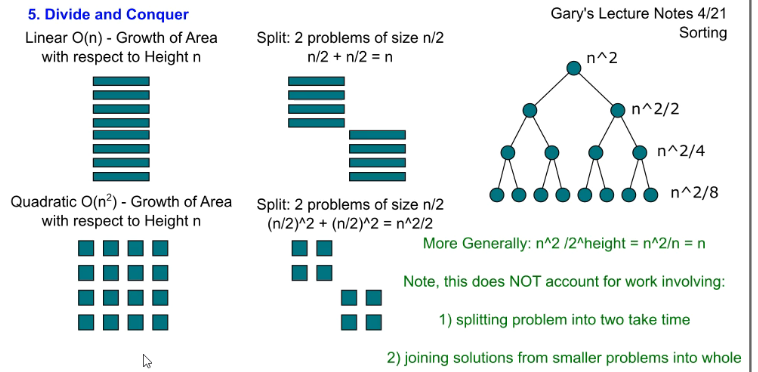
stable

把array二分直到每个子序只有一个element，再合并，每次合并都伴随sort

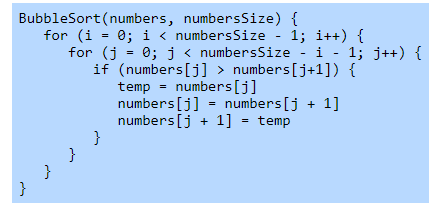




Merge sort是分治法(Divide and Conquer)的一种经典应用

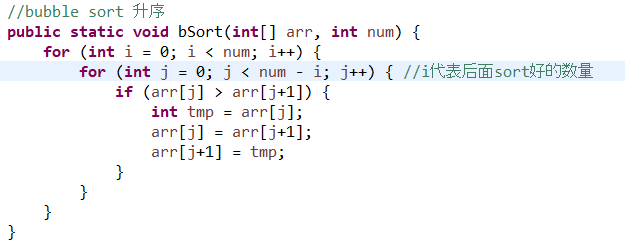


1. 冒泡排序 O(n^2) best O(n) 稳定



index012345，0和1先比较，再1和2比较，2和3比较，每次比较都会sort，这样一对对比较到最后一个index，最后一个index就是最大的element(sorted part)再从头开始开始一对对比较，每次比较都会放一个unsorted part里最大element到sorted part

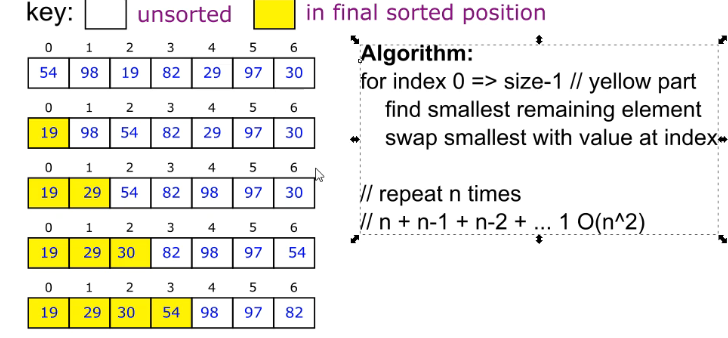




降序只要把if里 < 换成 >

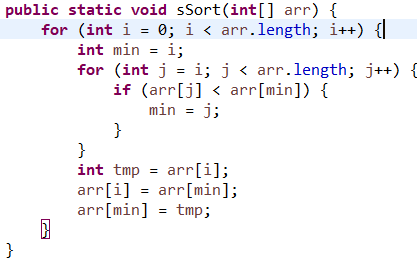
1. 选择排序 永远O(n^2) 不稳定

worst&best O(n^2) unstable(array时)， stable(linked list时)



54 98 19 82 29 97 30

把第一个数(54)作为sorted position，再往后search array里最小的数和54换位置，再把98作为放到sorted position，再重复搜索整个array里最小的数字和98换

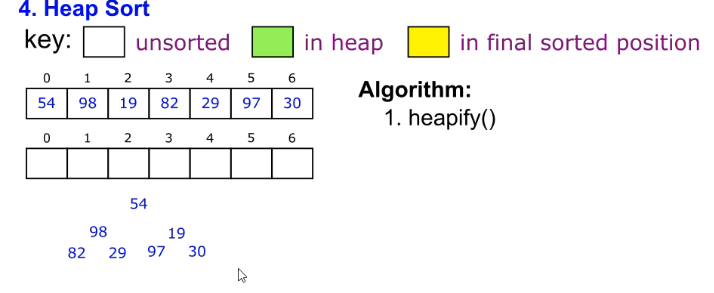


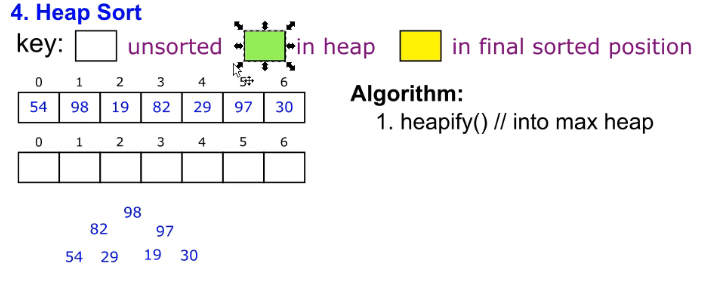
1. 堆排序 O(nlogn) 不稳定

简单的说就是把heap先heapify成max-heap，再把root(最大value)放到末尾处(sorted part)--和末尾处的element互换位置，（之后的操作不用考虑sorted part）对heap进行percolate down，这样root又是最大value，重复之前步骤

1.先转换成max-heap结构，再percolate down，再转换回array结构变成

98 82 97 54 29 19 30



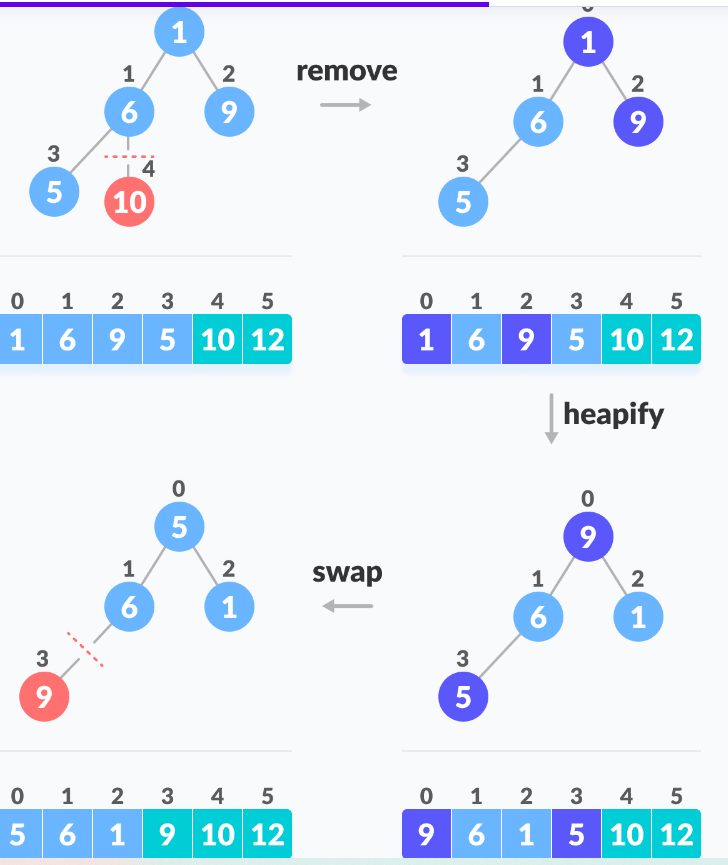
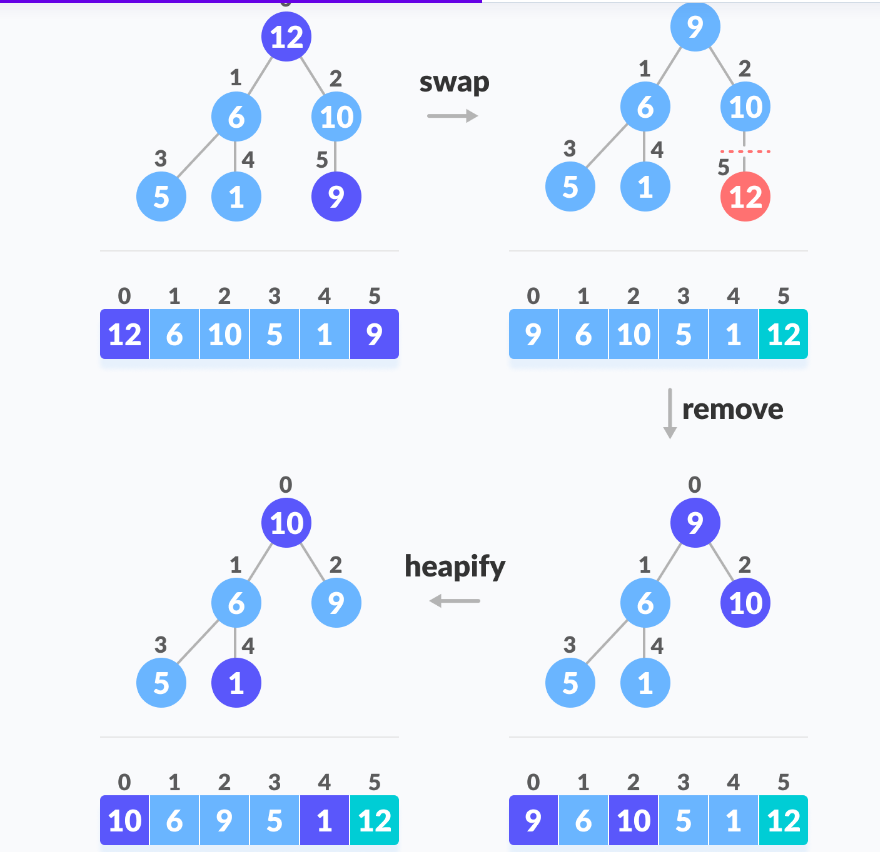


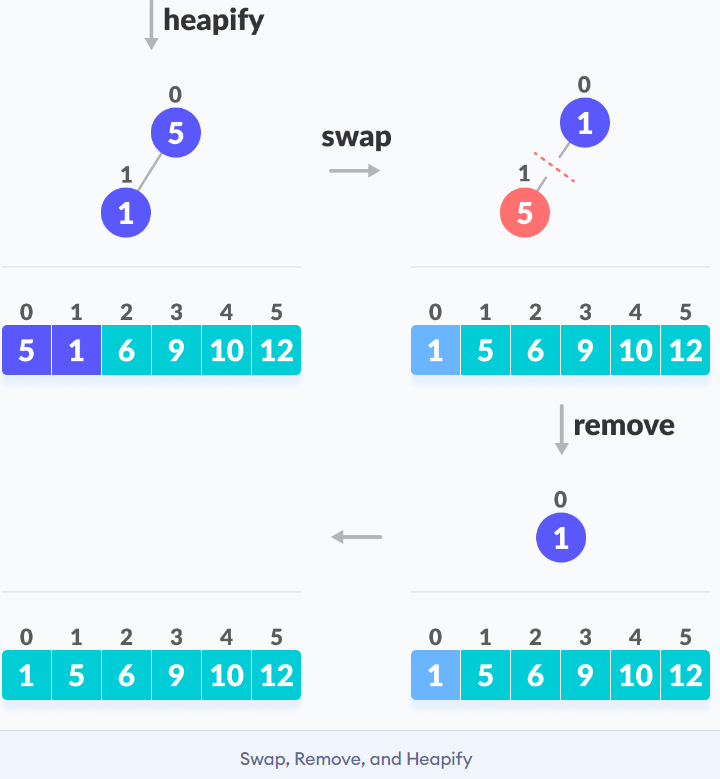
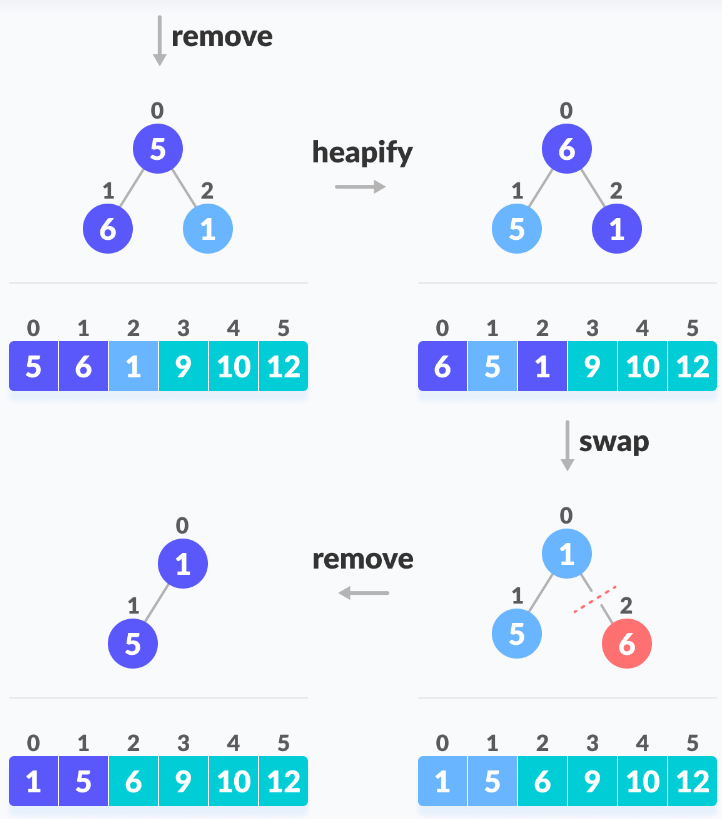
1. 开始heap sort，把最大的value放最后

图从左到右

12

34

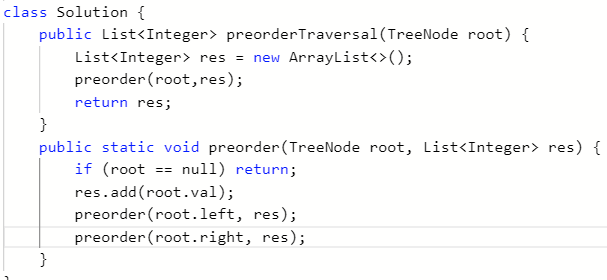




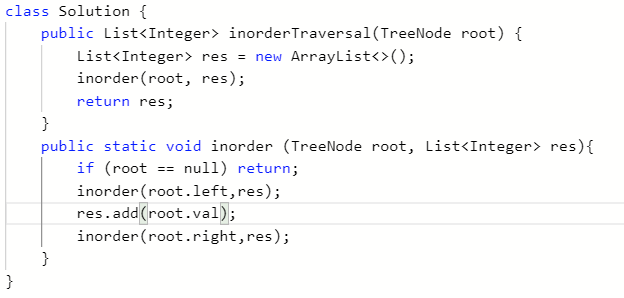
1. 树
2. 递归遍历 改顺序即可 迭代

前序遍历 中-左-右

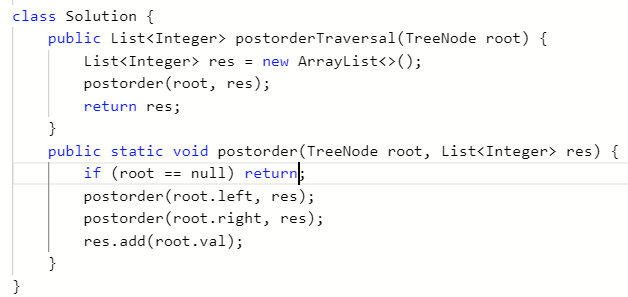
建arraylist -> 对root进行前序遍历 -> root是null 返回 否则加root的value并且前序遍历左 然后 右



中序遍历 左-中-右

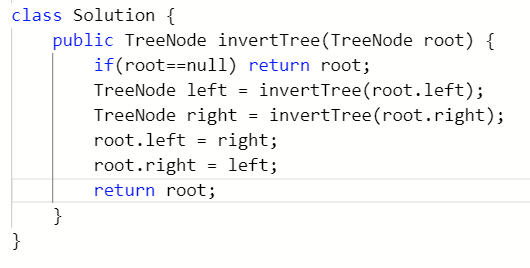


后序遍历 左-右-中



1. 226翻转二叉树

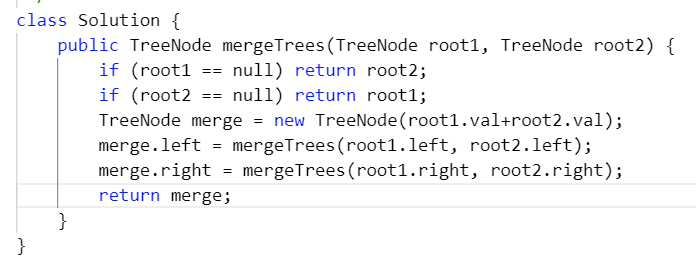
递归 保存左右节点，然后root的左右节点交换



1. 合并二叉树

合并两个树对应位置的节点值得到一个新的树

解法：如果一棵树为null则直接返回另一棵树，创建一个新的合并了值的节点，设置他 的左右儿子分别等于合并原来两个左右儿子

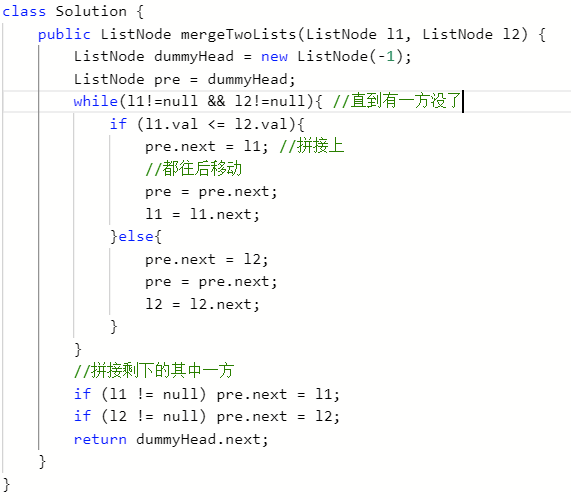


1. 剑指offer68-1二叉搜索树的最近公共祖先

解法：如果root比两个节点都大就去左边找—return 方法名(root.left,p,q)，大则反之

1. 链表
2. 合并两个链表 不去重 保持升序

创建一个dummyHead和pre两个node，用pre去traversal，用dumyHead.next在最后 返回。每次连接l1l2里节点小的，在l1和l2有一方没了就停止，把有的一方拼接上



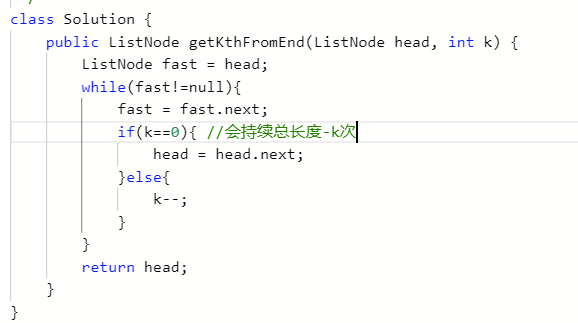
1. 分割链表 给一个链表头，和某个节点的值，把值小于这个节点放节点大/一样值的节点前面

解法：用两个dummyHead分别拿比几点大和小的节点然后拼接起来



1. 链表倒数第K个节点

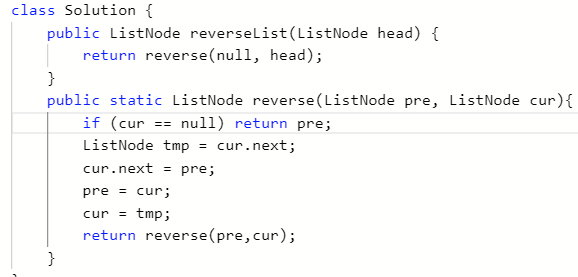
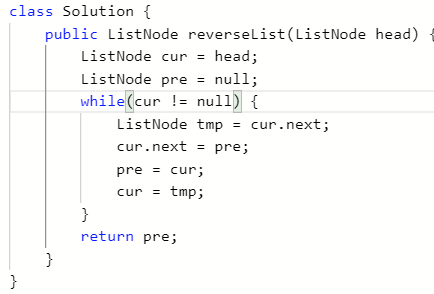
用快慢指针，快指针先走，每走一次k-1，k=0时候快慢指针才一起走



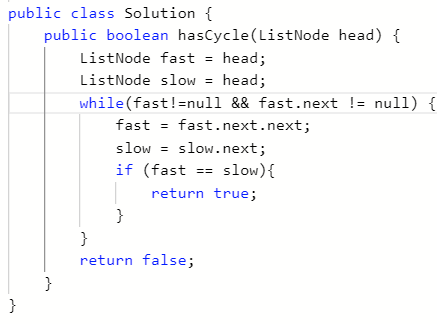
或者直接找链表长度，然后走长度-k次

1. 反转链表 力扣206

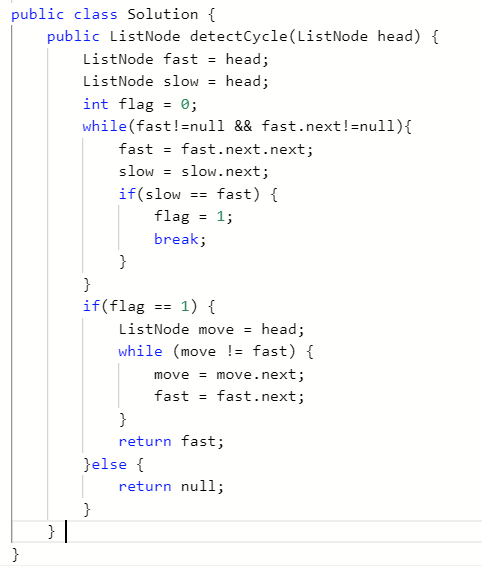
解法：用pre=null(在head前)，和一个cur(指向head)，每次把先存一下cur的下一个，然后把cur.next转向pre，再把pre和cur进行移动



1. 力扣141 判断是否有环 -- 快指针每次比慢指针多走一步，没环快指针直接到头了

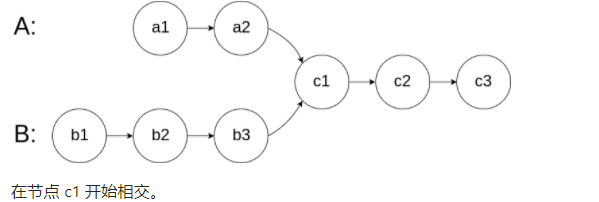


1. 力扣 找环的入口， 先找快慢相遇点，再用这个点和head的点同步运动，再次相遇点就是入口。 无环return null

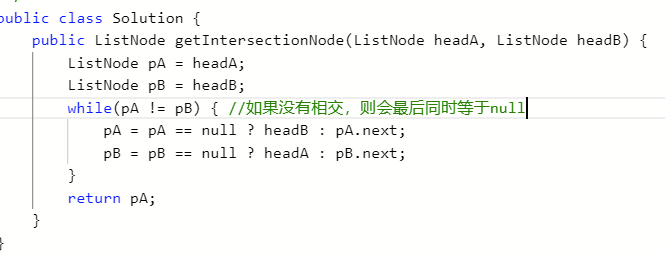


1. 力扣160 相交链表

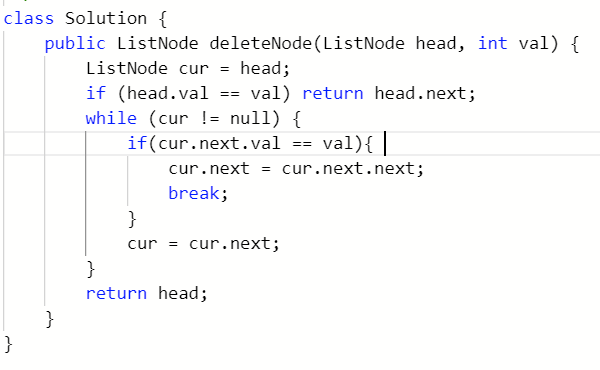
找链表相交的点



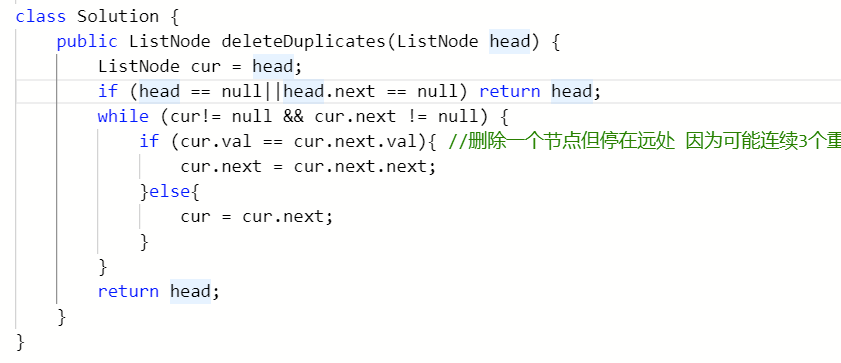
解法：快慢指针思想，一个链表结束了从另一个开始，这个各trace了一遍自己链表后 他们到相互链表后距离相交点的距离一样

s

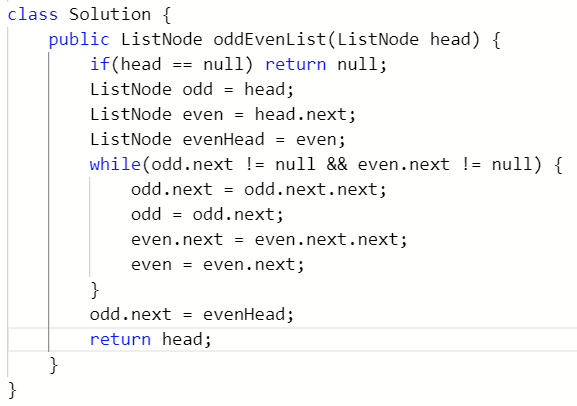
1. 去掉链表指定值



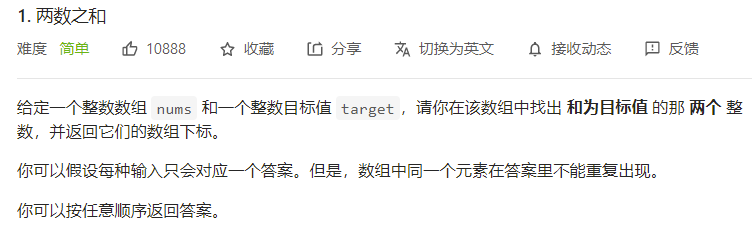
1. 力扣83 去掉链表重复值



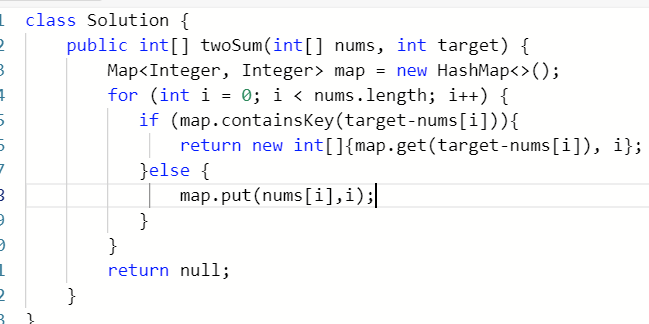
1. 奇偶链表



1. 回文链表
2. 力扣第一题



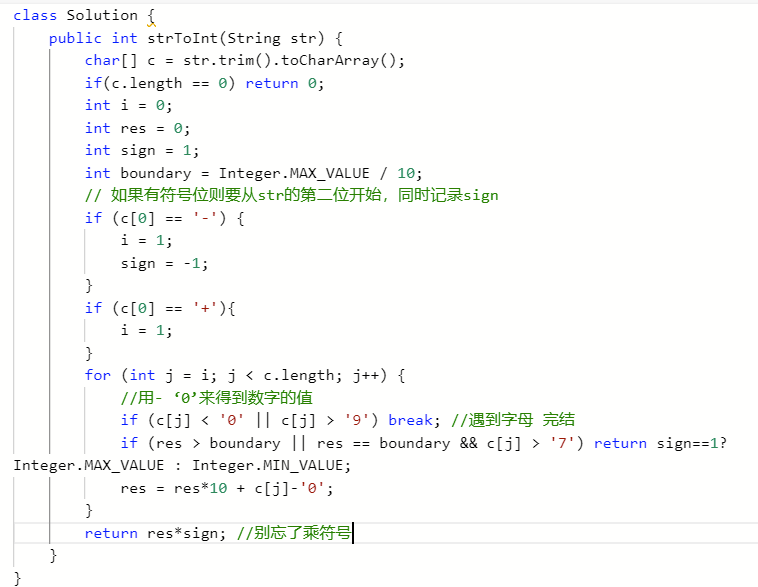
用哈希表放每个元素的值和对应的index，每次遍历用containsKey查找有没有对应的 target - 当前值的值，有则直接返回没有就put进



1. 字符串
2. 剑指offer67 字符串转整数



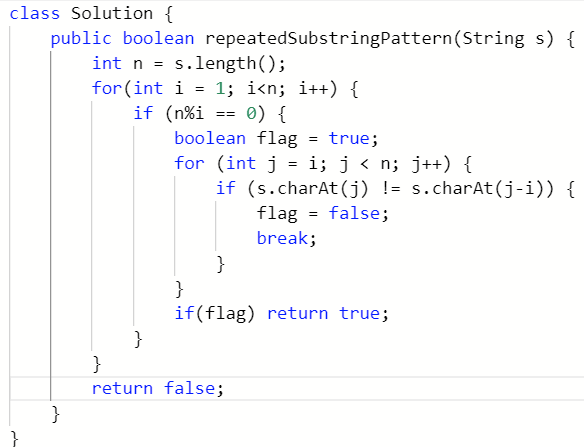
解法：去掉首部空格，看第一位是+，-，数字，如果都不是说明是字母要返回，对于每个字母 -‘0’得到对应数字，先对res\*10再加数字，如果res大于边界或者等于边界而且当前数字>7就越界了，越界返回integer最大值如果符号位是+，反之-



1. 力扣459 判断一个字符串是不是由一个字串重复多次构成

解法：从index1开始，用数组长度%i==0得到字符串倍数处，然后比较子串每一位

和字符串一不一样。外loop的目的是达到第二个子串开始然后不断和前一个 子串比较，内loop就是用第二个子串和第一个比，3和2比....



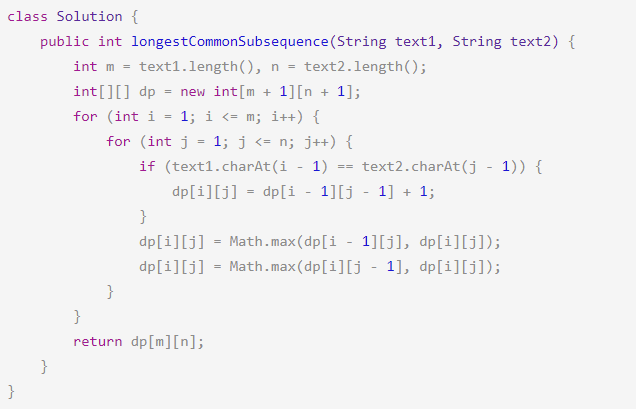
1. (optional)

力扣1143 找两个字符串的最长公共子序列 输入abcde，ace 得到3(ace)

解法:想象一个长方形，长和宽各是一条string，每个单位是一个char

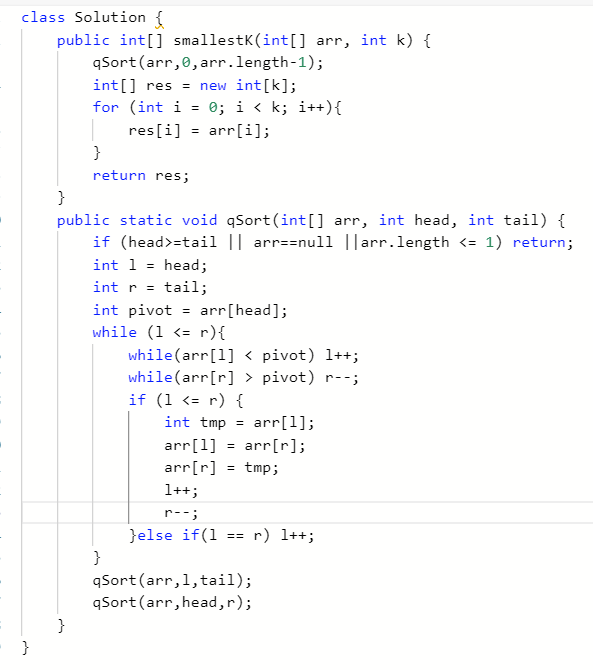
从第二行第二列开始，如果这个各自一样，则加上左上角格子的值，

否则取左和上的最大值，最后最右下角的为最大值

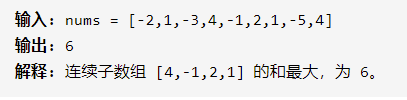


1. 数组
2. 找数组里最小k个数

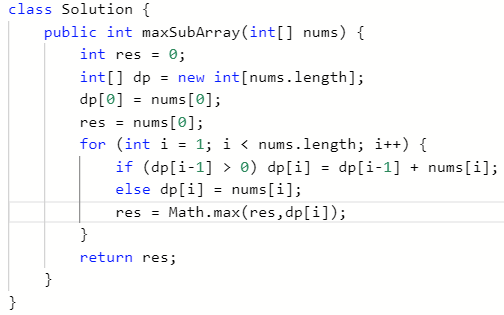
解法：先快排再遍历



1. 连续子数组的最大和 动态规划

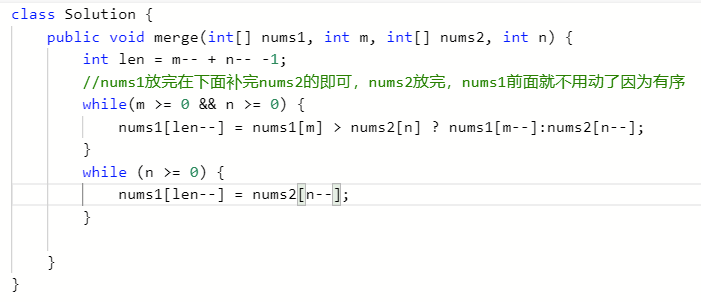


解法：用一个数组dp，index0放(nums的第一位), index1开始只要前面dp数组的值大 于0，就加上他和本次nums里的值，如果前面为0了，就=nums里的值重新开始， 每次res都取dp和nums位置上的最大值

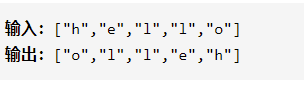


1. 力扣88 合并两个有序数组

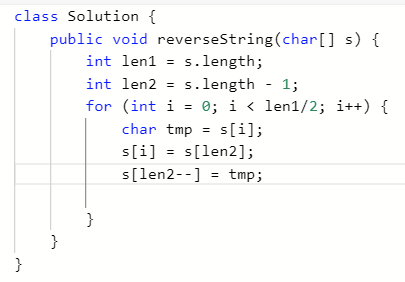
解法：从数组最后放最大的，合并进去的数组要确保补齐



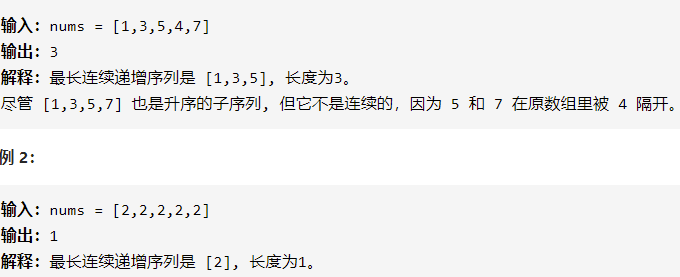
1. 字符串反转



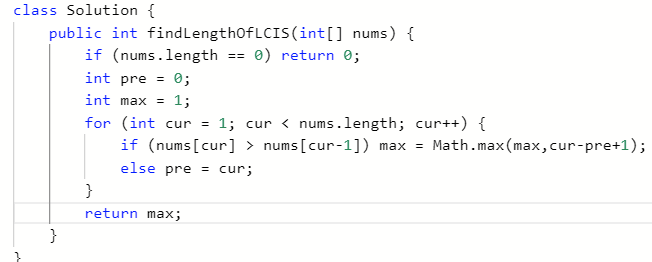
解法：对前半字符串与后半交换就行，遍历一半数组



1. 力扣674 最长连续递增序列长度

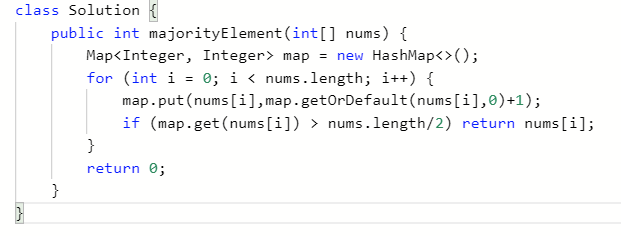


解法：用一个指针去记录是否连续，如果当前比前一个大了，说明连续则用当前i-这个指针位置-1得到现在 连续序列长度并和max比较后把他存到max里，否则指针从当前开始，重新开始标记子序位置



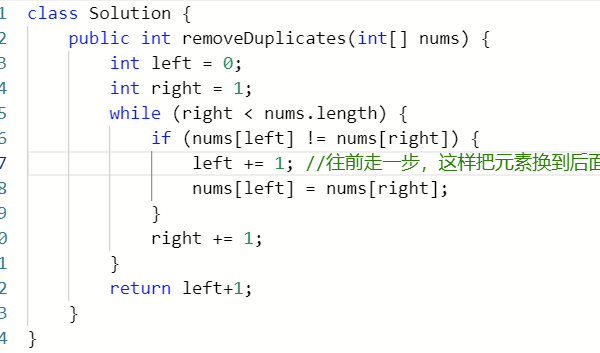
1. 剑指offer39 数组里出现次数超过一半的数字

解法：用hashmap装每各数和对应的次数+1，每次都比较出现次数有没有大于了 len/2



1. 力扣26 删除有序列表的重复项 返回数组最后一个index

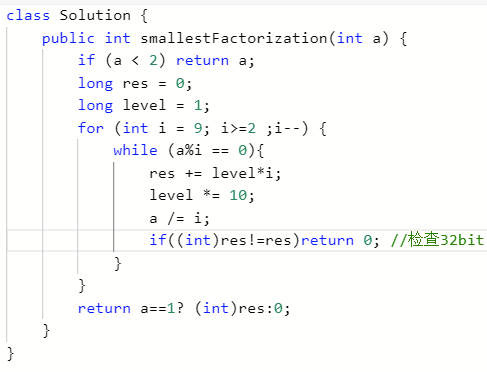
解法：双指针，用第二个指针去找和第一个所指的不一样的数，找到了就替换到前面



1. 其他
2. 力扣625

因式分解：输入一个数，返回的数各位乘积等于输入，返回的数据尽可能小，同时答案不在32-bit integer的话返回0

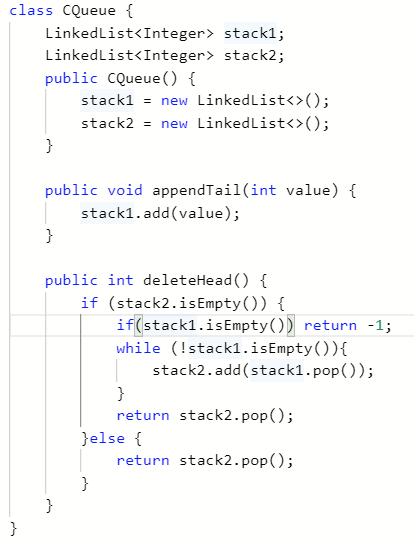
做法：输入的数比2小就返回自己，用for loop，i先=9去被输入的数除，一直--直到除得尽，除得尽的情况下去给返回结果res加上（i\*当前level 比如 1， 10）然后对level\*10和对输入数减小i倍数。最后a=1的话返回res，否则返回0



1. 剑指offer09

两个栈实现一个队列 -- 实现先进先出

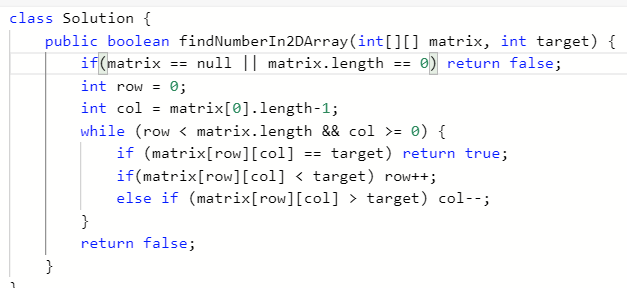
解法：创建两个链表，加的时候往1号加，第一次出的时候2为空，把1倒到2中，这样2的最上面就是最先进的，pop出就是最先进的，如果2号里不为空，则pop2号里的就行，因为1号里的肯定是后加的。要用isEmpty()，如果1和2都空返回-1



1. 剑指offer04 二分查找

给一个二维数组，每行每列都是升序，找到特定值

从右上角开始找，比target大，到下行，比target小，到前一列，相等返回true，越界 了还没找到，说明不存在，返回false 时间复杂度0(m+n)



力扣704 二分查找

