排序算法：

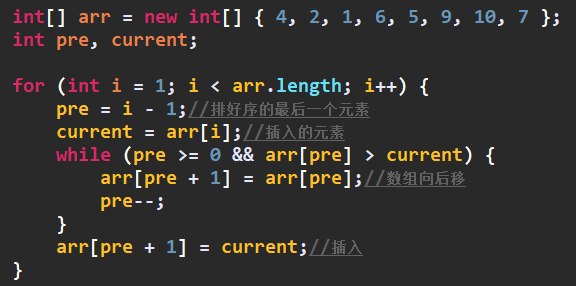
插入排序：把新数据插入到以排好序的数组中

4,2,1,6,5,9,10,7

2,4,1,6,5,9,10,7 4中插入2

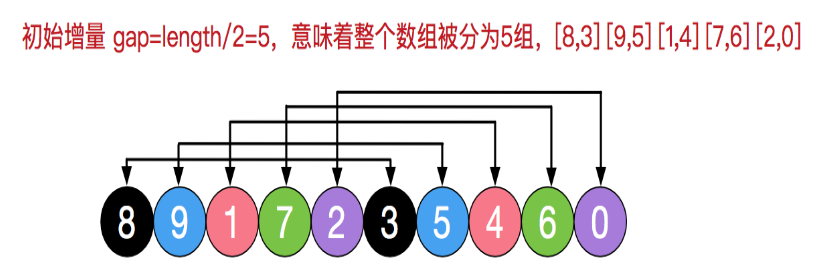
1,2,4,6,5,9,10,7 2,4中插入1

1,2,4,5,6,9,10,7 1,2,4,6中插入5

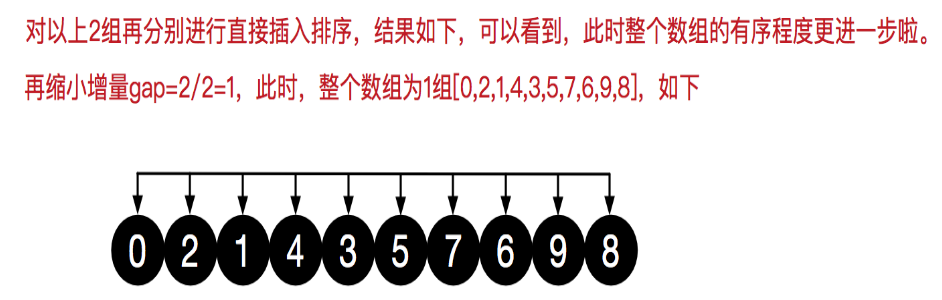


希尔排序：



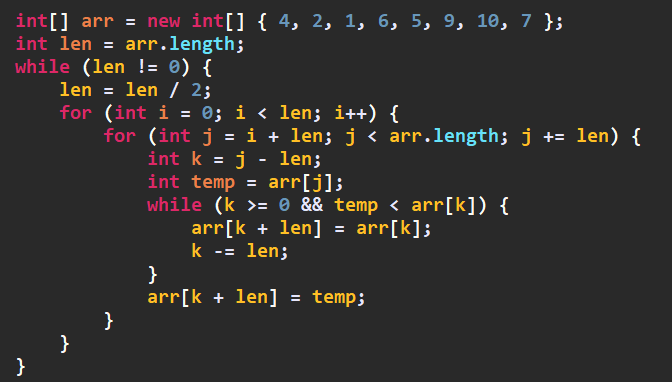






1,2,3,4,5,0在插入排序中，在插入0时，需要将2,3,4,5依次向后移，循环次数多，而希尔排序是先对数组进行处理处理后的结果是1,0,3,2,5,4这样就不必导致数组向后移，减少循环次数

希尔排序就是将大的数据放右边，小的放左边(只能是尽可能的)，这样在排序的时候，就只需要和周围的进行交换，减少数组的移动次数



通过len进行分组，然会对分组后的进行插入排序，然后再细化分组，再进行排序，当len=1时，再进行插入排序时，就会简单很多，只是和周围的数据进行比较

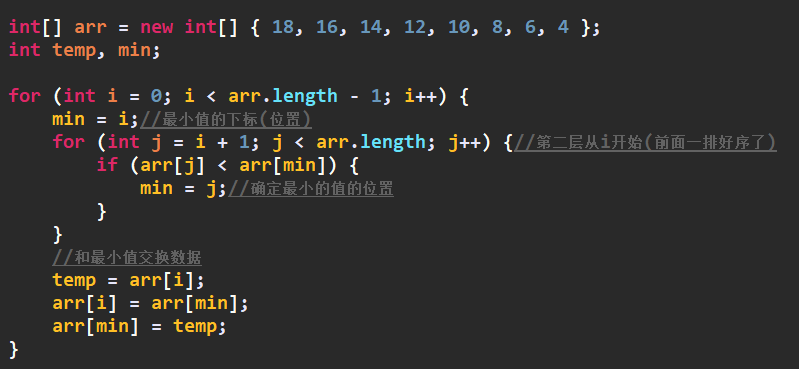
while循环是在进行分组，第一层for循环表示有多少组，第二层for循环为每个组进行插入排序，注意插入是按len，而不是1

选择排序：选出第一个数，然后第一个和其他进行比较，找出最小数的位置，遍历完后，第一个数和最小数进行交换位置

4,2,1,6,5,9,10,7

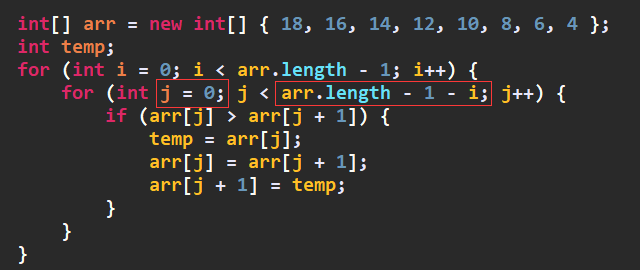
1,2,4,6,5,9,10,7 1和4交换位置后，2进行比较，发现2最小，没有比较的，同样4也是，取出6，和后面的比较(前面已经拍好序了)

1,2,4,5,6,9,10,7



先找到最小值的位置，再交换

冒泡排序：

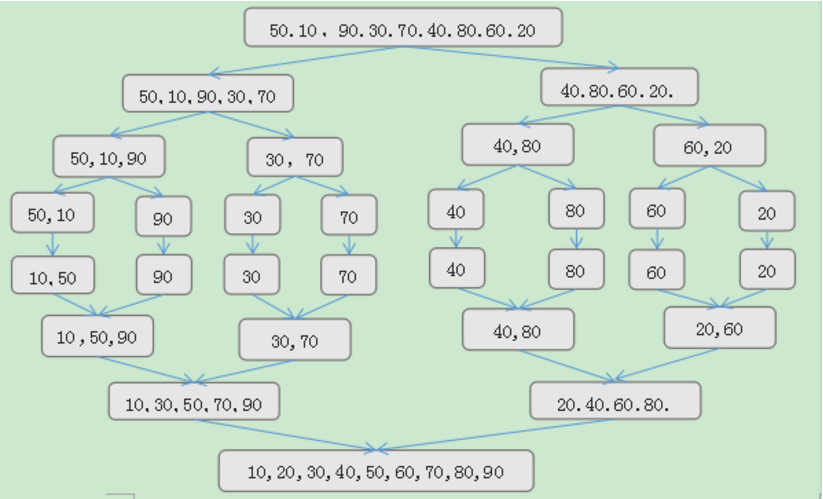


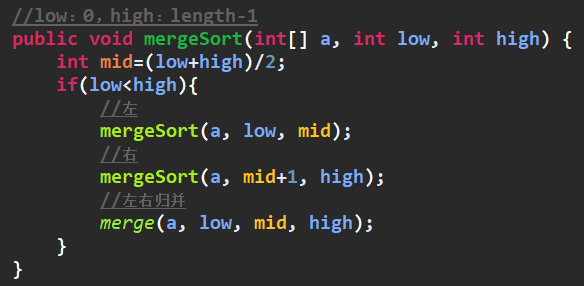
如果是升序j<len-1-I，如果是降序j=i

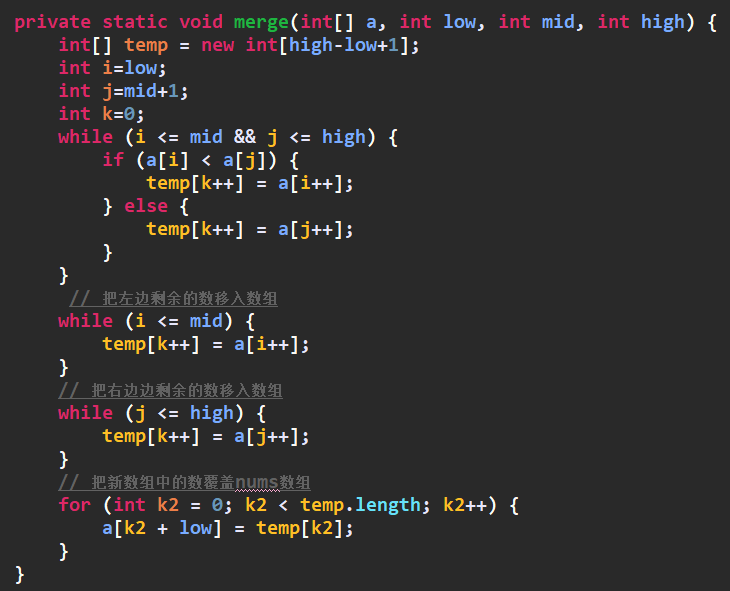
快速排序：找出一个基数，将大于基数的放左边，小于基数的放右边(只要左边小于基数，右边大于基数即可，而不必左右两边一个基数作为分解线，[10,19,8] [25,30,21,86]，基数是21，这样也可以)



归并排序：分治发







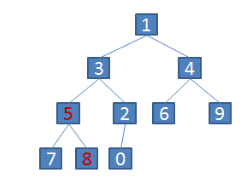
堆排序：

堆排序：建堆+排序

堆是一个二叉树，数组可以表示为堆，第k个节点的父节点(k-1)/2，左子节点2k+1，右子节点2k+2，比如这里的3，下标是1，

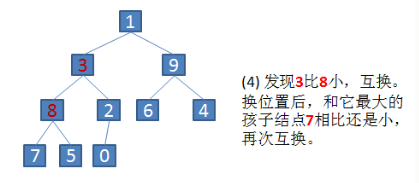
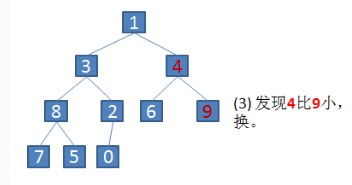
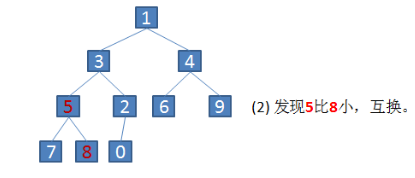
(k-1)/2=0，父节点为1，2k+1=3,2k+2=4，左节点为5，右节点为2

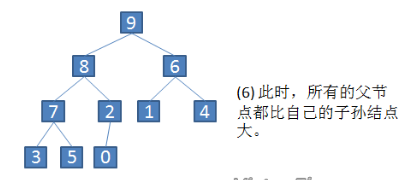
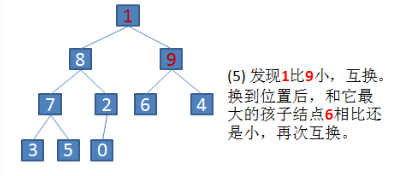




将数组表示为二叉树数后，开始建堆，堆分为大根堆：所以父节点大于子节点，小根堆：所有父节点小于子节点

建堆从非叶子节点开始，这里是从2开始，然后是5,4,3,1，所以需要建5次堆(堆的非叶子节点的个数=节点数/2)

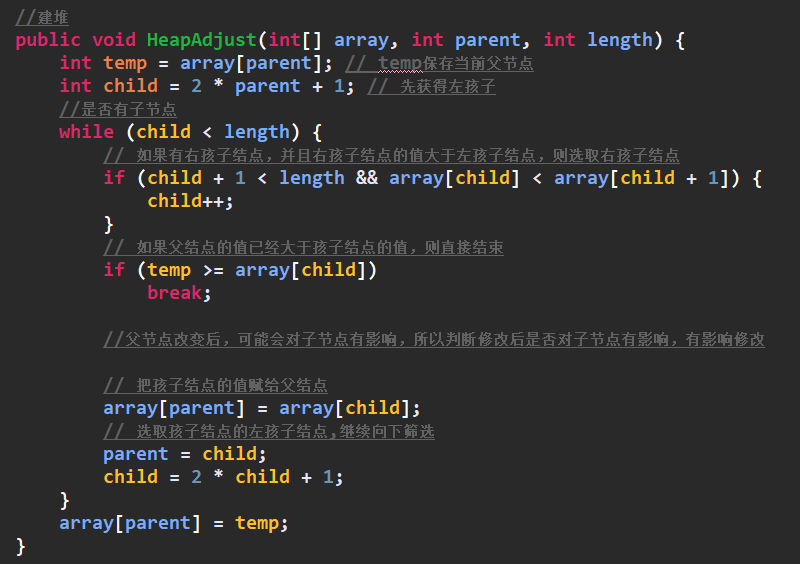


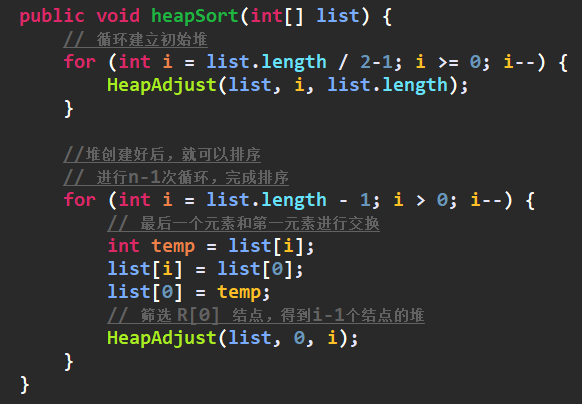


第4部，3和8换后，左支又不满足大堆，因此要判断变换后是否对子数有影响，判断都没有子节点为止，同样第5部也是

建立好堆后，那么堆顶元素肯定是最大的(大根堆)，可以将堆顶元素输出，然后将最后一个叶子节点作为堆顶，然后再建堆，输出堆顶元素，再将最后一个叶子节点作为堆顶，依次这样







稳定性：当数组中A和B的数相同，排序前，A在B前面而排序后B在A前面(1,5,3[A],6,3[B] 1,3[B],3[A],4,6)，这样就叫不稳定

稳定：冒泡，插入，归并

不稳定：选择，快排，希尔，堆排

时间复杂度：

O(n^2)：插入，选择，冒泡

O(nlogn)：快排，归并，希尔，堆排

快排最好，其次是归并和希尔，堆排序在数据量大时效果明显(快排和归并都是通过空间换时间的，递归)

数据规模小：插入排序，稳定性有要求选择插入或冒泡，稳定性无要求选择选择排序

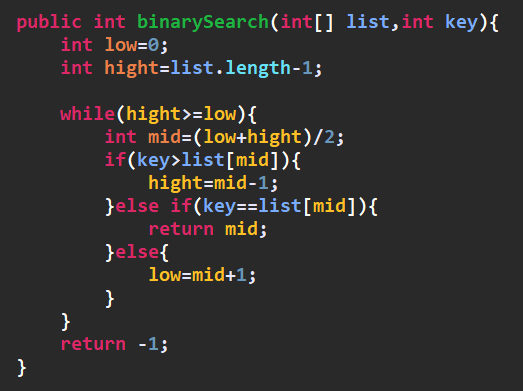
数据规模一般：稳定性无要求，空间大选择快排，序列本身有序，且要求稳定性，空间大选择归并

数据规模大：稳定性有要求选择归并，稳定性没要求选择堆排序

初始化序列有序，宜用插入，冒泡

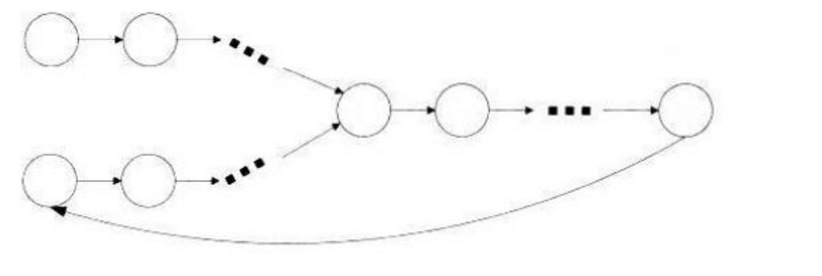


二分法：



链表是否相交：

L1,L2两天链表是否相交，可以将L1的表尾指向L2的表头，如果成环，L1和L2相交

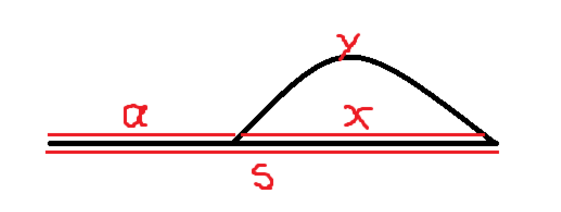


现在就转换为判断链表是否有环：采用追逐法，指针p1和p2都指向链表的头节点p1每次移动一步，p2每次移动两步，如果p1的next为null，那么链表无环，如果p1==p2，则链表有环

如果有环p2先进入环，p1后进入环，则p1和p2之间成追赶的方式，如果p1和p2相差1步，则p1走一步，p2走两步，p1和p2相遇，如果p1和p2相差2步，则下次相差一步，再下次就相遇，所以p1和p2肯定能相遇

拓展：成环后，找出成环的节点

当p1和p2相遇后，p2指向链表，p1不变，此后p1和p2都走一步，下次p1和p2相遇的点就是环的入节点



当p1和p2相遇后，p1还未走完环(极端情况，假设p1进入环后，p2就在p1前一步，则p2追赶上p1需要r-1次，r为环长度，所以p1和p2相遇时，p1并未走完环)，而p2可能已经走了n圈环，相遇时p2走的路是p1的两倍即2s=a+nr(r为环的长度)，a+x=s，x+y=r，最终a=(n-1)r+y，第一次相遇时，在a+x处，然后p1在表头，p2任然在a+x处，此后p1和p2都走一步，p1走a步时，p2走了(n-1)r+y，所以下次相遇的点就是环的入点

找到环后的入点后，就可以知道环的长度

扩展：求出环上距离任意一点最远的点(对面点)

同样采用快慢指针，fast和slow，fast和slow起始位置为p，当fast=p或fast->=p，此时slow所在的点就是对面点

环考虑使用快慢指针

二叉树