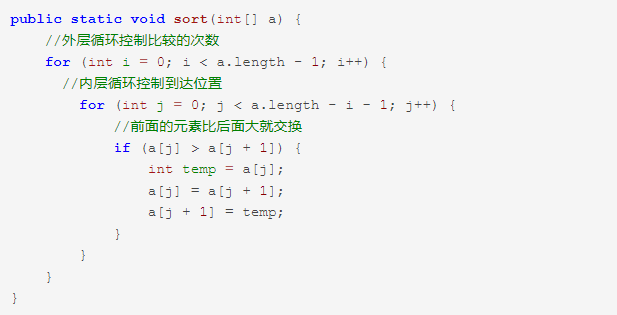
**一、排序算法**(按从小到大顺序)

1. **冒泡排序**

思想/原理：

从第一个元素开始，与之后面相连的元素比较，如果a[n]>a[n+1],则调换两个元素位置，然后接着比较a[n+1]与a[n+2],以此类推，即每一次外层循环，找出最大的数然后放到最右边。

代码示例：

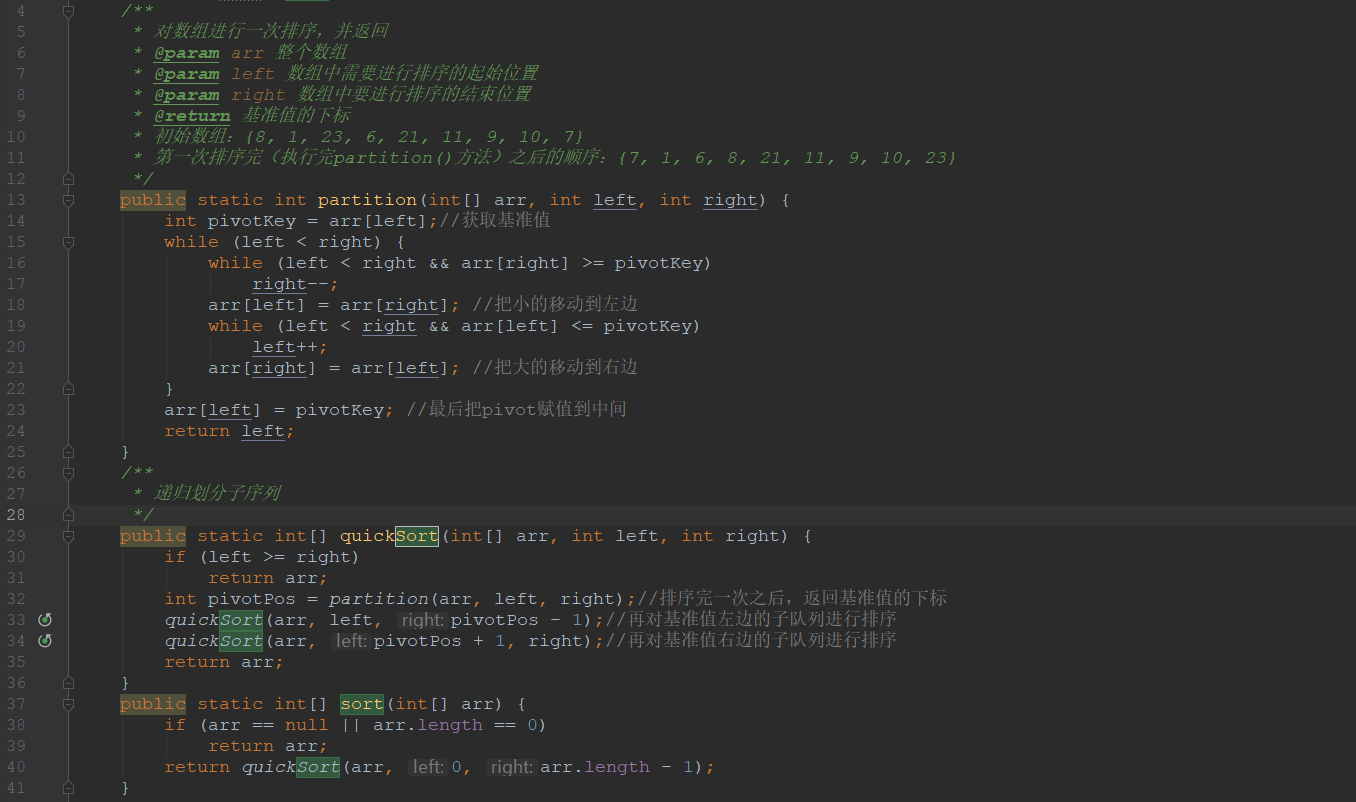


1. **快速排序**

思想/原理：

获取队列arr[]的第一个元素arr[0]=A，假设队列的下标是从[0,length-1]，令left=0,right=length-1,则队列的下标为[left, right]。从队列的最有一个元素往前与A判断大小时，如果如果元素>=A,则right--,否则交换元素与A的位置，然后再从队列的第一个元素与A判断大小，如果元素<=A，则left++,否则交换元素与A的位置。如果此时left<right,则继续循环判断。这样只是把小于A的元素放在了A的左边，把大于A的元素放在了A的右边，所以还需要记录执行完一次循环之后A的下标，然后递归调用该排序操作。

代码示例：

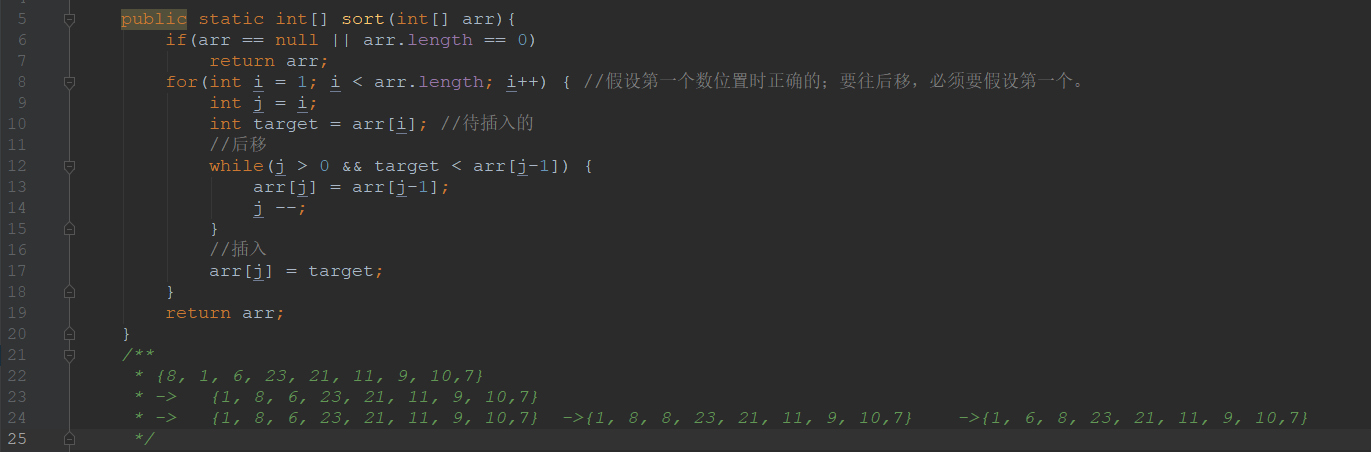


1. **插入排序**

思想/原理：

从队列的第二个元素起，依次从后往前与该元素前面的元素比较，如果当前元素小于前面的元素，则将前面的元素往后移一位，否则将当前元素插入到当前元素后面。

代码示例：

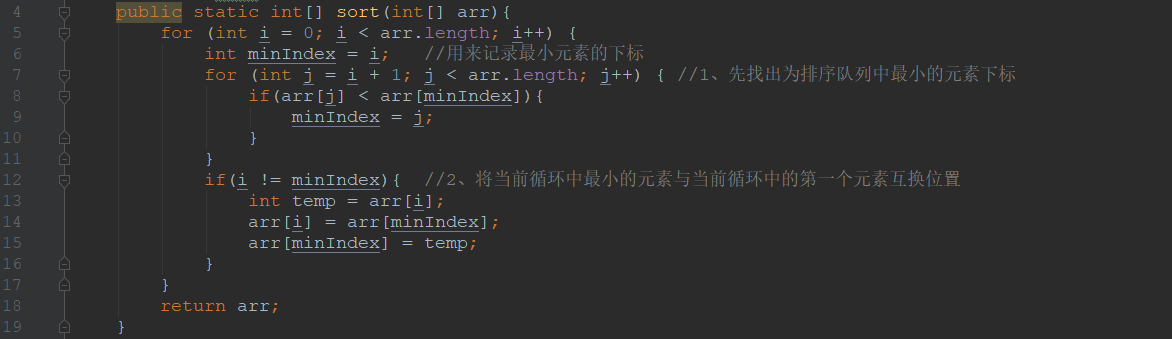


1. **选择排序**

思想/原理：

与冒泡排序有点相似，不同的是，选择排序是先找出最小值或者最大值，放到队列最前面，然后再找出剩下队列中的最小/大值依次放置。

代码示例：



1. **希尔排序(递减增量排序)**

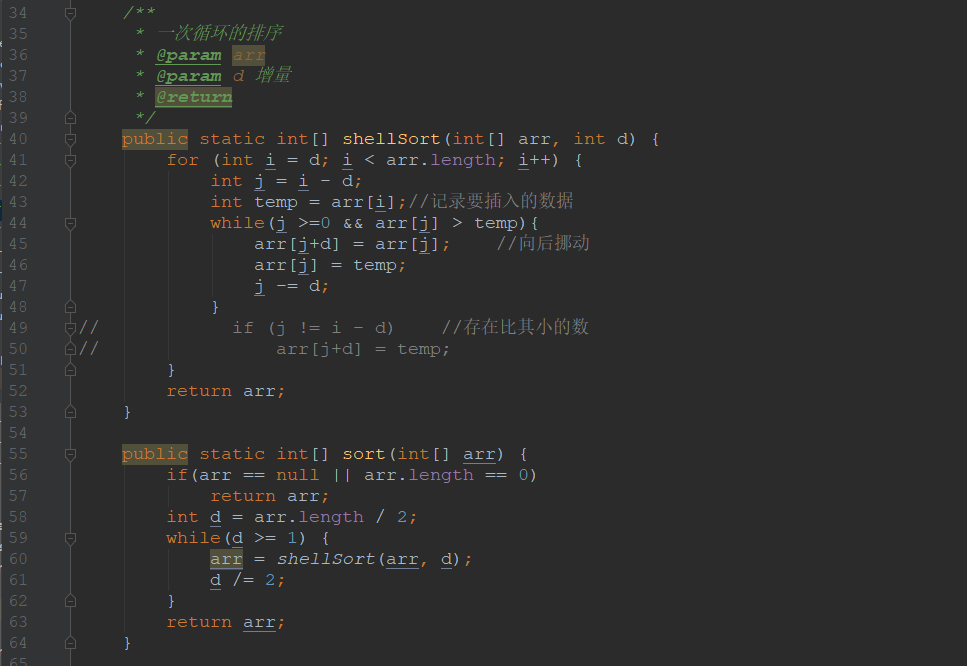
思想/原理：

1. 取下标为队列长度一半的元素arr[len/2],与a[0]比较,如果arr[len/2]<a[0]就交换两个元素位置，然后再依次比较arr[len/2+1]与arr[1]、arr[len/2+2]与arr[2]........arr[len](len为基数时是arr[len-1])与arr[len/2-1],如果前者大于后者，交换元素位置。这一步骤中，两个元素中的下标有一个间距为len/2,我们称之为增量d。
2. 取上一次增量的一半d/2即len/4,然后分步两两相连比较arr[0]与arr[len/4]、arr[len/2]、arr[3len/4]、arr[len]的大小，同样方法比较arr[1]与arr[len/4+1]、arr[len/2+1]、arr[3len/4+1]的大小，一直到比较arr[len/4-1]、arr[len/2-1]、arr[3len/4-1]的大小。经过此次步骤的循环排序，整个队列中从个增量的子队列(比如rr[0]、arr[len/4]、arr[len/2]、arr[3len/4]、arr[len])的顺序就排列好了。此次循环完之后增量d=len/4
3. 重复第二个步骤，每次增量都是上一次增量的一半，一直到d/2<1，整个排序就完成了，因为d/2=1是，相当于是对整个队列相邻的元素进行了一次排序，此时完成了整个队列的排序。

排序过程效果如下：



代码示例：



1. **归并排序**

思想/原理：

将待排序的序列分成若干个子序列，每个子序列是有序的，然后再把有序子序列合并成一个整体的有序序列。

代码示例：

1. **计数排序**

思想/原理：

先获取数组arr[]中最大的整数max，然后创建一个新数组count[]，长度为max+1，将新数组中每个元素设为0，然后设置count[arr[i]]++。然后循环遍历新数组count,下标为m,定义新下标k=0,如果count[]数组中元素不等于0,则arr[k++]=m。

代码示例：

