算法分析

**>>排序算法**

**>>>>>基于比较的排序算法：空间复杂度O(1)**

1. 冒泡排序：最坏时间复杂度为O(n^2)，最好时间复杂度为O(n)【需要在第一次遍历的时候适用一个旗标表示是否有交换的可能】，平均时间复杂度O(n^2)
   1. 定义：每次将数组中的一个数与前一个数相比，如果前一个数比自己大，则进行交换，结果就是每次遍历完后得到当前第i+1大的数放在数组的后面。
2. 选择排序：最坏时间复杂度O(n^2)，最好时间复杂度O(n^2)，平均时间复杂度O(n^2)，非稳定排序算法，但是选择排序交换的次数比冒泡排序交换的次数少，所有更优
   1. 定义:每次选择第i+1大的数，记录其下标，然后和下标为i的位置的数进行交换。
3. 插入排序：最坏时间复杂度O(n^2)，最好时间复杂度为O(n)【数组有序情况最优排序算法】，平均时间复杂度O(n^2)，稳定的排序算法
   1. 定义：从第二个元素开始，从其前一个数向前比较，如果比其小则顺次往前插入
4. 归并排序：最坏时间复杂度O(nlogn),最优时间复杂度O(n)，平均时间复杂度O(nlogn)，空间复杂度O(n)，稳定的排序
   1. 定义：将有序数组进行合并操作，从数组长度为1开始，逐渐变成2，4最后到所有数都完成合并操作
   2. 算法：一个可以合并数组两部分的方法merge(int[] arr, int left, int mid, int right),然后递归调用合并方法mergeSort(int[] arr, int left, int right)
5. 快速排序：最坏时间复杂度O(n^2),最优时间复杂度O(nlogn)，平均时间复杂度O(nlogn)，不稳定的排序
   1. 定义：以数组的最后一个元素作为基准，将数组分为两部分，一部分比基准大，一部分比基准小，然后调用递归方法继续对左右两部分进行划分。