算法基础

算法的效率（1星）

- 时间复杂度：指程序运行从开始到结束所需要的时间
- 空间复杂度：指对一个算法在运行过程中临时占用存储空间大小的度量

查找算法

- 顺序查找：将待查找的关键key元素从头到尾与表中元素进行比较，如果中间存在关键字key，则返回成功；否则，则查找失败。
- 二分查找（2星）【案例】：确定中间点位置mid=【（low+high）/2】，将待查找的k值与mid进行比较，若相等，则查找成功并返回此位置，否则需却的新的查找区间，继续二分查找。
- 哈希表查找（2星）：关键码序列相同的元素，如果该空间已经被占用，则自动转换到该关键码序列的下一个地址空间。

排序算法（5星）

- 排序算法的基本概念
  - 稳定的排序算法
  - 不稳定的排序算法
- 插入类排序
  - 直接插入排序：基本操作是将一条记录插入到已排好的有序表中，从而得到一个新的、记录数量增1的有序表。
    - 是稳定的排序算法，时间复杂度为O（n^2），空间复杂度为O（1）
  - 希尔排序：希尔排序是把记录按下标的一定增量分组，对每组使用直接插入排序算法排序；随着增量逐渐减少，每组包含的关键词越来越多，当增量减至 1 时，整个文件恰被分成一组，算法便终止。
    - 是一种不稳定的排序算法，时间复杂度约为O（n^1.3），空间复杂度为O（1）
- 选择类排序
  - 简单选择排序：在所有记录中选出排序码最小的记录，把它与第1个记录交换，然后在其余记录内选出排序码最小的记录，与第2个记录交换……依次类推，直到所有记录排完为止。
    - 是一种不稳定的排序算法，其时间复杂度为O（n^2），空间复杂度为O（1）
  - 堆排序：先将序列建立堆，然后输出堆顶元素，再将剩下的序列建立堆，然后再输出栈顶元素，依次类推，直到所有元素均输出为止，此时元素输出的序列就是一个有序序列。
    - 是一种不稳定的排序算法，时间复杂度为O（nlog2n），空间复杂度为O（1）
- 交换类排序【案例】
  - 冒泡排序：通过相邻元素之间的比较和交换，将排序码较少的元素逐渐从底部移向顶部。
    - 是一种稳定的排序算法，其时间复杂度为O（n^2），空间复杂度为O(1)
  - 快速排序：(1)首先设定一个分界值，通过该分界值将数组分成左右两部分。
    (2)将大于或等于分界值的数据集中到数组右边，小于分界值的数据集中到数组的左边。此时，左边部分中各元素都小于或等于分界值，而右边部分中各元素都大于或等于分界值。
    (3)然后，左边和右边的数据可以独立排序。对于左侧的数组数据，又可以取一个分界值，将该部分数据分成左右两部分，同样在左边放置较小值，右边放置较大值。右侧的数组数据也可以做类似处理。
    (4)重复上述过程，可以看出，这是一个递归定义。通过递归将左侧部分排好序后，再递归排好右侧部分的顺序。当左、右两个部分各数据排序完成后，整个数组的排序也就完成了。
    - 是一种不稳定的排序算法，平均时间复杂度为O（nlog2n），空间复杂度为O（log2n）
- 归并排序【案例】：将已有序的子序列合并，得到完全有序的序列；即先使每个子序列有序，再使子序列段间有序。若将两个有序表合并成一个有序表，称为二路归并。
  - 是一种稳定的排序算法，其时间复杂度约为O（nlog2n），空间复杂度为O(n)
- 基数排序：它是透过键值的部份资讯，将要排序的元素分配至某些"桶"中，藉以达到排序的作用
  - 是一种稳定的排序算法，时间复杂度约为O（d（n+rd）），空间复杂度为O（rd）
- 排序算法对比

算法策略【案例】

- 算法策略概述
- 分治法（1星）：特征：把一个问题拆分成多个小规模的相同子问题，一般可用递归解决。
  - 典型问题：归并排序、快速排序、二分搜索
- 贪心算法（5星）：特征：局部最优，但整体不见得最优。每步有明确的，既定的策略。
  - 典型问题：背包问题（装箱）、多机调度、找零钱问题
- 动态规划法（5星）：特征：划分子问题（最优子结构），并把子问题结果使用数组存储，利用查询子问题结果构造最终问题结果。
  - 典型问题：矩阵乘法、背包问题、LCS最长公共子序列
- 回溯法：特征：系统的搜索一个问题的所有解或任一解。有试探和回退的过程。
  - 典型问题：N皇后问题、迷宫、背包问题