1.认识复杂度和简单排序算法

- 时间复杂度
  - 常数操作
    - 与数据量无关，每次都是固定时间内完成的操作
    - int a = arr[i];
    - +-*/，位运算
  - 非常数操作
    - int b = list.get(i);
    - 链表，需要遍历
    - 评价一个算法流程的好坏：先看时间复杂度的指标，然后再分析不同数据样本下的实际运行时间，也就是"常数项时间"
- 额外空间复杂度
  - 只需要有限几个变量，O(l)
- 异或运算
  - a^b
    - 0^N=N
      - N^N=0
    - 满足交换律和结合律
      - a^b = b^a
      - a^b^c=a^(b^c)
    - 同一批数异或，结果是一样的（与顺序无关）
  - 相同为0，不同为1
  - 无进位相加
    - 偶数个1为0
    - 奇数个1为1
  - 交换两个数的值
    - 前提：a和b在内存里是两块独立的区域
    - 抖机灵，不推荐
    - int a = 17, b =23;
    - a = a^b;
    - b= a^b;
    - a = a^b;
  - 在一个整型数组中
    - 只有一种数出现了奇数次，其他数都出现了偶数次，如何找到这一种数
    - 有两种数出现了奇数次，其他数都出现了偶数次，如何找到这两种数
      - 提取一个不等于0的数的最右侧的1
        - int rightOne = err & (~err + 1);
    - 时间复杂度：O(N)
    - 空间复杂度：O(l)
- 选择排序
  - O(N^2)
  - 从数组中选择最小元素，将它与数组的第一个元素交换位置。再从数组剩下的元素中选择出最小的元素，将它与数组的第二个元素交换位置。不断进行这样的操作，直到整个数组排序。
- 冒泡排序
  - O(N^2)
  - 从左到右不断交换相邻逆序的元素，在一轮循环之后，可以让未排序的最大元素上浮到右侧。
  - 在一轮循环中，如果没有发生交换，说明数组已经是有序的，可以直接退出。
- 插入排序
  - O(N^2)
    - O 指的是上限
    - 算法可能遇到的最差情况
  - 每次都将当前元素插入到左侧已经排序的数组中，使得插入之后左侧数组依然有序。
  - 数据状况不同，会导致算法流程的时间复杂度不一样
- 二分法
  - O(logN)
    - 每次砍一半，需要多少次可以把数组砍没
  - 一个有序的数组，找num是否存在
  - 在一个有序数组中，找>=某个数最左侧的位置或<=某个数最右侧的位置
  - 局部最小值问题
    - 数组无序，任何两个相邻数一定不相等
- 对数器
  1. 有一个你想要测得方法a
  2. 实现复杂度不好但是容易实现的方法b
  3. 实现一个随机样本产生器
  4. 把方法a和方法b跑相同得随机样本，看看得到的结果是否一样
  5. 如果有一个随机样本使得对比结果不一致，打印样本进行人工干预，改对方法a或者方法b
  6. 当样本数量很多时，比对测试依然正确，可以确定方法a正确