排序

问题1: 排序

- 给n个数, 从小到大排序

思考: 篮球俱乐部

- 给定N个人的身高（1.95m~2.05m），要求将他们排成一列，使得任意选取两个人，他们中间如果存在K个人，并且身高和为S，那么S与K×2m的差的绝对值小于等于0.1m。

插入排序和选择排序

- 增量算法
  - 选择排序？取决于未来的输入？
  - 插入排序：来一个插一个
- 部分排序
  - 插入排序？如果前k大的是最后k个元素…
  - 选择排序：选的前k个就是前k大元素
- 结论：排序算法的选择要看实际应用

归并排序

- 二路归并
  - 1, 2, 4, 7, 9
  - 3, 5, 6, 10, 11
  - 合并：n
- 分治sort(i, j)，设时间为T(n)
  - 排前一半：sort(i, mid), 时间T(n/2)
  - 排后一半：sort(mid+1, j)，时间T(n/2)
  - 合并起来：n

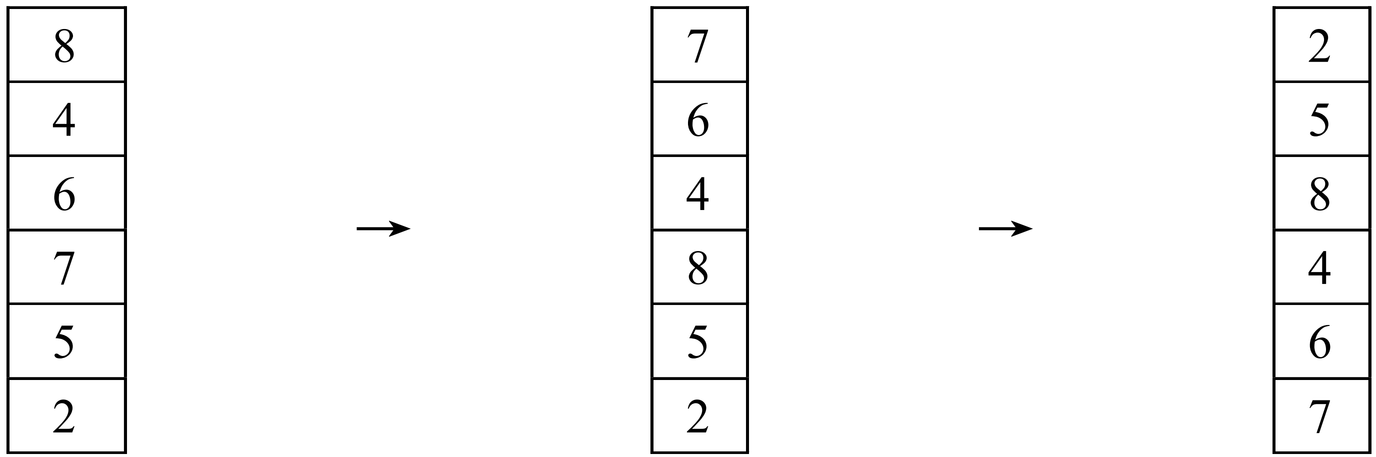
归并排序

- 算法分析
  - 时间: $T(n) = 2T(n/2) + n \rightarrow T(n) = O(n\log n)$
  - 空间: $S(n) = 2S(n/2) + n$? 空间可重用!
- 最好、最坏、平均是一致的

例题: 煎饼

- 目标：从小到大排序
- 从下数第k个开始往上翻
- (a) → (b) → (c)

3　　1



(a)　(b)　(c)

问题2: 逆序对数目

- 逆序对数目：i<j, a[i] > a[j]
- 枚举：$O(n^2)$. n <= 5000
- 利用归并排序的框架
  - j <= mid或i >= mid，递归处理
  - i < mid < j，合并的时候顺便求！
    - 1, 2, 4, 7, 9
    - 3, 5, 6, 10, 11
    - 取后一半元素时，前一半还剩多少个就是…
- 时间复杂度：O(nlogn)

快速排序

- 找一个数x
  – 让x左边的数都比x小
  – 让x右边的数都比x大
  – 分别给两边排序
  – 核心：如何调整x左右的数？
- 从两边往中间扫描
  – 在x左边第一个比x大的地方停下来
  – 在x右边第一个比x小的地方停下来
  – 两个交换，正好都满足要求

快速排序

- 例子：<u>1</u>, 8, 2, 9, 6, 4, 7, 3, <u>5</u>
  - 第一次交换8和5：1, <u>5</u>, 2, 9, 6, 4, 7, 3, <u>8</u>
  - 第二次交换9和3：1, 5, 2, <u>3</u>, 6, 4, 7, <u>9</u>, 8
  - 第三次交换6和4：1, 5, 2, 3, <u>4</u>, <u>6</u>, 7, 9, 8
  - 两个指针汇合，完成
  - 时间复杂度：O(n)

快速排序分析

- 最好情况：均分成两半，则是O(nlogn)
- 最坏情况：分成长度为1和n-1，则是$O(n^2)$
- 这是不是说明快速排序比归并排序差
  - 显然不是，否则就不会叫“快速排序”了嘛…
  - 加入随机数, 几乎可以保证是O(nlogn), 系数比归并排序小
  - 随机数让坏情况从数据转移到了随机运气

快速排序

- 一些小细节
  - n <= 10时用插入排序加速
  - x如何选？中间？随机（随机数产生开销）？
  - 警告！快速排序不稳定！原因？如何修改？
  - 最坏情况:数据→随机数

问题3: 求序列中第k大数

- 方法一: 先排序O(nlogn)
- 方法二: 借用快速排序的框架
  - 只需要根据k的大小只处理一边
  - 平均情况：O(n)

例题: 士兵排队问题

- n<=10,000个士兵在网格中
  - 位置用(x, y)表示
  - 士兵可以沿四个方向移动
  - 进入某一行且排在一起
  - 假设格子可以容纳所有士兵

分析

- 行：感觉在”中间”
  - 中位数 or 平均数？
  - 假设往下一行…
- 列？（请思考）

思考: 加权中位数

- X轴上有*n*个点，第i个点都的权值为正数*Vi*，要求在X轴上找出一点P，使sum{V_i*|x_p-x_i|}最小

计数排序

- 特殊的排序对象：0~100的整数(如分数)
- 开一个count[0..100]的数组，记录个数

```
for i:=1 to n do
  inc(count[score[i]]);
```

- 时间复杂度为O(n), 比快速排序还快
- 关键: 利用了关键码的范围

基于比较的排序时间下限

- 简单起见，不考虑相等的情形
- 可以获得多少信息？
  - 一次比较，两种结果
  - k次比较2^k种结果
- 需要获得多少信息？
  - n个数的排列有n!种
  - 最后只剩一种可能性时，排序完成

基于比较的排序时间下限

- 需要比较多少次？
  - 比较k次以后最好只能保证剩$n!/2^k$种可能性
  - $n!/2^k=1$时，即$k=\log(n!)$时排序完成
- $\log(n!) = \Omega(n\log n)$