9 文件与外排序

官方复习大纲中只有置换选择排序和多路归并两个板块, 因此笔记只记这些. 内外存和磁盘相关知识请移步 ics.

置换选择排序

思路

- 首先传入 m 个元素进内存建堆, 设置堆尾标志 $last = m - 1$, 然后只要 $last \geq 0$:
  - 堆的根节点传送到输出缓冲区, 记为 $mval$
  - 从输入缓冲区读入一个数 r :
    - 若 $r \geq mval$, 则把 r 放到根节点
    - 否则 $last$ 位置的元素放到根节点, $last- = 1$,
  - 刷新堆以获得新的 $mval$
- 算法结束后, 内存中也填满了未能处理的元素, 直接建堆等待下一顺串处理

分析

- 输出的一个顺串最小长度是 m, 最优长度为整个文件(正序输入), 平均情况为 $2m$

多路归并

- 对于每趟需要归并的 k 个顺串, 每步就是 k 个指针扫描, 直到全部 n 个顺串合并完成.
- 合并趟数 : $\lceil \log_k m \rceil$
- 优化:
  - 创建尽可能大的初始顺串
  - 把初始顺串长度作为权, 转化为Huffman树最优化问题, 得到一个 k叉Huffman 树, 称为**最佳归并树**

> 进一步优化? 每次比较 $k - 1$ 次开销还是太大了, 实际上一些比较可以保留

赢者树

- 完全二叉树结构, 每个叶子节点表示待归并顺串上的当前元素, 每个内部节点储存胜者 (最小值) 所在顺串的序号.

- 更新时只需输出对应叶子节点元素, 写入新元素后调整从叶子节点到根节点的路径, 时间复杂度为 $O(\log k)$.

败者树

- 胜者树的优化, 没有本质区别, 但在每个节点保留败者所在顺串的序号, 以及增加一个根节点储存最终胜者
- 也即优化了重构过程, 直接找父节点即可, 无需再与兄弟节点比较 (降低复杂度的常数)

时间复杂度

- 原始方法 : $O(nk)$
- 败者树方法 : $O(k + n\log k)$

> 为什么不用堆?
> 堆的每层调整都要对三个值 (父节点、两个子节点) 进行至少两次比较, 使用胜者败者树少一次比较, 败者树相对胜者树更少一次寻址