1 本质上还是要借助内排序完成,通过内外存之间的数据交换
2 外存永久存储,不易失,尽量减少访外存次数
3 外存访问分为的定位存取,以页块为单位
4 外存数据分成若干段,每次将一段读入内存排序->顺串/归并段
5 基本过程:置换选择排序->归并排序
6 置换选择排序:初始化最小堆(读 M 个记录,建立一个最小堆)->将最小值输出,如果下一个值不小于刚才的关键码,把它放到根节点,否则用 last 代替根节点,新的元素放到 last 位置,堆大小减一->重新建堆 形成的顺串平均长度 2M
7 归并排序:合并树高[logm + 1](向上取整),进行[logm]次扫描,2 个输入缓冲区(2 路归并),1 个输出缓冲区
8 减少归并次数:减少初始顺串的个数,增加归并的顺串数量
9 减少读写次数->huffman 最优化问题->最佳归并树,减少外存 IO 次数
10 赢者树/败者树
11 赢者树:用完全二叉树作为存储结构,叶节点用 L[1-n],内部结点用 B[1- (n - 1)](存放的是 L 的索引)
12 胜者树深度 s = [log2n] - 1,最底层(有内部结点的最底层)最左端内部结点 2^s,类似的最底层的内部结点数,最底层的外部结点数,最底层(真的最底层)外部结点之上的所有节点数(offset), 则有 p = i <= lowext ? (i + offset) / 2 : (i - lowext + n - 1) / 2;
13 通过两个选手比赛的分数确定比赛的赢家,树根记录全局的胜者,类似的修改胜者树(再比一轮)
14 败者树:父节点记录左右结点的败者,新增根节点,记录全局的胜者,简化重构过程
15 代码:生成败者树(沿右分支(结点编号为奇数)向上比赛) 重构败者树(根节点暂存胜者的索引,沿路径向上比赛)
16 k 路归并初始化败者树的时间为 O(k), 重构一次败者树的时间为 O(logk), 生成大小为 n 的顺串的时间为 O(k + nlogk)