第九章 外排序

1. FIFO(First in First out)方法和 LRU(Least recently used)方法是用来作为缓冲区替换算法的两种选择。FIFO 每次将最先进入缓冲区的结果替换出去；LRU 每次将距离上次使用时间最长的结果替换出去。假定我们有一个大小为 3 的缓冲区，和一个如下输入的输入序列：
2 3 1 4 0 3 2 4 1 2 4 3
(1) 请分别写出 FIFO 方法和 LRU 方法的运行结果（缓冲区的变化），并比较这两个算法，给出 FIFO 方法和 LRU 方法的主要区别。
(2) 想要减少访问内存的 miss rate，一个直观的想法是增大缓冲区的大小。请问这样做对于 LRU 和 FIFO 方法是否都是有效的？如果不是请举出反例。
(3) 描述一种你所知道的其他替换算法，或者你所设计的一种替换算法。并说明这个算法可能的优势和劣势。

解:
(1) FIFO: 4 1 3 LRU: 2 4 3
主要区别: LRU 方法会倾向于保留经常使用的数据,FIFO 方法则不考虑数据使用的次数,只关注数据第一次进缓存的时间
(2) 对于 FIFO 是有效的,因为在任何时刻,一个小的应用 FIFO 策略的缓存中的内容总会是一个大的 FIFO 缓存的真子集,miss rate 只可能减少
对于 LRU 也是有效的,小的 LRU 缓存也总会是大的 LRU 缓存的子集(大缓存驱逐的元素一定已经被小缓存驱逐了,小缓存当中的元素一定在大缓存中)
(3) LFU 方法,根据一个页面在缓存中被使用的次数(而不是最近一次使用的时间)来选择替换的页面,可能的优势:保留经常访问的页面,有些情况下会减少不命中次数,可能的劣势:对时间局部性差的程序应对不好.

2. 置换选择排序的核心思想是利用堆对数据进行处理。每输出一个值，就从缓冲区中读入下一个数。如果堆的大小是 M，一个顺串的最小长度就是 M 个记录，至少原来在堆中的那些记录将成为顺串的一部分。最好的情况下，例如输入已经被排序，有可能一次就把整个文件生成为一个顺串。
(1) 现在给出一组输入关键字(17, 2, 20, 40, 10, 19, 8, 13, 11, 25, 21, 7)，假设堆的大小是 5 且起始为空，请写出得到的初始顺串和最后堆的状态。排序时较小的元素在前。
(2) 置换选择排序一定要用堆来实现吗？请任意给出一个不同的实现，用上面的输入和设定比较一下两者的差异。

解:
(1) 初始顺串:2 10 17 19 20 25 40
最后堆的状态: 7 21 11 13 8
(2) 用有序的数组代替堆,其他的规则相同,如果数组的大小和堆的大小相同则输出的结果是一样的.
两者的差异在时间复杂度上,最小堆调整一次时间复杂度为 O(logn),有序数组每调整一次复杂度 O(n),并且最小堆建堆时间 O(n),数组排序时间最低 O(nlogn)(基于比较的排序).

3. 现在有 9 个长度不同的的顺串，其长度分别为 17, 40, 16, 55, 25, 11, 6, 21, 41。请用二路归并的方法对其进行归并。

(1) 构造最佳归并树。

(2) 根据最佳归并树计算每一趟和总的读记录次数。

(3) 在多路归并的时候，显然归并的路数 K 越大速度越快，那么是什么限制了 K 的大小？

解:

(1)

<table>
<tr><td>41</td><td>55</td><td>16</td><td>17</td><td>25</td><td>40</td><td>21</td><td>6</td><td>11</td></tr>
<tr><td colspan="2">96</td><td colspan="2">33</td><td>25</td><td>40</td><td>21</td><td colspan="2">17</td></tr>
<tr><td colspan="2">96</td><td colspan="3">58</td><td>40</td><td colspan="3">38</td></tr>
<tr><td colspan="2">96</td><td colspan="3">58</td><td colspan="4">78</td></tr>
<tr><td colspan="2">96</td><td colspan="7">136</td></tr>
<tr><td colspan="9">232</td></tr>
</table>

(2) 每一趟的读记录次数: 146 96 78 136 232

(3) 归并算法的复杂程度随着 k 的增加而上升

4. 胜者树和败者树都是完全二叉树，是树形选择排序的一种变型。每个叶子结点相当于一个选手，每个中间结点相当于一场比赛，每一层相当于一轮比赛。不同的是，胜者树的中间结点记录的是胜者的标号；而败者树的中间结点记录的败者的标号。

胜者树与败者树可以在 log(n)的时间内找到最值。任何一个叶子结点的值改变后，利用中间结点的信息，还是能够快速地找到最值。在 k 路归并排序中经常用到。

举例说明为什么败者树的访问内存次数要比胜者树少，并分析是什么原因造成的。

解:

假设外部结点的值分别为 1 2 3 4 5 6 7 8 则胜者树为 8 4 8 2 4 6 8 败者树为 8 4 2 6 1 3 5 7, 先假设取出 8 的结点,新的结点值为 0, 则胜者树的访存次数为:读六次写三次(三次比较,三次赋值), 败者树的访存次数也是一样的(三次比较,三次赋值),但是败者树的重构只利用了从树根到更新结点的值,而不是像胜者树一样还要用到兄弟节点的值

单纯的访存次数上没有数量级上的差别,但败者树访问的结点数目少于胜者树,意味着可以利用缓存的技术减少直接对内存的访问,同时败者树更新的过程中反复用到 B[0],它也可以在最后才写回内存,都会减少访问内存的次数

原因:只访问父节点而不涉及兄弟节点,反复应用 B[0]的值

5. 假设一个记录长为 32 个字节，一个块长 1024 个字节（每个块有 32 个记录），工作内存是 1MB（还有用于 I/O 缓冲区、程序变量等的其他存储空间）。使用置换选择和多路归并，其中归并算法只允许扫描两遍。预计能得到的文件最长为多少？并解释这个结果是怎样计算出来的。

解:
工作内存可以容纳 32k 个记录
平均情况下,置换选择会得到 64k 个记录的顺串
一个块为 1kb 则 1m 的内存可以进行 1024 路归并
一次归并得到长为(平均)2G 的顺串
两次扫描得到长为 2G*1024 是 2T 的顺串