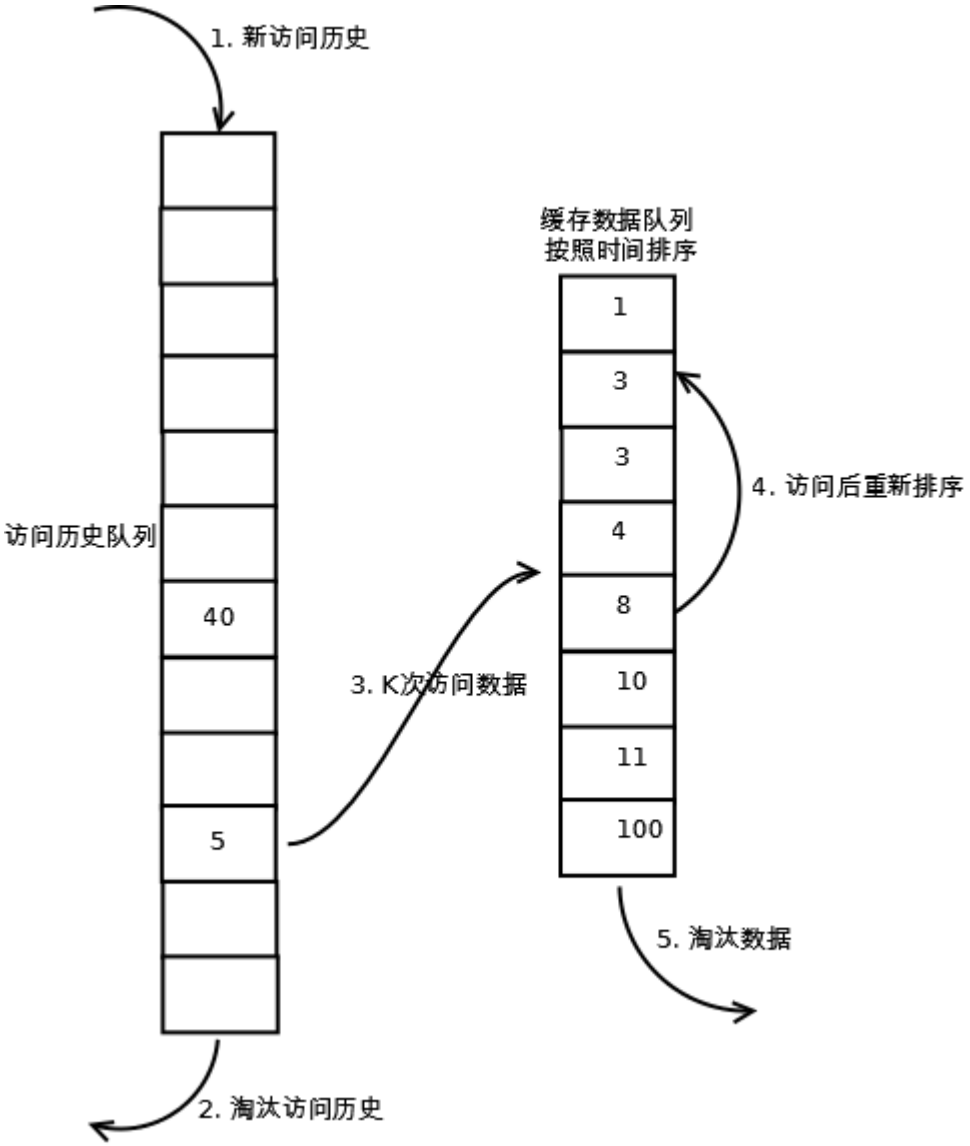
一、LRU-K算法

1、算法思想

LRU-K中的K代表最近使用的次数，因此LRU可以认为是LRU-1。LRU-K的主要目的是为了解决LRU算法“缓存污染”的问题，其核心思想是将“最近使用过1次”的判断标准扩展为“最近使用过K次”。

2、工作原理

相比LRU，LRU-K需要多维护一个队列，用于记录所有缓存数据被访问的历史。只有当数据的访问次数达到K次的时候，才将数据放入缓存。当需要淘汰数据时，LRU-K会淘汰第K次访问时间距当前时间最大的数据。详细实现如下



(1). 数据第一次被访问，加入到访问历史列表；

(2). 如果数据在访问历史列表里后没有达到K次访问，则按照一定规则（FIFO，LRU）淘汰；

(3). 当访问历史队列中的数据访问次数达到K次后，将数据索引从历史队列删除，将数据移到缓存队列中，并缓存此数据，缓存队列重新按照时间排序；

(4). 缓存数据队列中被再次访问后，重新排序；

(5). 需要淘汰数据时，淘汰缓存队列中排在末尾的数据，即：淘汰“倒数第K次访问离现在最久”的数据。

LRU-K具有LRU的优点，同时能够避免LRU的缺点，实际应用中LRU-2是综合各种因素后最优的选择，LRU-3或者更大的K值命中率会高，但适应性差，需要大量的数据访问才能将历史访问记录清除掉。

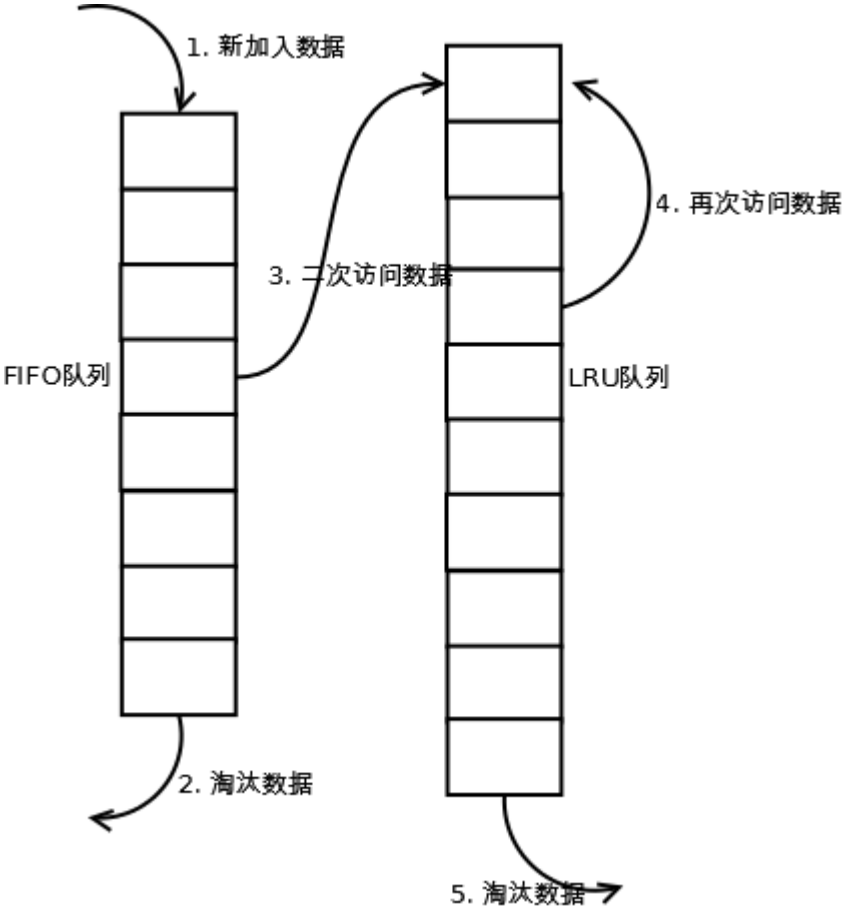
二、Two queues（2Q）

1、算法思想

该算法类似于LRU-2，不同点在于2Q将LRU-2算法中的访问历史队列（注意这不是缓存数据的）改为一个FIFO缓存队列，即：2Q算法有两个缓存队列，一个是FIFO队列，一个是LRU队列。

2、工作原理

当数据第一次访问时，2Q算法将数据缓存在FIFO队列里面，当数据第二次被访问时，则将数据从FIFO队列移到LRU队列里面，两个队列各自按照自己的方法淘汰数据。详细实现如下：



(1). 新访问的数据插入到FIFO队列；

(2). 如果数据在FIFO队列中一直没有被再次访问，则最终按照FIFO规则淘汰；

(3). 如果数据在FIFO队列中被再次访问，则将数据移到LRU队列头部；

(4). 如果数据在LRU队列再次被访问，则将数据移到LRU队列头部；

(5). LRU队列淘汰末尾的数据。

参考资料：

- The LRU-K Page Replacement Algorithm For Database Disk Buffering
- 2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm