第0类：没有被访问，没有被修改。

第1类：没有被访问，已被修改。

第2类：已被访问，没有被修改。

第3类：已被访问，已被修改。

尽管第1类初看起来似乎是不可能的，但是一个第3类的页面在它的R位被时钟中断清零后就成了第1类。时钟中断不清除M位是因为在决定一个页面是否需要写回磁盘时将用到这个信息。清除R位而不清除M位产生了第1类页面。

NRU（Not Recently Used，最近未使用）算法随机地从类编号最小的非空类中挑选一个页面淘汰之。这个算法隐含的意思是，在最近一个时钟滴答中（典型的时间是大约20ms）淘汰一个没有被访问的已修改页面要比淘汰一个被频繁使用的“干净”页面好。NRU主要优点是易于理解和能够有效地被实现，虽然它的性能不是最好的，但是已经够用了。

3.4.3 先进先出页面置换算法

另一种开销较小的页面置换算法是FIFO（First-In First-Out，先进先出）算法。为了解释它是怎样工作的，我们设想有一个超级市场，它有足够的货架能展示k种不同的商品。有一天，某家公司介绍了一种新的方便食品——即食的、冷冻干燥的、可以用微波炉加热的酸乳酪，这个产品非常成功，所以容量有限的超市必须撤掉一种旧的商品以便能够展示该新产品。

一种可能的解决方法就是找到该超级市场中库存时间最长的商品并将其替换掉（比如某种120年以前就开始卖的商品），理由是现在已经没有人喜欢它了。这实际上相当于超级市场有一个按照引进时间排列的所有商品的链表。新的商品被加到链表的尾部，链表头上的商品则被撤掉。

同样的思想也可以应用在页面置换算法中。由操作系统维护一个所有当前在内存中的页面的链表，最新进入的页面放在表尾，最久进入的页面放在表头。当发生缺页中断时，淘汰表头的页面并把新调入的页面加到表尾。当FIFO用在超级市场时，可能会淘汰剃须膏，但也可能淘汰面粉、盐或黄油这一类常用商品。因此，当它应用在计算机上时也会引起同样的问题，由于这一原因，很少使用纯粹的FIFO算法。

3.4.4 第二次机会页面置换算法

FIFO算法可能会把经常使用的页面置换出去，为了避免这一问题，对该算法做一个简单的修改：检查最老页面的R位。如果R位是0，那么这个页面既老又没有被使用，可以立刻置换掉；如果是1，就将R位清0，并把该页面放到链表的尾端，修改它的装入时间使它就像刚装入的一样，然后继续搜索。

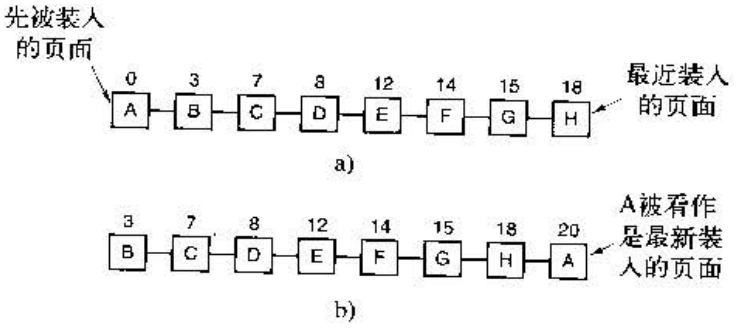

图3-15 第二次机会算法的操作（页面上面的数字是装入时间）：a) 按先进先出的方法排列的页面；b) 在时间20发生缺页中断并且A的R位已经设置时的页面链表

这一算法称为第二次机会（second chance）算法，如图3-15所示。在图3-15a中我们看到页面A到页面H按照进入内存的时间顺序保存在链表中。

假设在时间20发生了一次缺页中断，这时最老的页面是A，它是在时刻0到达的。如果A的R位是0，则将它淘汰出内存，或者把它写回磁盘（如果它已被修改过），或者只是简单地放弃（如果它是“干净”的）；另一方面，如果其R位已经设置了，则将A放到链表的尾部并且重新设置“装入时间”为当前时刻（20），然后清除R位。然后从B页面开始继续搜索合适的页面。

第二次机会算法就是寻找一个最近的时钟间隔以来没有被访问过的页面。如果所有的页面都被访问过了，该算法就简化为纯粹的FIFO算法。特别地，想象一下，假设图3-15a中所有的页面的R位都被设置了，操作系统将会一个接一个地把每个页面都移动到链表的尾部并清除被移动的页面的R位。最后算法又将回到页面A，此时它的R位已经被清除了，因此A页面将被淘汰，所以这个算法总是可以结束的。