**1）最佳置换算法（OPT）（理想置换算法）**

**最佳置换法（OPT）：选择将来不再使用或在最远的将来才被访问的页调换出去（不便于实现）**

　　这是一种理想情况下的页面置换算法，但实际上是不可能实现的。该算法的基本思想是：发生缺页时，有些页面在内存中，其中有一页将很快被访问（也包含紧接着的下一条指令的那页），而其他页面则可能要到10、100或者1000条指令后才会被访问，每个页面都可以用在该页面首次被访问前所要执行的指令数进行标记。最佳页面置换算法只是简单地规定：**标记最大的页应该被置换**。这个算法唯一的一个问题就是它无法实现。当缺页发生时，操作系统无法知道各个页面下一次是在什么时候被访问。虽然这个算法不可能实现，但是最佳页面置换算法可以用于对可实现算法的性能进行衡量比较。  
**2）先进先出置换算法（FIFO）**  
　　最简单的页面置换算法是先入先出（FIFO）法。这种算法的实质是，**总是选择在主存中停留时间最长（即最老）的一页置换，即先进入内存的页，先退出内存。**理由是：最早调入内存的页，其不再被使用的可能性比刚调入内存的可能性大。建立一个FIFO队列，收容所有在内存中的页。被置换页面总是在队列头上进行。当一个页面被放入内存时，就把它插在队尾上。  
　　这种算法只是在按线性顺序访问地址空间时才是理想的，否则效率不高。**因为那些常被访问的页，往往在主存中也停留得最久，结果它们因变“老”而不得不被置换出去**。  
　　FIFO的另一个缺点是，它有一种异常现象，即在增加存储块的情况下，反而使缺页中断率增加了。当然，导致这种异常现象的页面走向实际上是很少见的。  
　　3）最近最久未使用（LRU）算法

　　FIFO算法和OPT算法之间的主要差别是，**FIFO算法利用页面进入内存后的时间长短作为置换依据，而OPT算法的依据是将来使用页面的时间**。如果以最近的过去作为不久将来的近似，那么就可以把过去最长一段时间里不曾被使用的页面置换掉。**它的实质是，当需要置换一页时，选择在最近一段时间里最久没有使用过的页面予以置换。这种算法就称为最久未使用算法**（LeastRecentlyUsed，LRU）。LRU算法是与每个页面最后使用的时间有关的。当必须置换一个页面时，LRU算法选择过去一段时间里最久未被使用的页面。

**4）最少使用（LFU）置换算法**

在采用最少使用置换算法时，应为在内存中的每个页面设置一个移位寄存器，用来记录该页面被访问的频率。该置换算法选择在之前时期使用最少的页面作为淘汰页。由于存储器具有较高的访问速度，例如100 ns，在1 ms时间内可能对某页面连续访问成千上万次，因此，通常不能直接利用计数器来记录某页被访问的次数，而是采用移位寄存器方式。每次访问某页时，便将该移位寄存器的最高位置1，再每隔一定时间(例如100 ns)右移一次。这样，**在最近一段时间使用最少的页面将是最小的页。**

**LFU置换算法的页面访问图与LRU置换算法的访问图完全相同；**或者说，利用这样一套硬件既可实现LRU算法，又可实现LFU算法。应该指出，LFU算法并不能真正反映出页面的使用情况，因为在每一时间间隔内，只是用寄存器的一位来记录页的使用情况，因此，访问一次和访问10 000次是等效的。

**5）第二次机会算法（类似LRU2）**

**第二次机会算法的基本思想是与FIFO相同的，但是有所改进，避免把经常使用的页面置换出去。**当选择置换页面时，检查它的访问位。如果是0，就淘汰这页；如果访问位是1，就给它第二次机会，并选择下一个FIFO页面。当一个页面得到第二次机会时，它的访问位就清为0，它的到达时间就置为当前时间。如果该页在此期间被访问过，则访问位置1。这样给了第二次机会的页面将不被淘汰，直至所有其他页面被淘汰过（或者也给了第二次机会）。因此，如果一个页面经常使用，它的访问位总保持为1，它就从来不会被淘汰出去。

第二次机会算法可视为一个环形队列。用一个指针指示哪一页是下面要淘汰的。当需要一个存储块时，指针就前进，直至找到访问位是0的页。随着指针的前进，把访问位就清为0。在最坏的情况下，所有的访问位都是1，指针要通过整个队列一周，每个页都给第二次机会。这时就退化成FIFO算法了。