地址映射过程中，若在页面中发现所要访问的页面不在内存中，则产生缺页中断，当发生缺页中断时，如果操作系统内存中没有空闲页面，则操作系统必须在内存选择一个页面将其移除内存，以便为即将调入的页面让出空间。而用来选择淘汰哪一页的规则叫做页面置换算法。

目标：尽可能减少页面的调动次数。

FIFO先进先出：置换先进来的页面

LRU最近最久未使用：选择最长时间没被引用的页面。

LFU最不常用：置换访问次数最少的页面。

页面的频繁更换，导致整个系统效率急剧下降，这个现象称为内存抖动。

抖动一般是内存分配算法不好，内存太小或者程序的算法不佳引起的页面频繁从内存调入调出。

belady现象：采用FIFO算法时，如果对一个进程未分配它所要求的全部页面，有时就会出现分配的页面数增多但缺页率反而提高的异常现象。

1.最佳置换算法：从主存中移出永远不再需要的页面，如无这样的页面存在，则选择最长时间不需要访问的页面。于所选择的被淘汰页面将是以后永不使用的，或者是在最长时间内不再被访问的页面，这样可以保证获得最低的缺页率。最佳置换算法可以用来评价其他算法。

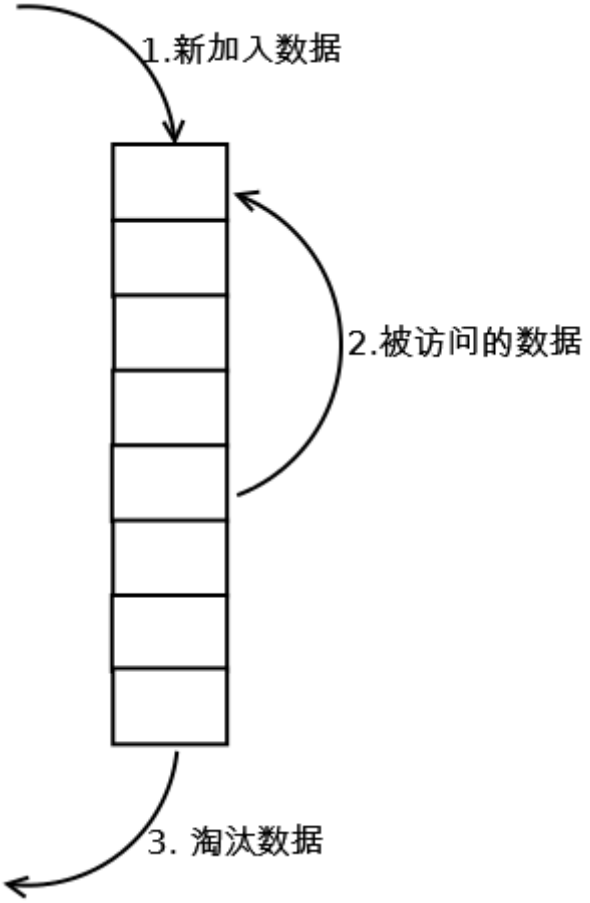
2.先进先出置换算法：是最简单的页面置换算法。这种算法的基本思想是：当需要淘汰一个页面时，总是选择驻留主存时间最长的页面进行淘汰，即先进入主存的页面先淘汰。其理由是：最早调入主存的页面不再被使用的可能性最大。

3.LRU最近最久未使用算法：这种算法利用局部性原理，根据一个作业在执行过程中过去的页面访问历史来推测未来的行为。它认为过去一段时间里不曾被访问过的页面，在最近的将来可能也不会再被访问。所以，这种算法的实质是：当需要淘汰一个页面时，总是选择在最近一段时间内最久不用的页面予以淘汰。

4.LFU最近最少使用：它是基于“如果一个数据在最近一段时间内使用次数很少，那么在将来一段时间内被使用的可能性也很小”的思路。

LRU算法：

LRU算法是最近最少使用算法，其核心思想就是“如果数据最近被访问过，那么它将来被访问的几率会更高”。最常见的就是使用一个链表保存缓存数据，详细算法实现如下：



1.新数据插入到链表的头部。

2.每当缓存命中时，则将数据移动到链表的头部

3.当链表满的时候，将链表尾部的数据丢弃。

命中率问题分析：当存在热点数据的时候，LRU的效率会很好，但偶发性、周期性的批量操作会使得命中率下降很快，缓存中脏数据会很多。

复杂度：实现比较简单

代价：命中时需要遍历链表，找到命中的数据块索引，然后需要将数据移动到头部