1. **假设磁盘块大小为 8 KB，块中存储 200 字节的定长记录，块首部只包括一个 8 字节的模式指针和一个偏移量表。对于插入块内的每条记录，在偏移量表中都增加一个 2 字节的指针指向该记录。假设每天向块内插入 4 条记录（空间不足时允许插入部分记录后结束全部操作），删除 2 条记录。假设每天的删除记录操作总是发生在插入记录之前，删除记录使用一个"删除标记"代替记录在偏移量表中的指针。给定一个磁盘块，如果刚开始块是空的，则几天后不能再向该块内插入记录？此时，该块内一共有多少条记录？**

磁盘块共有 8192 B，模式指针占 8 B，用于存储记录和偏移量表的空间还剩

$$8192 - 8 = 8184 \text{ B}$$

每插入一条记录，需要消耗定长记录空间 200 B + 偏移量表指针 2 B = 202 B

删除一条记录时，只释放偏移量表中的指针 2 B，不释放定长记录占用的 200 B 空间

因此，每天一共会消耗 $202 * 4 - 2 * 2 = 804$ B

$8184 / 804 = 10$ (天)（向下取整）

因此，十天后这个磁盘块就会变满，不能再向块内插入记录

每天增加 2 条记录，10 天后一共增加了 20 条记录，最后一天在插入第一条记录的时候磁盘块就已经满了，因此块内一共有 20 条记录

2. 假设我们采用 LRU 作为缓冲区置换策略，当我们向 Buffer Manager 发出一个读页请求时，请讨论一下：

（1） 如果页不在缓冲区中，我们需要从磁盘中读入该页。请问如何才能在缓冲区不满的时候快速地返回一个 free 的 frame？请给出至少两种策略，并分析一下各自的时间复杂度。

i. 维护一个空闲链表，在初始化时将缓冲区的所有 frame 都放入这个链表，每次分配时将链表头的 frame 分配给缓冲区，当释放某个 frame 时，将其加入链尾。

优点：查询空闲 frame 的时间复杂度为 O(1)，查询快。
缺点：需要额外的内存来维护链表，缓冲区满时，该链表冗余。

ii. 维护一个位图来标记空闲 frame，用位图来表示 frame 是否空闲。每一位对应一个 frame，1 表示占用，0 表示空闲。分配 frame 时，从位图中找到第一个 0 的位置。

优点：占用空间更小，结构更紧凑。
缺点：查找的时间复杂度为 O(n)，查找偏慢。

（2） 如何才能快速地判断所请求的页是否在缓冲区中？如果请求的页在缓冲区中，如何快速返回该页对应的 frame 地址？请给出至少两种策略，并分析一下各自的时间复杂度。

i. 维护一个哈希表来存储每个 frame 的映射关系，键为页号，值为 frame 地址。

优点：查找、插入速度的时间复杂度为 O(1)，查找速度快。
缺点：需要更多的内存来存储哈希表，并且如果哈希表冲突过多可能出现性能退化。

ii. 维护一个平衡二叉搜索树来反映每个 frame 的映射关系

优点：在缓冲区很大的情况下稳定性也较好，一般不会出现性能退化的情况
缺点：管理成本和空间开销稍高，且查找略慢，查找的时间复杂度为 O(logn)

（3） 我们在讲义上介绍了 SSD 感知的 CF-LRU 算法，即 Clean-First LRU 算法。该算法虽然看起来可以减少对 SSD 的写操作，但依然存在一些问题。请分析一下该算法的主要缺点有哪些？给出两点，并简要解释你的理由。

i. CF-LRU 依赖于优先替换干净页的策略，但当缓存中干净页数量较少或几乎没有时，CF-LRU 的效果会显著下降。此时，它不得不选择脏页进行替换，导致写回开销依然存在。特别是在高频写入的场景下，页面很容易变成脏页，CF-LRU 的优化优势可能变得微乎其微，从而难以有效减少写操作。

ii. CF-LRU 优先选择干净页来替换，但它并不主动清理脏页（例如，提前将脏页写回存储），导致脏页可能会在缓存中堆积，影响缓存的命中率和替换效率。此外，如果系统突然需要大量空间，CF-LRU 可能面临大量脏页写回的延迟，增加了系统负担和响应时间。