11.3

a. 由于非叶节点在内存中，因此查找插入所需叶页的页码的成本可以忽略不计。在叶级上，读取需要一次随机磁盘访问，更新需要一次随机磁盘访问，以及写入一个页面的成本。导致叶子节点分裂的插入需要额外的页写入。因此，要构建一个包含任何条目的 B+树，它最多需要 2 *nr 次随机磁盘访问和 nr+2 * (nr/f)页写入。第二部分的成本来自于在最坏的情况下每个叶子都被填满了一半，所以出现的分裂次数是 2*nr/f。
上面的公式忽略了编写非叶节点的代价，因为我们假设它们在内存中，但实际上它们最终也会被写入。这个代价近似于 2* (nr/f)/f，这是叶子上方的内部节点的数量；我们可以添加更多的项来考虑更高级别的节点，但是这些项比叶节点的数量要小得多，可以忽略。

11.4

a.重建索引来恢复顺序性；

b.
1.B+树的 n 快单元和每个节点都被填了一半，充满度为 1/4；

2.不可能，因为拆分是有顺序的；

3. 在常规的 B+树结构中，叶页可能不是顺序的，因此在最坏的情况下，每个叶页需要一次搜索。
使用一次一块的方法，对于每 n 个节点块，我们将至少有 n/2 个叶子。每个 n 个节点的块都可以用一个码来读取。所以最坏的情况是减少 n / 2 倍；

4. 允许在同一块的节点之间进行再分配，不需要额外的查找。而在常规的 B+-树中，我们需要的查找数量与涉及再分配的叶页数量一样多。这使得叶块的再分配在这个方案中更加有效。最糟糕的情况是入住率接近 50%；