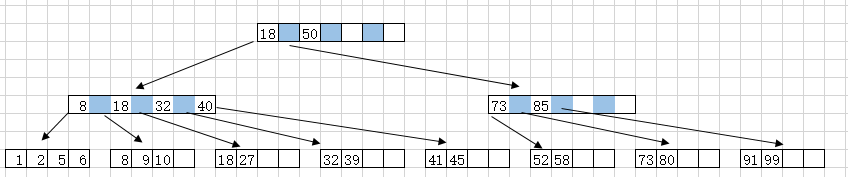
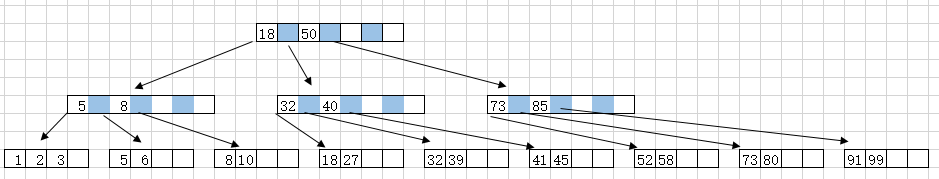
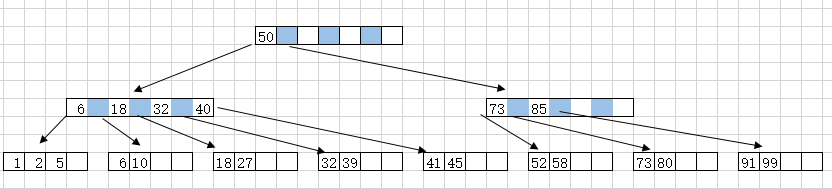
**10.1**

1. 如图：

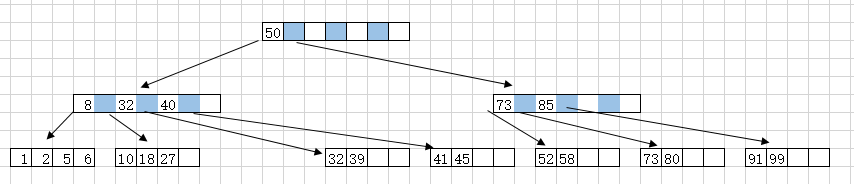




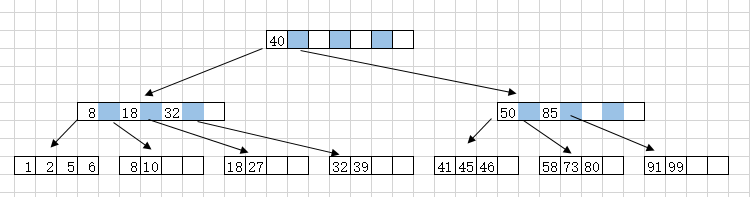
总共要进行2次新建页，4次页读取和5次页写入操作。



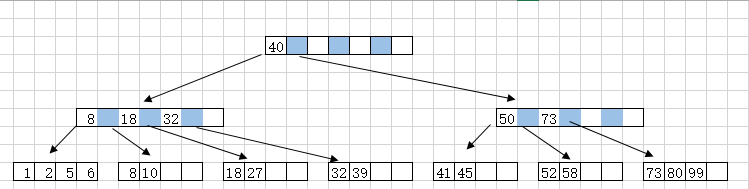
4.



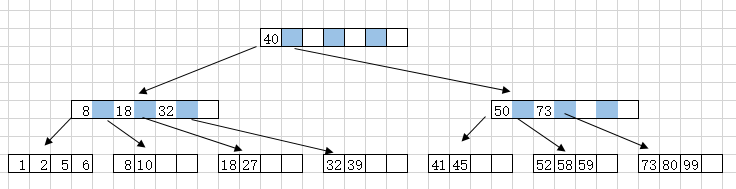
5.



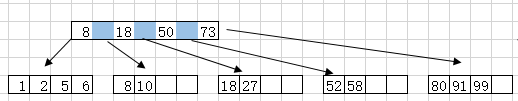
6.



7.



8.

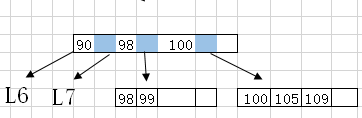


**10.2**

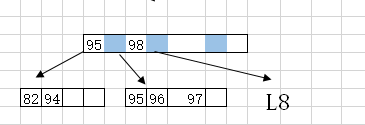
1.

I1、I2、L2、L3、L4……L8

2.



3.



4.

在[65,79)范围内的数均可以。

5.

A < 10，10 <= B <20 ，20 <= C <30 ;

6.

由于ISAM没有最小和最大节点容量，故不会发生结构变化。故插入任何值都不会导致树的高度增加。

7.

只要对L4、L5、L7、L8中的一个节点插入9个值就可以形成3个溢出页。

前四次插入形成第一个溢出页，第5~8次插入形成第二个溢出页，第9个插入形成第三个溢出页。

**13.1**

|  |  |  |  |
| --- | --- | --- | --- |
|  | A | B | C |
| 1 | ⌈10000/3⌉= 3334 | ⌈20000/5⌉=4000 | ⌈2000000/17⌉=117648 |
| 2 | ⌈log2 3334⌉ + 1 = 13 | ⌈log4 4000⌉ + 1 = 7 | ⌈log16 117648⌉ + 1 = 6 |
| 3 | 2\*10000\*13 = 260 000 | 2\*20000\*7 = 280 000 | 2\*2000000\*6 = 24 000 000 |
| 4 | 101 | 142 | 1415 |

第四题中，B需满足 B − 1 ≥⌈N/B⌉。

**13.3**

1. 每个数据页可用的空间为512-12=500 bytes。故每页可存储 ⌊500/48⌋=10个数据，总共需要有4500/10=450页来存储所有文件。总共会生成⌈450/4⌉=113个有序文件，除最后一个文件是2页长外，其他都是4页长。

2.⌈log3 133 ⌉+ 1=6

3. 每轮有450页被读写了两次，共有六轮排序，故总的I/O为：2 ∗ 450 ∗ 6 = 5400。

4. 由B − 1 ≥⌈N/B⌉，B=4有N=12，故最多有12个文件，每个文件可存放10个元组，故最多可处理含有120个元组的文件。当缓冲区数目为257时，这个数目是657920

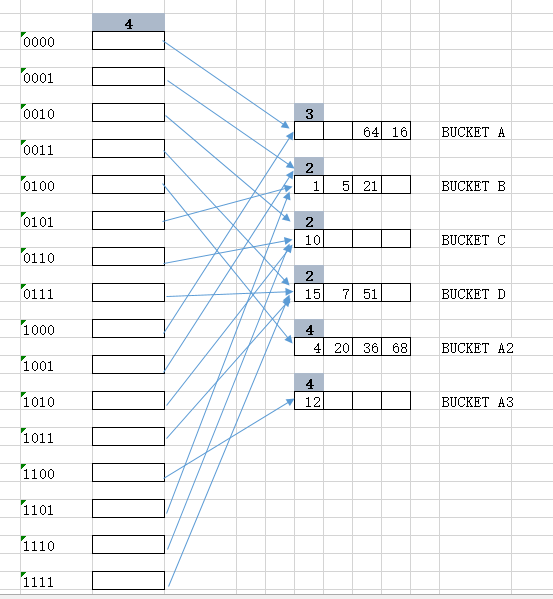
5.

1. 假设空间利用率为67%，则需顺序读取450/0.67 = 600个页。
2. 共有4500个元组，这里就有4500次I/O，另外索引的读取共需：⌈(4500 ∗ 12)/(512 ∗ 0.67))⌉=158次（假设空间利用率为67%）故总的I/O为4500+158=4658。
3. 聚簇索引时的I/O为：65792/0.67 = 98197

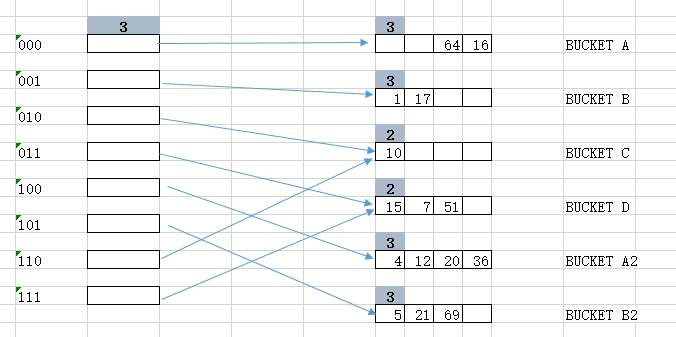
非聚簇索引的I/O为：657920 + ⌈(657920 ∗ 12)/(512 ∗ 0.67))⌉ = 680935.

**11.1**

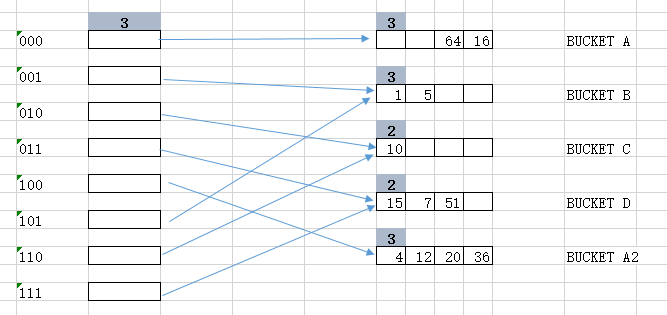
1. 在可能有删除操作的情况下，所有数据都有可能是最后一个插入的数据。
2. 在桶A2中的数据不可能是最后一个插入的，因为A和A2的数据数目总和是6，所以最后一次插入不会导致桶分裂。
3. 如果是从2个桶开始插入的话，导致最后一次分裂的数据项应该在桶A和A2中，如果是从1个桶开始分裂的话，唯独桶C中的10不可能是最后一个导致桶分裂的数据项，因为C桶原来可能是和B或D桶一起，但这两种情况合成的桶的数据项数目都是4，达不到分裂的条件。



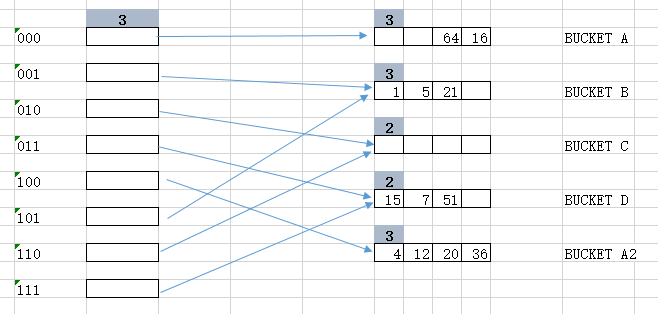






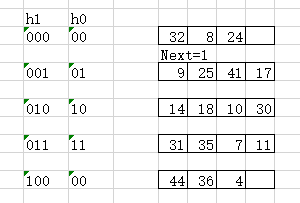


1. 不需要合并，因为此时010和110都指向桶C。

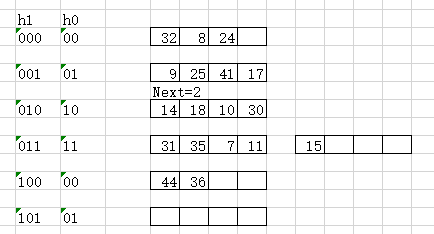


**11.2**

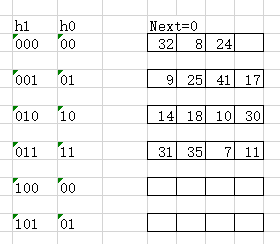
1. 所有项都有可能。
2. 如果插入的最后一项有h0=00,那么会导致分裂。
3. 其必须具有h0=00的性质。













插入f0=11的桶：63, 127, 255, 511, 1023。在能减少溢出链长度的插入之前最多可插入3个数据。

**11.3**

1. 用来在动态操作的时候判断是否需要进行桶分裂操作或者合并操作。
2. 2个桶。当目录加倍完成后引发这个加倍操作的桶分裂出来的两个桶分别只有一个指针指向他们。删除操作后目录的大小不会变化，如果规定对空桶不做删除合并处理的话。
3. 不能，如果目录不在内存中的话就要从磁盘读取，而目录文件又可能很大。
4. 假设插入一系列形式为2m的数其中m>k，则在第一次分裂时会形成2k，但其中只有一个桶存储着数据， 空间使用率为2-k
5. 不需要。
6. 没有溢出页，那么所有的重复值都会存储到同一个桶中，如果重复值的个数大于桶的深度的话就会很难处理，可能需要扩展桶深度。

**11.4**

1. 如果哈希函数好的话溢出页会很少，查找到之后大多数情况下可以通过一次I/O把相应的数据页读出。
2. 不能，有可能有溢出页。
3. 当所有数据都在一个桶中的时候则是实际代价。
4. 依然假设每次都插入同一个桶，共插入n个溢出页，则每产生一个溢出页就会有2个数据页产生，其中一个为空。设开始有m个数据页，则空间利用率为。