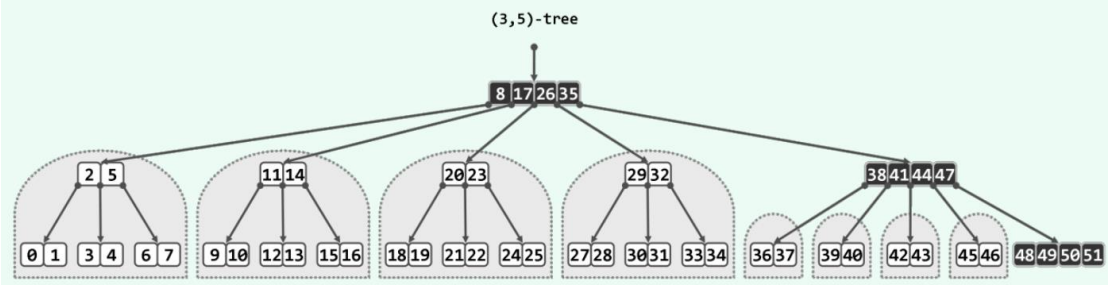
Q: 现有一组各异的关键码，插入到一颗初始为空的 m 阶 B-树中，设 m<<n

1）按照何种顺序插入这批关键码，能使得得到的 B-树高度最大?

2） 按照何种顺序插入，能使高度最小?

1） 保证 B-树达到最大高度的一种简明方法，就是按单调次序插入所有关键码。不妨设 m 为奇数(偶数的情况方法类似)。比如，按单调递增次序将: { 0, 1, 2, ..., 51 }插入初始为空的 5 阶 B-树，所生成 B-树的结构应如图所示。



一般地，不难验证:在按递增次序插入各关键码的过程中，最右侧通路(沿途节点在图中以 黑色示意)以下的所有子树(以虚框包围的各组白色节点)，始终都属于“稀疏临界”状态。在处于这种状态的子树中，任一节点的删除，都将引起持续的合并操作，并导致高度的下降。

因此，若阶次为 m，则此类子树中的每个节点均有⌈m/2⌉分支;若其高度为 h，则其下所含的 外部节点总数应为⌈m/2⌉h，内部节点总数应为⌈m/2⌉h - 1。在上例中 m = 5，于是高度为 h = 1 的(4 棵)此类子树必然包含 3 个外部节点和 2 个内部节点，高度为 h = 2 的(4 棵)此类子树必然 包含 9 个外部节点和 8 个内部节点。

实际上若采用单调递增的次序，则每次插入的关键码在当前都属最大。因此，插入算法必然 沿着最右侧通路做查找并确定其插入位置;而一旦出现上溢现象，也只能沿最右侧通路实施分裂 操作。如此，尽管最右侧通路下属的子树可能会增加，但它们始终保持稀疏临界状态。

可知:如此插入[0, n)而生成的 m 阶 B-树，高度应为: h = hmax = log⌈m/2⌉⌊(n + 1)/2⌋ + 1

仍以上述 B-树为例，m = 5，n = 52，故树高应为: h = log⌈5/2⌉⌊(52+1)/2⌋ +1 = 3

若继续插入下一关键码 52, 则在持续分裂 3 次之后, 树高将增至: h = log⌈5/2⌉⌊(53+1)/2⌋ +1 = 4

依然是此时所能达到的最大树高。

2） 在尽可能均匀插入时 B-Tree 的高度最小。顺序不一，这里提供一种插入的顺序:

可推导得到 B- Tree 的最小高度为 ceil(logm(N+1)),可以只考虑 n = m^k-1 的情况，将这 n 个关键码排序，记录其顺序序号分别为 num = 1，2，…，n

将序号 num 用 m 进制表示，得到的数应为 k 位 m 进制数，将该数字左右翻转后得到数字 num',然后将关键码按照 num'的顺序插入即可，下为例子:

考虑阶次为 3 的树，要插入一组共 26 个互异的关键码，为方便起见，记 26 个关键码的值为 1，2，3，…，26，他们排序后的序号分别为 1，2，3，…，26。将这些关键码用 3 进制表示后左右翻转，有:

num 十进制表示	num 三进制表示	左右翻转得到 num' 三进制表示	num'十进制表示
1	001	100	9

2	002	200	18
3	010	010	3
4	011	110	12
5	012	210	21
6	020	020	6
7	021	120	15
8	022	220	24
9	100	001	1
10	101	101	10
11	102	201	19
12	110	011	4
13	111	111	13
14	112	211	22
15	120	021	7
16	121	121	16
17	122	221	25
18	200	002	2
19	201	102	11
20	202	202	20
21	210	012	5
22	211	112	14
23	212	212	23
24	220	022	8
25	221	122	17
26	222	222	26

按照关键码对应的 num'顺序将关键码排序，得到这 26 个关键码的插入顺序应当为：9，18，3，12，21，6，15，24，1，10，19，4，13，22，7，16，25，2，11，20，5，8，17

插入后得到的树为：