

7.1.1. 哈希表问题

哈希函数 $\text{Hash}(\text{Key}) = \text{Key} \bmod 13$ ，处理冲突函数 $H_i = (\text{Hash}(\text{key}) + d_i) \bmod m$ (m 为哈希表长度， $d_i = 1, 2, 3, 4, 5, \dots$)。

给定关键字序列：14, 02, 24, 29, 01, 33, 79, 41, 53, 46, 68, 90。

试在 $S.\text{elem}[0..15]$ 存储空间上，解答如下问题：

- (1) 构造如上给定关键字序列的哈希表，统计每个关键字的查找次数
- (2) 计算其查找成功时的平均查找长度 ASL
- (3) 计算其查找不成功时的平均查找长度 ASL

7. 1. 1 解：处理冲突函数 $H_i=(Hash(key)+d_i) \bmod m$ (m 为哈希表长度 16， $d_i=1,2,3,\cdots$)

(1) 构造哈希表及每个关键字的查找次数

哈希表 S.elem[0..15]

地址	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字		14	02	29	01	79	41	33	53	46	68	24	90			
查找成功时的 比较次数		1	1	1	4	5	5	1	8	3	8	1	1			

(2) 计算关键字查找成功时的平均查找长度 $ASL=(\sum \text{各关键字查找成功时的比较次数})/(\text{关键字的个数})$

查找成功时的 $ASL=(1+1+1+4+5+5+1+8+3+8+1+1)/12=39/12=13/4=3.25$

(3) 计算关键字查找不成功时的平均查找长度 $ASL=(\sum \text{各地址查找不成功时的比较次数})/(\text{不成功地址个数})$

根据哈希函数 $Hash(Key)=Key \bmod 13$ ，则第一次计算的哈希地址值只可能是 0~12，这些哈希地址查找不成功的比较次数为它到第一个没有存储关键字的地址的比较次数。因此查找不成功的比较次数如下表所示：

地址	0	1	2	3	4	5	6	7	8	9	10	11	12
查找不成功时的 比较次数	1	13	12	11	10	9	8	7	6	5	4	3	2

查找不成功时的 $ASL=(1+13+12+11+10+9+8+7+6+5+4+3+2)/13=91/13=7$

7.1.2 哈希表问题

哈希函数 $\text{Hash}(\text{Key}) = \text{Key} \text{ MOD } 13$ ，处理冲突函数 $H_i = (\text{Hash}(\text{key}) + d_i) \text{ mod } m$ （若 $H_i < 0$ ，则 $H_i = H_i + m$ ）， m 为哈希表长度 16， $d_i = 1, -1, 2, -2, 3, -3, 4, -4, \dots$ ）。

给定关键字序列：14, 02, 24, 29, 01, 33, 79, 41, 53, 46, 68, 90。

试在 $S.\text{elem}[0..15]$ 存储空间上，解答如下问题：

- （1）构造如上给定关键字序列的哈希表，统计每个关键字的查找次数
- （2）计算其查找成功时的平均查找长度 ASL
- （3）计算其查找不成功时的平均查找长度 ASL

7.1.2. 解：处理冲突函数 $H_i = (\text{Hash}(\text{key}) + d_i) \bmod m$ (若 $H_i < 0$, 则 $H_i = H_i + m$), m 为哈希表长度 16, $d_i = 1, -1, 2, -2, 3, -3, \dots$

(1) 构造哈希表及每个关键字的查找次数。给定关键字序列为 14, 02, 24, 29, 01, 33, 79, 41, 53, 46, 68, 90。

哈希表 $S.\text{elem}[0..15] = \{01, 14, 02, 29, 41, 68, \#, 33, 46, \#, \#, 24, 90, \#, 53, 79\}$

地址	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字	01	14	02	29	41	68		33	46			24	90		53	79
查找成功时的 比较次数	3	1	1	1	4	4		1	2			1	1		7	5

(2) 计算关键字查找成功时的平均查找长度 $ASL = (\sum \text{各关键字查找成功时的比较次数}) / (\text{关键字的个数})$

查找成功时的 $ASL = (3+1+1+1+4+4+1+2+1+1+7+5) / 12 = 31/12 \approx 2.58$

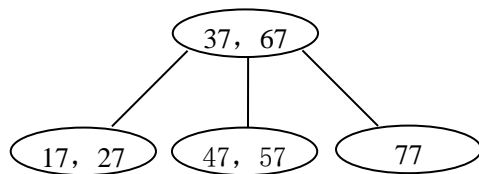
(3) 计算关键字查找不成功时的平均查找长度 $ASL = (\sum \text{各地址查找不成功时的比较次数}) / (\text{不成功地址个数})$

根据哈希函数 $\text{Hash}(\text{Key}) = \text{Key} \bmod 13$, 则第一次计算的哈希地址值只可能是 0~12, 这些哈希地址查找不成功的比较次数为它到第一个没有存储关键字的地址的比较次数。因此查找不成功的比较次数如下表所示:

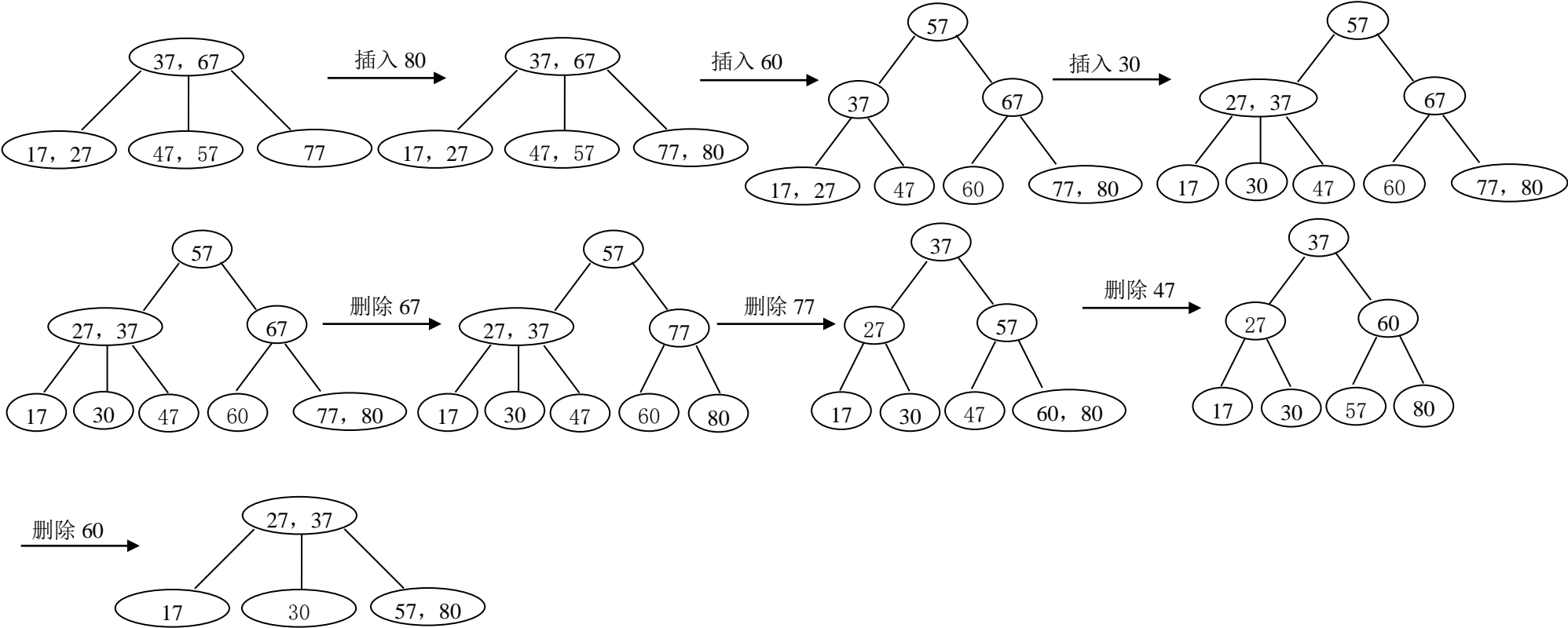
地址	0	1	2	3	4	5	6	7	8	9	10	11	12
查找不成功时的 比较次数	7	9	8	6	4	2	1	3	2	1	1	3	2

查找不成功时的 $ASL = (7+9+8+6+4+2+1+3+2+1+1+3+2) / 13 = 49/13 \approx 3.77$

7.2. 请在下面的 3 阶（2-3）B_树上依次插入关键字 80、60、30，然后再依次删除关键字 67、77、47、60。试画出每次操作后的 B_树结构。



7.2. 解:



7.3. 平衡二叉树

已知关键字序列为 {47, 57, 87, 77, 67, 27, 07, 97, 57, 37, 17}，其中，x 表示值为 x 的另一关键字。按如下要求完成本题。

(1) 针对给定关键字序列的不同排列，所构造出的不同形态的二叉排序树中，在最好和最坏情况下，该二叉排序树的高度各是多少？

(2) 根据给定的关键字序列，构造一棵平衡二叉排序树。

(3) 在等概率的情况下，计算查找成功时该平衡二叉排序树的 $ASL_{成功}$ 。

(4) 在等概率的情况下，计算查找不成功时该平衡二叉排序树的 $ASL_{不成功}$ 。

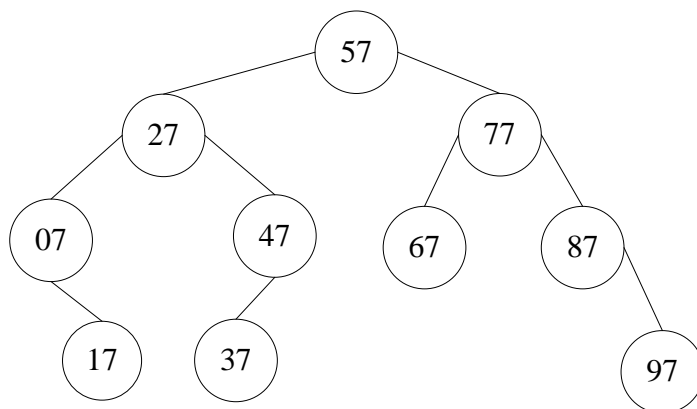
解：

(1) 最好情况下的二叉排序树的高度为：4，最坏情况下的二叉排序树的高度为：

10

注意：二叉排序树最好情况下的高度为平衡二叉树的高度，最坏情况下的高度为不同的关键字个数。

(2) 构造一棵平衡二叉排序树



详细构造过程如下页：

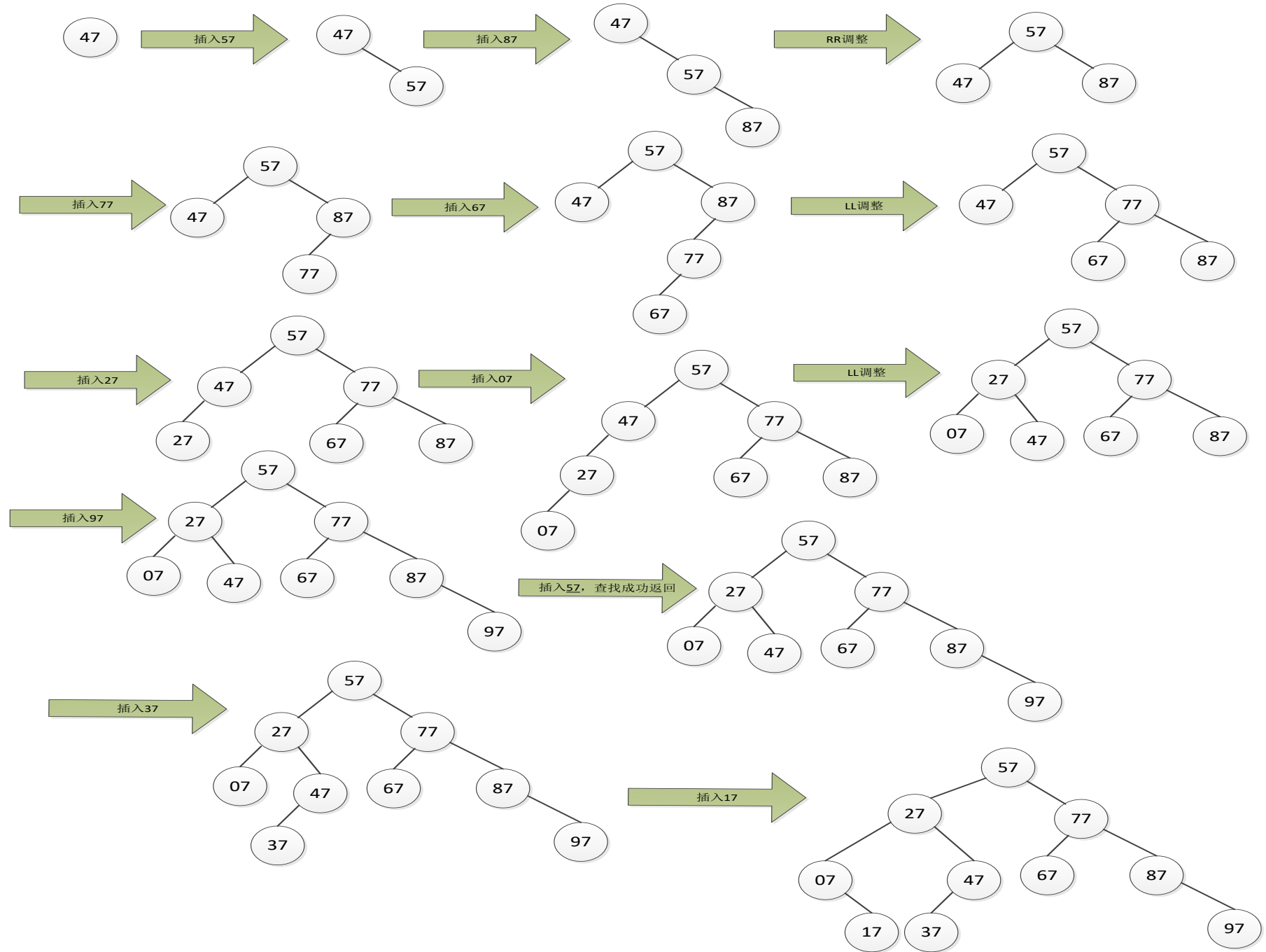
(3) 查找成功时的平均查找长度

$$ASL_{成功} = (3 \times 4 + 4 \times 3 + 2 \times 2 + 1 \times 1) / 10 = 2.9$$

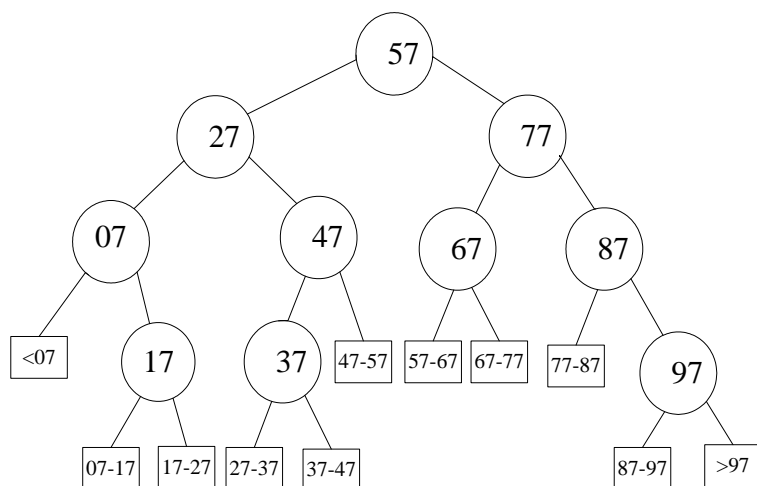
(4) 查找不成功时的平均查找长度

$$ASL_{不成功} = (5 \times 3 + 6 \times 4) / 11 = 39 / 11$$

平衡二叉树构造过程：关键字序列为 {47, 57, 87, 77, 67, 27, 07, 97, 57, 37, 17}，构造一棵平衡二叉树。



7.4. 求折半查找判定树的平均查找长度



根据上图的折半查找判定树，解答如下问题：

- (1) 在等概率的情况下，计算查找成功时该判定树的平均查找长度 **ASL**。
- (2) 在等概率的情况下，计算查找不成功时该判定树的平均查找长度 **ASL**。

解：

折半查找判定树满足以下两点：

1. 折半查找判定树是一棵二叉排序树，即每个结点的值均大于其左子树上所有结点的值，小于其右子树上所有结点的值；

2. 折半查找判定树中的结点（**圆形结点**）都是查找成功的情况，将每个结点的空指针指向一个实际上并不存在的结点——称为外结点，所有外结点即是查找不成功的情况（**方框形状的结点**）。如果有序表的长度为 n ，则外结点一定有 $n+1$ 个。

- (1) 在等概率的情况下，查找**成功**时该判定树的 **ASL**。

在折半查找判定树中，某结点所在的层数即是查找该结点的比较次数，整个判定树代表的有序表的平均查找长度即为查找每个结点的比较次数之和除以有序表的长度。

$$ASL=(1\times 1+2\times 2+3\times 4+4\times 3)/10=29/10$$

- (2) 在等概率的情况下，查找**不成功**时该判定树的 **ASL**。

在折半查找判定树中，查找不成功时的比较次数即是查找相应外结点时与内结点的比较次数。整个判定树代表的有序表在查找失败时的平均查找长度即为查找每个外结点的比较次数之和除以外结点的个数。

$$ASL=(3\times 5+4\times 6)/11=39/11$$

7.5. 试证明：在由 $n(n>0)$ 个结点构成的有序表中进行折半查找，其最坏情况下与关键字比较的次数不超过由 n 个结点所构成的完全二叉树的深度。

证明：

根据完全二叉树的性质 4，可以知道：

具有 n 个结点的完全二叉树的深度为 $\lfloor \log_2 n \rfloor + 1$ 。

考虑具有 n 个关键字的有序表 $S[1..n]$ ，对 S 采用折半查找方法查找表中的元素是，首先查找的是 $S[(1+n)/2]$ 。

当查找不成功时，继续在左或右边的一半中继续查找。如此重复，直到查找成功或查找不成功时，查找过程结束。

当查找成功时，查找过程结束在一个内部结点上；

当查找不成功时，查找过程结束在一个外部结点上。

根据上述查找过程，可以得到一棵具有 n 个结点的判定树。

设判定树的高度为 h ，则外部结点全部出现在第 h 层和第 $h+1$ 层上

考虑最坏的情况，查找不成功时需要比较的最多次数 h

由于具有 n 个结点的二叉判定树的深度为 $\lfloor \log_2 n \rfloor + 1$ ，即 $h = \lfloor \log_2 n \rfloor + 1$ 。

所以，最坏情况下，与关键字比较的次数不超过由 n 个结点所构成的完全二叉树的深度。

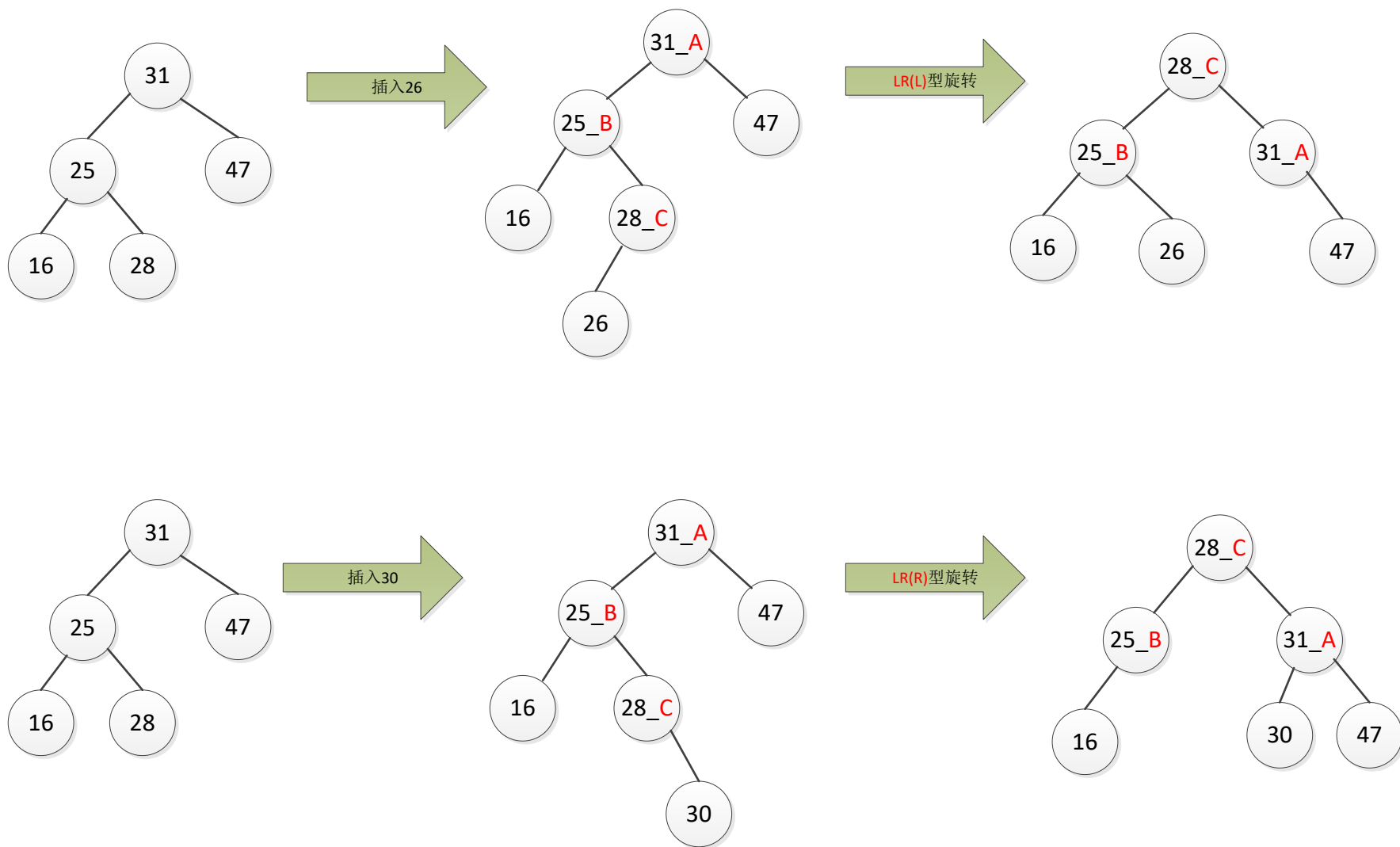
证毕。

要点：1. 具有 n 个结点的完全二叉树的深度为 $\lfloor \log_2 n \rfloor + 1$ ；

2. 具有 n 个结点的二叉判定树的深度为 $\lfloor \log_2 n \rfloor + 1$ 。

7.6 平衡二叉树（教材例题第 207 页~209 页）

（1）LR 型旋转（LR(L)型、LR(R)型）（教材第 208 页图 7.18）



(2) RL 型旋转 (RL(L)型、RL(R)型) (教材第 209 页图 7.20)

