

■ 一个有序链表搜索、添加、删除的平均时间复杂度是多少？

□  $O(n)$



■ 能否利用二分搜索优化有序链表，将搜索、添加、删除的平均时间复杂度降低至  $O(\log n)$ ？

□ 链表没有像数组那样的高效随机访问（ $O(1)$  时间复杂度），所以不能像有序数组那样直接进行二分搜索优化

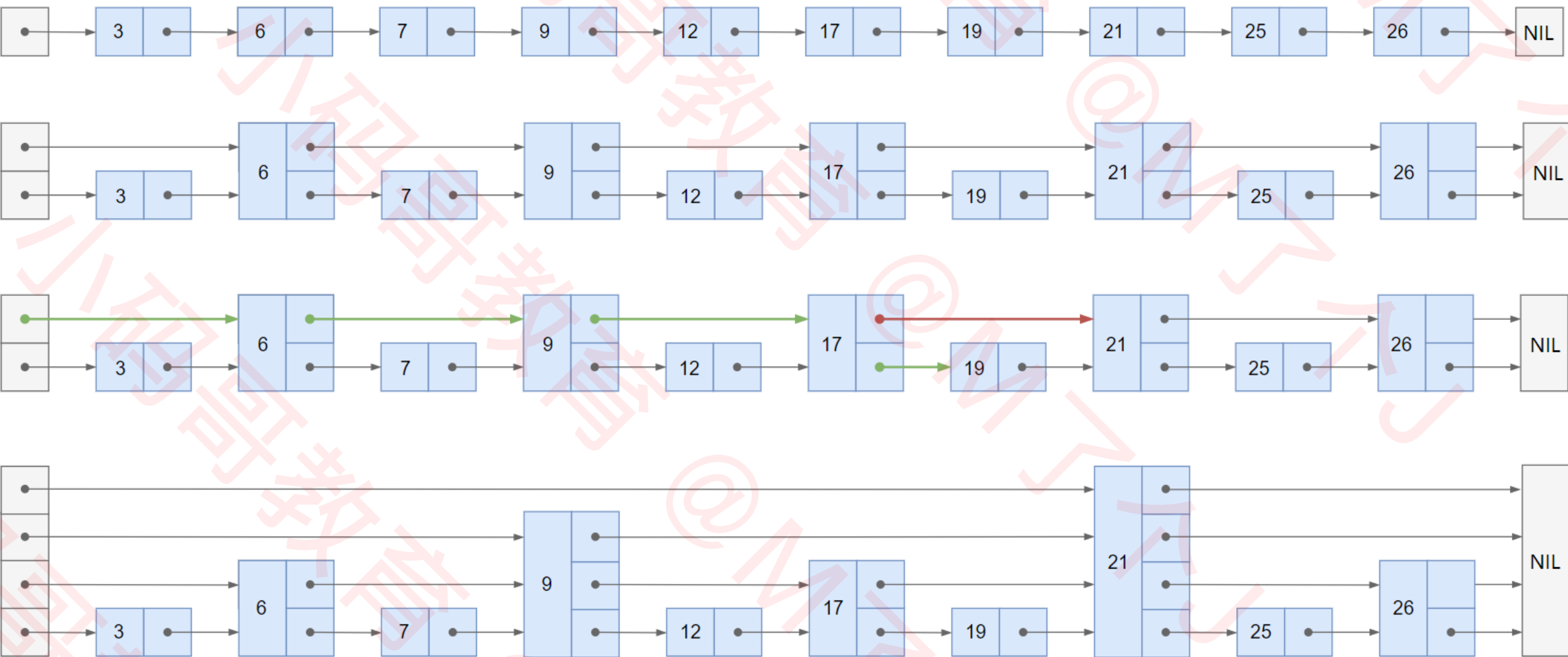
■ 那有没有其他办法让有序链表搜索、添加、删除的平均时间复杂度降低至  $O(\log n)$ ？

□ 使用跳表（SkipList）

# 跳表 (SkipList)

- 跳表，又叫做跳跃表、跳跃列表，在有序链表的基础上增加了“跳跃”的功能
- 由William Pugh于1990年发布，设计的初衷是为了取代平衡树（比如红黑树）
- Redis中的 SortedSet、LevelDB 中的 MemTable 都用到了跳表
- Redis、LevelDB 都是著名的 Key-Value 数据库
- 对比平衡树
- 跳表的实现和维护会更加简单
- 跳表的搜索、删除、添加的平均时间复杂度是  $O(\log n)$

# 使用跳表优化链表（图片来源于网络）

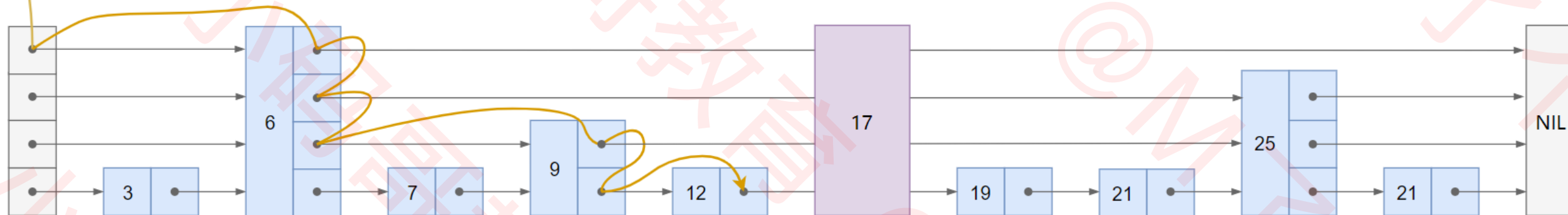


# 跳表的搜索

- ① 从顶层链表的首元素开始，从左往右搜索，直至找到一个大于或等于目标的元素，或者到达当前层链表的尾部
- ② 如果该元素等于目标元素，则表明该元素已被找到
- ③ 如果该元素大于目标元素或已到达链表的尾部，则退回到当前层的前一个元素，然后转入下一层进行搜索

# 跳表的添加、删除

Search Path



## ■ 添加的细节

□ 随机决定新添加元素的层数

## ■ 删除的细节

□ 删除一个元素后，整个跳表的层数可能会降低

# 跳表的层数

- 跳表是按层构造的，底层是一个普通的有序链表，高层相当于是低层的“快速通道”
- 在第  $i$  层中的元素按某个固定的概率  $p$ （通常为  $\frac{1}{2}$  或  $\frac{1}{4}$ ）出现在第  $i + 1$  层中，产生越高的层数，概率越低
- ✓ 元素层数恰好等于 1 的概率为  $1 - p$
- ✓ 元素层数大于等于 2 的概率为  $p$ ，而元素层数恰好等于 2 的概率为  $p * (1 - p)$
- ✓ 元素层数大于等于 3 的概率为  $p^2$ ，而元素层数恰好等于 3 的概率为  $p^2 * (1 - p)$
- ✓ 元素层数大于等于 4 的概率为  $p^3$ ，而元素层数恰好等于 4 的概率为  $p^3 * (1 - p)$
- ✓ .....
- ✓ 一个元素的平均层数是  $1 / (1 - p)$

$$1 \times (1-p) + 2p(1-p) + 3p^2(1-p) + 4p^3(1-p) + \dots = (1-p) \sum_{k=1}^{+\infty} kp^{k-1} = (1-p) \cdot \frac{1}{(1-p)^2} = \frac{1}{1-p}$$

- 当  $p = \frac{1}{2}$  时，每个元素所包含的平均指针数量是 2
- 当  $p = \frac{1}{4}$  时，每个元素所包含的平均指针数量是 1.33

# 跳表的复杂度分析

## ■ 每一层的元素数量

- 第 1 层链表固定有  $n$  个元素
- 第 2 层链表平均有  $n * p$  个元素
- 第 3 层链表平均有  $n * p^2$  个元素
- 第  $k$  层链表平均有  $n * p^k$  个元素
- ...

## ■ 另外

- 最高层的层数是  $\log_{1/p} n$ , 平均有个  $1/p$  元素
- 在搜索时, 每一层链表的预期查找步数最多是  $1/p$ , 所以总的查找步数是  $-(\log_p n / p)$ , 时间复杂度是  $O(\log n)$