跳表：为什么Redis一定要用跳表来实现有序集合？

二分查找底层依赖的是数组随机访问的特性，所以只能用数组来实现。

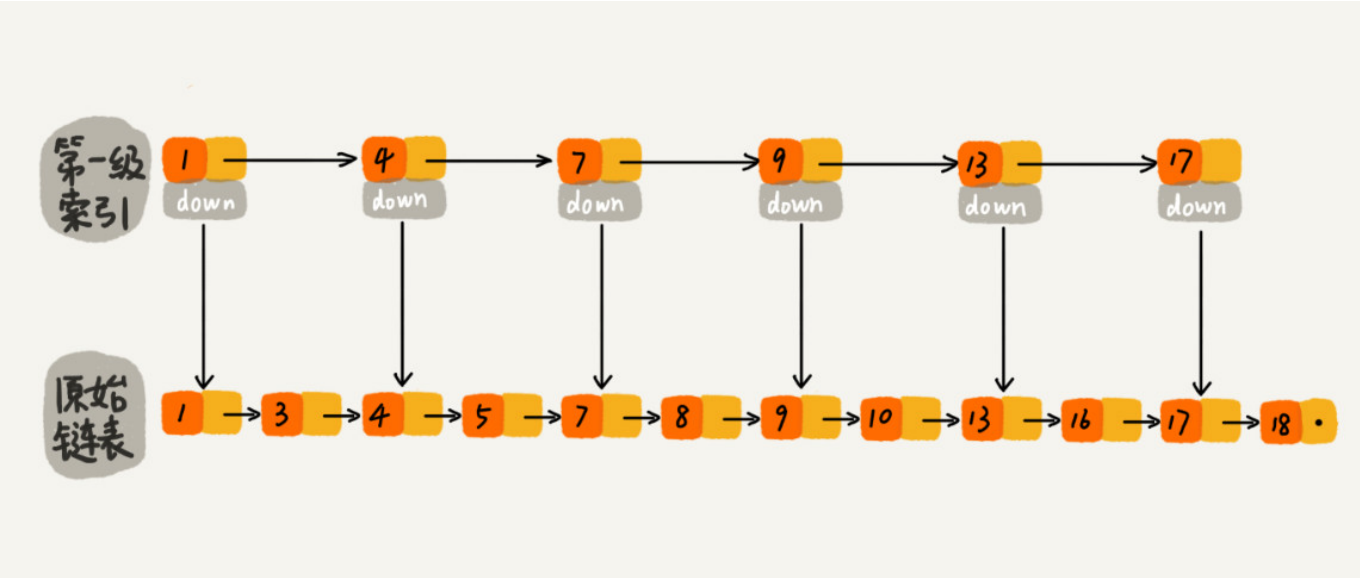
只需要对链表稍加改造，就可以支持类似“二分”的查找算法。改造之后的数据结构叫做**跳表**

那 Redis 为什么会选择用跳表来实现有序集合呢？ 为什么不用红黑树呢？

如何理解“跳表”

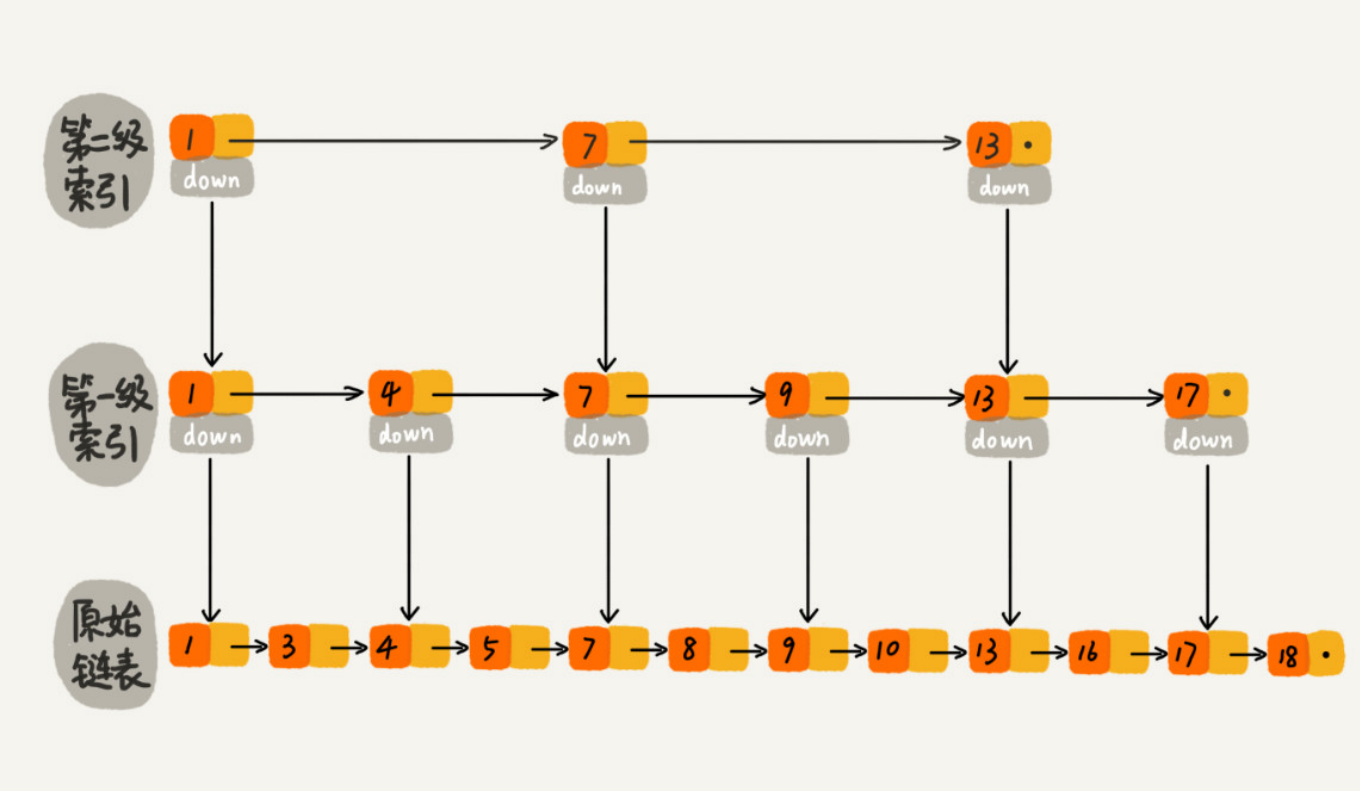
单链表查找时间复杂度是O(n);

给单链表，每两个结点提取一个结点到上一级，我们吧抽出来的那一级叫做索引或者索引层。如图所示：

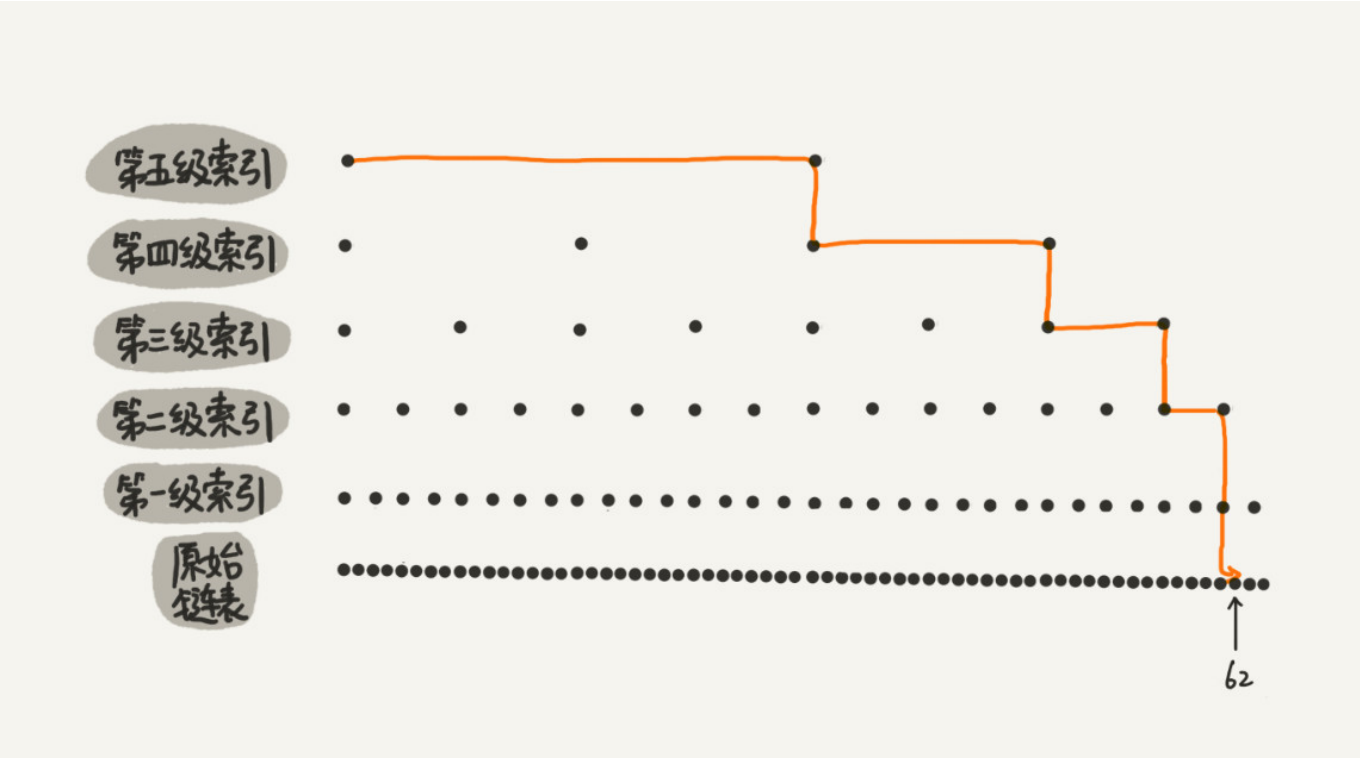


查找某个结点，比如 16。我们可以先在索引层遍历，当遍历到索引层中值为 13 的结点时，我们发现下一个结点是 17，那要查找的结点 16 肯定就在这两个结点之间。然后我们通过索引层结点的 down 指针，下降到原始链表这一层，继续遍历;

加了一层索引之后，查找一个结点需要遍历的节点个数减少了，也就是说查找效率提高了。



为了让你能真切地感受索引提升查询效率。我画了一个包含 64 个结点的链表，按照前面讲的这种思路，建立了五级索引。



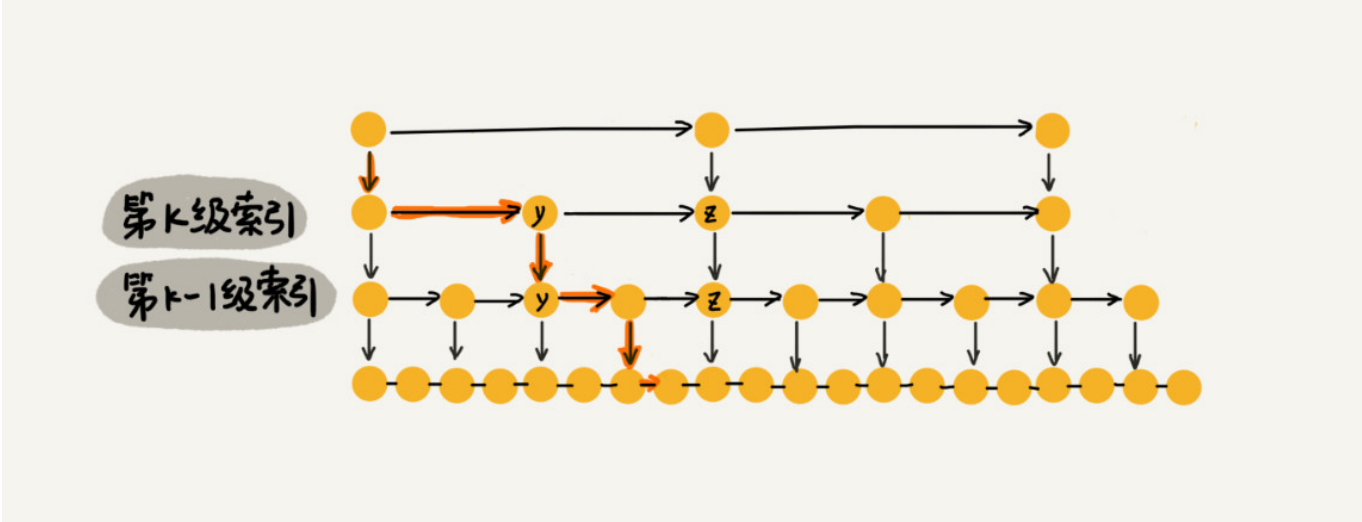
图中我们可以看出，原来没有索引的时候，查找 62 需要遍历 62 个结点，现在只需要遍历 11 个结点;

用跳表查询到底多快

单链表查找时间复杂度是O(n),那么多级索引的跳表中，查询某个数据的时间复杂度是多少呢?

首先，按照每两个结点抽出一个作为上一级索引的节点，那么第一季索引的结点个数大约就是n/2，第二级索引结点数就是n/4，以此类推，第k级索引的结点个数就是第k-1级索引的结点个数的1/2，那第k级索引结点的个数就是n/(2^k)。

假设索引有h级，最高级索引有2个结点，通过上面的公式，我们可以得到n/(2^h)=2,所以$h=\log_2 n-1$; 如果包含原始链表这一层，整个跳表的高度就是$h=\log_2 n$。我们在跳表中查询某个数据的时候，如果每一层都要遍历m个结点，那么跳表中查询一个数据的时间复杂度就是O(m*logn);



上述中，每一层遍历m个结点，加入要查找的数据x，在第k级索引，遍历到y结点之后，发现x大于y，小于z，我们通过y的down指针，从第k级索引下降到第k-1级索引。在第k-1级索引中，y和z之间只有3个结点（包括y和Z），所以我们在k-1级索引中，最多只需要遍历3个结点，以此类推，每一级索引最多只需要遍历3个结点。所以m=3，所以在跳表中查询任意数据的时间复杂度就是 O(logn)。

跳表是不是很浪费内存

空间复杂度。 每两个结点抽一个索引的话，每层的结点数是：n/2,n/4,n/8 ..... 8,4,2； 所以总和是：n/2 + n/4 + n/8 + ... + 4 + 2 = n - 2; 所以空间复杂度是O(n)

如果我们每三个结点或五个结点，抽一个结点到上级索引，是不是就不用那么多索引结点了呢?

总的索引结点大约就是 n/3+n/9+n/27+...+9+3+1=n/2。尽管空间复杂度还是 O(n)，但比上面的每两个结点抽一个结点的索引构建方法，要减少了一半的索引结点存储空间。

在实际的软件开发中，原始链表中存储的有可能是很大的对象，而索引结点只需要存储关键值和几个指针，并不需要存储对象，所以当对象比索引结点大很多时，那索引占用的额外空间就可以忽略了。

高效动态的插入和删除

际上，跳表这个动态数据结构，不仅支持查找操作，还支持动态的插入、删除操作，而且插入、删除操作的时间复杂度也是 O(logn)。

对于跳表来说，查找某个结点的的时间复杂度是 O(logn)，所以这里查找某个数据应该插入的位置，方法也是类似的，时间复杂度也是 O(logn)；

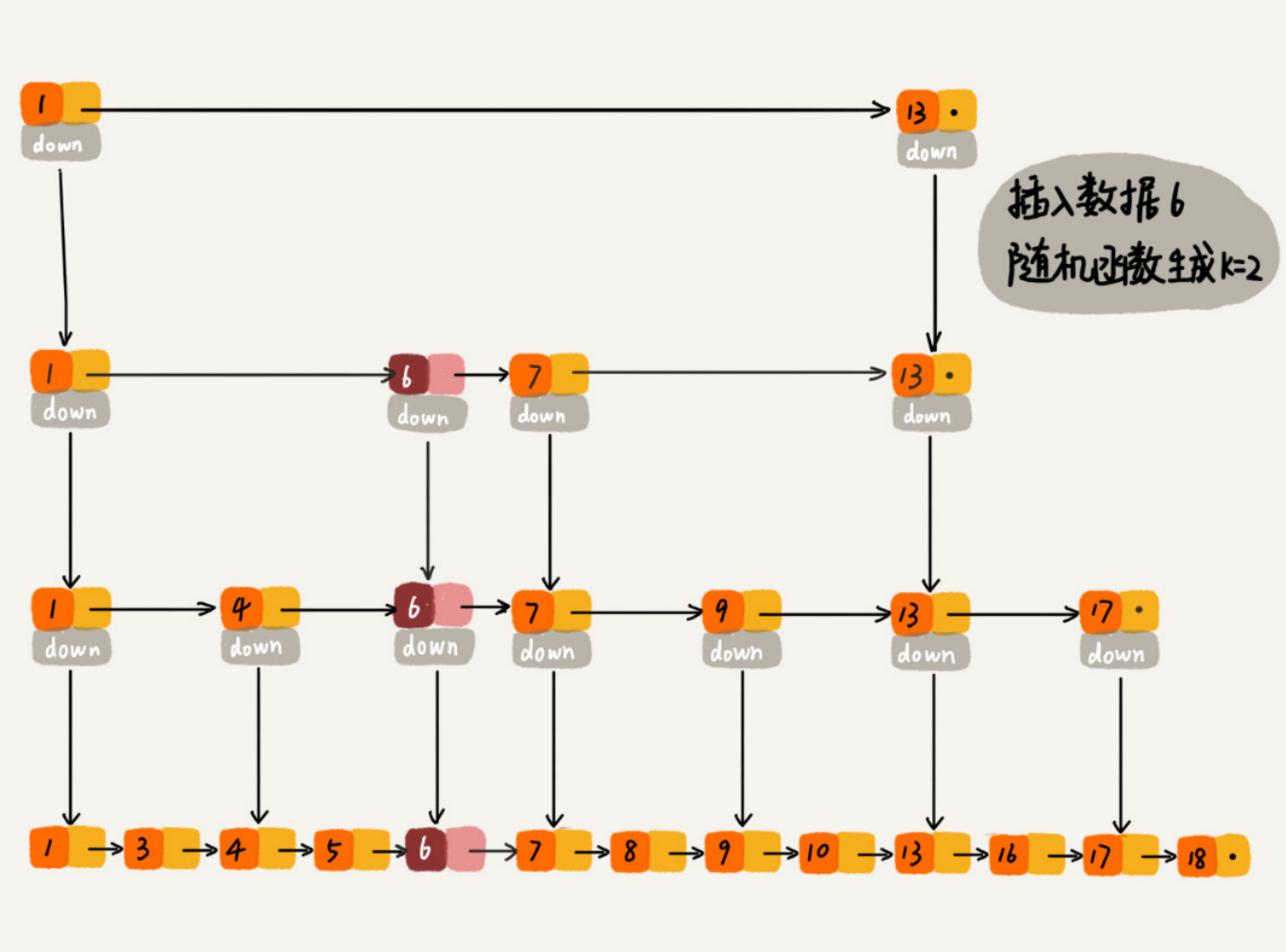
删除操作： 如果这个结点在索引中也有出现，我们除了要删除原始链表中的结点，还要删除索引中的。因为单链表中的删除操作需要拿到要删除结点的前驱结点，然后通过指针操作完成删除。所以在查找要删除的结点的时候，一定要获取前驱结点。当然，如果我们用的是双向链表，就不需要考虑这个问题了。

跳表索引动态更新

如果我们不更新索引，就可能出现某两个索引结点之间数据非常多，极端情况下，跳表还会退化成单链表。

当我们往跳表中插入数据的时候，我们可以选择同时将这个数据插入到部分索引层中。如何选择加入哪些索引层呢?

我们通过一个随机函数，来决定将这个结点插入到那几层索引中，比如随机函数生成了值K，那我们就将这个结点添加到第一级到第K级，这K级索引中。



随机函数的选择很有讲究，从概率上来讲，能保证跳表的索引大小和数据的大小平衡性，不至于性能过度退化。

解答开篇

Redis中的有序集合支持的核心操作主要用以下几个： 1、插入一个数据 2、删除一个数据 3、查找一个数据 4、按照区间查找数据 5、迭代输出有序序列

插入、删除、查找以及迭代输出有序序列这几个操作，红黑树也可以完成，时间复杂度跟跳表是一样的。但是，按照区间来查找数据这个操作，红黑树的效率没有跳表高。

对于按照区间查找数据这个操作，跳表可以做到 O(logn) 的时间复杂度定位区间的起点，然后在原始链表中顺序往后遍历就可以了。

Redis 之所以用跳表来实现有序集合，还有其他原因，比如，跳表更容易代码实现。虽然跳表的实现也不简单，但比起红黑树来说还是好懂、好写多了，而简单就意味着可读性好，不容易出错。

内容小结

跳表使用空间换时间的设计思路，通过构建多级索引来提高查询的效率，实现了基于链表的“二分查找”。跳表是一种动态数据结构，支持快速的插入、删除、查找操作，时间复杂度都是 O(logn)。