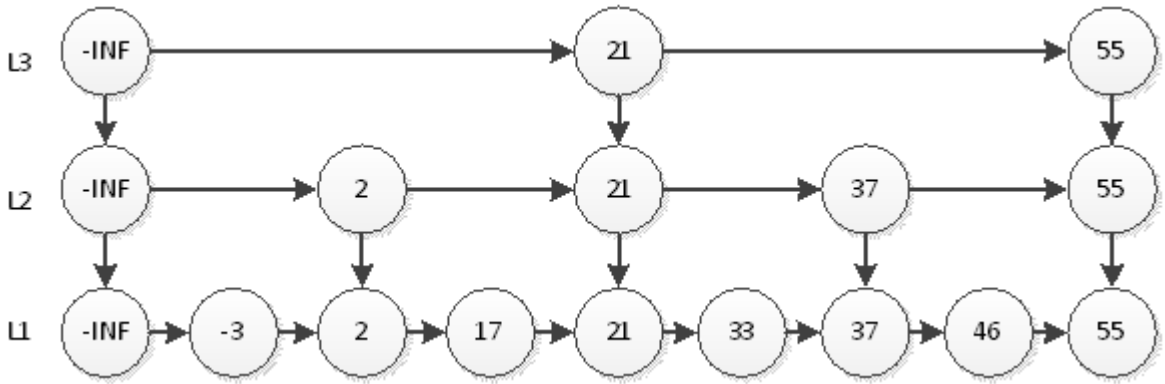
跳跃表

我们知道，普通单链表查询一个元素的时间复杂度为O(n)，即使该单链表是有序的，我们也不能通过2分的方式缩减时间复杂度。

跳跃表通过开辟捷径的方式快速访问到查询元素。查询过程中，如果元素值大于当前层结点值，小于下一个结点的值，那么就从下一层链表开始查找，否则仍在当前层查找。

跳跃表基本结构如下：



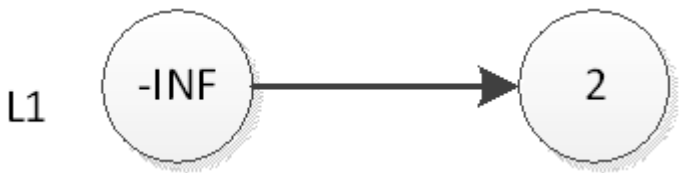
跳跃表实现

容易实现的跳跃表采用抛硬币方式插入杀出元素，并提供O(logn)的查询时间负责度。

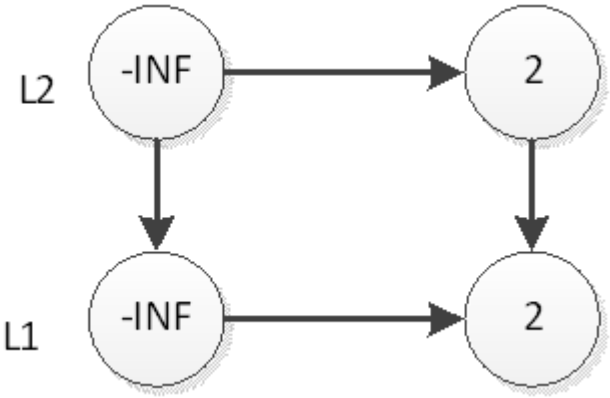
先讨论插入，我们先看理想的跳跃表结构，L2层的元素个数是L1层元素个数的1/2，L3层的元素个数是L2层的元素个数的1/2，以此类推。从这里，我们可以想到，只要在插入时尽量保证上一层的元素个数是下一层元素的1/2，我们的跳跃表就能成为理想的跳跃表。那么怎么样才能在插入时保证上一层元素个数是下一层元素个数的1/2呢？很简单，抛硬币就能解决了！假设元素X要插入跳跃表，很显然，L1层肯定要插入X。那么L2层要不要插入X呢？我们希望上层元素个数是下层元素个数的1/2，所以我们有1/2的概率希望X插入L2层，那么抛一下硬币吧，正面就插入，反面就不插入。那么L3到底要不要插入X呢？相对于L2层，我们还是希望1/2的概率插入，那么继续抛硬币吧！以此类推，元素X插入第n层的概率是(1/2)的n次。这样，我们能在跳跃表中插入一个元素了。

在此还是以上图为例：跳跃表的初试状态如下图，表中没有一个元素：

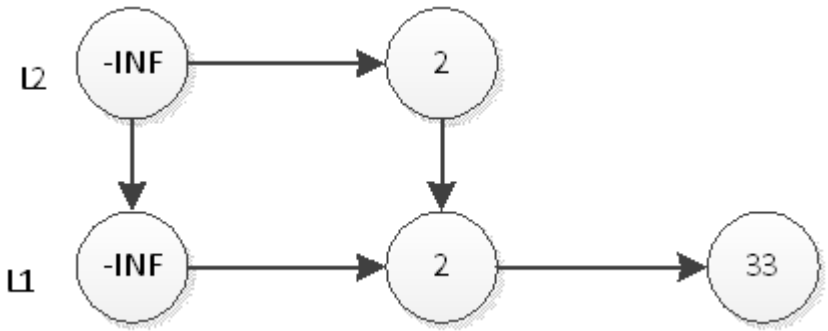
如果我们要插入元素2，首先是在底部插入元素2，如下图：



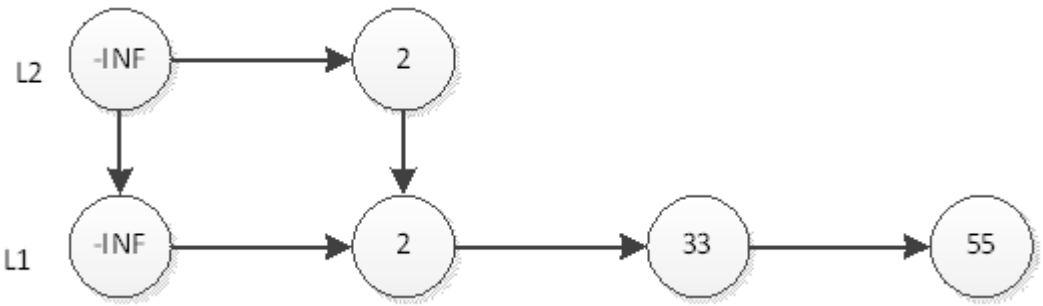
然后我们抛硬币，结果是正面，那么我们要将2插入到L2层，如下图



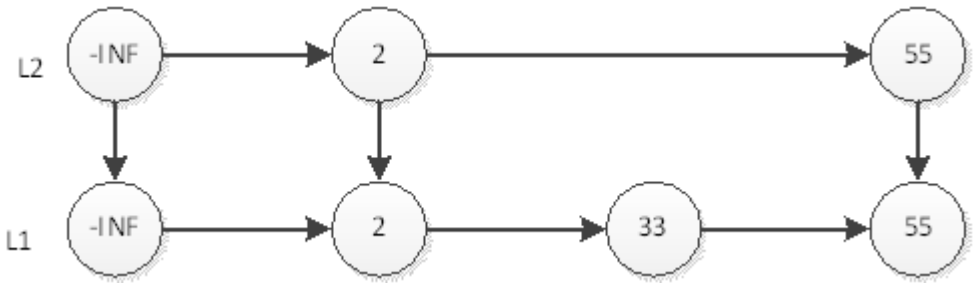
继续抛硬币，结果是反面，那么元素2的插入操作就停止了，插入后的表结构就是上图所示。接下来，我们插入元素33，跟元素2的插入一样，现在L1层插入33，如下图：



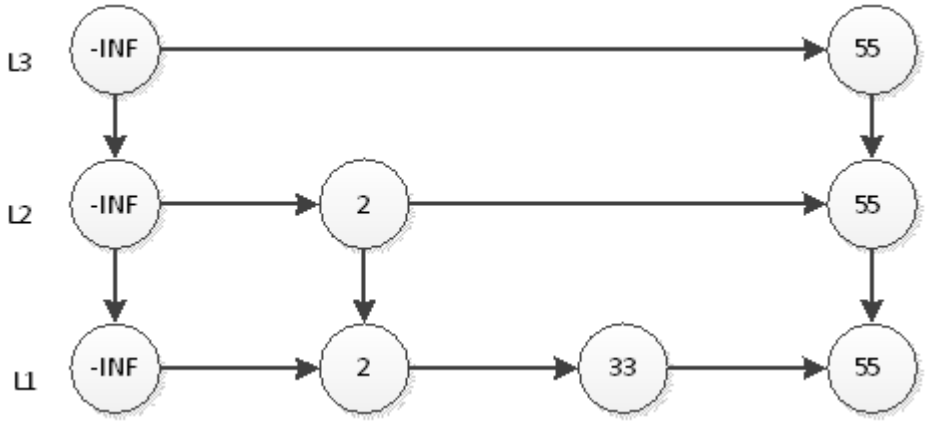
然后抛硬币，结果是反面，那么元素33的插入操作就结束了，插入后的表结构就是上图所示。接下来，我们插入元素55，首先在L1插入55，插入后如下图：



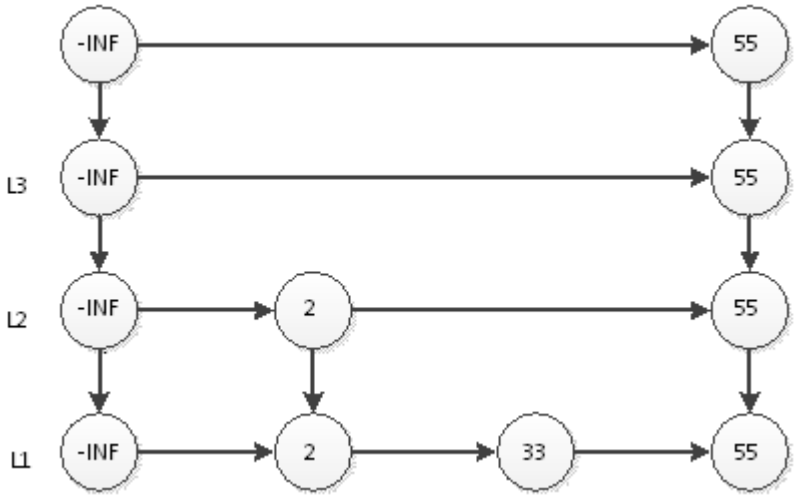
然后抛硬币，结果是正面，那么L2层需要插入55，如下图：



继续抛硬币，结果又是正面，那么L3层需要插入55，如下图：



继续抛硬币，结果又是正面，那么要在L4插入55，结果如下图：



继续抛硬币，结果是反面，那么55的插入结束，表结构就如上图所示。

以此类推，我们插入剩余的元素。当然因为规模小，结果很可能不是一个理想的跳跃表。但是如果元素个数n的规模很大，学过概率论的同学都知道，最终的表结构肯定非常接近于理想跳跃表。