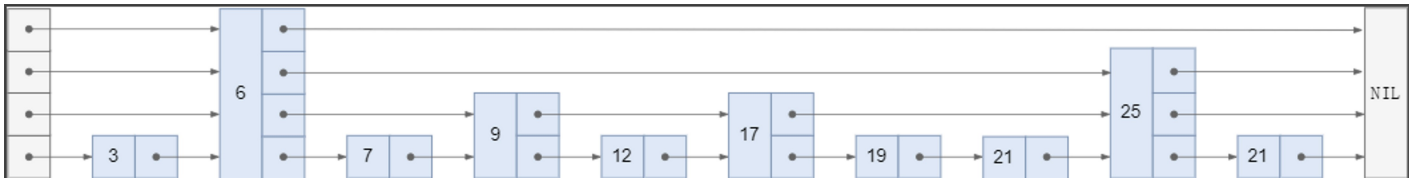
跳表(Skip-List)

2020年11月13日 15:51

跳跃表是一种随机化的数据结构，目前开源软件 Redis 和 LevelDB 都有用到它，它的效率和红黑树以及 AVL 树不相上下。



Skip-List主要由以下部分构成：

- 表头（head）：负责维护跳跃表的节点指针。
- 跳跃表节点：保存着元素值，以及多个层。
- 层：保存着指向其他元素的指针。高层的指针越过的元素数量大于等于低层的指针，为了提高查找的效率，程序总是从高层先开始访问，然后随着元素值范围的缩小，慢慢降低层次。
- 表尾：全部由 NULL 组成，表示跳跃表的末尾。

跳跃表的特点：

1. 每个跳跃表由很多层结构组成。
2. 每一层都是一个有序链表，且第一个节点是头节点。
3. 最底层的有序链表包含所有节点。
4. 每个节点可能有多个指针，这与节点所包含的层数有关。
5. 跳跃表的查找、插入、删除的时间复杂度均为O(log N)， 空间复杂度为O(N)。
6. 维持结构平衡成本低， 完全靠随机

跳跃表的操作：

1. Insert:
   a. 新节点和各层索引节点逐一比较，确定原链表的插入位置。O（logN）
   b. 把索引插入到原链表。O（1）
   c. 利用抛硬币的随机方式，决定新节点是否提升为上一级索引。结果为“正”则提升并继续抛硬币，结果为“负”则停止。O（logN）

   总体上，跳跃表插入操作的时间复杂度是O（logN），而这种数据结构所占空间是2N，既空间复杂度是 O（N）。

2. Delete:
   a. 自上而下，查找第一次出现节点的索引，并逐层找到每一层对应的节点。O（logN）
   b. 删除每一层查找到的节点，如果该层只剩下1个节点，删除整个一层（原链表除外）。O（logN）

   总体上，跳跃表删除操作的时间复杂度是O（logN）。

Q & A

1. 为什么要用概率的方法决定节点是否要放入上层索引?
   因为跳表的添加和删除节点是不可预测的， 很难用一种算法来保证跳表的索

引始终均匀。而用概率的方式可以让大致上趋于均匀分布。

2. 跳表和平衡树的区别?
   维持结构平衡成本低， 完全靠随机。 平衡树（AVL tree等）插入删除要Rebalance, 从而使结构平衡