1. 跳跃表
2. 概述

跳跃表是一种有序数据结构，通过在节点中维持多个指向其他节点的指针，从而达到快速访问节点的目的。支持平均O(logN)，最坏O(N)复杂度的节点查找，效率与平衡树差不多，由于事先更加简单，是平衡树的良性替代品。Redis只在两个地方用到了跳跃表，一个是实现有序集合键，另一个是在集群节点中用作内部数据结构。

1. 跳跃表的实现

Redis的跳跃表由zskiplistNode结构和zskiplist结构定义，zskiplistNode结构表示跳跃表节点，zskiplist结构用于保存跳跃表节点的相关信息。

1. zskiplist

该结构包含以下属性：

header: 跳跃表的表头节点；

tail: 跳跃表的表尾节点；

level： 记录目前跳跃表内，层数最大的那个节点的层数，不包括表头节点；

length： 跳跃表的长度

1. zskiplistNode

该结构包含以下属性：

level数组：跳跃表节点的各个层，每个层带有两个属性，一个是前进指针，，另一个是跨度，程序可以通过这些层来加快访问其他节点的速度，层的数量越多，速度越快。每次创建一个新的跳跃表节点时，随机生成一个介于1-32之间的值作为level数组的大小，即层高。

前进指针：用于访问位于表尾方向的其他节点

跨度：记录了前进指针所指向节点和当前节点的距离，跨度用于计算排位的，在查找某个节点的过程中，将沿途访问过的所有层的跨度累计起来得到的结果就是目标节点在跳跃表中的排位。

后退指针：指向位于当前节点的前一个节点；

分值：各个节点的分支，按照分支从小到大排序；

成员对象：各个节点保存了成员对象，成员对象是一个指针，指向一个字符串对象（SDS），在同一个跳跃表中，各个节点保存的成员对象必须是唯一的，但是多个节点保存的分值可以是相同的，分值相同的节点将按照成员对象在字典序中的大小进行排序。

1. 整数集合
2. 概述

整数集合是集合键的底层实现之一，当一个集合中只包含整数值元素，并且这个集合的元素数量不多时，Redis就会使用整数集合作为集合键的底层实现。

1. 实现

整数集合是Redis用于保存整数值的集合抽象数据结构，可以保存类型为int16\_t, int32\_t, int64\_t的整数值，并且有去重操作。

结构如下：

typedef struct intset{

uint32\_t encoding;

uint32\_t length;

int8\_t contents[];

}intset;

contents数组中的每个元素就是集合中的每个整数，并按照值的大小升序排列，且没有重复项；

length属性记录了整数集合中包含的元素数量；

encoding属性决定了contents数组中存储的项的类型；

1. 升级

每当我们要将一个新元素添加到整数集合中，并且新元素的类型比整数集合中现有元素的类型都要长时，整数集合需要先进行升级，才能让新元素添加到整数集合中。步骤如下：

1. 根据新元素的类型，扩展整数集合底层数组的空间大小，并为新元素分配空间；
2. 将底层数组现有的所有元素转变为与新元素相同的类型，并将转换后的元素放置到正确的位上，注意由于扩展了元素的大小，新元素的位置确定在最后一个，那么就从后往前重新分配位置，以保持有序性不变；
3. 将新元素添加到底层数组中；

由于有升级操作，因此向整数集合中添加新元素的时间复杂度为O(N)。注意，整数集合不支持降级操作，一旦升级了数组，就会保持升级后的状态了。

1. 升级的优势
2. 提升灵活性

由于C语言是静态类型语言，为了避免类型错误，我们不会将不同类型的值放在以同一个数据结构中，使用整数集合的升级可以自动适应新元素，不用担心出现类型错误。

1. 节约内存

升级只会在需要的时候，而不是大包大揽地直接用最大的类型，尽量地节省了内存。