散列表上：Word文档中单词拼写检查功能是如何实现的?

Word的单词拼写检查功能，小巧实用，这个功能如何实现呢?

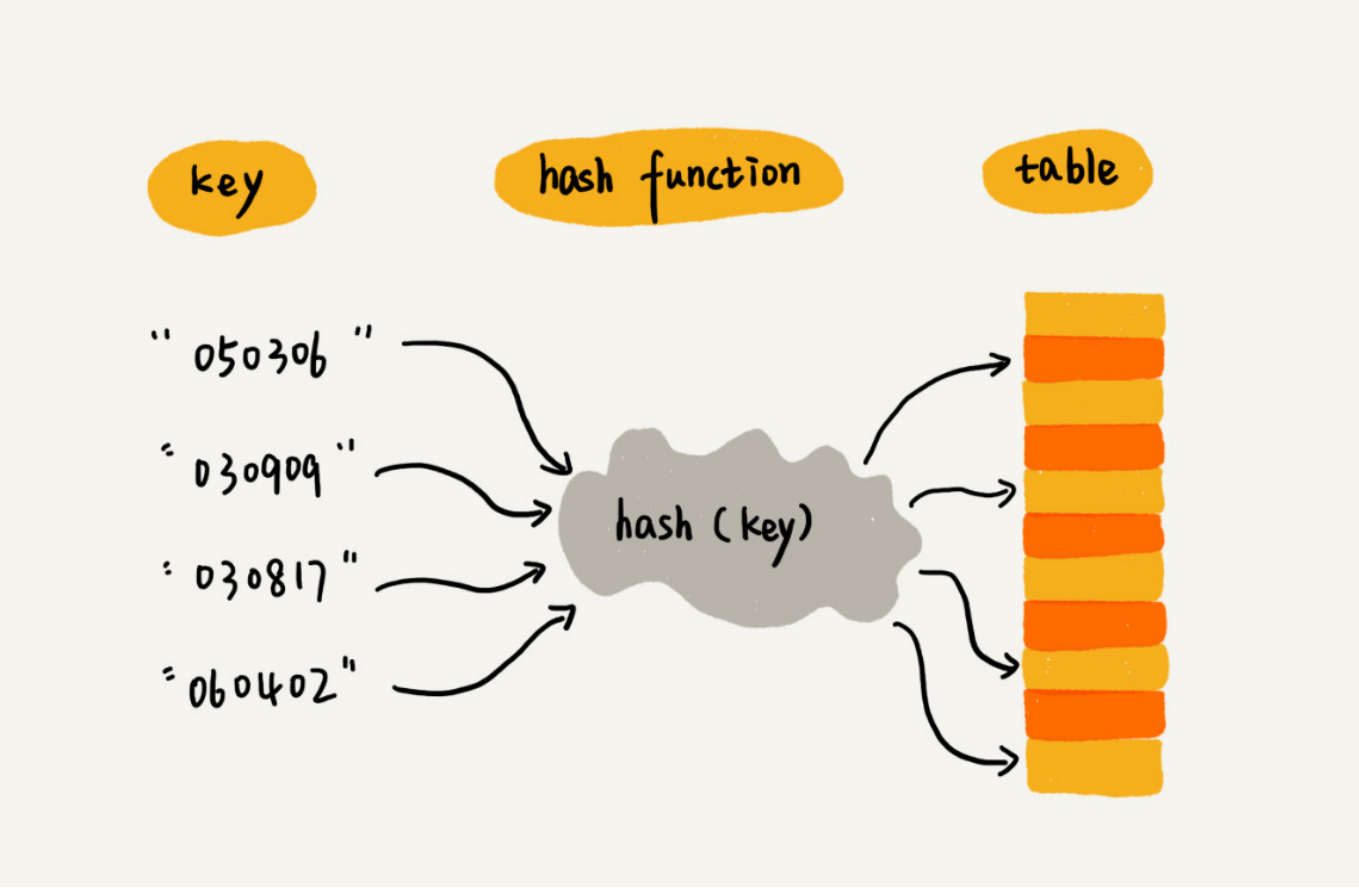
散列思想

散列表的英文“Hash Table”，也叫哈希表或者“Hash 表”。

散列表用的是数组支持按照下表访问数据的特性，所以散列表其实就是数组的一种扩展，由数组演化而来。可以说，如果没有数组就没有散列表。

运动会编号，参赛编号是自然数，并且与数组的下标形成一一映射，所以利用数组支持根据下标随机访问的时候，时间复杂度是O(1)。

其中参赛选手的编号我们叫做键，或者关键字。 我们把参赛编号转化为数组下标的映射方法就叫做散列函数，而散列函数计算得到的值就叫做散列值。



总结出这样的规律： 散列表用的就是数组支持按照下标随机访问的时候，时间复杂度是 O(1) 的特性。我们通过散列函数把元素的键值映射为下标，然后将数据存储在数组中对应下标的位置。

散列函数

散列函数，顾名思义，他就是一个函数。我们可以把它定义成hash(key)，其中key表示元素的键值，hash(key)的值表示经过散列函数计算得到的散列值。

三点散列函数设计的基本要求

1. 散列函数计算得到的散列值是一个非负整数；
2. 如果key1 = key2，那hash(key1) == hash(key2);
3. 如果key1 != key2，那hash(key1) != hash(key2);

上述要求第三点，几乎不可能实现，业界著名的MD5，SHA，CRC等哈希算法，也无法完全避免散列冲突。

散列冲突

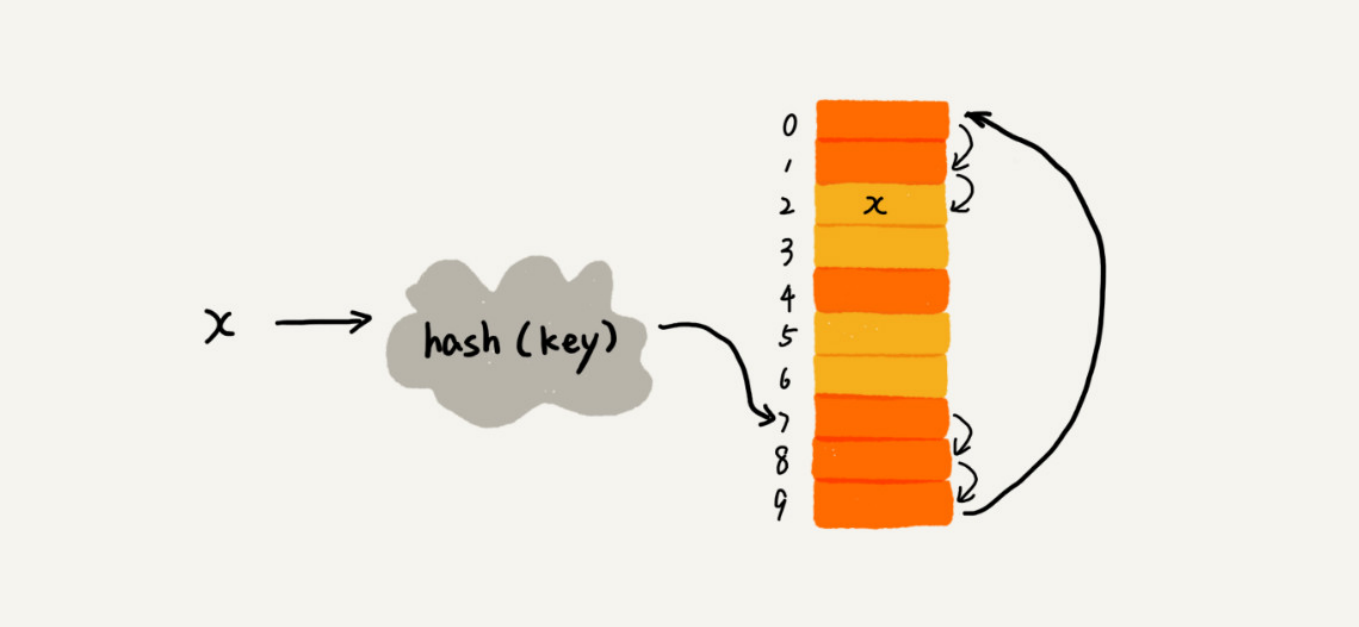
两类解决散列冲突的方法，开放寻找法和链表法；

1、开放寻址法 核心思想，如果出现散列冲突，我们就重新探测一个空闲位置，将其插入。

如何探测，一个比较简单的探测方法，**线性探测**；

插入

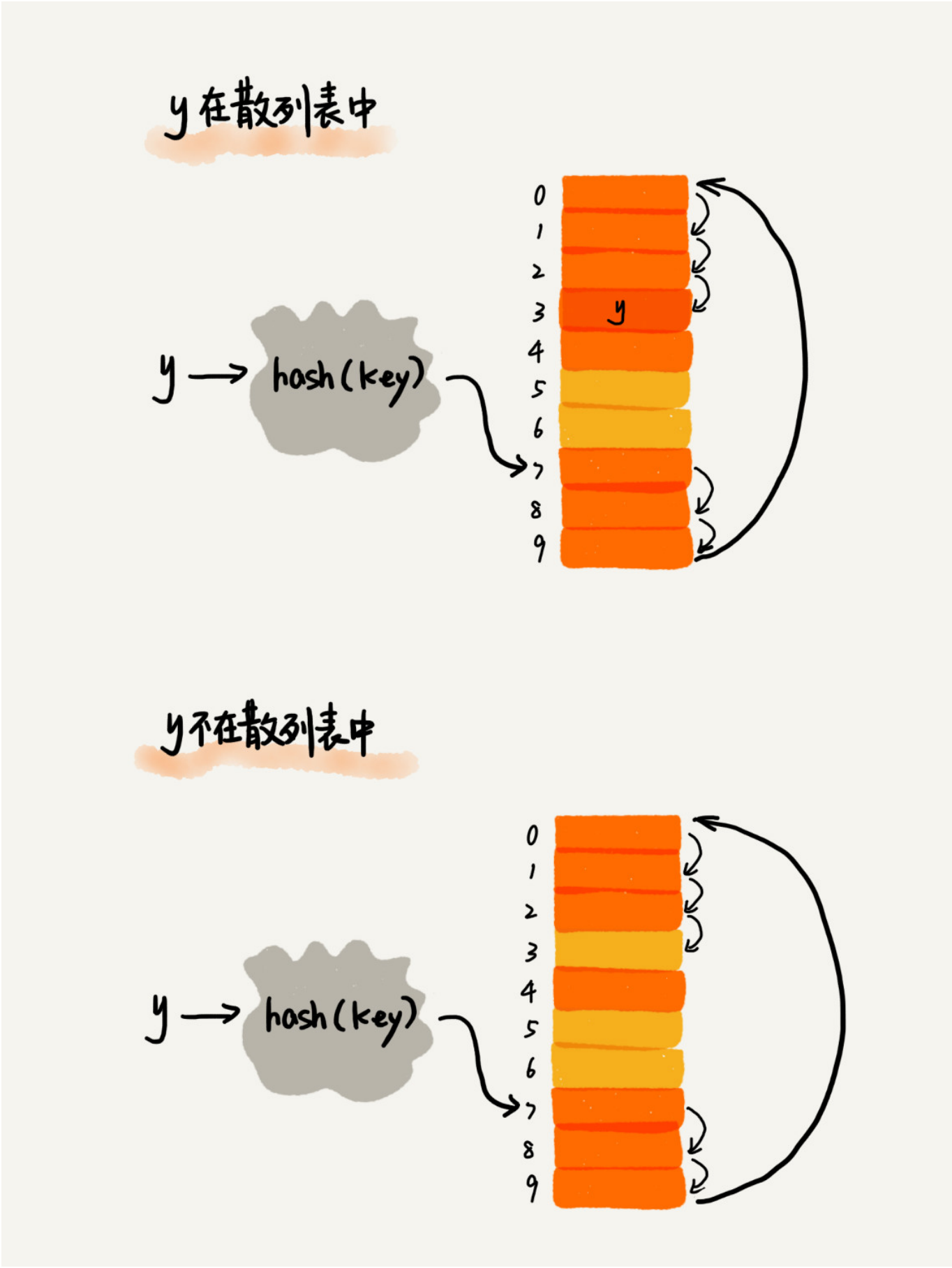
当我们往散列表中插入数据时，如果某个数据经过散列函数散列之后，存储位置已经被占用了，我们就从当前位置开始，依次往后查找，开是否有空闲位置，直到找到为止。



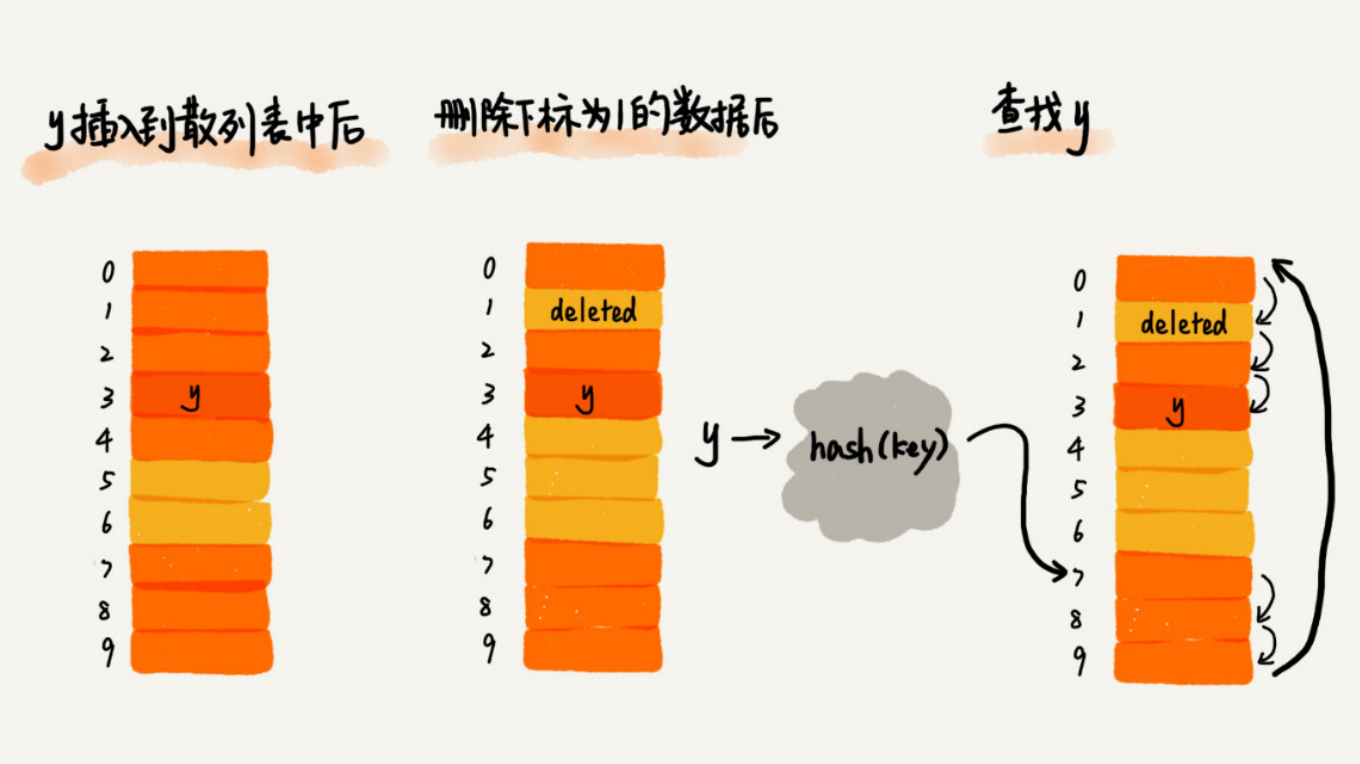
x经过第一次散列到位置7处，但是已经被占用，就往下查找，看是否有空闲位置，遍历到尾部没有，再从表头开始查找，最后在下标为2处找到。

查找

查找数据类似于插入过程，通过散列函数求出查找元素的键值对应的散列值，然后比较数组中下标为散列值的元素和要查找的元素，相等就找到了，不相等，就往后依次查找，如果遍历到空闲位置还没有找到，就说明要查找的元素并没有在散列表中。



删除 我们删除操作时，不能仅仅将元素设置为空，这样会使得查找算法失效。 而应该标记为deleted，当线性探测查找的时候，标记为deleted的空间，并不是停下来，而是往下探测。



线性探测法其实存在很大问题，当散列表中插入数据越来越多时，散列冲突发生的可能性就越来越大，空闲位置会越来越少，线性探测的时间就会越来越久。

经典探测方法还有，**二次探测和双重探测**

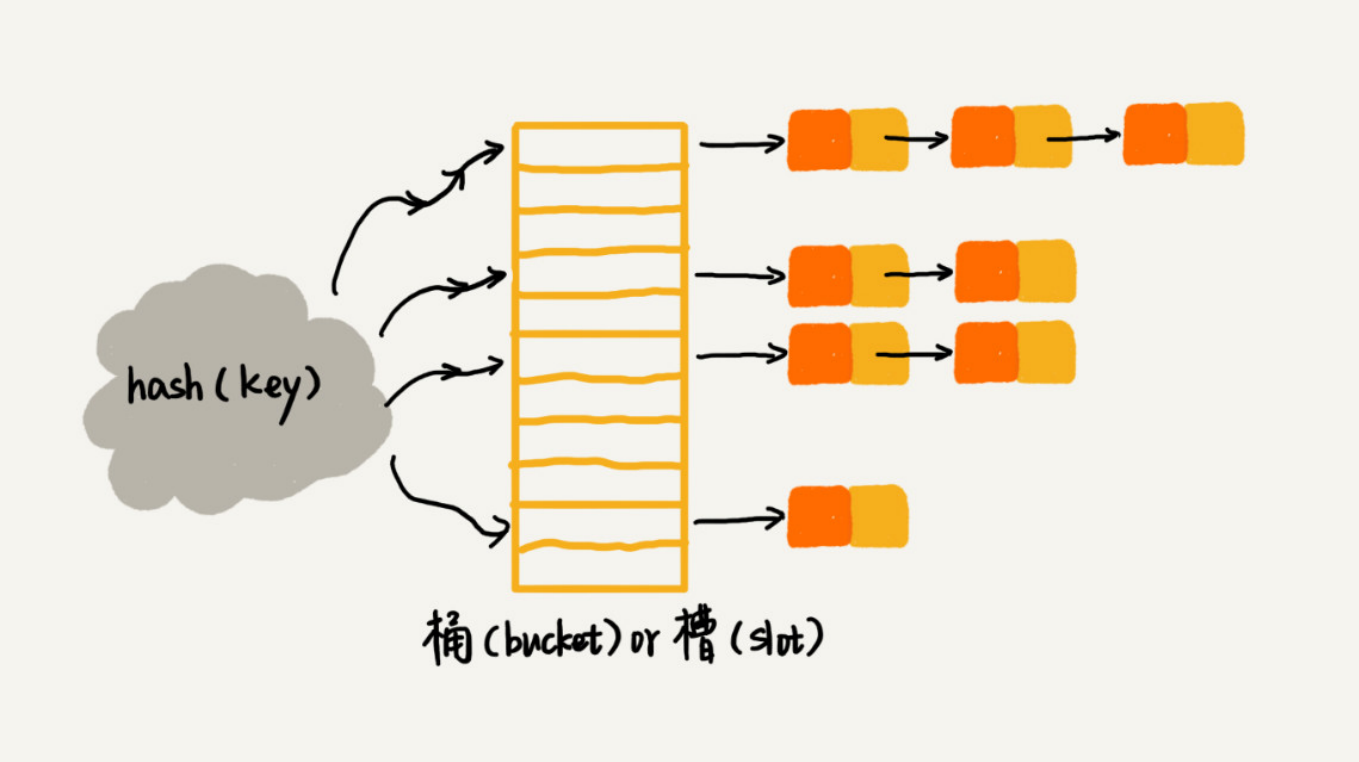
二次探测探测的步长就变成了原来的“二次方”，也就是说，它探测的下标序列就是 hash(key)+0，hash(key)+1^2，hash(key)+2^2.....

双重散列，不仅要使用一个散列函数。我们使用一组散列函数 hash1(key)，hash2(key)，hash3(key).....我们先用第一个散列函数，如果计算得到的存储位置已经被占用，再用第二个散列函数，依次类推，直到找到空闲的存储位置。

> 散列表的装载因子 = 填入表中的元素个数 / 散列表的长度

2、链表法

链表法是一种更加查用的散列冲突解决办法，相比开发寻址法，他要简单很多。



插入 只需要计算出对应的散列槽位，将其插入到对应的链表中即可，所以插入时间复杂度是O(1); 当查找、删除一个元素的时候，我们同样通过散列函数计算出对应的槽，然后遍历链表查找或者删除。那查找或者删除操作的时间复杂度是多少呢?

这两个操作的时间复杂度跟链表的长度 k 成正比，也就是 O(k)。理论上讲，k=n/m，其中 n 表示散列中数据的个数，m 表示散列表中“槽”的个数。

解答开篇

常用的英文单词有 20 万个左右，假设单词的平均长度是 10 个字母，平均一个单词占用 10 个字节的内存空间，那 20 万英文单词大约占 2MB 的存储空间，就算放大 10 倍也就是 20MB。对于现在的计算机来说，这个大小完全可以放在内存里面。所以我们可以用散列表来存储整个英文单词词。

当用户输入某个英文单词时，我们拿用户输入的单词去散列表中查找。如果查到，则说明拼写正确；如果没有查到，则说明拼写可能有误，给予提示。

课后思考

1. 假设我们有 10 万条 URL 访问日志，如何按照访问次数给 URL 排序?

遍历 10 万条数据，以 URL 为 key，访问次数为 value，存入散列表，同时记录下访问次数的最大值 K，时间复杂度 O(N)。

如果 K 不是很大，可以使用桶排序，时间复杂度 O(N)。如果 K 非常大（比如大于 10 万），就使用快速排序，复杂度 O(NlogN)。

2. 有两个字符串数组，每个数组大约有 10 万条字符串，如何快速找出两个数组中相同的字符串?

以第一个字符串数组构建散列表，key 为字符串，value 为出现次数。再遍历第二个字符串数组，以字符串为 key 在散列表中查找，如果 value 大于零，说明存在相同字符串。时间复杂度 O(N)。