Hash冲突有什么解决方式

题目标签

学习时长：20分钟

题目难度：中等

知识点标签：Hash

题目描述

Hash冲突有什么解决方式

面试题分析

先说出hash表的概念，再说出不同的解决方法，再之处各种方法的优缺点

一.哈希表简介

非哈希表的特点：关键字在表中的位置和它自检不存在一个确定的关系，查找的过程为给定值一次和各个关系自进行比较，查找的效率取决于给定值进行比较的次数。
哈希表的特点：关键字在表中位置和它自检存在一种确定的关系。
哈希函数：一般情况下，需要在关键字与它在表中的存储位置之间建立一个函数关系，以（key）作为关键字为key的记录在表中的位置，通常称这个函数f(key)为哈希函数。
hash：翻译为"散列表"，就是把任意长度的输入，通过散列算法，变成固定长度输出，该输出结果是散列值。
这种转换是一种压缩映射，散列表的空间通常小鱼输入的空间，不同的输入可能会散列成相同的输出，所以不能从散列表来唯一的确定输入值。
简单的说就是一种将任意长度的消息压缩到固定长度的消息摘要函数。
hash冲突：就是根据key即经过一个函数f(key)得到的结果的作为地址去存放当前的key value键值对(这个是hashmap的存值方式)，但是却发现算出来的地址上已经有人先来了。就是说这个地方要挤一挤啦。这就是所谓的hash冲突啦

回到顶部

二.哈希函数处理冲突的方法

1）开放定址法：

这种方法也称再散列法，其基本思想是：当关键字key的哈希地址p=H（key）出现冲突时，以p为基础，产生另一个哈希地址p1，如果p1仍然冲突，再以p为基础，产生另一个哈希地址p2，…，直到找出一个不冲突的哈希地址pi ，将相应元素存入其中。这种方法有一个通用的再散列函数形式：

Hi=（H（key）+di）% m i=1，2，…，n

其中H（key）为哈希函数，m 为表长，di称为增量序列。增量序列的取值方式不同，相应的再散列方式也不同。主要有以下三种：

线性探测再散列

dii=1，2，3，…，m-1

这种方法的特点是：冲突发生时，顺序查看表中下一单元，直到找出一个空单元或查遍全表。

二次探测再散列

di=12，-12，22，-22，…，k2，-k2 (k<=m/2)

这种方法的特点是：冲突发生时，在表的左右进行跳跃式探测，比较灵活。

伪随机探测再散列

di=伪随机数序列。

具体实现时，应建立一个伪随机数发生器，（如i=(i+p) % m），并给定一个随机数做起点。

例如，已知哈希表长度m=11，哈希函数为：H（key）= key % 11，则H（47）=3，H（26）=4，H（60）=5，假设下一个关键字为69，则H（69）=3，与47冲突。

如果用线性探测再散列处理冲突，下一个哈希地址为H1=（3 + 1）% 11 = 4，仍然冲突，再找下一个哈希地址为H2=（3 + 2）% 11 = 5，还是冲突，继续找下一个哈希地址为H3=（3 + 3）% 11 = 6，此时不再冲突，将69填入5号单元。

如果用二次探测再散列处理冲突，下一个哈希地址为H1=（3 + 12）% 11 = 4，仍然冲突，再找下一个哈希地址为H2=（3 - 12）% 11 = 2，此时不再冲突，将69填入2号单元。

如果用伪随机探测再散列处理冲突，且伪随机数序列为：2，5，9，……..，则下一个哈希地址为H1=（3 + 2）% 11 = 5，仍然冲突，再找下一个哈希地址为H2=（3 + 5）% 11 = 8，此时不再冲突，将69填入8号单元。

2) 再哈希法

这种方法是同时构造多个不同的哈希函数：

Hi=RH1（key） i=1，2，…，k

当哈希地址Hi=RH1（key）发生冲突时，再计算Hi=RH2（key）……，直到冲突不再产生。这种方法不易产生聚集，但增加了计算时间。

3）链地址法

这种方法的基本思想是将所有哈希地址为i的元素构成一个称为同义词链的单链表，并将单链表的头指针存在哈希表的第i个单元中，因而查找、插入和删除主要在同义词链中进行。链地址法适用于经常进行插入和删除的情况。

4）建立公共溢出区

这种方法的基本思想是：将哈希表分为基本表和溢出表两部分，凡是和基本表发生冲突的元素，一律填入溢出表。

优缺点

开放散列表（open hashing）/拉链法（针对桶链结构）

1）优点：①对于记录总数频繁可变的情况，处理的比较好（也就是避免了动态调整的开销） ②由于记录存储在结点中，而结点是动态分配，不会造成内存的浪费，所以尤其适合那种记录本身尺寸（size）很大的情况，因为此时指针的开销可以忽略不计了 ③删除记录时，比较方便，直接通过指针操作即可

2）缺点： ①存储的记录是随机分布在内存中的，这样在查询记录时，相比结构紧凑的数据类型（比如数组），哈希表的跳转访问会带来额外的时间开销 ②如果所有的 key-value 对是可以提前预知，并之后不会发生变化时（即不允许插入和删除），可以人为创建一个不会产生冲突的完美哈希函数（perfect hash function），此时封闭散列的性能将远高于开放散列 ③由于使用指针，记录不容易进行序列化（serialize）操作

封闭散列（closed hashing）/ 开放定址法

1）优点： ①记录更容易进行序列化（serialize）操作 ②如果记录总数可以预知，可以创建完美哈希函数，此时处理数据的效率是非常高的

2）缺点： ①存储记录的数目不能超过桶数组的长度，如果超过就需要扩容，而扩容会导致某次操作的时间成本飙升，这在实时或者交互式应用中可能会是一个严重的缺陷 ②使用探测序列，有可能其计算的时间成本过高，导致哈希表的处理性能降低 ③由于记录是存放在桶数组中的，而桶数组必然存在空槽，所以当记录本身尺寸（size）很大并且记录总数规模很大时，空槽占用的空间会导致明显的内存浪费 ④删除记录时，比较麻烦。比如需要删除记录a，记录b是在a之后插入桶数组的，但是和记录a有冲突，是通过探测序列再次跳转找到的地址，所以如果直接删除a，a的位置变为空槽，而空槽是查询记录失败的终止条件，这样会导致记录b在a的位置重新插入数据前不可见，所以不能直接删除a，而是设置删除标记。这就需要额外的空间和操作。