哈希表

哈希表，也称散列表，是一种高效的数据结构。它的最大优点就是把数据存储和查找所消耗的时间大大降低，几乎可以看成是 O（1）的，而代价是消耗比较多的内存。因此，`空间换时间`是哈希表的主要特点。

常用哈希函数构造方法

> 直接定址法和除留取余法

哈希表的基本原理是使用一个下标范围比较大的数组 A 来存储元素，设计一个函数 h，对于要存储的线性表的每个元素 node，取一个关键字 key，算出一个函数值 h(key)，把 h(key) 作为数组下标，用 A[h(key)] 这个数组单元来存储 node。也可以简单的理解为，按照关键字为每一个元素“分类”，然后将这个元素存储在相应“类”所对应的地方(这一过程称为“`直接定址`”)。

但是，不能够保证每个元素的关键字与函数值是一一对应的，因此极有可能出现对于不同的元素，却计算出了相同的函数值，这样就产生了“冲突”，换句话说，就是把不同的元素分在了相同的“类”之中了。假设一个结点的关键字值为key，把它存入哈希表的过程是根据确定的函数h计算出h(key)的值，如果以该值为地址的存储空间还没有被占用，那么就把结点存入该单元；如果此值所指单元里已存储了别的结点(即发生了冲突)，那么就再用另一个函数I进行映射，计算出I(h(key)),再看用这个值作为地址的单元是否已被占用了，若已被占用，则再用I映射，直至找到一个空位置将结点存入为止。当然，这只是解决“冲突”问题的一种简单方法，如何避免、减少和处理“冲突”是使用哈希表的一个难题。

> 例：用哈希表存储以下线性表（18，75，60，43，54，90，46）

假设选取的哈希函数为 h(K) = K mod m，K 为元素的关键字，m 为哈希表的长度，用余数作为存储该元素的哈希地址。K 和 m 均为正整数，并且 m 要大于或等于线性表的长度 n。此例n = 7，故取 m = 13 就已经足够，得到的每个元素的哈希地址为：

- h(18)=18 mod 13 = 5
- h(75)=75 mod 13 = 10
- h(60)=60 mod 13 = 8
- h(43)=43 mod 13 = 4
- h(54)=54 mod 13 = 2
- h(90)=90 mod 13 = 12
- h(46)=46 mod 13 = 7

字符串哈希

判断两个字符串是否相等，可以使用字符串哈希算法，即判定两个字符串的哈希值是否相等。

字符串哈希假定不会发生冲突，然而冲突是无法避免的，只是我们根据经验值调整参数，使得冲突概率尽量趋近于0。

- 字符串S = $c_1 c_2$ ..... c_m
- 空间：unsigned long long
- 质数：P = 131 或 13331
- 前k个字符构成的字符串哈希值递推式：H(S,k)=H(S,k-1)*P+c_k

例如：字符串S="ACDA"，令A=1，B=2，C=3，依次类推

H(S,1)=1*P^0

H(S,2)=1*P^1+3

H(S,3)=1*P^2+3*P^1+4

H(S,4)=1*P^3+3*P^2+4*P^1+1

假定给出字符串区间[L,R]，求这个区间子串S'哈希值。

已知：H(S,R) H(S,L-1)

hash(S')=H(S,L~R)=H(S,R)-H(S,L-1)*P^{R-L+1}

快乐刷题

- P50 分身数对
- P49 模拟哈希表
- P51 整数集合
- P48 字符串哈希