

Data Structures for Dictionaries: Hash Tables

Problem 1: Read a grade file, where grades are integers between 0 to 99. Keep track of number of occurrences of each grade.

Fastest Solution: Create an array T of size 100. $T[i]$ stores the number of occurrences of grade i .

Problem 2: Read a data file, keep track of number of occurrences of each integer value (from 0 to $2^{32} - 1$).

Fastest Solution: Create an array of size 2^{32} , as above.

Wasteful use of memory, especially when data are files relatively small.

Problem 3: Read a text file, keep track of number of occurrences of each word.

Cannot use **keys as indices** anymore!

1. We need to be able to convert any type of key to an integer.
2. We need to map the universe of keys into a small number of slots.

csc263 lec4 ppt pg3

研究：

- ➔ 增（插入 insert）
- ➔ 删（delete）
- ➔ 查（寻址/search）

List 是 ADT ADT，其两种基本的实现方式有 array 数组，linkedList 链表

array[4]=60, 4 是 index 索引，60 是其对应的数据、数值、value
index 索引是提前设置好的

HashTable 基本原理包含了 array，随机顺序访问数据/查询，**访问查询速度最快**，快速根据 key(index 索引) $O(1)$

Array 是简单的 HashTable

LinkedList 链式一般是从第一个进行 traverse, $O(n)$

数据源 (file)– 存储(array)

将数据源里的 type of key 转换成 integer（通过 hash function 转换成具有整数性质的散列值），引出 Hash Function（universe of keys

- ➔ slots, slot 里面即每个槽位里存的还是原关键字 key）

Hash Table（散列表/哈希表）：元素，关键字（key），散列值（ $h(k)$ ）

承载因子（load factor）一般是 0.75

目标：使数据尽可能均匀放在 slots 里，ensure uniform

Expected Run Time in a Successful Search (under SUHA):

That is k is a key that exists in the hash table.

Let $k_1, k_2, k_3, \dots, k_n$ be the order of insertion into the hash table.

k could be k_1 , or k_2 , or k_3 , or ..., or k_n .

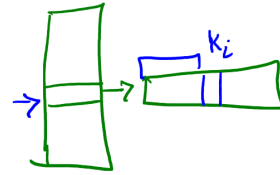
The probability that k is k_i ($1 \leq i \leq n$) is: $\frac{1}{n}$

So the expected number of steps to find k is the sum over:

the probability that k is k_i , times the number of steps required to find k_i

$$\begin{aligned} E[t_{m,n}(k)] &= \frac{1}{n} \times S_1 + \frac{1}{n} \times S_2 + \frac{1}{n} \times S_3 + \dots + \frac{1}{n} \times S_n \\ &= \frac{1}{n} \sum_{i=1}^n S_i \end{aligned}$$

S_i denotes the expected number of steps to find k_i .



S_i = expected number of steps to find k_i

. = number of elements examined during search for k_i

. = 1+ number of elements **before** k_i in the linked list stored at $h(k_i)$

. = 1+ number of keys that hash same as k_i and are inserted after k_i .

csc263 lec4 ppt pg14

注意: **链接法 (chaining)** 中的插入和搜索的顺序 (不同)

“在对元素 x 的一次成功查找中, 所检查的元素就是 x 所在的链表中 x 前面的元素数多 1。在该链表中因为, 新的元素都是在表头插入的, 所以出现在 x 之前的元素都是 x 之后插入的” -教材 p146

chaining search avg - successful case

$$E[t_{m,n}(k)] = \frac{1}{n} \sum_{i=1}^n S_i$$

$$= \frac{1}{n} \left[1 + \sum_{j=i+1}^n E[X_{i,j}] \right]$$

$$= \frac{1}{n} \left[1 + \sum_{j=i+1}^n \left(\frac{1}{m} \right) \right]$$

\rightarrow k_i slot 的概率

$$= \frac{1}{n} \left[n + \sum_{i=1}^n \frac{n-i+1}{m} \right]$$

slot for probability

$$= \frac{1}{n} \left[n + \frac{n}{m} \cdot \frac{n+1}{2} \right]$$

$$= 1 + \frac{n}{m} - \frac{1}{m} - \frac{(n+1)}{2m}$$

$$= 1 + \frac{n}{2m} - \frac{1}{2m} \in \Theta(1 + \frac{n}{m})$$

search 的方向 $\rightarrow \rightarrow \rightarrow$
但并非 insert 的方向

insert 的方向 $\leftarrow \leftarrow \leftarrow$

first (key) element
second element
third element

表头为 linked list 链表的树式:

first
second
third
 $j=i+1$

定理 11.1 在 SUHA（简单均匀散列）的假设下，对于用链接法解决冲突的散列表，一次不成功查找的平均时间为 $\Theta(1+\alpha)$

定理 11.2 在 SUHA（简单均匀散列）的假设下，对于用链接法解决冲突的散列表，一次成功查找的平均时间为 $\Theta(1+\alpha)$