Hash tables

Data Structures & Algorithms

Motivation

- Many applications require a dynamic set that supports only the dictionary operations INSERT, SEARCH, and DELETE.
- A hash table is an effective data structure for implementing dictionaries.
- Under reasonable assumptions, the average time to search for an element in a hash table is O(1)

Direct-address tables

DIRECT-ADDRESS-SEARCH(T, k)

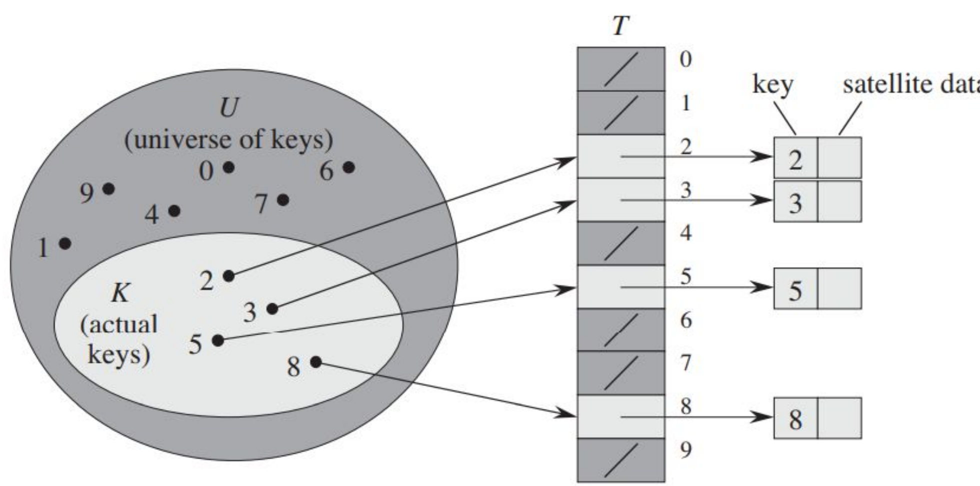
return $T[k]$

DIRECT-ADDRESS-INSERT(T, x)

$T[x.key] = x$

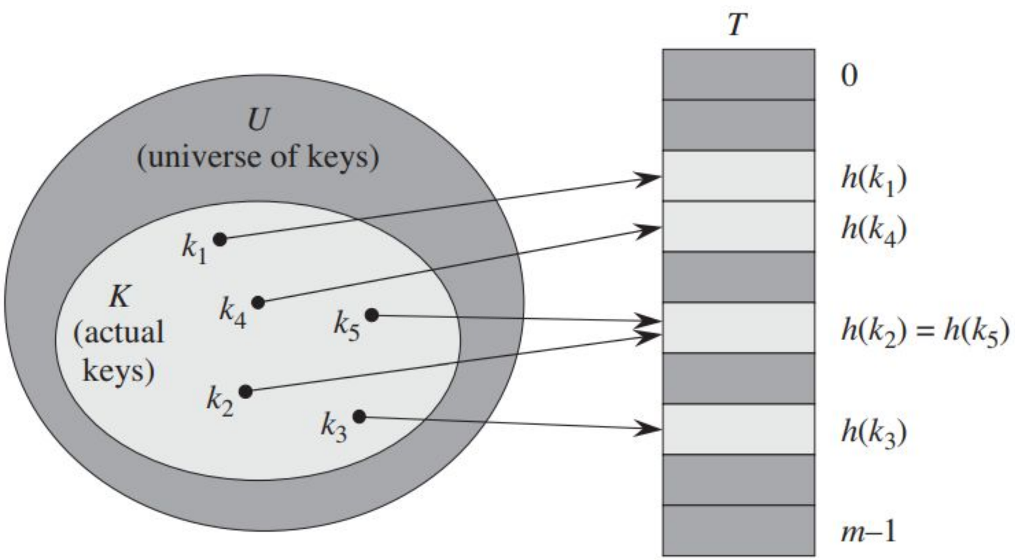
DIRECT-ADDRESS-DELETE(T, x)

$T[x.key] =$ NIL



Hash tables

two keys may hash to the same slot. We call this situation a **collision**



Collision resolution by chaining

CHAINED-HASH-INSERT(T, x)

insert x at the head of list $T[h(x.key)]$

CHAINED-HASH-SEARCH(T, k)

search for an element with key k in list $T[h(k)]$

CHAINED-HASH-DELETE(T, x)

delete x from the list $T[h(x.key)]$

Analysis of hashing with chaining

Load factor(alpha: the average number of elements stored in a chain) = n/m

n = # of elements

m =# of slots

In a hash table in which collisions are resolved by chaining, an unsuccessful search takes average-case time $\Theta(1+\alpha)$, under the assumption of simple uniform hashing.

Analysis of hashing with chaining

we take the average, over the n elements x in the table, of 1 plus the expected number of elements added to x's list after x was added to the list. Let xi denote the ith element inserted into the table, for i = 1, 2,..., n

$$X_{ij} = \mathrm{I}\{h(k_i) = h(k_j)\}$$

$$\Pr\{h(k_i) = h(k_j)\} = 1/m$$

$$\begin{aligned}
\mathrm{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left(1 + \sum_{j=i+1}^{n} X_{ij}\right)\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(1 + \sum_{j=i+1}^{n} \mathrm{E}[X_{ij}]\right) \quad \text{(by linearity of expectation)} \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(1 + \sum_{j=i+1}^{n} \frac{1}{m}\right) \\
&= 1 + \frac{1}{nm}\sum_{i=1}^{n}(n-i) \\
&= 1 + \frac{1}{nm}\left(\sum_{i=1}^{n} n - \sum_{i=1}^{n} i\right) \\
&= 1 + \frac{1}{nm}\left(n^2 - \frac{n(n+1)}{2}\right) \quad \text{(by equation (A.1))} \\
&= 1 + \frac{n-1}{2m} \\
&= 1 + \frac{\alpha}{2} - \frac{\alpha}{2n}\,.
\end{aligned}$$

Hash functions

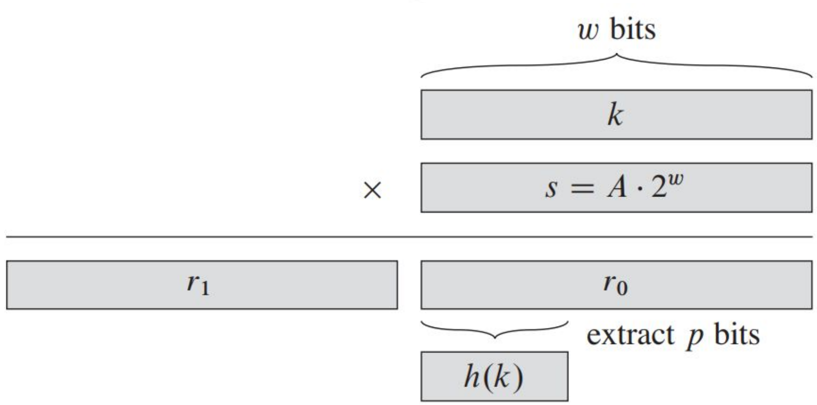
- The division method

$$h(k) = k \bmod m$$

- The multiplication method

$$h(k) = \lfloor m\,(kA \bmod 1) \rfloor$$

"$kA \bmod 1$" means the fractional part of kA, that is, $kA - \lfloor kA \rfloor$



Open addressing

```
HASH-INSERT(T, k)
i = 0
repeat
    j = h(k, i)
    if T[j] == NIL
        T[j] = k
        return j
    else i = i + 1
until i == m
error "hash table overflow"
```

```
HASH-SEARCH(T, k)
i = 0
repeat
    j = h(k, i)
    if T[j] == k
        return j
    i = i + 1
until T[j] == NIL or i == m
return NIL
```

Linear probing

Given an ordinary hash function $h' : U \rightarrow \{0, 1, \ldots, m-1\}$, which we refer to as an ***auxiliary hash function***, the method of ***linear probing*** uses the hash function

$$h(k, i) = (h'(k) + i) \bmod m$$

primary clustering: Clusters arise because an empty slot preceded by i full slots gets filled next with probability (i + 1)/m.

Long runs of occupied slots tend to get longer, and the average search time increases

Quadratic probing

Quadratic probing uses a hash function of the form

$$h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m ,$$

where h' is an auxiliary hash function, c_1 and c_2 are positive auxiliary constants,

secondary clustering: the initial probe determines the entire sequence, and so only m distinct probe sequences are used

Double hashing

$$h(k, i) = (h_1(k) + i h_2(k)) \bmod m$$

$$\begin{aligned} h_1(k) &= k \bmod m \,, \\ h_2(k) &= 1 + (k \bmod m') \,, \end{aligned}$$

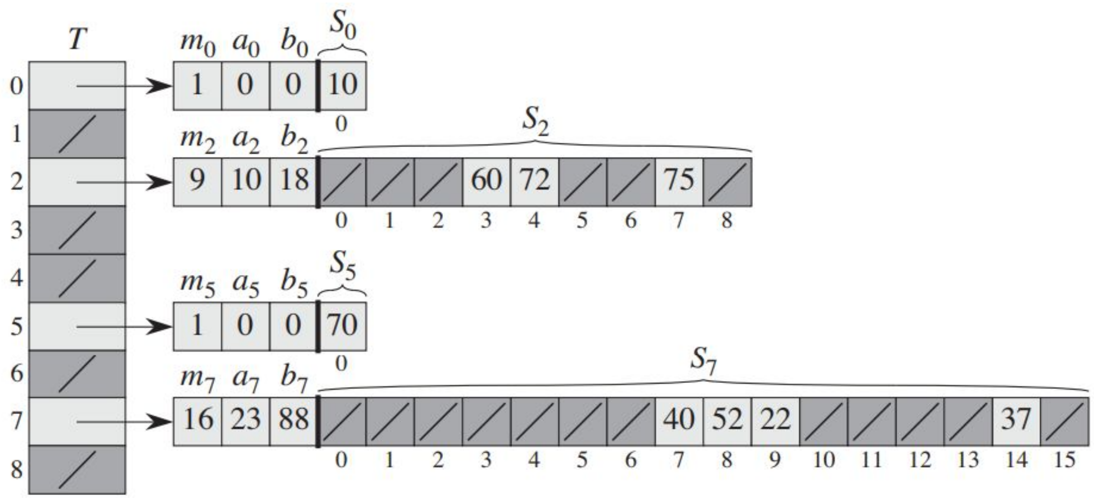
- The value h2(k) must be relatively prime to the hash-table size m for the entire hash table to be searched.

Exercise

11.4-1

Consider inserting the keys $10, 22, 31, 4, 15, 28, 17, 88, 59$ into a hash table of length $m = 11$ using open addressing with the auxiliary hash function $h'(k) = k$. Illustrate the result of inserting these keys using linear probing, using quadratic probing with $c_1 = 1$ and $c_2 = 3$, and using double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (k \bmod (m - 1))$.

Perfect hashing



Probability of collision

Theorem 11.9

Suppose that we store n keys in a hash table of size $m = n^2$ using a hash function h randomly chosen from a universal class of hash functions. Then, the probability is less than $1/2$ that there are any collisions.

$$\begin{aligned} \mathrm{E}[X] &= \binom{n}{2} \cdot \frac{1}{n^2} \\ &= \frac{n^2 - n}{2} \cdot \frac{1}{n^2} \\ &< 1/2 . \end{aligned}$$