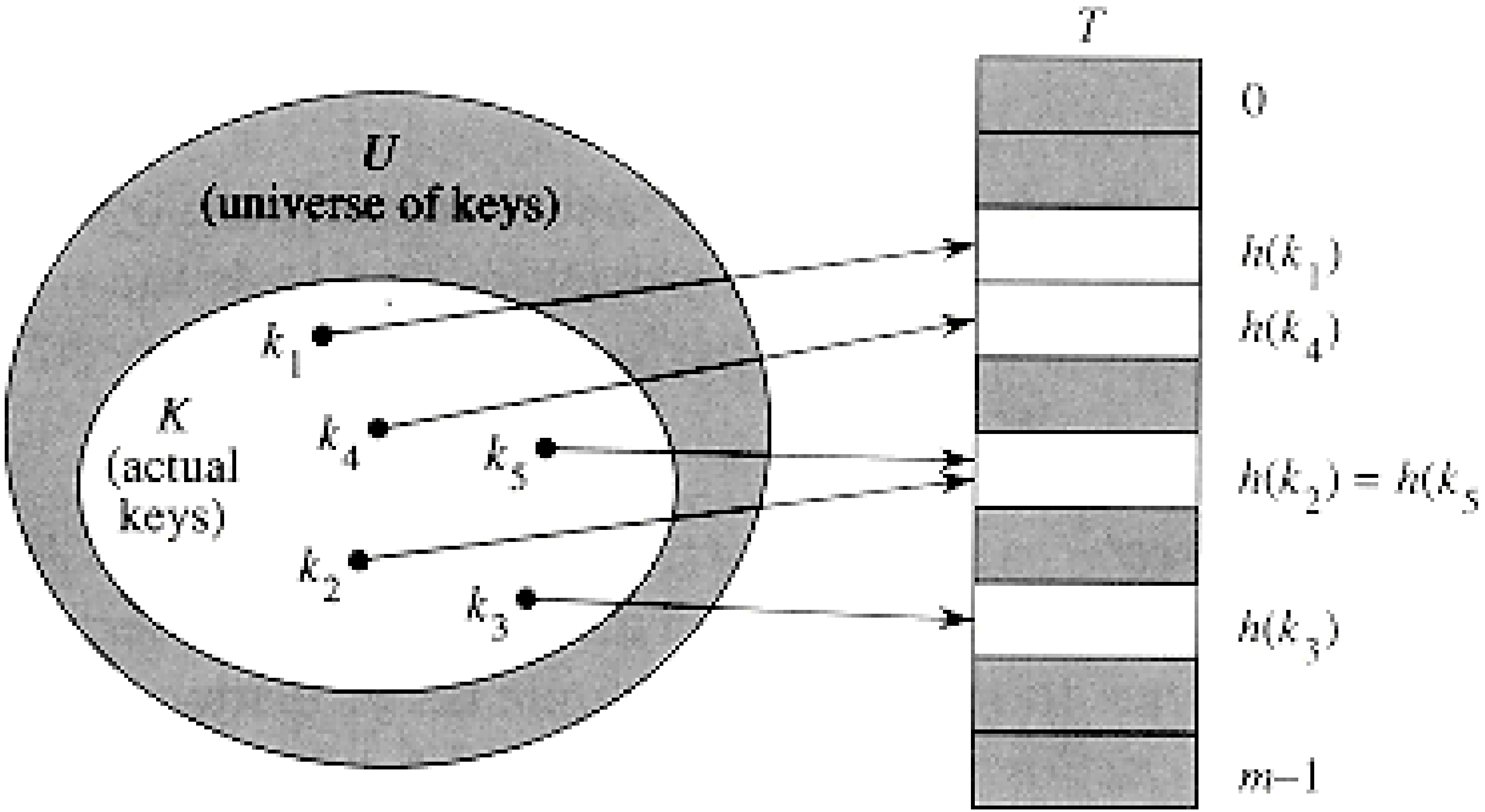
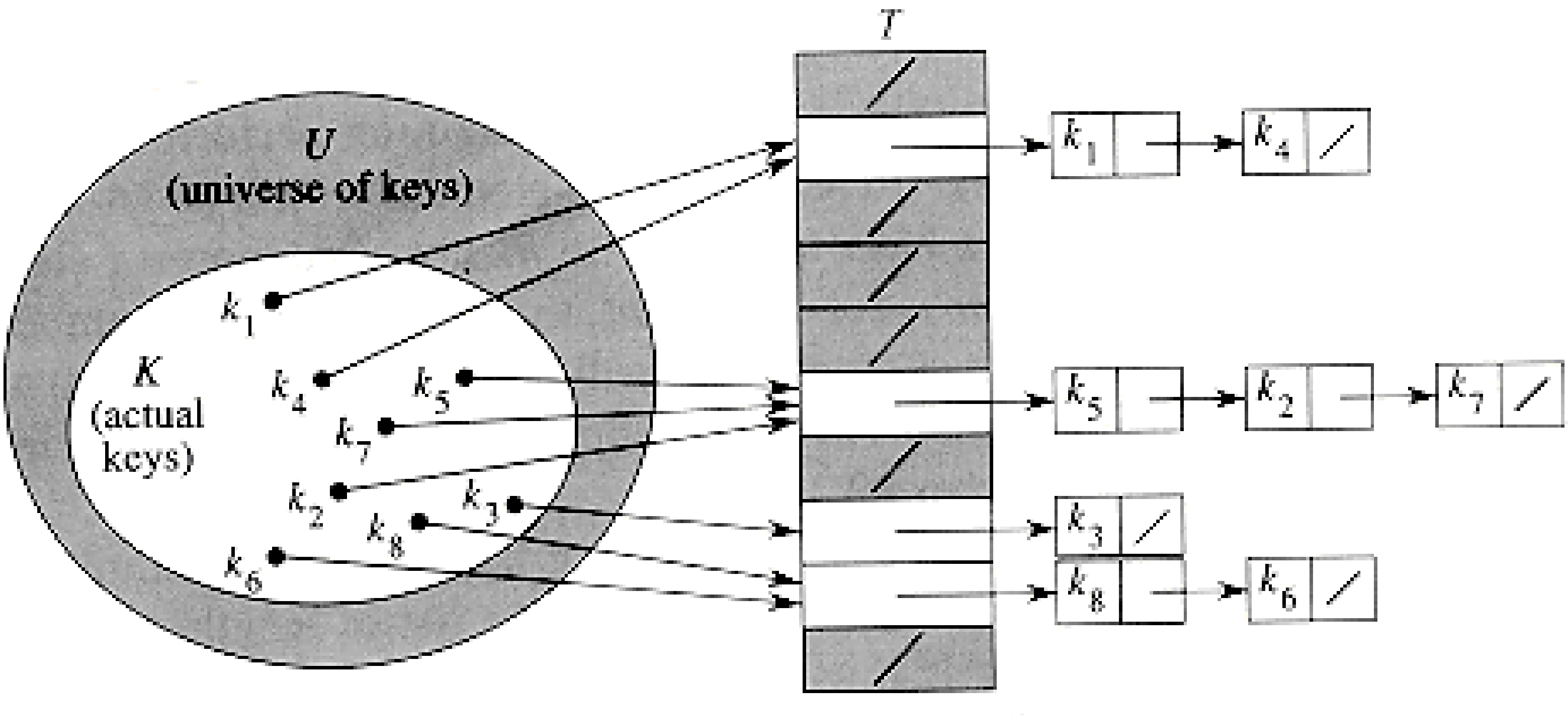
# Hash table



T
U
(universe of keys)
K
(actual
keys)
$k_1$
$k_4$
$k_5$
$k_2$
$k_3$
0
$h(k_1)$
$h(k_4)$
$h(k_2) = h(k_5)$
$h(k_3)$
$m-1$

# Collision resolution by chaining



# Collision resolution by chaining

operations on a hash table $T$

insert($T$,$x$)

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

search($T$,$k$)

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

delete($T$,$x$)

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

Running time

| | |
|---|---|
| insert : | $O(1)$ |
| search: | proportional to the length of the list |
| delete: | ( ***if the lists are singly linked*** ) |
| | proportional to the length of the list |

***if the lists are doubly linked and***
***when we know position***: $O(1)$

# Open addressing

store the records directly within the array

**probing**: search through alternate locations in the array (the probe sequence)

**Collisions** (solutions)

- linear probing

  the interval between probes is fixed - often at 1.
- quadratic probing

  the interval between probes increases proportional to the hash value (the interval increase linearly)
- double hashing

  the interval between probes is computed by another hash function

# Open addressing

**Formal:**

hash function is defined as follows:

- $h$: $U \times \{0, 1, \ldots, m-1\} \rightarrow \{0, 1, \ldots, m-1\}$

the ***probe sequence***

$\langle h(k, 0), h(k, 1), \ldots, h(k, m-1) \rangle$

*important:* acces every hash-table position

Assume that *(for the next examples)*

- each entry contains either a key or $\perp$

# Open addressing: linear probing

Given hash function $h'$: $U \rightarrow \{0, 1, . . . , m - 1\}$

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

Slot probed: $\boldsymbol{T[h'(k)]}$, $T[h'(k) + 1]$, … $T[m - 1]$,
$T[0]$, $T[1]$, . . . , until $T[h'(k) - 1]$.

Problem : ***primary clustering***

long runs of occupied slots build up, increasing the average search time.

**<u>Example</u>**

Consider keys: 53, 151, 54, 55, 56

illustrate their positioning in an initially empty hash table,

when m = 97 and $h'(k)=k \bmod m$

# Open addressing: quadratic probing

Given hash function $h'$: $U \rightarrow \{0, 1, \ldots, m - 1\}$,

$$\boldsymbol{h(k,i) = (h'(k) + c1*i + c2*i^{2}) \bmod m}$$

$c1$ and $c2$ <> 0 are auxiliary constants,

and $i = 0, 1, \ldots, m - 1$.

Problem: ***secondary clustering***

if two keys have the same initial probe position,

then their probe sequences are the same:

$$h(k1, 0) = h(k2, 0) \Rightarrow h(k1, i) = h(k2, i).$$

# Open addressing: double hashing

Given hash functions

$$h1, h2: U \Rightarrow \{0, 1, \ldots, m - 1\},$$

$$\boldsymbol{h(k,i) = (h1(k) + i*h2(k)) \bmod m}$$

- *h*1 and *h*2 - auxiliary hash func.

<u>Remark:</u>
one of the best methods for open addressing

# Open addressing: double hashing

**Choosing h1 and h2**

if $m$ and $h2(k)$ have greatest common divisor $d > 1$ for some key $k$, then a search for key $k$ would examine only $(1/d)$th of the hash table.

$h2(k)$ - relatively prime to the hash-table size $m$

Convenient ways to ensure this condition:

- $m$ be a power of 2
  design $h2$ so that it always produces an odd number
- let $m$ be prime
  and design $h2$ so that it always returns a positive integer less than $m$.

Example:

choose $m$ prime

$h1(k) = k \bmod m$ ,

$h2(k) = 1 + (k \bmod m')$,

where $m'$ slightly less than $m$ (say, $m - 1$ or $m - 2$).

# Open addressing

**Write subalg. for search, insert, delete .**

**Delete**

move the data

mark position with a special value DELETED

- modify SEARCH

    so that it keeps on looking when it sees the value DELETED,

- modify INSERT

    would treat DELETED slot as if it were empty (a new key can be inserted)

# Coalesced hashing

| | h1 |
|---|---|
| A.L. | 11 |
| AUDREY | 3 |
| AL | 5 |
| TOOTIE | 3 |
| DONNA | 10 |
| MARK | 4 |
| JEFF | 10 |
| DAVE | 9 |