Set ADT Implemantations (Part II)

Note Title 10/9/2007

Open Addressing

Use first empty slot defined by probe sequence $\langle s_0, \ldots, s_{m-1} \rangle$

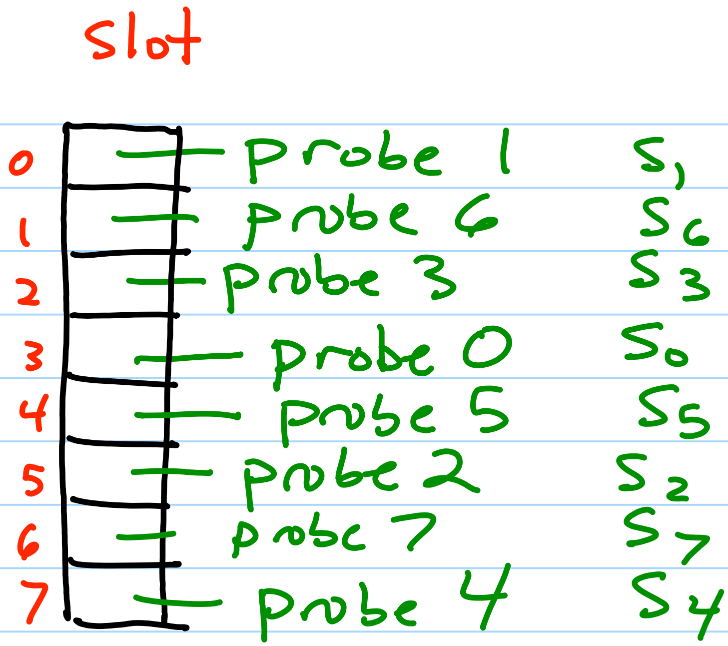
Let $x = \text{element.hashcode()}$

$s_0 = \text{hash}(x)$

$i = 1, \ldots, m-1 \quad s_i = (s_{i-1} + \text{stepHash}(x)) \% m$

mod

def of probe seq



$m = 8$

hash table size

$hash(x) = 3$

$stepHash(x) = 5$

Locating an element - Go through hash table in order of the probe sequence until you either reach the desired element or an empty (unused) slot is reached

element is not in the set

Adding an element

Go through hash table in order given by probe sequence until the element is found or an empty slot s is reached

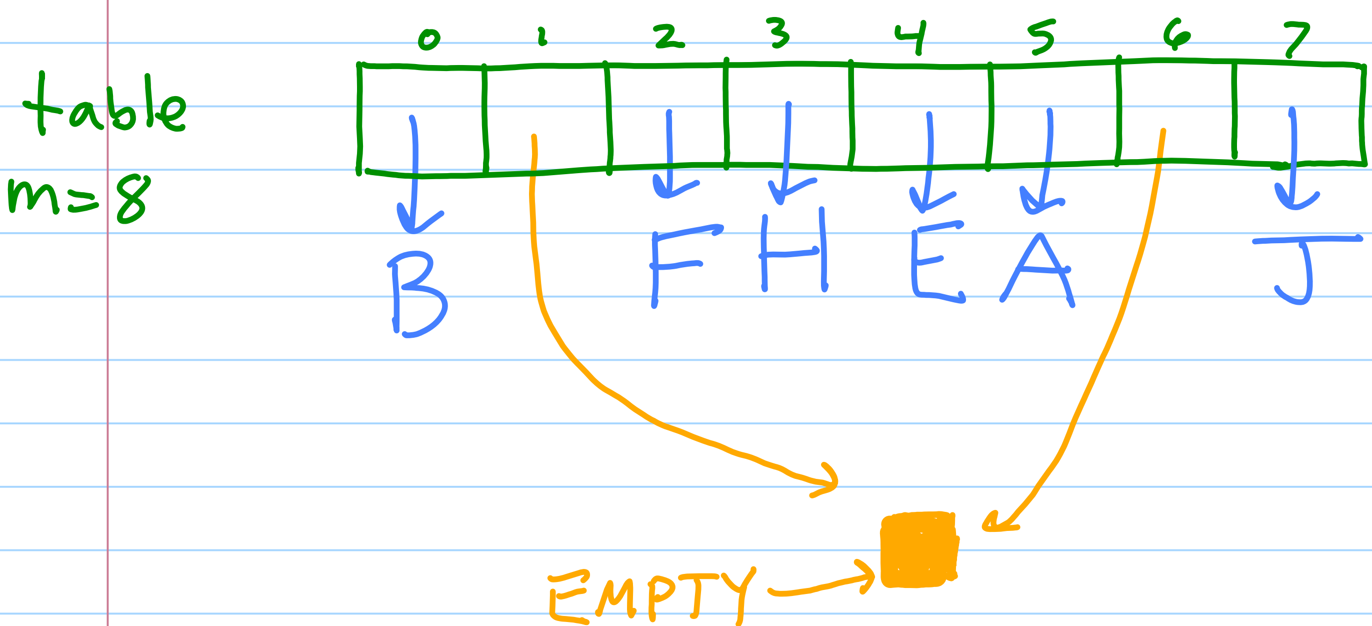
take appropriate action

element was not there & we will place it at slot s   table[s] = e

Let's look at an example.

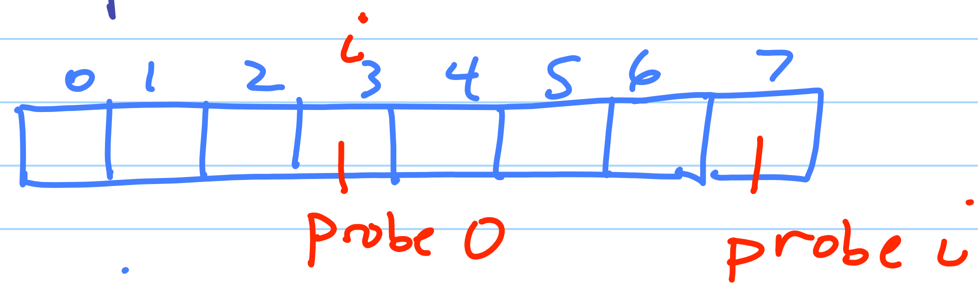
element e	A	B	E	F	H	J
$hash(e.hashCode())$	5	0	4	0	5	5
$stepHash(e.hashCode())$	7	1	5	5	3	5

Insert in order A, B, E, F, H, J



Care must be selected in how m and stepHash(x) relate. Why?

$m=8$



hash $= i$

stepHash $= 4$

Avoid this bad behavior & guarantee probe sequence is a permutation of $\langle 0, \ldots, m-1 \rangle$ by making m & stepHash relatively prime

Pick m to be a power of 2, make stepHash odd

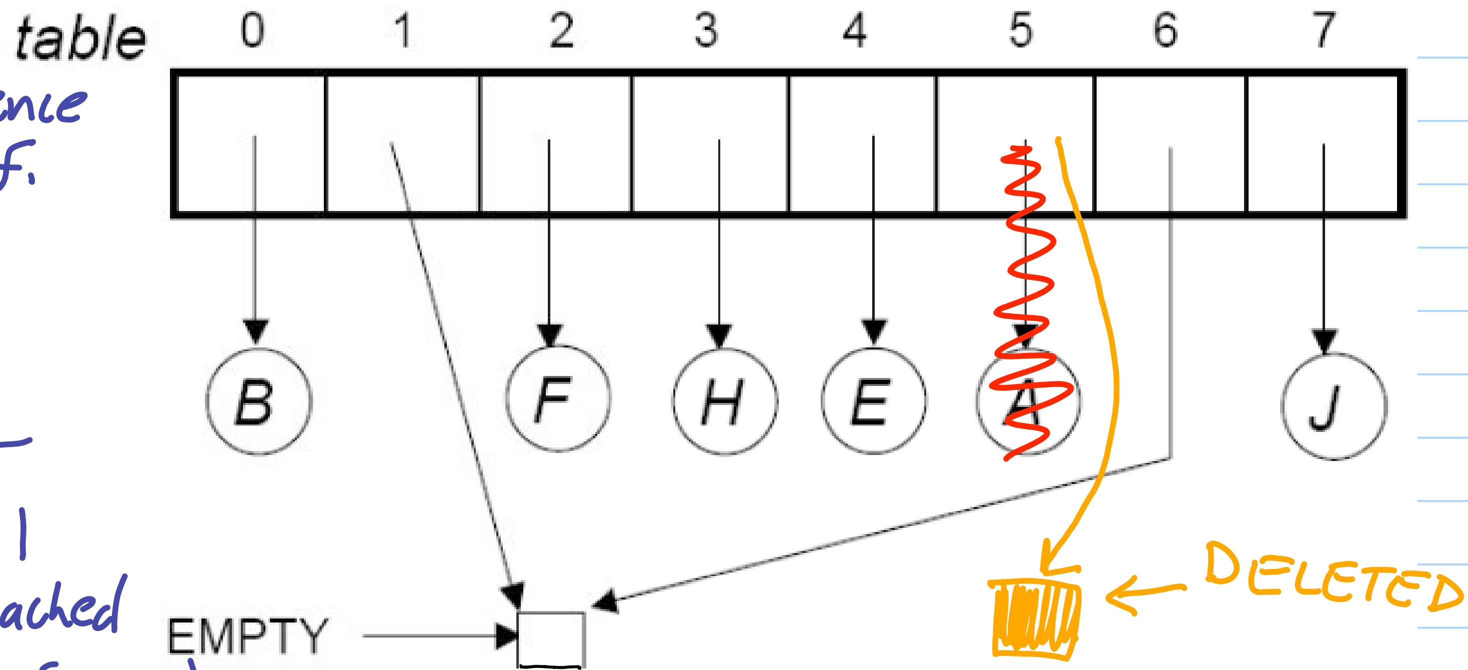
How can you remove an element?

element e	A	B	E	F	H	J
$hash(e.hashCode())$	5	0	4	0	5	5
$stepHash(e.hashCode())$	7	1	5	5	3	5

Let's delete A.

Now search for F.

Replace reference to A, by a ref. to DELETED sentinel.



Search must continue until EMPTY is reached (or element is found)

Deleting an element

Problem: hash table can fill up with slots "marked as deleted" (ref to DELETED)

Partly address this problem by re-using a deleted slot along probe sequence when inserting a new element.

only stop if element found or reach EMPTY slot

In a internal locate method (to check if element is in set), remember the 1st DELETED slot found & return that (or first empty slot if there were no deleted slots)

Actual load versus target load

Load factor $\alpha = \dfrac{n+d}{m}$

elements in set (n)

slots marked as deleted (d)

table size (m)

During an unsuccessful search, α is fraction of slots that will cause search to continue.

desired load factor α_* (e.g., $1/2$)

actual load is current value of $\dfrac{n+d}{m}$

actual load

desired load

Goal: Keep α close to α_*

Limit frequency of resizing ← expensive

Double table size (m) when

α reaches $\dfrac{1+\alpha_*}{2}$ } halfway between α_* + 1

when expected search cost is twice as large as desired

$\alpha_* = 1/2$, resize when $\alpha = 3/4$

hash functions change + you must rebuild by re-inserting all elements (in order)

go through slots, move to next if EMPTY or DELETED, reinsert elements in new table

The hash table could be oversized
(+ cluttered with deleted slots)

half table size when $\frac{n}{m}$ drops down to $\frac{\alpha^*}{2}$

reduce hash table size by a factor of 2.

Analysis

$$E\left[\begin{array}{c}\#\text{ probes in an}\\ \underline{\text{unsuccessful search}}\end{array}\right] = 1\cdot \begin{array}{c}\text{prob.}\\ \text{probe 0}\\ \text{occurs}\end{array} + 1\cdot \begin{array}{c}\text{prob}\\ \text{probe 1}\\ \text{occurs}\end{array} + \dots$$

$$\leq 1 + \alpha + \alpha^2 + \alpha^3 + \dots$$

probe 0 always occurs

n/m prob that a collision occurs on probe 0

very, very close to prob probe 2 occurs.

$$\frac{n}{m}\cdot\frac{(n-1)}{(m-1)} \leq \alpha^2$$

$$\leq \sum_{i=0}^{\infty} \alpha^i = \boxed{\frac{1}{1-\alpha}}$$

$$E\left[\begin{array}{c}\#\text{ probes in a}\\ \text{successful search}\end{array}\right] = \frac{1}{\alpha}\ln\frac{1}{1-\alpha}$$

Tradeoff between space + search time

as we increase α_*

use less space but higher expected cost per search

this is a pretty large value

$\alpha_* = 7/8 \qquad \frac{1}{1-7/8} = 8$

As we decrease α_*, use more space but have lower expected cost per search

$\alpha_* = 1/4 \qquad \frac{1}{1-1/4} = 4/3$

space $4m$