# Resolving collisions w/ open addressing assuming $n \leq m$

The point is to avoid auxilliary linked lists (and pointers). Use that space for table.

Instead, create a probe sequence as a function of key value.

↳ permutation of slots to try.

| | |
|---|---|
| 1 | 36 |
| 2 | 43 |
| 3 | |
| 4 | 78 |
| 5 | |
| 6 | 5 |
| 7 | 103 |
| 8 | |
| 9 | 2014 |
| 10 | |
| 11 | |

ex: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.$

Insert(64):

Try T[9] : full

Try T[2] : full

Try T[4] : full

Try T[8] : ok

Search(64) follows same sequence.

Would return "not found" after 4 attempts.

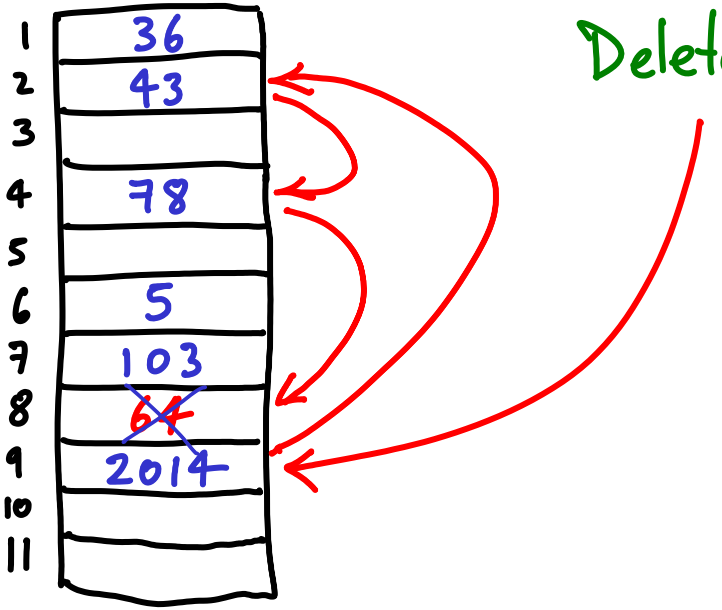
ex: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.$

Really this is $h(k, i)$

$h(64,1) = 9$ / $h(64,2) = 2$ / $h(64,3) = 4$ / etc

ex: $h(64) \rightarrow$ 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.

Really this is $h(k, i)$ $h(64,1) = 9$ / $h(64,2) = 2$ / $h(64,3) = 4$ / etc




Delete(64) : $h(64,1) = 9$, occupied by 2014

$h(64,2) = 2$, occupied by 43

$h(64,3) = 4$, occupied by 78

$h(64,4) = 8$, found 64, DELETE IT.

OK?

ex: $h(64) \to 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.$

Really this is $h(k, i)$ $h(64,1) = 9$ / $h(64,2) = 2$ / $h(64,3) = 4$ / etc

| | |
|---|---|
| 1 | 36 |
| 2 | 43 |
| 3 | |
| 4 | 78 |
| 5 | |
| 6 | 5 |
| 7 | 103 ✓ |
| 8 | DEL |
| 9 | 2014 |
| 10 | |
| 11 | |

Delete(64) : $h(64,1) = 9$, occupied by 2014
$h(64,2) = 2$, occupied by 43
$h(64,3) = 4$, occupied by 78
$h(64,4) = 8$, found 64, DELETE IT.

what if $h(103) \to 4, 8, 2, 7, 1, 10, 11, 3, 5, 9, 6$ ?

Search(103) : $h(103,1) = 4$, occupied by 78
$h(103,2) = 8$, empty : declare 103 not in T.

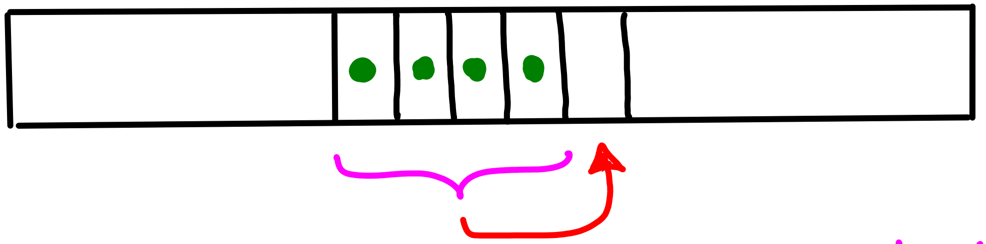
Could use special "deleted" markers, but search time increases

(Consider deleting all but one element, and then searching for it)

# Typical probing sequences

Linear probing : $h(k,i) = (h(k,0) + i) \bmod m$ ~ $h(k)$ and wrap around.

... tends to generate clusters.



$$\underbrace{\text{probability of extending a cluster} = \frac{|\text{cluster}|}{m}}_{\text{slows down search}}$$

# Typical probing sequences

Linear probing : $h(k,i) = (h(k,0) + i) \bmod m$ ~ $h(k)$ and wrap around.

... tends to generate clusters.

Quadratic probing : $h(k,i) = (h(k,0) + \underbrace{c \cdot i}_{\text{linear}} + \underbrace{d \cdot i^2}_{\text{make it look more random}}) \bmod m$

Less clustering, need to make sure sequence hits all slots

Both generate $m$ probe sequences in total

Double hashing : $h(k,i) = (h_1(k) + \underbrace{i \cdot h_2(k)}_{\text{each } k \text{ has "random" offset}}) \bmod m$

Generates $O(m^2)$ probe sequences: better

Heuristic : choose $m = 2^r$ & $h_2(k)$ : odd.

# ANALYSIS OF OPEN ADDRESSING

ASSUMING UNIFORM HASHING : each key is equally likely to have any of the $m!$ permutations as probe sequence (independent of other keys)

even though all we have so far is $O(m^2)$

$\neq$

simple uniform hashing

For a random $h$, every slot is equally likely

# ANALYSIS OF OPEN ADDRESSING

ASSUMING UNIFORM HASHING : each key is equally likely to have any of the $m!$ permutations as probe sequence (independent of other keys)

Recall $n < m$, so $\alpha < 1$. Claim: $E[\#\text{probes}] \le \frac{1}{1-\alpha}$ $\left(\frac{m}{m-n}\right)$
(search)

If true, then for $n \ll m$ $E[\#\text{probes}] = O(1)$

↳ $n = \frac{1}{2}m \rightarrow 2$ probes

↳ 90% full table → 10 probes

Works well if you can afford a table ~ data ×2

but keep in mind: we're using a very strong assumption

Claim: $E[\#\text{probes}] \leq \frac{1}{1-\alpha}$

Look at unsuccessful search

$P[\text{1st probe collides}] = \frac{n}{m}$ → need 2nd probe

$P[\text{2nd probe collides}] = \frac{n-1}{m-1}$ → need 3rd probe

$\vdots$

$\frac{n-i}{m-i} < \frac{n}{m} = \alpha$

Remember, probe sequence is a permutation. Never check one slot twice.

$$E[\#\text{probes}] = 1 + \frac{n}{m}\left(1 + \frac{n-1}{m-1}\left(1 + \frac{n-2}{m-2}\left(\cdots\left(1 + \frac{0}{m-n}\right)\right)\right)\right)$$

$$\leq 1 + \alpha\left(1 + \alpha\left(1 + \alpha\left(\cdots\left(1 + \alpha\right)\right)\right)\right) \quad \text{... n terms}$$

$$\leq 1 + \alpha + \alpha^2 + \alpha^3 + \cdots \quad \text{.... } \infty \text{ terms}$$

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

see CLRS for alternate analysis incl. successful search

Suggested reading:

perfect hashing — Family of hash functions, pick one randomly. Beats adversaries.

universal hashing — Fixed input. Create h( ) based on this.

cuckoo hashing ! — Cuckoos Use Mafia Tactics, And They Work

April 18, 2014 | by Stephen Luntz