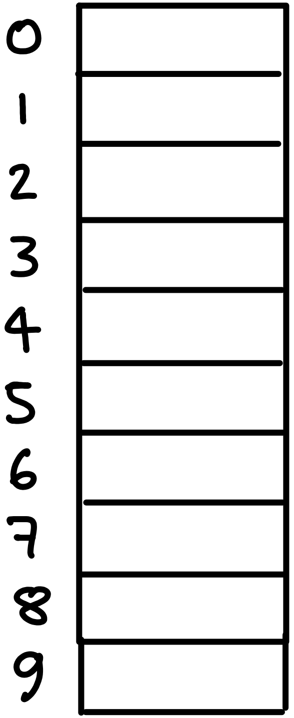
OPEN ADDRESSING

Avoid using pointers in linked lists. Use that space for a larger table.



To be clear, for the same number of keys, chaining uses extra pointers that take more space

OPEN ADDRESSING

Require $n \le m$

Avoid using pointers in linked lists. Use that space for a larger table.

We use a probe sequence → permutation of all slots.

$m = 10$

0	
1	36
2	43
3	
4	78
5	5
6	103
7	
8	
9	2014

e.g., $h(64) = \{9, 2, 4, 8, 1, 3, 0, 7, 5, 6\}$

Search(64)

Try T[9] : not 64

Try T[2] : not 64

Try T[4] : not 64

Try T[8] : "not found"

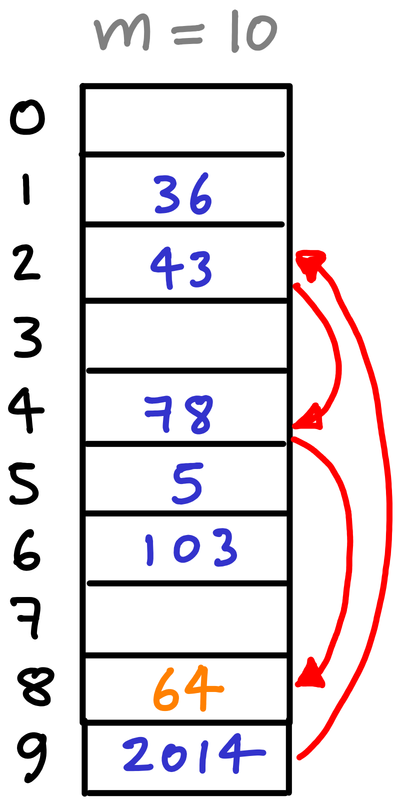
OPEN ADDRESSING

Require $n \le m$

Avoid using pointers in linked lists. Use that space for a larger table.

We use a probe sequence → permutation of all slots.

e.g., $h(64) = \{9, 2, 4, 8, 1, 3, 0, 7, 5, 6\}$



Search(64)	Insert(64)
Try T[9] : not 64	Try T[9] : full
Try T[2] : not 64	Try T[2] : full
Try T[4] : not 64	Try T[4] : full
Try T[8] : "not found"	Try T[8] : OK

$h(64) = \{9, 2, 4, 8, 1, 3, 0, 7, 5, 6\}$ i = iteration. Then $h(k) \to h(k,i)$

the probe sequence has to be generated somehow

via function $h(k,i)$

$h(64) = \{9, 2, 4, 8, 1, 3, 0, 7, 5, 6\}$ i = iteration. Then $h(k) \to h(k, i)$

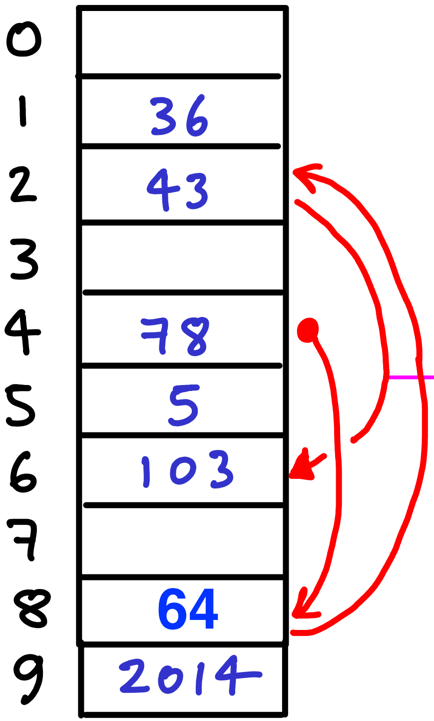
Delete(64) :

$h(64,1) = 9$ $h(64,2) = 2$ $h(64,3) = 4$ $h(64,4) = 8$

found 64, can delete

0	
1	36
2	43
3	
4	78
5	5
6	103
7	
8	64
9	2014

$h(64) = \{9, 2, 4, 8, 1, 3, 0, 7, 5, 6\}$ i = iteration. Then $h(k) \rightarrow h(k,i)$



Delete(64) :

$h(64,1) = 9$ $h(64,2) = 2$ $h(64,3) = 4$ $h(64,4) = 8$

found 64, can delete

Problem: what if $h(103) = \{4, 8, 2, 6, \ldots$

(103 was inserted after 64)

$h(64) = \{9, 2, 4, 8, 1, 3, 0, 7, 5, 6\}$ $\quad i =$ iteration. Then $h(k) \to h(k, i)$

0	
1	36
2	43
3	
4	78
5	5
6	103
7	
8	
9	2014

Delete(64) :

$h(64,1) = 9 \quad h(64,2) = 2 \quad h(64,3) = 4 \quad h(64,4) = 8$

found 64, can delete

Problem: what if $h(103) = \{4, 8, 2, 6, \ldots$

(103 was inserted after 64)

Now, search(103)

after having deleted 64

$h(64) = \{9, 2, 4, 8, 1, 3, 0, 7, 5, 6\}$ i = iteration. Then $h(k) \to h(k,i)$

0	
1	36
2	43
3	
4	78
5	5
6	103
7	
8	
9	2014

Delete(64) :

$h(64,1) = 9$ $h(64,2) = 2$ $h(64,3) = 4$ $h(64,4) = 8$

found 64, can delete

Problem: what if $h(103) = \{4, 8, 2, 6, \ldots$

(103 was inserted after 64)

Now, search(103) : $h(103,1) = 4$, $h(103,2) = 8$

"not found"

$h(64) = \{9, 2, 4, 8, 1, 3, 0, 7, 5, 6\}$ i = iteration. Then $h(k) \to h(k, i)$

0	
1	36
2	43
3	
4	78
5	5
6	103
7	
8	DEL
9	2014

Delete(64) :

$h(64,1) = 9$ $h(64,2) = 2$ $h(64,3) = 4$ $h(64,4) = 8$

found 64, can delete

Problem: what if $h(103) = \{4, 8, 2, 6, \ldots$

(103 was inserted after 64)

Now, search(103) : $h(103,1) = 4$, $h(103,2) = 8$

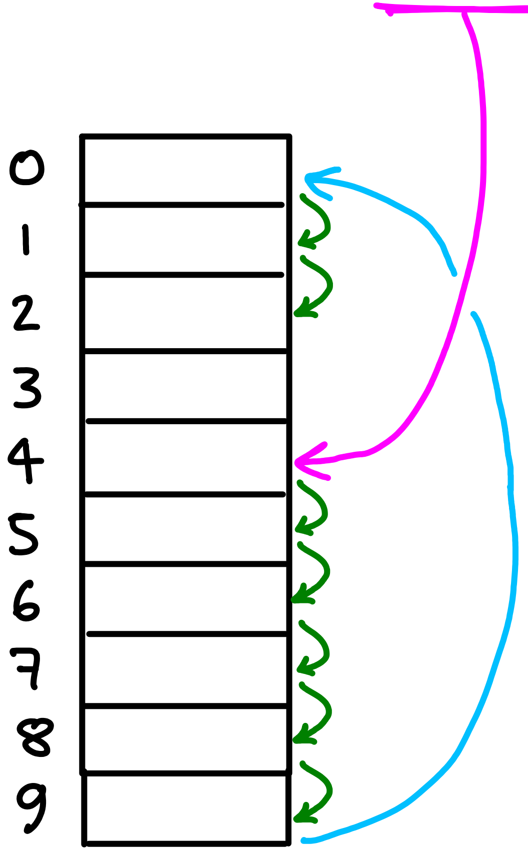
"not found"

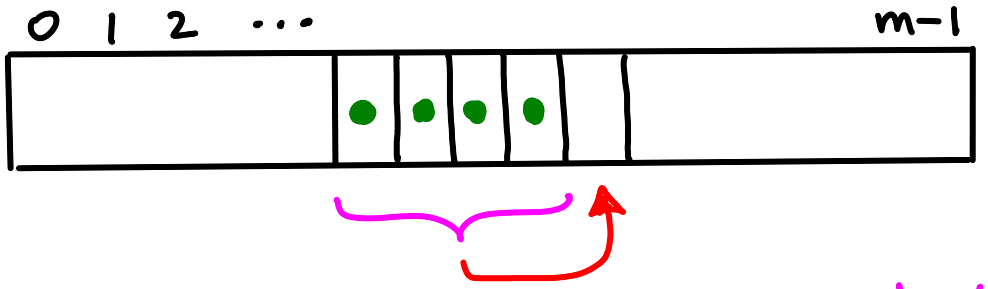
Could use special "deleted" markers, but search becomes inefficient.

e.g., insert n elements, delete n-1, search for last remaining.

Typical probing sequences

Linear probing: $h(k,i) = \big(h(k,0) + i\big) \bmod m$ $\quad\sim h(k)$ & scan





$$\underbrace{\text{probability of extending a cluster} = \frac{|\text{cluster}|}{m} \gg \frac{1}{m}}_{\text{slows down search}}$$

Typical probing sequences

Linear probing: $h(k,i) = (h(k,0) + i) \bmod m$ ~ h(k) & scan

...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

(number theory)

Both generate m probe sequences

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

Can generate up to m^2 probe sequences: better

ANALYSIS of OPEN ADDRESSING

ASSUMPTION: UNIFORM HASHING

Every key is equally likely to have any of the $m!$ permutations as a probe sequence

(and all probe sequences are independent)

- The common probing methods that we saw don't even come close

---

Don't confuse with Simple Uniform Hashing

(assumption for Chaining)

ANALYSIS of OPEN ADDRESSING with UNIFORM HASHING ASSUMPTION

Recall, $n < m$, so $\alpha < 1$

Claim: Expected #probes when searching $\leq \frac{1}{1-\alpha}$ $\left(\frac{m}{m-n}\right)$

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

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

↳ 90% full table $\rightarrow$ 10 probes

Claim: $E[\#\text{probes}] \le \frac{1}{1-\alpha}$ e.g., consider 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

$\vdots$

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

$E[\#\text{probes}] = 1 + \frac{n}{m}\left(1 + \underbrace{\frac{n-1}{m-1}}_{\text{probability of needing to probe more}}(\cdots)\right)$

2nd probe

probability of needing to probe more

Claim: $E[\#\text{probes}] \le \frac{1}{1-\alpha}$ e.g., consider 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

$$\vdots$$

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

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

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

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

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

see CLRS for further analysis including successful search