CS180 - Hash Tables p.3

Note Title 11/23/2009

Announcements

- Program due tomorrow
- Homework is out - due by midnight on 12/1

Dictionaries:

A structure which supports the following:

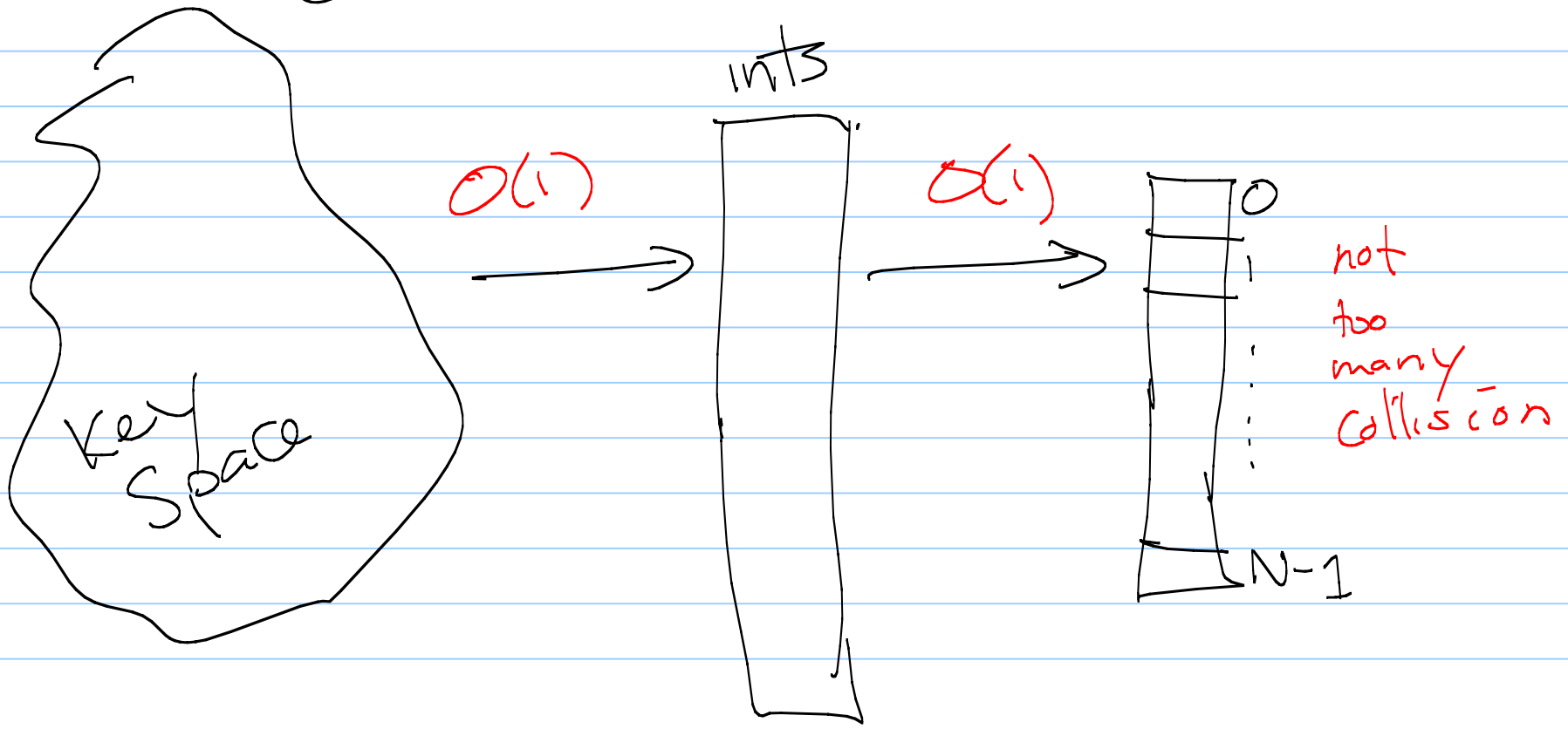
```
void insert(keyType &k, dataType &d)
dataType find(keyType &k)
void remove(keyType &k)
```

Examples: (key, data)

- Locker number & name
- Flight # & arrival info

Hashing for fast lookups

Hashing - big picture



Collisions

Can we ever totally avoid collisions?

NO

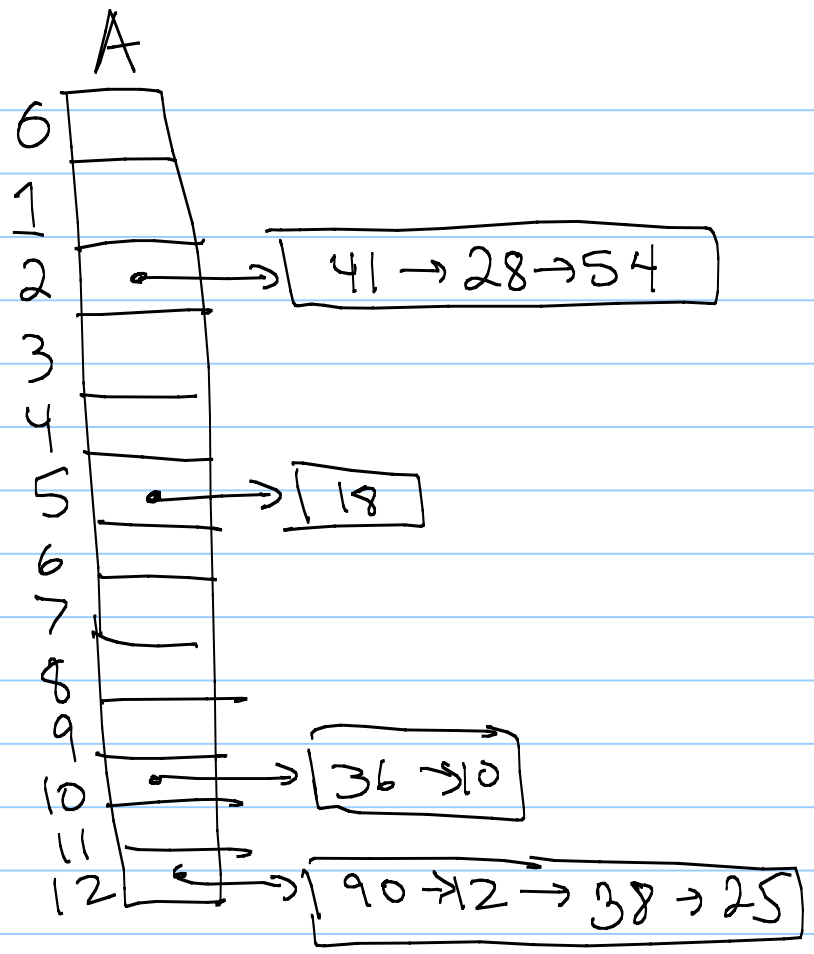
Keyspace is larger than our array!

How can we handle collisions?

(Do we have data structures to store more than one thing??)

- vectors
- lists
- tree

Ex:



find(38)

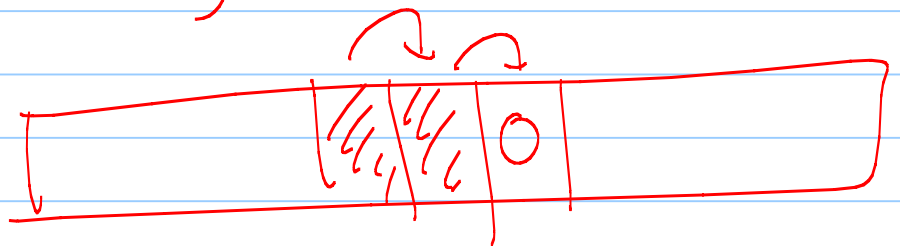
worst case - O(N)
everything hashes to same #

insert → O(1)

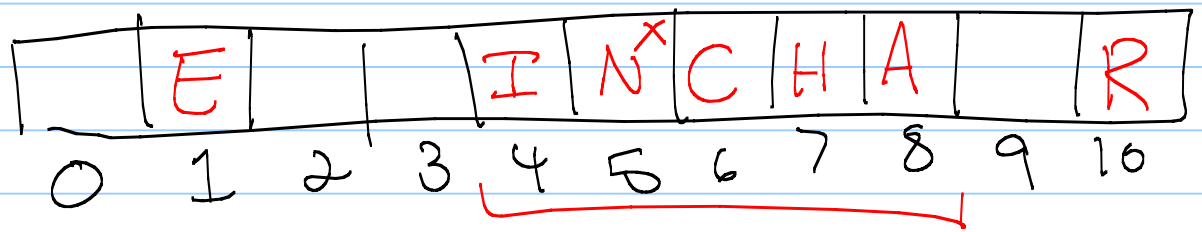
Linear Probing:

Instead of lists, if we hash to a full spot, just keep checking next spot until it is empty.

(assuming table never fills up)



Example: use Linear probing



find(49, X)

- Map is $h(k) = k \bmod 11$

key, data

insert(12, E)
insert(21, R)
insert(37, I)
insert(26, N)
insert(16, C)
insert(5, H)
insert(15, A)

$12 = 1 \bmod 11$
$21 \bmod 11 = 10$
$37 \bmod 11 = 4$
$26 \bmod 11 = 4$
$16 \bmod 11 = 5$
$5 \bmod 11 = 5$
$15 \bmod 11 = 4$

N = size of table

Running times:

Find? $O(n)$ n = #items inserted

Insert? $O(n)$

Remove? naive – $O(n+N)$ assuming key & data are stored

or $O(n^2)$

Usually, take a shortcut & mark a cell as "deleted".

Quadratic Probing.

Notice: Linear Probing checks spot $A[h(i)+1 \bmod N]$ if $A[h(i)]$ is full.

To avoid clusters, instead try

$A[(h(i) + j^2) \bmod N]$ where $j = 0, 1, 2, 3 \ldots$

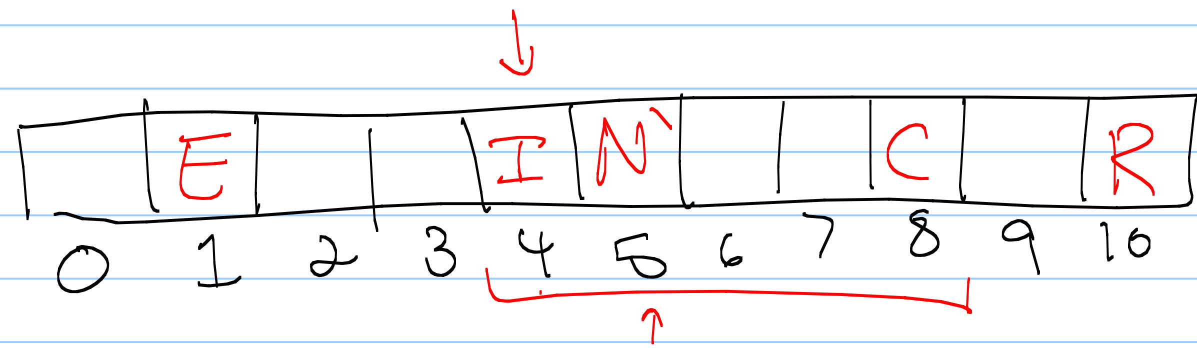
$A[h(i)]$

$A[h(i)+1 \bmod N]$

$A[h(i)+4 \bmod N]$

$A[h(i)+9 \bmod N]$

⋮



(48, C)

4

↓

4+1

↓

4 + 4 ← 2^2

Quadratic Probing issues:

- still cause "secondary clustering"
- N really must be prime for this to work
- Even with N prime, may fail if array is half full

may actually fail to ever find an open spot

Double Hashing

Try $A[h(i)]$

$+$ $+1$ for linear
$+j^2$ for quadratic

$$A[h(i) + f(j) \bmod N]$$

$$f(j) = j \cdot h'(k)$$

h' is another hash function

k is key of data already stored in $A[h(i)]$

Load Factors

Most of these techniques only work well if $\frac{n}{N} < .5$

Even chaining gets worse if $\frac{n}{N} > .9$

A lot of code periodically checks $\frac{n}{N}$, + rehashes if it is $> .5$