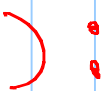


CS 180 - Hash Tables 2

Announcements

- Midterm 2 back today

Average: 36.9
Std Dev: ~10



max: 55
min: 19

- Program 5 - due Tuesday
- Next homework out - this weekend, break due when we get back from break

Dictionaries:

A structure which supports the following;

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

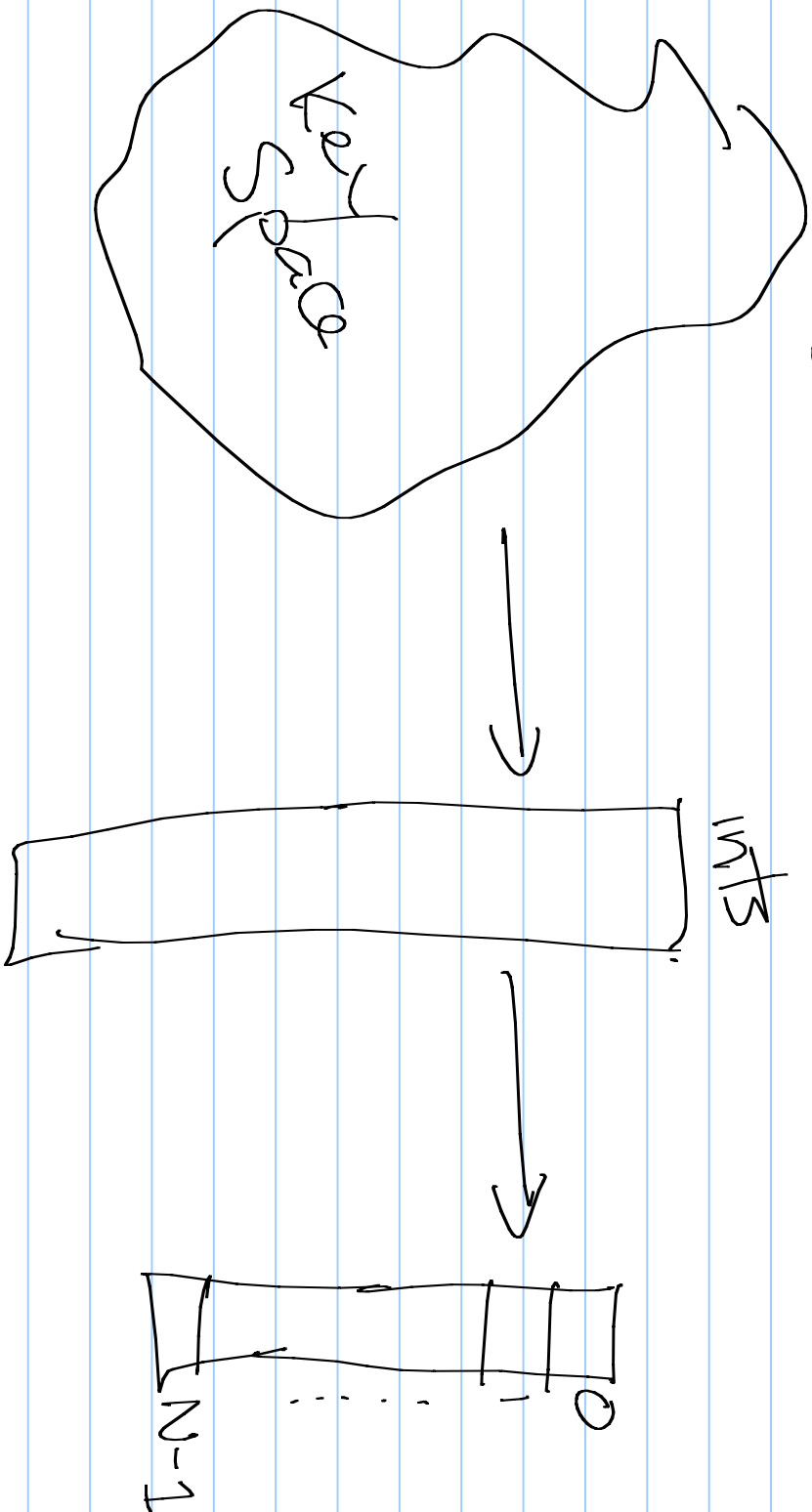
Examples:

~ Locker number & name

key data

- Flight # of arrival info

Hashing - big picture



Strategies for converting non-numeric keys to numeric keys

- Depends on type:

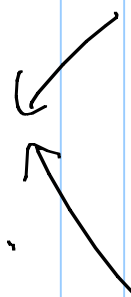
char: cast to an int

long: add first 32 bits to 2nd 32 bits

string: Polynomial hashing

temp01 temp10

Same int



Compressing numbers down to something
between 0 and $N-1$

Issues?

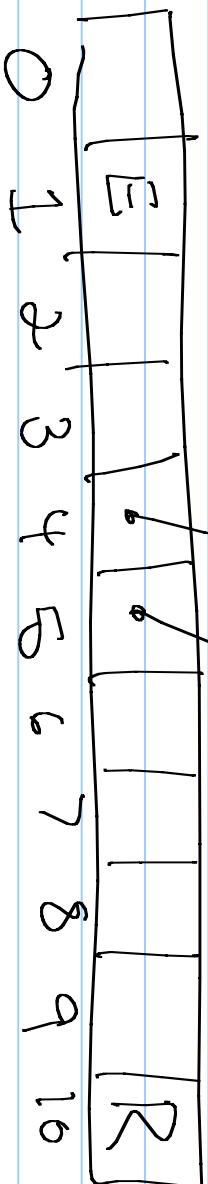
Modular arithmetic

$$\underbrace{h(x)}_{\text{int}} \bmod N$$

$$x \bmod n$$

Example

$I \rightarrow N$
 $C \rightarrow H$



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

- insert(12, E) $12 \bmod 11 = 1$
- insert(21, R) $21 \bmod 11 = 10$
- insert(37, I) $37 \bmod 11 = 4$
- insert(26, N) $26 \bmod 11 = 4$
- insert(16, C) $16 \bmod 11 = 5$
- insert(5, H) $5 \bmod 11 = 5$

Some comments:

This works better if size of table is a prime number.

Why?

Go take number theory

100
↓
101

The MATH (multiply add + divide) method is a bit better:

$$h(x) = |ax + b| \bmod N$$

where a, b are - not equal

- relatively prime

$$\gcd(a, b) = 1$$

21, 16

1, 3, 7, 21

1, 2, 4, 8, 16

- less than N

Goal: Simple Uniform Hashing Assumption:

For all $k \in \text{keyspace}$,

$$\Pr[h(k) = i] = \frac{1}{N}$$

(Essentially, elements are "thrown randomly" into the buckets)

Collisions

Can we ever totally avoid collisions?

NO

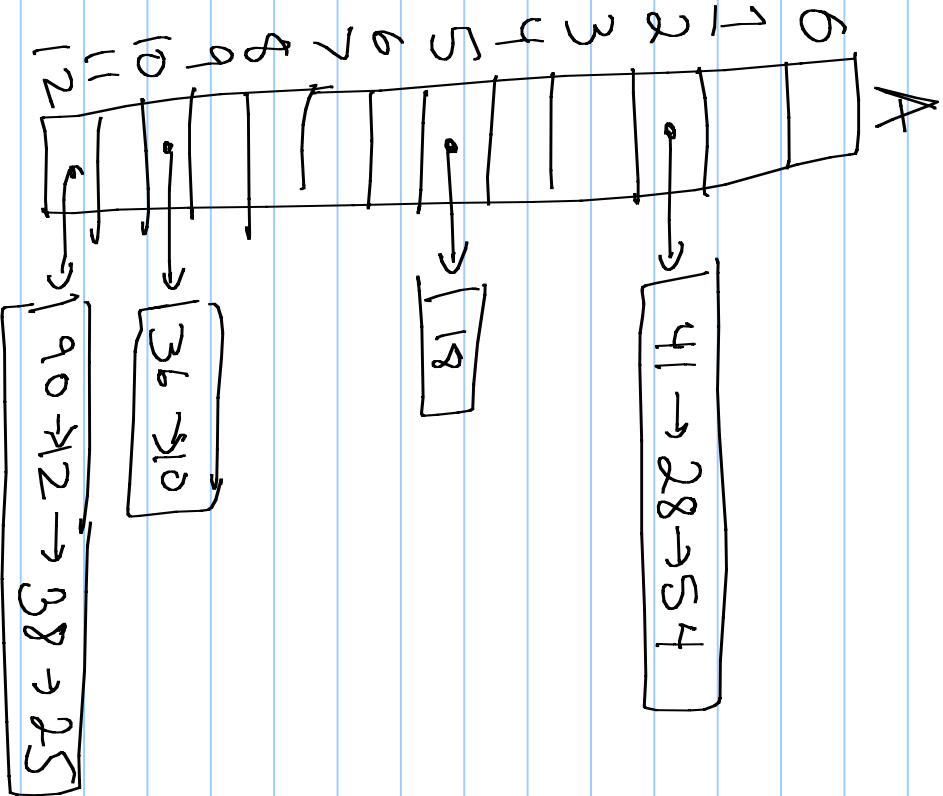
Key space is larger than our
array!

How can we handle collisions?

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

- linked list
- tree
- heaps
- vector

Ex:



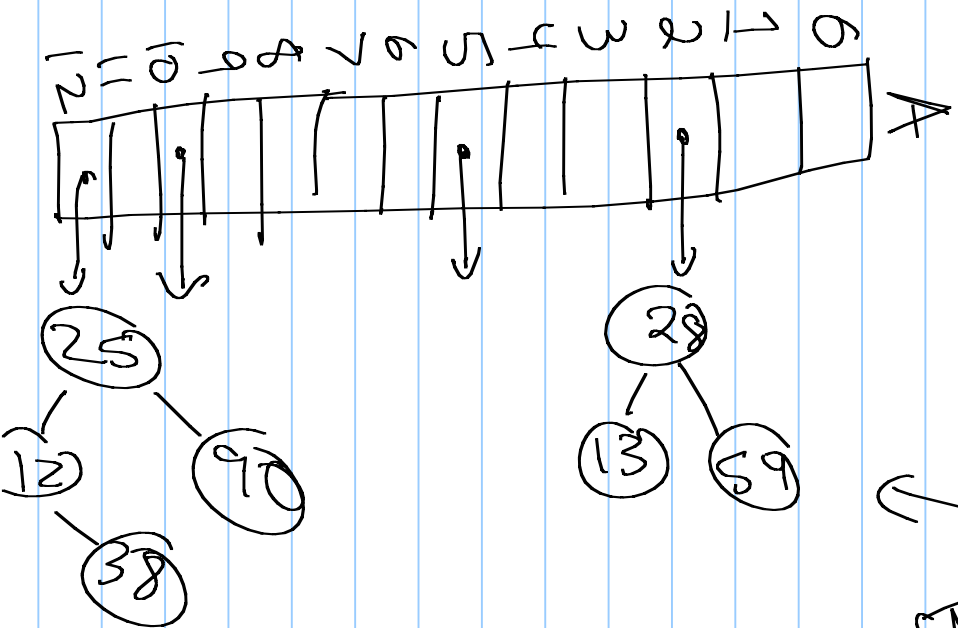
find (38)

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

insert $\sim O(1)$

Ex

balanced (ex AVL)
↓
BST



insert

$O(\log n)$

find

$O(\log n)$

