Announcements

- Check point today

- Program is due Sat. by midnight
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  a name
  - flight  #  arrival time
Hashing for fast lookups
Hashing - big picture

Key Space

Taking keys if hashing into O(1)

Polynomial computation

Avoid collisions

h(k)

O(1)

0

...
Collisions

Can we ever totally avoid collisions?

No

Keyspace is larger than our array!

Yesterday - strategies to deal with collisions.
How can we handle collisions?

**Strategy #1:**

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

- vectors
- lists
- tree
Ex:

N = size of table
n = # of elements in the table

Running times:

Find: $O(n)$
(find every key hashed to same spot)
$O(\log n)$

Insert: $O(1)$
$O(\log n)$
Linear Probing:

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

[Diagram of linear probing with arrows indicating the process]
Example:

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

  - insert(12, E)
    - 12 mod 11 = 1
  - insert(21, R)
    - 21 mod 11 = 10
  - insert(37, I)
    - 37 mod 11 = 4
  - insert(26, N)
    - 26 mod 11 = 4
  - insert(16, C)
    - 16 mod 11 = 5
  - insert(15, H)
    - 15 mod 11 = 5
  - insert(15, A)
    - 15 mod 11 = 4
Remove? Instead of deleting, mark as deleted.
Quadratic Probing.

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

To avoid clusters, instead try $A[(h(i) + j^2) \mod N]$ where $j = 0, 1, 2, 3, ...$

$A[h(i) \mod N]$ if full

1. $A[h(i) + 1 \mod N]$ if full
2. $A[h(i) + 2^2 \mod N]$ if full
3. $A[h(i) + 3^2 \mod N] + 4$
\[
\begin{align*}
\text{insert}(12, E) & \quad 12 \mod 11 = 1 \\
\text{insert}(21, R) & \quad 21 \mod 11 = 10 \leftarrow \\
\text{insert}(37, I) & \quad 37 \mod 11 = 4 \\
\text{insert}(26, N) & \quad 26 \mod 11 = 4 \leftarrow \\
\text{insert}(16, C) & \quad 16 \mod 11 = 5 \leftarrow \\
\text{insert}(5, H) & \quad 5 \mod 11 = 5 \leftarrow \\
\text{insert}(15, A) & \quad 15 \mod 11 = 4 \leftarrow 
\end{align*}
\]
Quadratic Probing issues:

- Still cause “secondary clustering”
- $N$ really must be prime for this to work (can’t have a lot of divisors)
- Even with $N$ prime, may fail if array is half full
Double Hashing

Try \[ ACh(i) \]

if full:

\[ A \left[ h(i) + f(j) \mod N \right] \]

In linear probing, \( +j \)

In quadratic, \( +j^2 \)

\[ f(j) = j \cdot h'(k) \]

\( h' \) is another hash function

\( k' \) is key of data already stored in \( ACh(i) \)
Load Factors

Most of these techniques only work well if \( \frac{n}{N} < 0.5 \)

Even chaining gets worse if \( \frac{n}{N} > 0.9 \)

A lot of code periodically checks \( \frac{n}{N} \), + rehashes if it is > .5
+ checks if many things have been removed