# CSSE 230 Day 19

More hash tables

EditorTrees

Check out from SVN:
HashSetExercise (individ repos)

# Announcements

- See schedule page

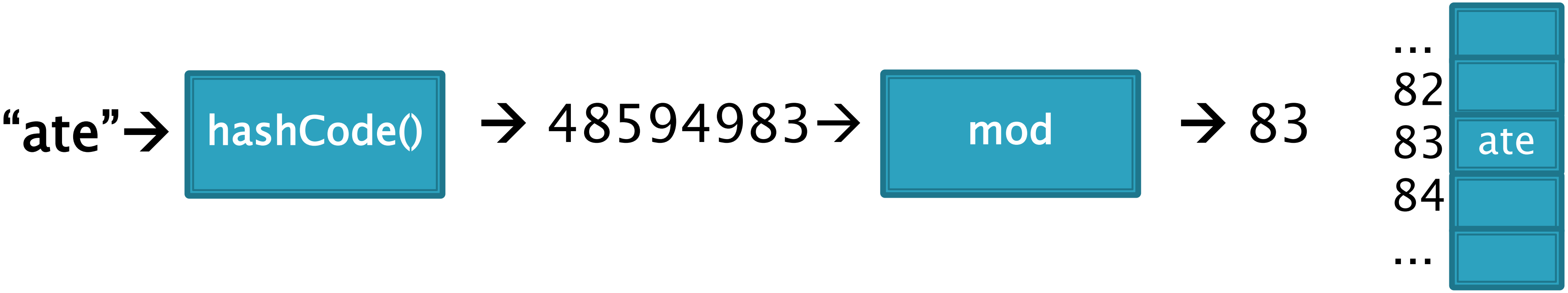
- Google created a new hash function for Strings, reported to be 30–50% faster than others:

http://google-opensource.blogspot.com/2011/04/introducing-cityhash.html

- Questions?

# Review: hash codes distribute keys across an array

- But if there’s already an element at (hashCode() % m), we have a **collision**!



# Collision Resolution: Linear Probing

- Collision? Use the next available space:
  - Try H+1, H+2, H+3, …
  - Wraparound at the end of the array

- Problem: Clustering

- Animation:
  - http://www.cs.auckland.ac.nz/software/AlgAnim/hash_tables.html

# Linear Probing Efficiency

- Expected number of probes =
  - $\frac{1}{1-\lambda}$ ignoring clustering:
  - $\frac{1}{2}\left(1 + \frac{1}{(1-\lambda)^2}\right)$ taking clustering into account
  - Recall λ is the load Factor
- Can we do better?

# Quadratic Probing

- Linear probing:
    - Collision at H? Try H, H+1, H+2, H+3,...

- Quadratic probing:
    - Collision at H? Try H, $H+1^2$. $H+2^2$, $H+3^2$, ...
    - Eliminates primary clustering, but can cause "secondary clustering"

# Quadratic Probing Tricks (1/2)

- Choose a prime number p for the array size
- Then if $\lambda \le 0.5$:
  - Guaranteed insertion
    - If there is a “hole”, we’ll find it
  - No cell is probed twice
- See proof of Theorem 20.4:
  - Suppose that we repeat a probe before trying more than half the slots in the table
  - See that this leads to a contradiction
    - Contradicts fact that the table size is prime

# Quadratic Probing Tricks (2/2)

- Use an algebraic trick to calculate next index
  - Replaces mod and general multiplication
  - Difference between successive probes yields:
    - Probe i location, $H_i = (H_{i-1} + 2i - 1) \% M$
  - Just use bit shift to “multiply” i by 2
  - Don’t need mod, since i is at most M/2, so

    -
      ```
      probeLoc= probeLoc+ (i << 1) - 1;
      if (probeLoc >= M)
              probeLoc -= M;
      ```

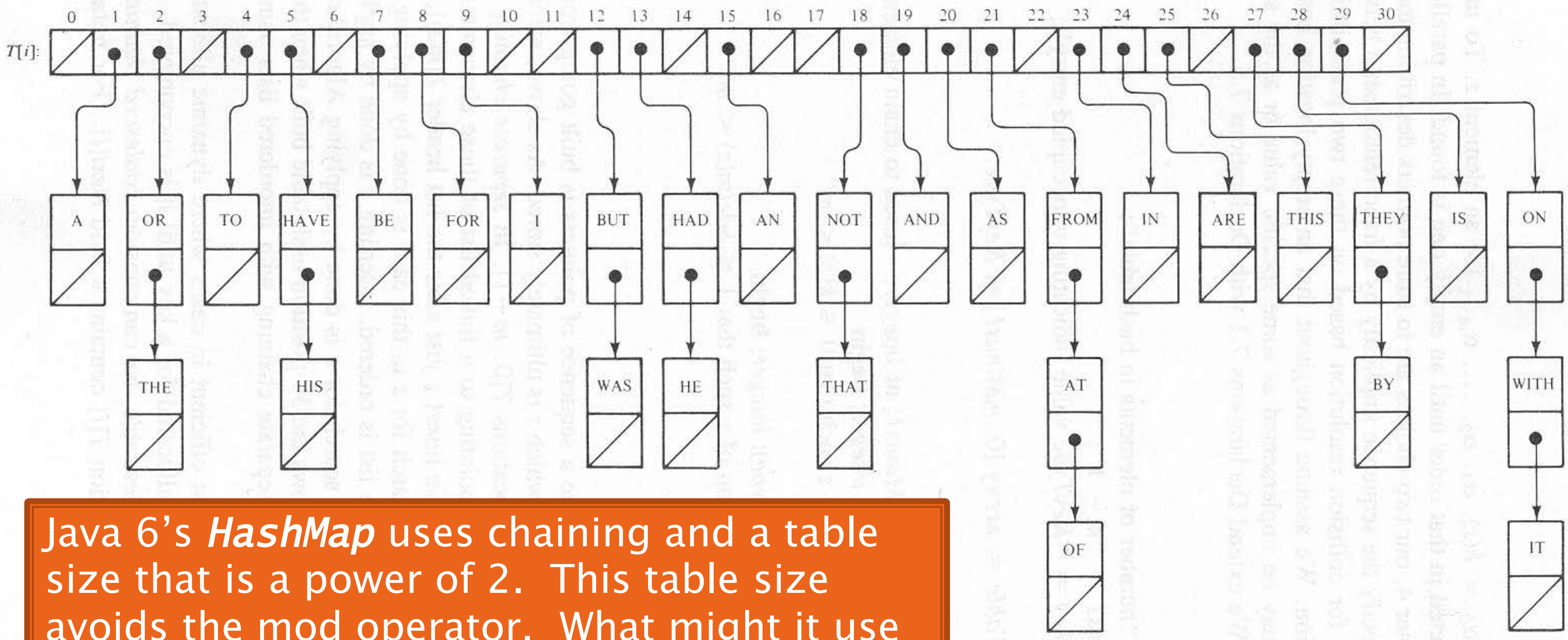
# Quadratic probing analysis

- No one has been able to analyze it!
- Experimental data shows that it works well
  - Provided that the array size is prime, and is the table is less than half full

# Another Approach: Separate Chaining

- Use an array of **linked lists**
- How would that help resolve collisions?

# Hashing with Chaining



Java 6’s ***HashMap*** uses chaining and a table size that is a power of 2. This table size avoids the mod operator. What might it use instead to make hashCodes() point to table locations?
(http://www.javaspecialists.eu/archive/Issue054.html)

# Hash Table Exercise

~40 minutes

On a handout and in your repository

Do it with your "EditorTrees" team

There's a handout for everyone, but only one submission per team

Check out from SVN:
HashSetExercise (individ repos)