CS5112: Algorithms and Data Structures for Applications

Lecture 4.1: Applications of hashing

Ramin Zabih

Some figures from Wikipedia/Google image search

Administrivia

- Web site is: https://github.com/cornelltech/CS5112-F18
  - As usual, this is pretty much all you need to know
- HW 1 at Thursday 11:59PM
  - Also very high tech!

CORNELL TECH

Quiz 1 comments

- Overall people did pretty well
  - 6/6: 73 people
  - 5/6: 58 people
  - 4/5: 22 people
- There was a Dijkstra question that required some thought
  - Might be on prelim/final in some form?
- High tech solution seemed to work well
- Reminder: we will drop your lowest quiz

Homework comments

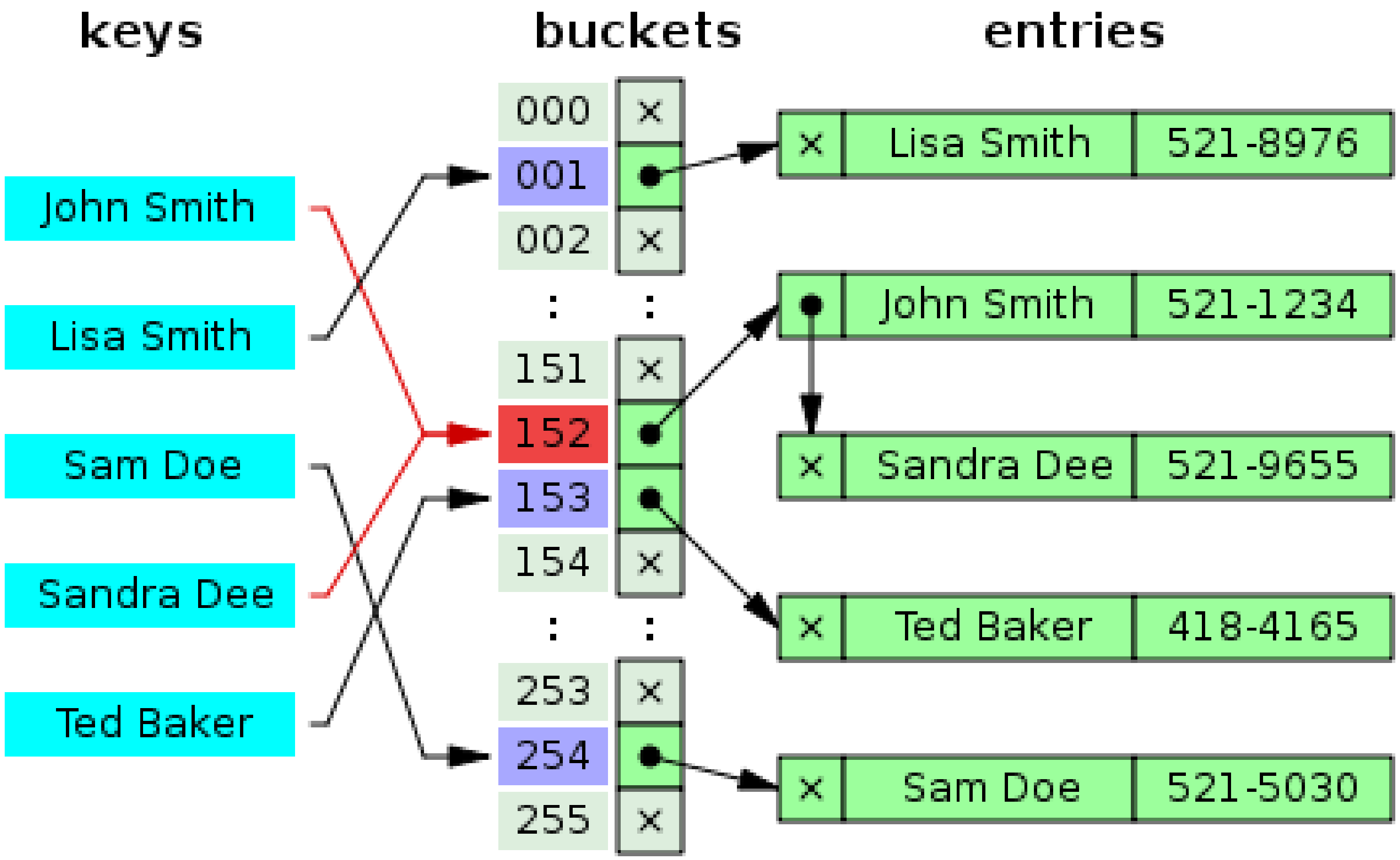
- Getting the course staff is slower than we hoped
- Slack should be your primary contact
- Each student has 1 slip day over the semester
- For a pair, you can use a single day (not 2 days)
  - You need to tell us which student to charge it to
  - If you don’t, we will ask you, and eventually charge it to both of you

Today

- Fun applications of hashing!
  - Lots of billion-dollar ideas
- Greg on cryptocurrency

Collisions are an issue

- See HW 1



CORNELL TECH

Can you determine if there are collisions?

- Given a hashing function h from bit strings to bit strings
  - No limits on the length of input or output
- Recall: cryptographic hash functions shouldn't have collisions
  - Two inputs with same output: $h(s) = h(s')$
- Can we tell this by inspecting h?

Different excuses for failure

“I can’t find an efficient algorithm, but neither can all these famous people.”

Garey & Johnson, *Computers and Intractability*

Uncomputable vs intractable

- Uncomputable: proven to be this is impossible
  - Determine if h has any collisions
  - Almost any question about a program
  - Some very subtle problems where the input size is unbounded
- Intractable: proven at least as hard as famous open problems
  - Technically "NP-hard"
  - Almost any question about a graph, such as coloring
    - A tractable graph problem is pure gold!
  - Many problems in cryptography

Uses in CS of hardness results

- Very important in many applications
- Use case 1: hard problems can help you
  – Want to show that if you could break a code you could also solve a famous open problem (e.g. factoring efficiently)
  – Mostly shows up in adversarial situations
- Use case 2: hard problems avoid wasting time
  – Showing that a problem is hard will keep people from working on it
  – Amusingly enough, sometimes it shows publications are wrong

Back to the fun part…

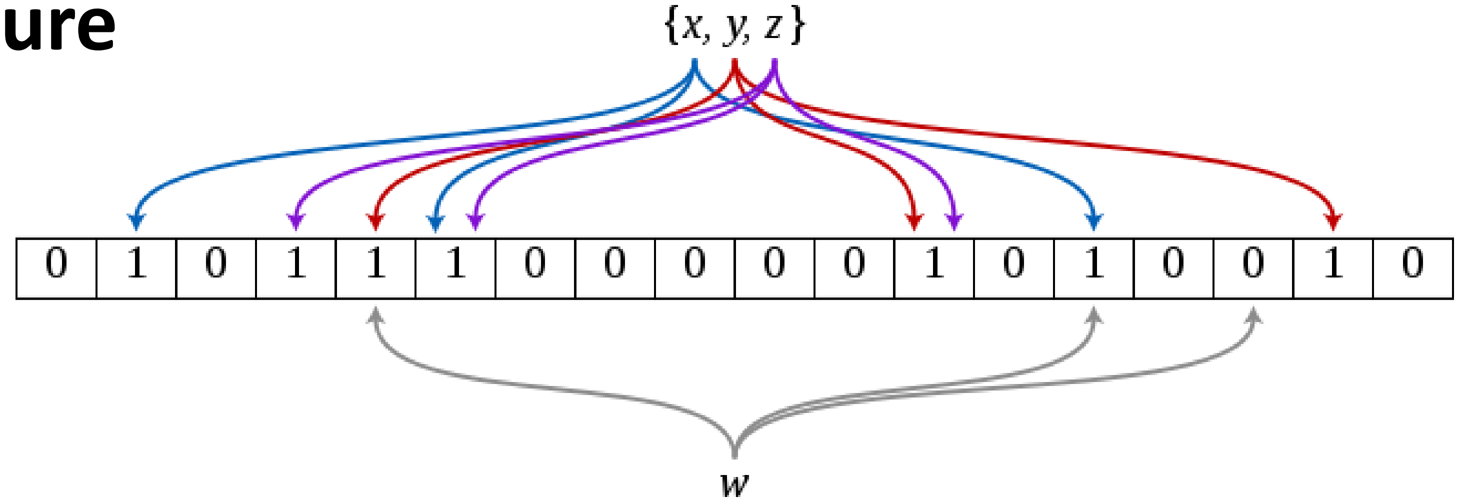
What cool stuff can we do with hashing?

Bloom filters

- Suppose you are processing items, most of them are cheap but a few of them are very expensive.
  - Can we quickly figure out if an item **is** expensive?
  - Could store the expensive items in an associative array
  - Or use a binary valued hash table?
    - Efficient way to find out if an item **might be** expensive
- We will query set membership but allow *false positives*
  - I.e. the answer to $s \in S$ is either 'possibly' or 'definitely not'
- Use a few hash functions h_i and bit array A
  - To insert s we set $A[h_i(s)] = 1 \; \forall \; i$

Bloom filter example

- Example has 3 hash functions and 18 bit array
- $\{x, y, z\}$ are in the set, w is not
- Bits are (sort of) **signature**



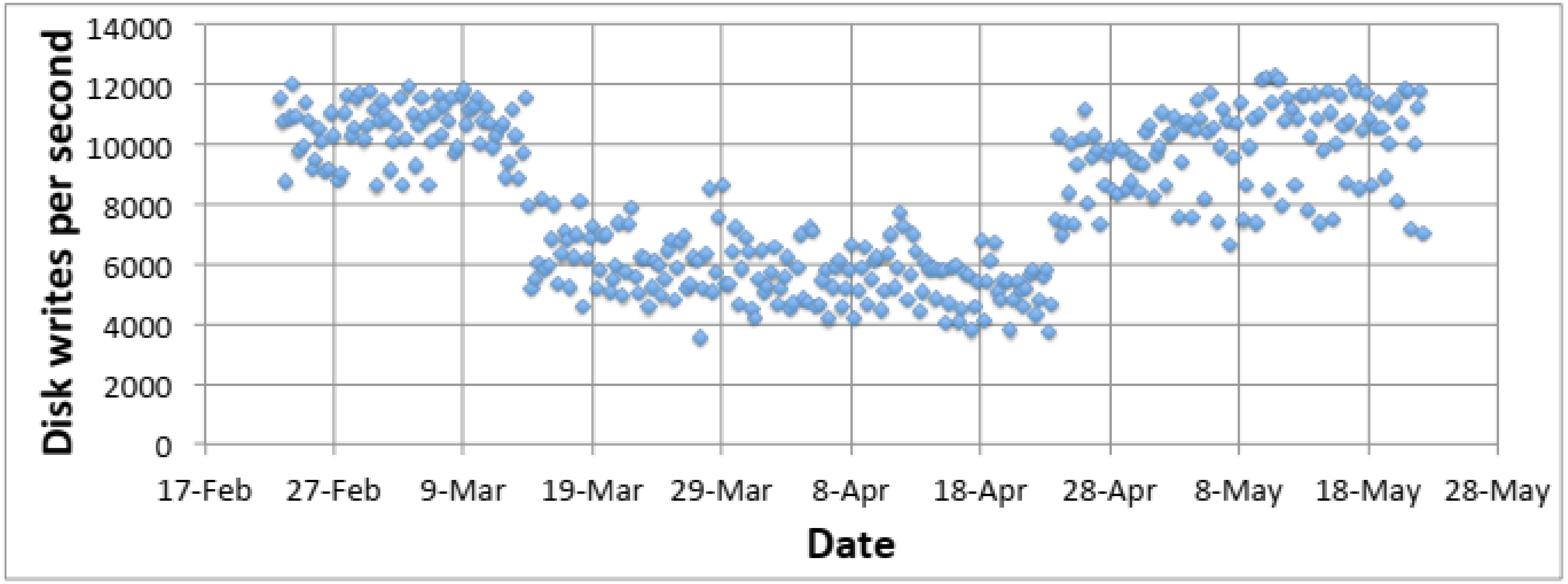
- Figure by David Eppstein, https://commons.wikimedia.org/w/index.php?curid=2609777

Application: web caching

- CDN's, like Akamai, make the web work (~70% of traffic)
- About 75% of URL's are 'one hit wonders'
  - Never looked at again by anyone
  - Let's not do the work to put these in the disk cache!
    - Cache on second hit
- Use a Bloom filter to record URL's that have been accessed
- A one hit wonder will not be in the Bloom filter
- See: Maggs, Bruce M.; Sitaraman, Ramesh K. (July 2015), "Algorithmic nuggets in content delivery" (PDF), *SIGCOMM Computer Communication Review*, New York, NY, USA,**45** (3): 52–66

CORNELL TECH

Bloom filters really work!



- Figures from: Maggs, Bruce M.; Sitaraman, Ramesh K. (July 2015), "Algorithmic nuggets in content delivery" (PDF), *SIGCOMM Computer Communication Review*, New York, NY, USA,**45** (3): 52–66

Cool facts about Bloom filters

- You don’t need to build different hash functions, you can use a single one and divide its output into fields (usually)
- Can calculate probability of false positives and keep it low
- Time to add an element to the filter, or check if an element is in the filter, is independent of the size of the element (!)
- You can estimate the size of the union of two sets from the bitwise OR of their Bloom filters

MinHash

- Suppose you want to figure out how similar two sets are
  - Jacard similarity measure is $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$
  - This is 0 when disjoint and 1 when identical
- Define $h_{min}(S)$ to be the element of S with the smallest value of the hash function h, i.e. $h_{min}(S) = \arg\min_{s \in S} h(s)$
  - This uses hashing to compute a set's "signature"
- Probability that $h_{min}(A) = h_{min}(B)$ is $J(A,B)$
- Do this with a bunch of different hash functions

MinHash applications

- Plagiarism detection in articles
- Collaborative filtering!
  - Amazon, NetFlix, etc.

Distributed hash tables (DHT)

- BitTorrent, etc.
- Given a file name and its data, store/retrieve it in a network
- Compute the hash of the file name
- This maps to a particular processor, which holds the file