Bloom Filters

A Simple Problem

- **Design Chrome Malicious URL Detector**
  - Lets say you work for Google, in the Chrome team, and you want to add a feature to the browser which notifies the user if the url he has entered is a malicious URL.

  - Given a database of about known 1 million malicious URLs, the size of any dictionary for matching will be around 50MB (**size of 1 million urls with 50 average string length**). 50MB is too heavy for a browser so this cannot be locally done!!

  - Anything locally should be something < 2MB memory. (25x far)

    Wait you cannot even compress the strings (Huffman Coding) to that size.

Universal hashing review

- Goals
  - Design $h: objects \rightarrow [0 - R]$ such that
  - If $O_1 \neq O_2 \ \ then \ h(O_1) \neq h(O_2)$ with high probability.
  - h is cheap to compute and requires almost no memory.

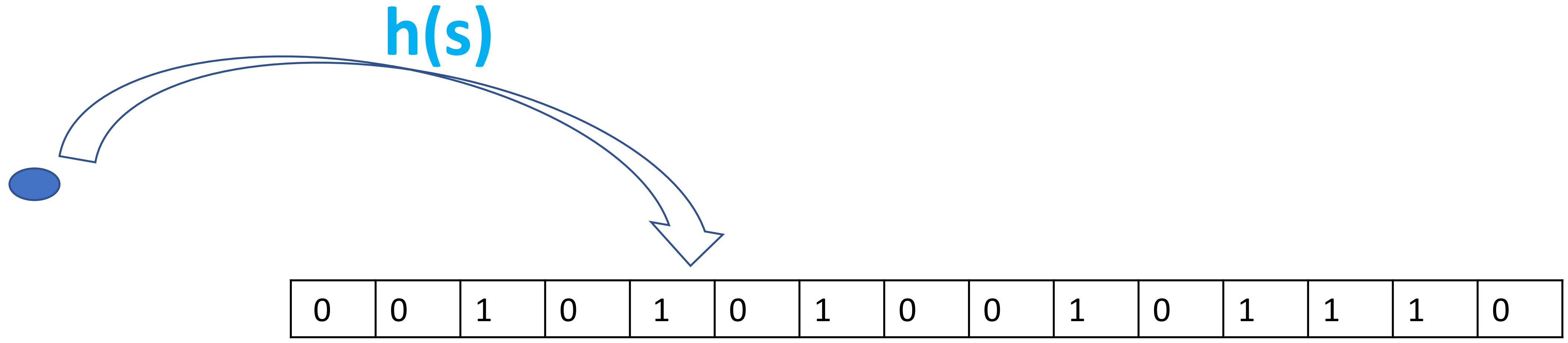
- Classical Example:
  - Randomly generate a and b (seed), choose a prime P >> R.
  - $h(x) = ax + b \ mod \ P \ mod \ R$ (faster if $R = 2^{32}$ why?)
  - For strings or other objects, use bitarrays, randomly shuffle bits based on a some seed then take mod.
    - **example:** See Murmurhash.
  - **Memory**: A seed per hash function!

More concrete problem

- Given a universal hashing
  - $h: strings \rightarrow \ [0 - 999]$
  - Given 100 queries (strings) (average length 50 characters). Space is 100x50x8 = 40,000 bits.
  - Same Task: Given a new query, answer whether we have seen it?

- Given a new string, answer no if it is not seen and if it is seen than answer no with small probability
  - What can we do in 1000 bits?

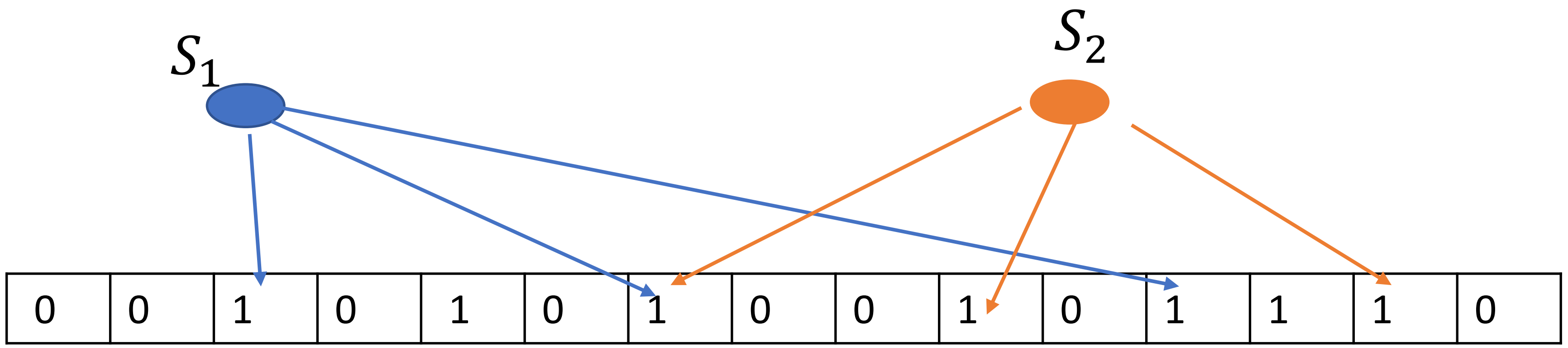
Bit-Maps and Universal Hashing



- Given a query q, if h(q) = 1 return seen, else not seen.
- What is chance for false positive? Given there are N Strings.
- $< 1 - \left(1 - \frac{1}{\mathrm{R}}\right)^{N}$ $\quad (R\ is\ size\ of\ bitmap)$

Can we do better? Bloom Filters

- Use K-independent hash functions instead of 1.
  - Just select seeds independently.
- K = 3 illustration



- Given a query q, if all $h_1(q), h_2(q), h_3(q)$ is set, return seen else no.
- False Positive Rate?

Illustration of Bloom Filters (K = 2)

- $h_1(x) = x \ mod \ 5, h_2(x) = (2x + 3) \ mod \ 5$
- Initialize Bloom Filters

0	0	0	0	0

- Insert 9 (4 and 1)

0	1	0	0	1

- Insert 11 (1 and 0)

1	1	0	0	1

Illustration of Bloom Filters

- $h_1(x) = x \; mod \; 5, h_2(x) = (2x + 3) mod \; 5$

1	1	0	0	1

- Query 15 : 0 and 3 → No
- Query 16: 1 and 0 → Yes (False Positive)

Properties

- If the query was inserted before, bloom filters always return true.
  - No false negatives.
- However, it can return true for an element which was not inserted
  - Chances of false positives.
- If false positives are rare then caches still work.
  - Why and How?

A bit of Analysis

- Pr a bit is not set given N strings are inserted?
  - $\left(1 - \frac{1}{R}\right)^{KN}$
  - Pr that a given bit is set 1 - $\left(1 - \frac{1}{R}\right)^{KN}$

- Pr that all the k-bits $h_1(q), h_2(q) \ldots h_K(q)$ is set without seeing q
  - $\left(1 - \left(1 - \frac{1}{R}\right)^{KN}\right)^{K} \approx \left(1 - e^{\frac{KN}{R}}\right)^{K}$
  - Minimized at $K = \ln(2) \times \frac{R}{N}$. If R is say 10N, then K = 6.9 or 7.
  - Optimum false positive is approx. $0.618^{\frac{R}{N}}$ which is < 0.008
  - Compare that with 0.1 with k =1.

- **So with N strings we need 10N bits space. Compare with hash table of $\boldsymbol{N \times 8 \times 50}$ (assuming 50 characters mean) ⇒ a reduction of 40x in memory.**

Generic set compression

- Given a set S of n objects with each object being heavy such as strings, etc.
- Bloom filter can compress S to less memory around 10 bit per object and still answer membership queries efficiently with rare chances of false positives.
- It can up updated dynamically on the fly.
- How about deletions?

Use of Bloom Filters from Wikipedia

- The servers of Akamai Technologies, a content delivery provider, use Bloom filters to prevent "one-hit-wonders" from being stored in its disk caches. One-hit-wonders are web objects requested by users just once, something that Akamai found applied to nearly three-quarters of their caching infrastructure. Using a Bloom filter to detect the second request for a web object and caching that object only on its second request prevents one-hit wonders from entering the disk cache, significantly reducing disk workload and increasing disk cache hit rates.
- Google Bigtable, Apache HBase and Apache Cassandra, and Postgresql use Bloom filters to reduce the disk lookups for non-existent rows or columns. Avoiding costly disk lookups considerably increases the performance of a database query operation.
- The Google Chrome web browser used to use a Bloom filter to identify malicious URLs. Any URL was first checked against a local Bloom filter, and only if the Bloom filter returned a positive result was a full check of the URL performed (and the user warned, if that too returned a positive result).
- The Squid Web Proxy Cache uses Bloom filters for cache digests.
- Bitcoin uses Bloom filters to speed up wallet synchronization.
- The Venti archival storage system uses Bloom filters to detect previously stored data.
- The SPIN model checker uses Bloom filters to track the reachable state space for large verification problems.
- The Cascading analytics framework uses Bloom filters to speed up asymmetric joins, where one of the joined data sets is significantly larger than the other (often called Bloom join in the database literature).
- The Exim mail transfer agent (MTA) uses Bloom filters in its rate-limit feature.
- Medium uses Bloom filters to avoid recommending articles a user has previously read.
- Ethereum uses Bloom filters for quickly finding logs on the Ethereum blockchain.

Popular Use: One-hit-wonders

- Content delivery networks deploy web caches around the world to cache and serve web content to users with greater performance and reliability.
- A key application of Bloom filters is their use in efficiently determining which web objects to store in these web caches.
- Nearly three-quarters of the URLs accessed from a typical web cache are "one-hit-wonders" that are accessed by users only once and never again.
- To prevent caching one-hit-wonders, a Bloom filter is used to keep track of all URLs that are accessed by users.
- A web object is cached only when it has been accessed at least once before, i.e., the object is cached on its second request.
- The use of a Bloom filter in this fashion significantly reduces the disk write workload, since one-hit-wonders are never written to the disk cache.

Deletion: Option 1

- Use two bloom filters
    - One to keep track of added elements
    - One to keep track of deleted elements

- What are the chances of false negatives?

- What are the chances of false positives?
    - Decreased!

A Popular Alternative

- **Counting filters**
  - Fan, Li; Cao, Pei; Almeida, Jussara; Broder, Andrei (2000), "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol“

- In a counting filter the array positions (buckets) are extended from being a single bit to being an n-bit counter.
- **Addition**: Increment
- **Deletion**: Decrement

- Chances of both false positive and false negatives.

Union of two bloom filters?

- Given bloom filter B_1 for set S_1 and another bloom filter B_2 of set S_2 with same hash functions.

- What is the bloom filter of $S_1 \cup S_2$?
  - Just OR the two bloom filters.

- Bloom filters can be organized in distributed data structures to perform fully decentralized computations of aggregate functions. Decentralized aggregation makes them ideal for several application by avoiding costly communication.

Shrink Size of Bloom Filters?

- Can we shrink the bloom filter to half of its size?

- How about doubling its size?

Weakness of Bloom Filters

- Needs full independent hash functions. (Hard)
- The space usage is 1.44x more than the information theoretically best possible.
- Dynamically growing bloom filters is hard. Best size depends on false positive rate and number of insertions.

A problem to ponder on

- You want to know if two people share friends on facebook.

- For privacy and memory reasons, Facebook only gave you a compressed bloom filter of their graph.
  - What should you ask Facebook to compress.

- How to do it and how good will it be?