2023-10-28 - System Design Algorithms

<u>1. Hash functions (refresher)</u>

- Convert any input to a fixed-length, deterministic, chaotic output.
- Small change in input ⇒ drastically different hash.
- Hash functions are one-way functions: impractical to reverse. Cryptographic hash functions make it infeasible to find any input matching a given hash.

Properties of a good hash function:

collision resistance ("uniformity")	Irreversibility	speed

Hash tables implementation idea

A hash function maps items to array positions and store something at that position (e.g. a linked list of associated values)

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Value																				

Why hash? because of O(1) look-ups. **Expected problems**: hash collisions.

2. Bloom filters

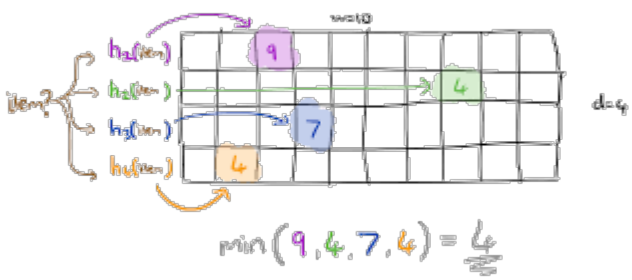
tells us that the element either definitely is not in the set or may be in the set.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Operation:	Features:
• Set membership ∈X is summarised in a bit array of length **n** • Choose **k** hash functions (mapping elements ∈X to integers {0…n}) ; The **k** hashes associate each **x∈X** with **k** locations in the bit array. • To add an element **x**: set the **k** bits associated with **x** to 1. • To check **x** is in the filter: check if the associated **k** bits are all 1.	• Fast query speed: O(**k**) • Low storage: O(**n**) • No false-negatives

<u>**3. Count-Min Sketch**</u>

Applications: Estimate the number of appearances of an item, Data range estimation



Operation:

- Original set has a large number of elements **|X|**
- Choose **K** hash functions that map to **D** different positions
- Use a **K×D** matrix **A** to estimate occurences of items in the original set
- Where $\mathbf{A}_{\mathbf{k,d}}$ is the number of times hash function **k** mapped to position **d**

4. Linear counter

Estimates the cardinality (number of unique elements) of a set X

- Initialise a bit array D with zeros.
- Hash each x∈X to a location in D, set those bits to 1.
- cardinality(X)≈$\sum_i$ D [i]

x	$h(x)$
Zoubin	4
Carl	2
Zoubin	4
Richard	3
Máté	4
Zoubin	4
Carl	2

Bit array

Position	Value	Items mapped
1	0	
2	1	Carl
3	1	Richard
4	1	Zoubin, Máté
5	0	

5. Loglog / Hyperloglog

5.1. **Summary -** https://engineering.fb.com/2018/12/13/data-infrastructure/hyperloglog/

Estimates the cardinality (number of unique elements) of a set X

- Map each x∈X to a hash.(The hash is a random bit string.)
- Out of |X| random bit strings, there are $|X|\ /\ 2^k$ with **k** leading zeros
- Find K, the largest number of leading zeros found in any of the |X| hashes.
- Use K to compute a very noisy estimate of |X|

5. 1 Probabilistic counter (powers of 2)

uses based on the number of zeros seen at the end for given data items. 00 01 11 here we say 2^2 = 4 as max unique count. **Disadvantage:** only gives estimates in powers of 2, nothing in between	Example of where it doesn't work 1001 0000 1100 1101 0001 max zeros = 4 = 2 ^ 4 = 16, whereas the number of entries is 5 only.

5. 2 Loglog: uses memory of log (log n)

if we divide previous into buckets 'm' with 2 digits

10 - max zeros at end =0

00 - max zeros = 2

11 - max zeros = 2

10 - max zeros = 0

avg = 0+2+2+0 / 4 = 1

==> constant * m * 2 ^ avg = 0.79 * 4 * 2^1 = 6.32

5. 3 Hyperloglog

uses harmonic mean instead of arithmetic mean in loglog.

unique entries

$$= constant * m * \frac{m}{\sum_{i=1}^{m} \frac{1}{2^{bucket[i]}}}$$

$$= 0.79 * 4 * \frac{4}{\frac{1}{2^2} + \frac{1}{2^0} + \frac{1}{2^0} + \frac{1}{2^2}}$$

= 5.056