CSC4524: Streaming algorithms

HyperLogLog

Mini-project 2

- **What is the number of unique (domain name, page title) couples visited the 01-08-2016 ?**
  - Output: Input cardinality
  - Work due to 25-11- 2019 23:59

Solution

HyperLogLog

HyperLogLog

- **HyperLogLog** is an algorithm for the count-distinct problem, approximating the number of distinct elements in a multiset. Calculating the *exact* cardinality of a multiset requires an amount of memory proportional to the cardinality, which is impractical for very large data sets. HyperLogLog algorithm, uses significantly less memory than this, at the cost of obtaining only an approximation of the cardinality.

Intuition

- **Experience: We toss a coin multiple times**
- **Observation: We observed a series of 20 consecutive heads**
- **Question: How many times was the coin tossed before observing these 20 consecutive heads ?**
  - P(o | n tosses) = $\frac{1}{2}^{20}(n - 20 + 1)$
  - P(o | 2^{20}tosses) = $\frac{1}{2}^{20}(2^{20} - 20 + 1) \approx 1$

Intuition

- The basis of the HyperLogLog algorithm is the observation that the cardinality of a multiset of uniformly distributed random numbers can be estimated by calculating the maximum number of leading zeros in the binary representation of each number in the set. If the maximum number of leading zeros observed is n, an estimate for the number of distinct elements in the set is 2^n

Hypothesis

- Hypothesis: **Uniformly distributes random numbers multiset.**

- A hash function is applied to each element in the original multiset to obtain a multiset of uniformly distributed random numbers with the same cardinality as the original multiset.

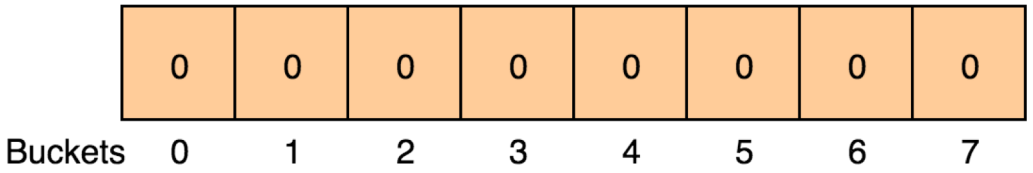
Large variance

- The simple estimate of cardinality obtained using the number of leading zeros over all items has the disadvantage of **large variance**. HyperLogLog minimizes, the variance is by splitting the multiset into numerous subsets, calculating the maximum number of leading zeros in the numbers in each of these subsets, and using a **harmonic mean** to combine these estimates for each subset into an estimate of the cardinality of the whole set.

HyperLogLog

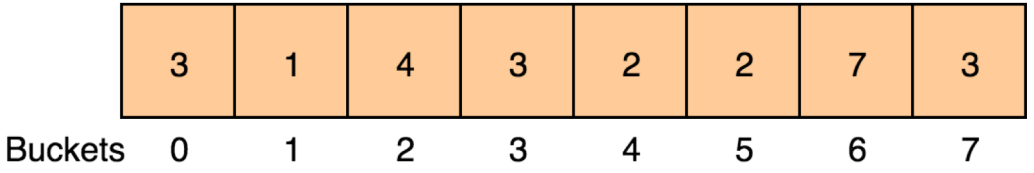
- Structure: *m* buckets initialized to 0.
- Stream cardinality: *n*

Example: *m* = 8

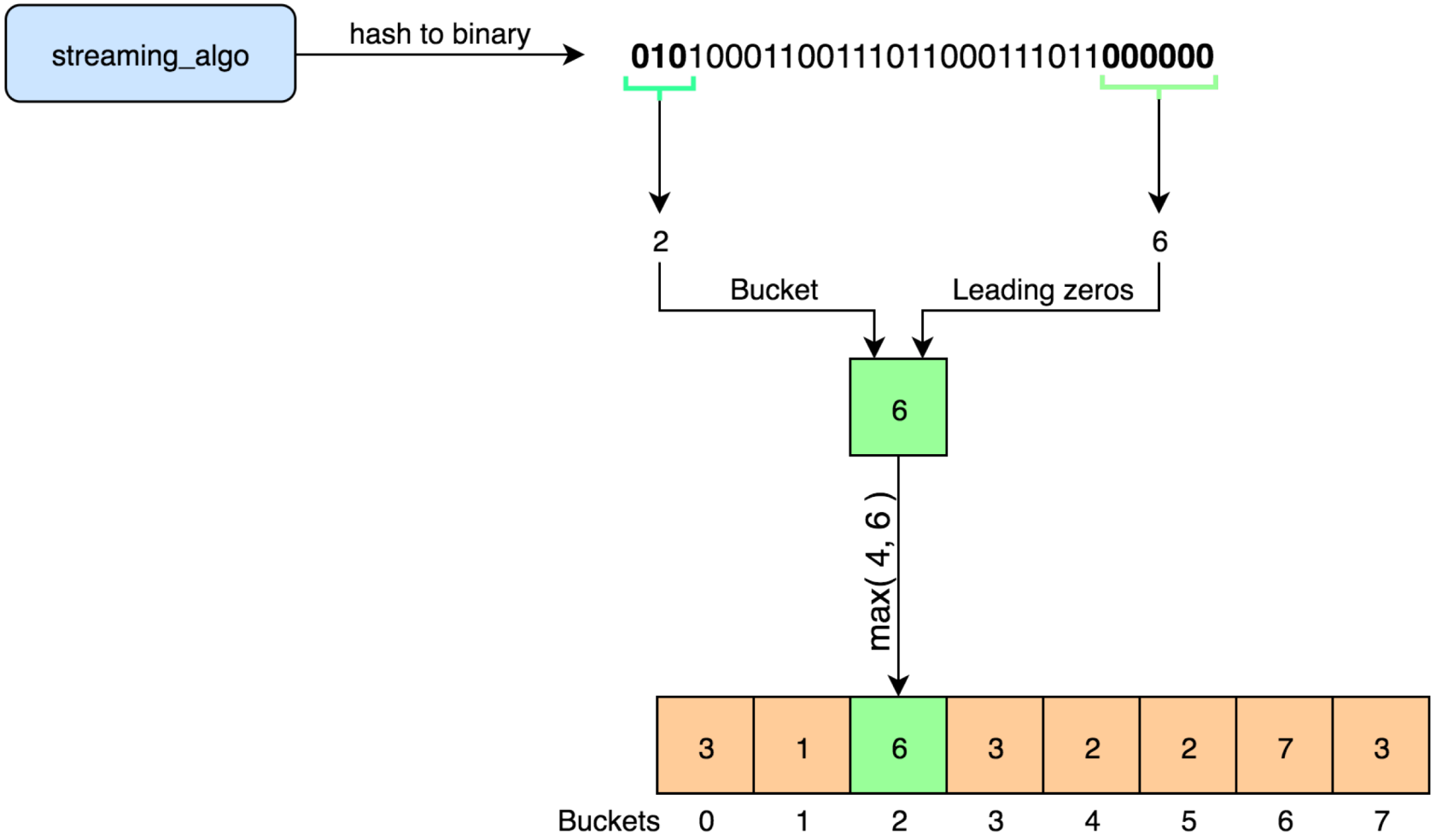


Add item

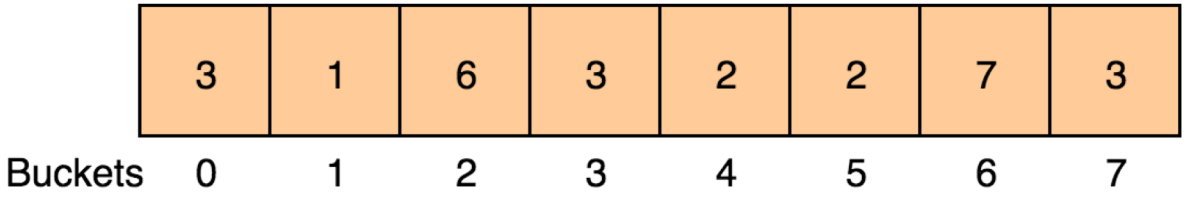
- $m = 8$
- We added items and would like to add ”streaming_algo”



Add item


streaming_algo
hash to binary
01010001100111011000111011000000
2
6
Bucket
Leading zeros
6
max(4, 6)
3
1
6
3
2
2
7
3
Buckets
0
1
2
3
4
5
6
7

Estimate cardinality



- Intuition: each bucket has $\frac{n}{m}$ cardinality

$$m\,(\sum_{1}^{m} 2^{-bucket_i})^{-1} \approx \frac{n}{m}$$

Harmonic mean

$$n \approx m^2\,(\sum_{1}^{m} 2^{-bucket_i})^{-1}$$

Estimate cardinality

$$n \approx \alpha_m \, m^2 \, (\sum_{1}^{m} 2^{-bucket_i})^{-1}$$

$$\alpha_m = \left(m \int_0^\infty \left(\log_2 \left(\frac{2+u}{1+u} \right) \right)^m du \right)^{-1}$$

- The constant α_m is introduced to correct a systematic multiplicative bias present due to hash collisions.

Complexity

- **Time**

$$O(\#elements)$$

- **Memory**

$$O(m)$$

Applications

- Google's BigQuery uses HyperLogLog to efficiently count unique elements in a table(APPROX_COUNT_DISTINCT()).
- Reddit uses HyperLogLog to retrieve the number of distinct views a post has gathered.

Mini-project 3

- **Compute the number of views for each (domain name, page title) couple**
  - Input: domain name + page title
  - Output: #views
  - Work due to 01-12- 2019 23:59