SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE NAME : 19CS508 - BIG DATA ANALYTICS

III YEAR / V SEMESTER

Unit 4- STREAM MEMORY

Topic 3 : Filtering Streams

Filtering Streams

- The randomized algorithms and data structures we have seen so far always produce the correct answer but have a small probability of being slow. In this lecture, we will consider randomized algorithms that are always fast, but return the wrong answer with some small probability.
- More generally, we are interested in tradeoffs between the (likely) efficiency of the algorithm and the (likely) quality of its output.
- Specifically, we introduce an error rate δ and analyze the running time required to guarantee
- the output is correct with probability 1 − δ. For "high probability" correctness, we need δ < 1/nc for some constant c. In practice, it may be sufficient (or even necessary) to set δ to a small
- constant; for example, setting δ = 1/1000 means the algorithm produces correct results at least 99.9% of the time..

Bloom filters

- Bloom filters are a natural variant of hashing proposed by Burton Bloom in 1970 as a mechanism for supporting membership queries in sets.
- In strict accordance with Stigler's Law of Autonomy, Bloom filters are identical to Zatocoding, a coding system for library cards developed by Calvin Mooers in 1947.
- (Mooers was the person who coined the phrase "information retrieval".) A probabilistic analysis of Zatocoding appears in the personal notes of cybernetics pioneer W. Ross Ashby from 1960. (or)

What is Bloom Filter?

A Bloom filter is a **space-efficient probabilistic** data structure that is used to test whether an element is a member of a set.

For example, checking availability of username is set membership problem, where the set is the list of all registered username. The price we pay for efficiency is that it is probabilistic in nature that means, there might be some False Positive results. **False positive means**, it might tell that given username is already taken but actually it's not.

A Bloom filter consists of:

1. An array of n bits, initially all 0's.

2. A collection of hash functions h1, h2, . . . , hk. Each hash function maps —key|| values to n buckets, corresponding to the n bits of the bit-array.

3. A set S of m key values.

The purpose of the Bloom filter is to allow through all stream elements whose keys are in S, while rejecting most of the stream elements whose keys are not in S.

- To initialize the bit array, begin with all bits 0. Take each key value in S and hash it using each of the k hash functions. Set to 1 each bit that is hi(K) for some hash function hi and some key value K in S.
- To test a key K that arrives in the stream, check that all of h1(K), h2(K), . . . , hk(K) are 1's in the bit-array. If all are 1's, then let the stream element through. If one or more of these bits are 0, then K could not be in S, so reject the stream element..

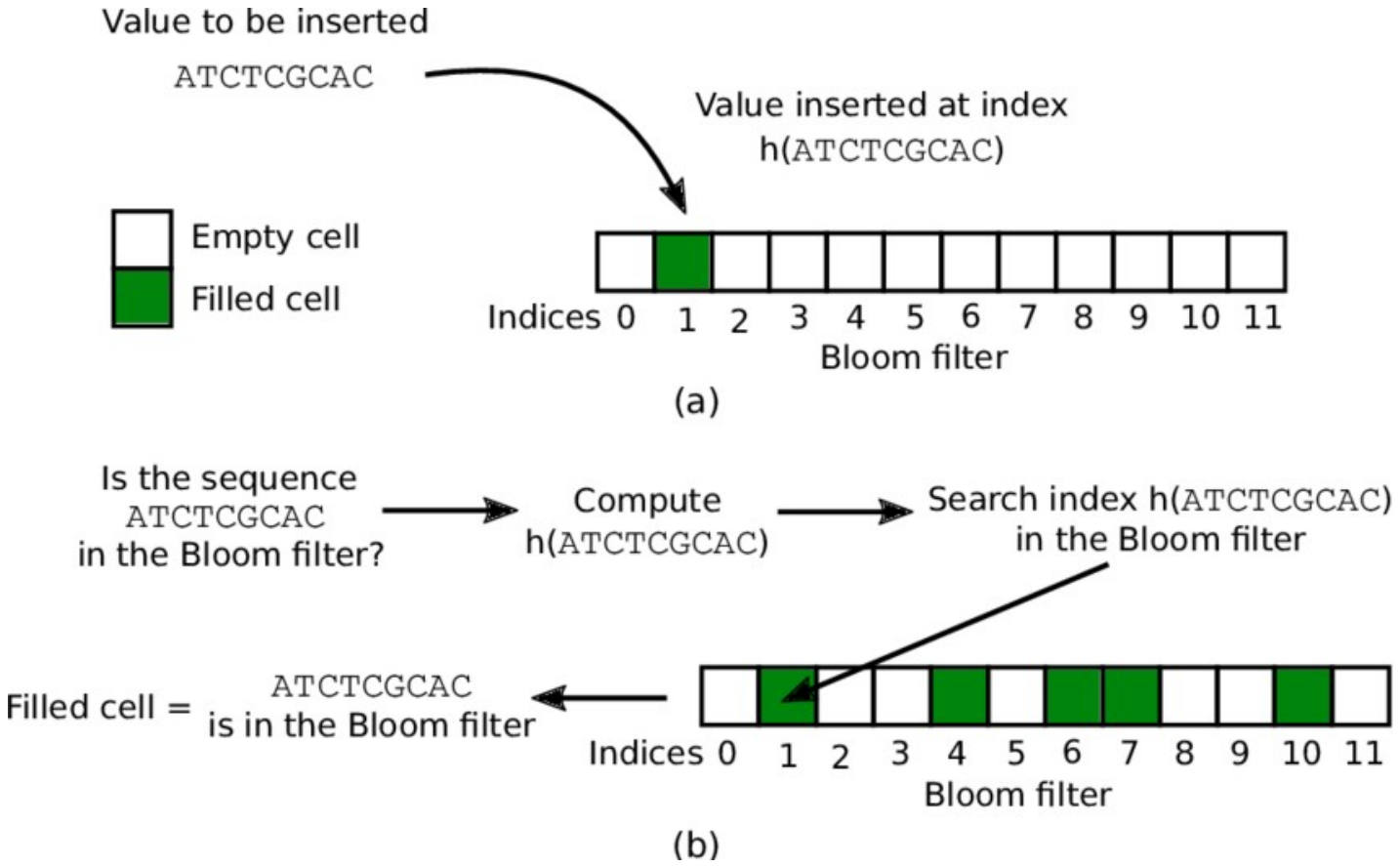
Suppose you are creating an account on Geekbook, you want to enter a cool username, you entered it and got a message, “Username is already taken”. You added your birth date along username, still no luck. Now you have added your university roll number also, still got “Username is already taken”. It’s really frustrating, isn’t it?

But have you ever thought about how quickly Geekbook checks availability of username by searching millions of username registered with it. There are many ways to do this job –

•Linear search : Bad idea!

•Binary Search : Store all username alphabetically and compare entered username with middle one in list, If it matched, then username is taken otherwise figure out, whether entered username will come before or after middle one and if it will come after, neglect all the usernames before middle one(inclusive). Now search after middle one and repeat this process until you got a match or search end with no match. This technique is better and promising but still it requires multiple steps.

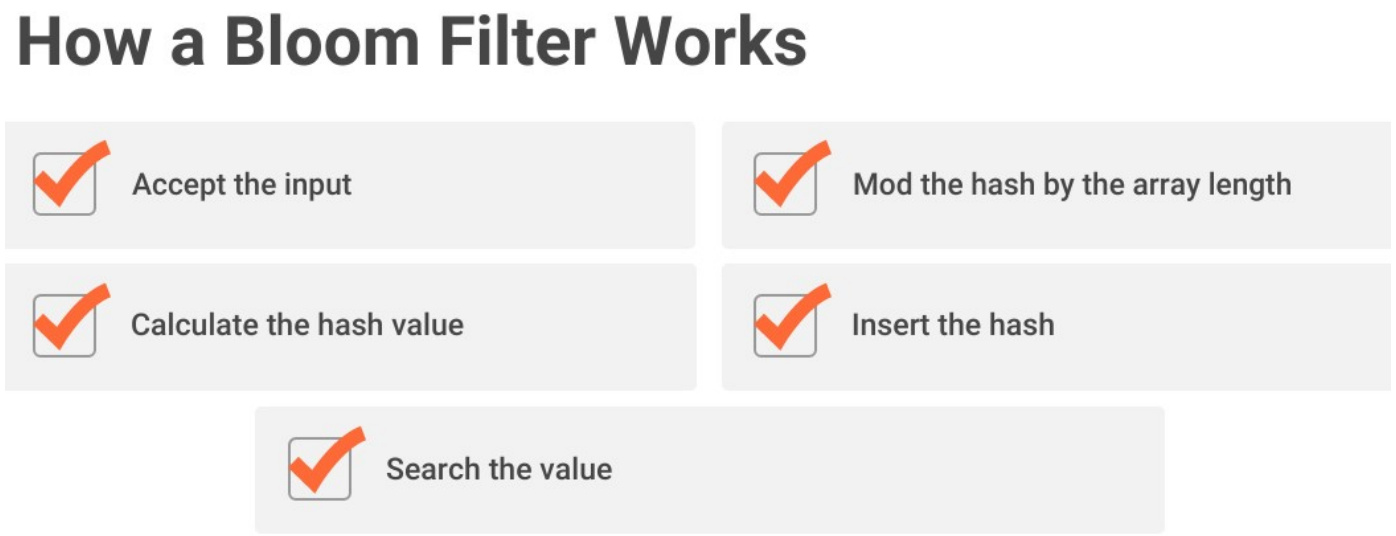
But, there must be something better!!



Bloom Filter

Bloom filters can be of various types:

- Compressed Bloom filters
- Spectral Bloom filters
- Space code Bloom filters
- Decaying Bloom filters



Applications of Bloom Filters

1. Checking for email ID availability
2. Ensuring the security level of a suspicious URL
3. Recommending new content
4. Saving storage space on social media platforms
5. Detecting weak passwords
6. Tracing IP addresses
7. Supporting P2P networks

Activity

Advantages

- During entry and searches, the time complexity of the Bloom filter data framework is 0(k), where k is the maximum number of hash functions implemented. In computing, the complexity of time is the computational challenge defining the time required to execute an algorithm on a computer.
- Bloom filters have a space complexity of 0(m), wherein m is the total array capacity. The space complexity of a formula or computer program is the memory required to address a specific case of a computational challenge. Space complexity is generally determined contingent on the input's characteristics.
- Unlike hash tables, which use a single hash function, Bloom filters employ numerous hash functions to avoid hash collisions. However, this is not a failsafe.

- **Disadvantages**
- There are incorrect outcomes. This indicates that the method cannot always accurately determine whether an element exists in the collection. It never produces a false negative, however.
- Only the probability can be retrieved from the array, not the original data.
- The greater the number of hash functions, the slower the Bloom filter. However, if you have a small number, you may experience excessive false positives.

Assessment 1

1. List out the advantages of Filtering Streams
   a)_____
   b)_____
   c)_____
   d)_____

2. Identify the disadvantages of Filtering Streams
   a)_____
   b)_____
   c)_____
   d)_____

REFERENCES

1. Anand Rajaraman and Jeffrey David Ullman, "Mining of Massive Datasets", Cambridge University Press, 2012.
2. David Loshin, "Big Data Analytics: From Strategic Planning to Enterprise Integration with Tools, Techniques, NoSQL, and Graph", Morgan Kaufmann/Elsevier Publishers, 2013
3. Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data", Wiley publishers, 2015.

THANK YOU