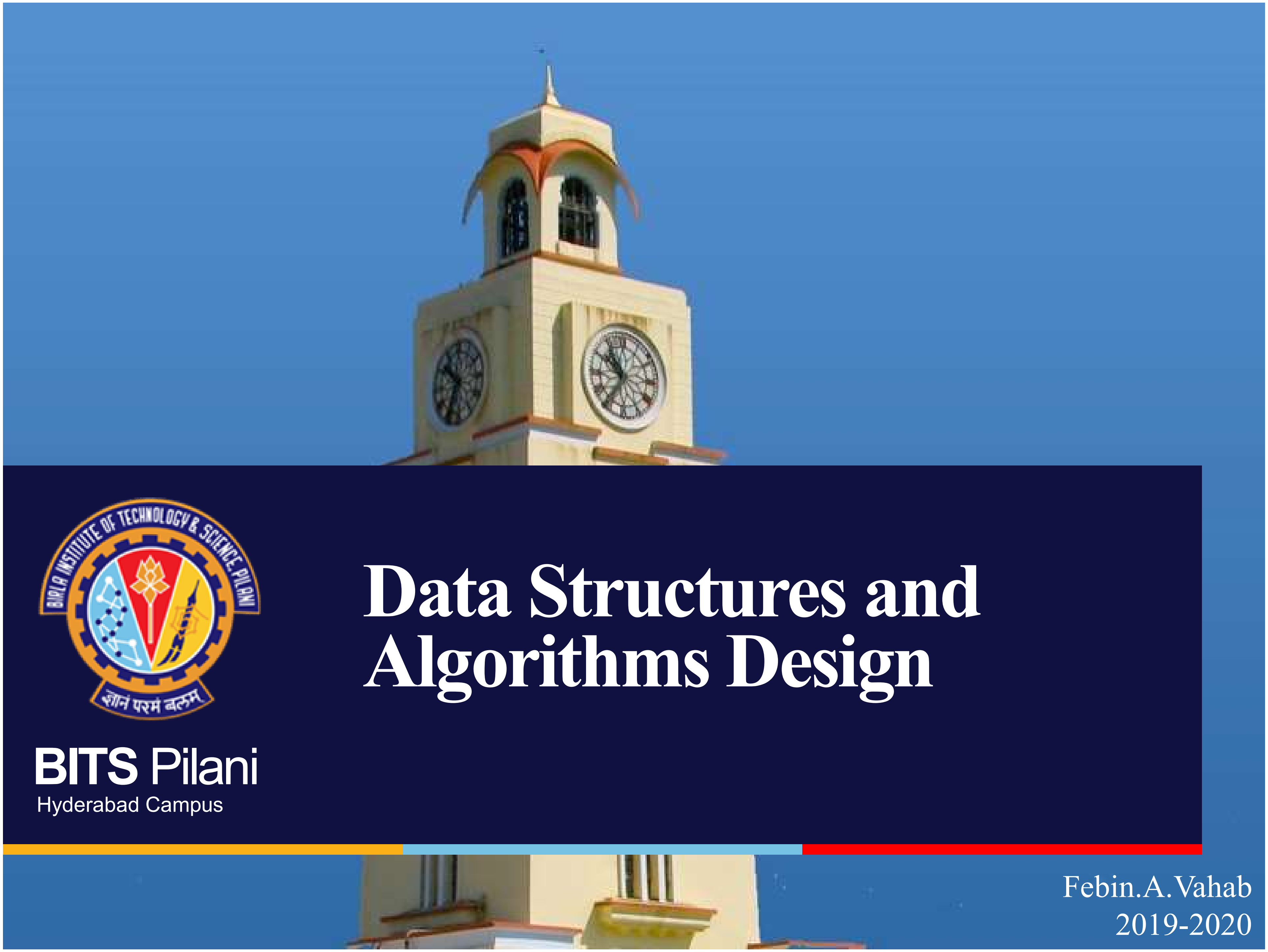


BITS Pilani
Hyderabad Campus

Data Structures and Algorithms Design

Febin.A.Vahab
2019-2020

SESSION 8 -PLAN

$\{m^2\}$ $\{m\}$

$\underbrace{h(k)}_{1} + i(\underbrace{h'(k)}_{2})$

$2 \quad 1$

Sessions(#)	List of Topic Title	Text/Ref Book/external resource
8	Ordered Dictionary: ADT, Applications Bloom Filters - Motivation, Properties and Operations Types of Bloom Filters, Applications	T1: 3.1 R3: 10

Bloom filter

- Suppose you are creating an account on Facebook, 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 how quickly Facebook check availability of username by searching millions of username registered with it.

Bloom filter

- Burton Bloom introduced Bloom filters in the 1970s [Bloom 70]
- 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.

Bloom filter

- **False positive means**, it might tell that given username is already taken but actually it's not.
- It only supports two operations: insertion and membership querying
- Strings are stored using multiple hash functions
- It can be queried to check the presence of a string
- Membership queries result in rare false positives but never false negatives

Bloom filter

- For many applications, a low rate of false positives is tolerable.
- For instance, the job of a network traffic shaper is to throttle bulk transfers (e.g. BitTorrent) so that interactive sessions (games) see good response times.
- A traffic shaper might use a Bloom filter to determine whether a packet belonging to a particular session is bulk or interactive.
- If it misidentifies one in ten thousand bulk packets as interactive and fails to throttle it, nobody will notice

Bloom filter

- A empty bloom filter is a **bit array** of **m** bits, all set to zero, like this –

1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0

Bloom filter

- We need **k** number of **hash functions** to calculate the hashes for a given input.
- When we want to add an item in the filter, the bits at k indices h1(x), h2(x), … hk(x) are set, where indices are calculated using hash functions.

- Example – Suppose we want to enter "Movie" in the filter, we are using 3 hash functions and a bit array of length 10, all set to 0 initially. First we'll calculate the hashes as following

Bloom filter

k = 3 } m = 10

- h1("movie") % 10 = 1 ✓
- h2("movie") % 10 = 4 ✓
- h3("movie") % 10 = 7 ✓

- Now we will set the bits at indices 1, 4 and 7 to 1

1	2	3	4	5	6	7	8	9	10
1	0	0	1	0	0	1	0	0	0
↑			↑			↑			

Bloom filter

- Again we want to enter "drama", similarly we'll calculate hashes
- h1("drama") % 10 = 3 ✓
- h2("drama") % 10 = 5 ✓
- h3("drama") % 10 = 4 ✓

- Set the bits at indices 3, 5 and 4 to 1

1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	0	1	0	0	0

Bloom filter

- Now if we want to check "movie" is present in filter or not.
- We calculate respective hashes using h1, h2 and h3 and check if all these indices are set to 1 in the bit array.

→ 1 4 7

- If all the bits are set then we can say that "movie" is **probably present**.

false positive

- If any of the bit at these indices are 0 then "movie" is **definitely not present**.

False Positive in Bloom Filters

- The question is why we said **"probably present"**, why this uncertainty.
- Let's understand this with an example. Suppose we want to check whether "cat" is present or not.
- We'll calculate hashes using h1, h2 and h3
- h1("cat") % 10 = 1 ✓
- h2("cat") % 10 = 3 ✓
- h3("cat") % 10 = 7 ✓

false positive

✓ ✓
1, 4, 7 → mw
3, 4, 5 → d

Bloom filter

- If we check the bit array, bits at these indices are set to 1 but we know that "cat" was never added to the filter.
- Bit at index 1 and 7 was set when we added "movie" and bit 3 was set we added "drama".

1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	0	1	0	0	0

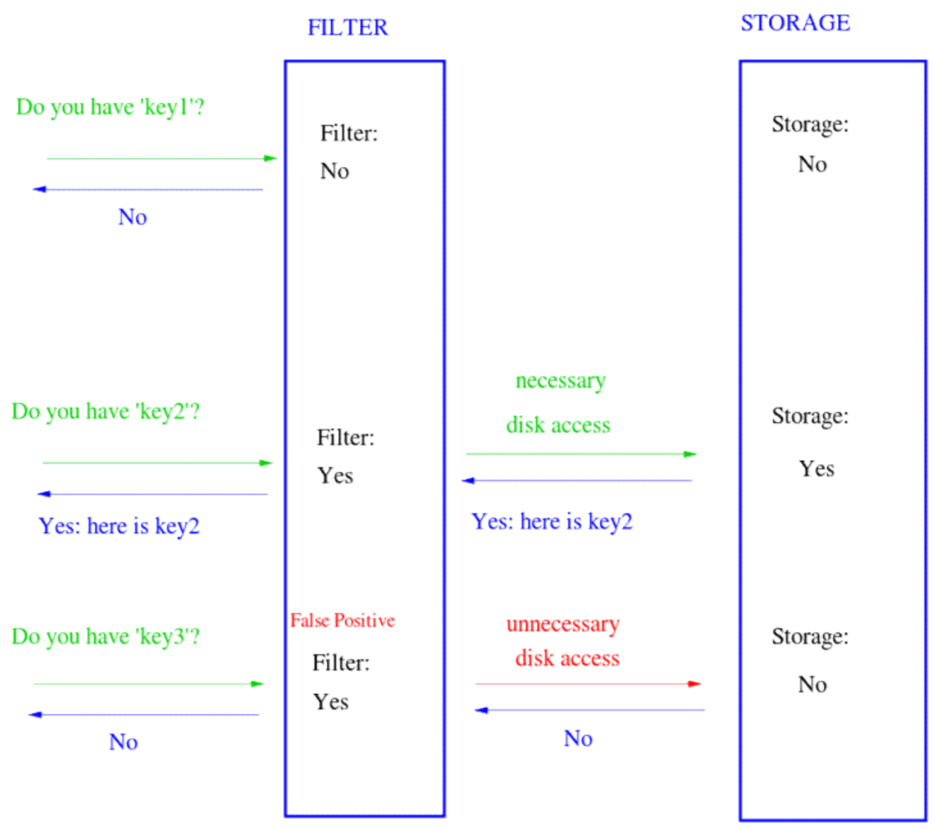
Bloom filter

- So, because bits at calculated indices are already set by some other item, bloom filter erroneously claim that “cat” is present and generating a false positive result.
- Depending on the application, it could be huge downside or relatively okay.

Bloom filter

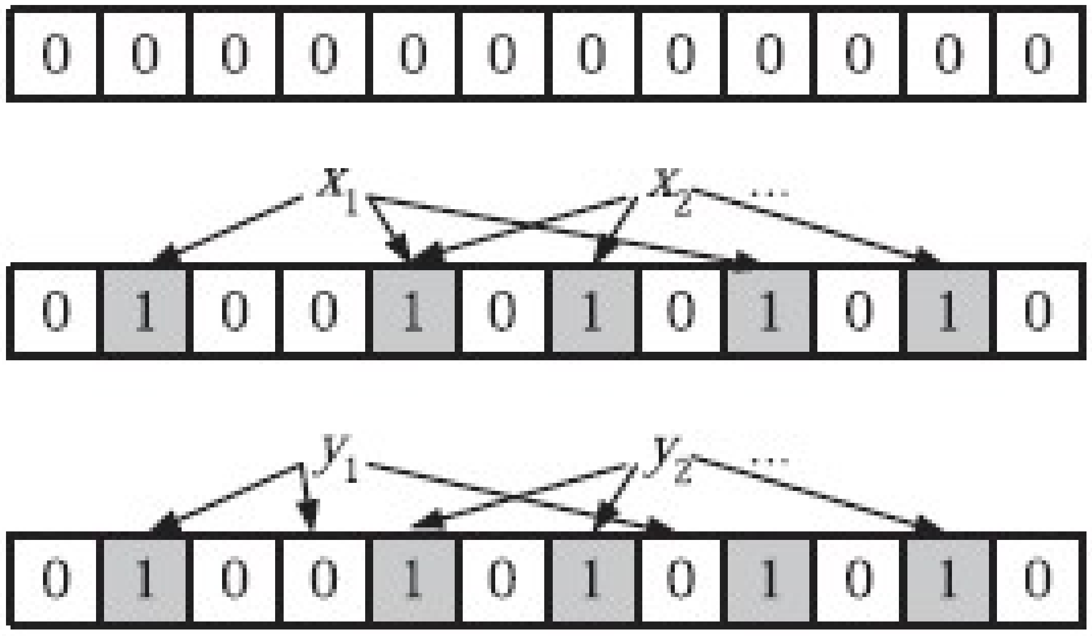
- We can control the probability of getting a false positive by controlling the size of the Bloom filter.
- More space means fewer false positives.
- If we want decrease probability of false positive result, we have to use more number of hash functions and larger bit array.
- This would add latency in addition of item and checking membership.

Bloom filter-Example 1



https://en.wikipedia.org/wiki/Bloom_filter

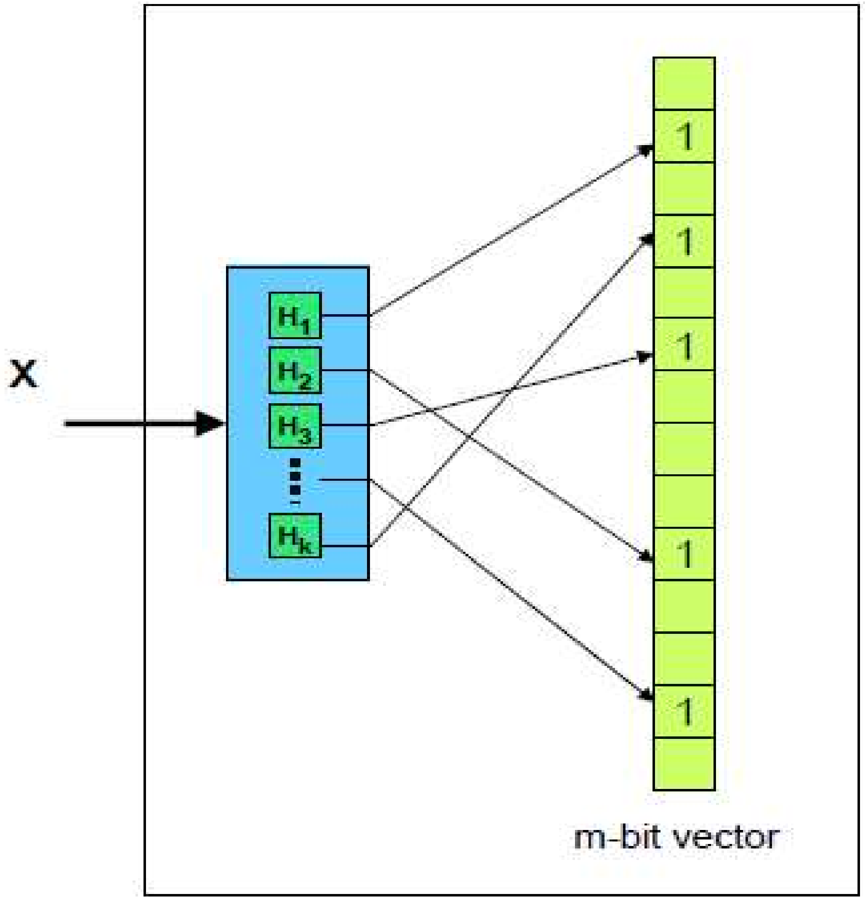
Bloom filter-Example 2



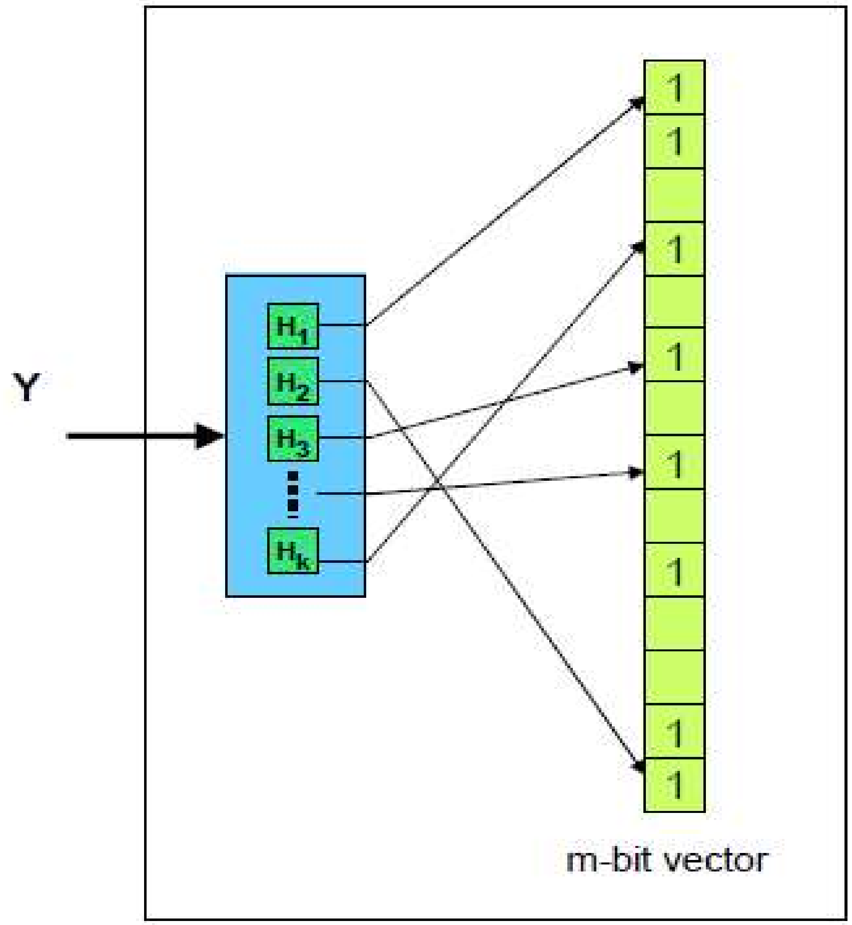
Bloom filter-Example

- An example of a Bloom filter.
- The filter begins as an array of all 0s.
- Each item in the set x_i is hashed k times, with each hash yielding a bit location ,these bits are set to 1.
- To check if an element y is in the set, hash it k times and check the corresponding bits.
- The element y1 cannot be in the set, since a 0 is found at one of the bits.
- The element y2 is either in the set or the filter has yielded a false positive.

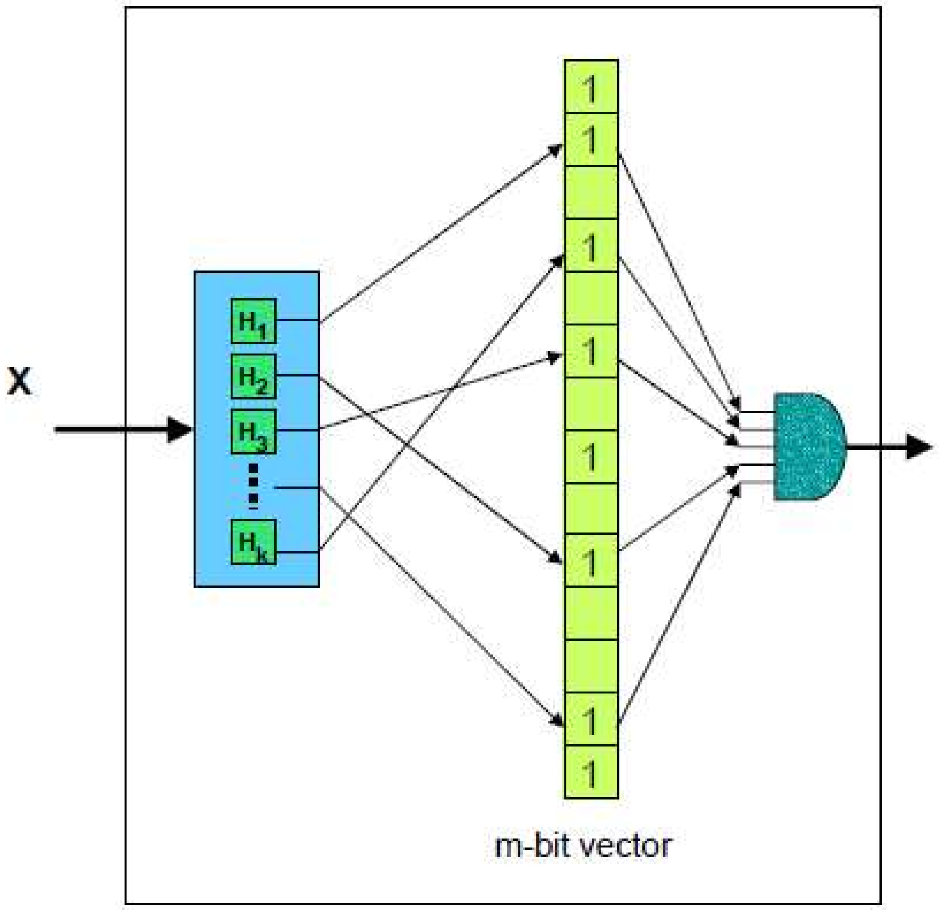
Programming a Bloom filter



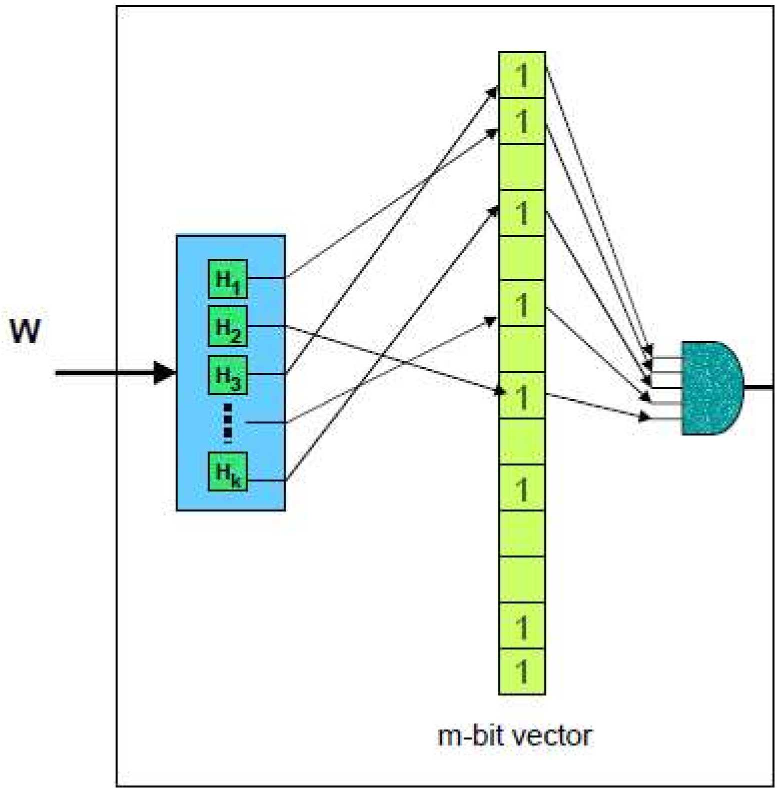
Programming a Bloom filter



Querying



Bloom filter-False Positive



Bloom filter

- Given a Bloom filter with *m* bits and *k* hashing functions, both insertion and membership testing are *O(k)*. ✓
- That is, each time you want to add an element to the set or check set membership, you just need to run the element through the the *k* hash functions and add it to the set or check those bits.

3

↑ ↑ ↑

Bloom filter

- The space advantages are more difficult to sum up; again it depends on the error rate you're willing to tolerate.
- It also depends on the potential range of the elements to be inserted; if it is very limited, a deterministic bit vector can do better.
- If you can't even ballpark estimate the number of elements to be inserted, you may be better off with a hash table or a scalable Bloom filter.
- http://pages.cs.wisc.edu/~cao/papers/summary-cache/node8.html

Bloom filter

- Applications include :
  - –Dictionaries-Hyphenation
  - – Databases
  - – Summary Caches
  - – P2P networks
  - – Packet Routing
  - – Multicast

Bloom filter-Deletion

- Deleting elements from filter is **not possible** because, if we delete a single element by clearing bits at indices generated by k hash functions, it might cause deletion of few other elements.

Bloom filter-TRY OUT!!

- Use N = 11 bits for our filter.
- Stream elements = integers.
- Use two hash functions:
  - **$h_1(x)$** =
    - Take odd-numbered bits from the right in the binary representation of x.
    - Treat it as an integer i.
    - Result is i modulo 11.
  - **$h_2(x)$** = same, but take even-numbered bits.

Bloom filter-TRY OUT!!

Stream element	h1	h2	Filter Contents
			00000000000
25 = 11001	5	2	00100100000
159 = 10011111	0	7	10100101000
585 = 1001001001	7	9	10100101010

Lookup element y = 118

Bloom filter-TRY OUT!!

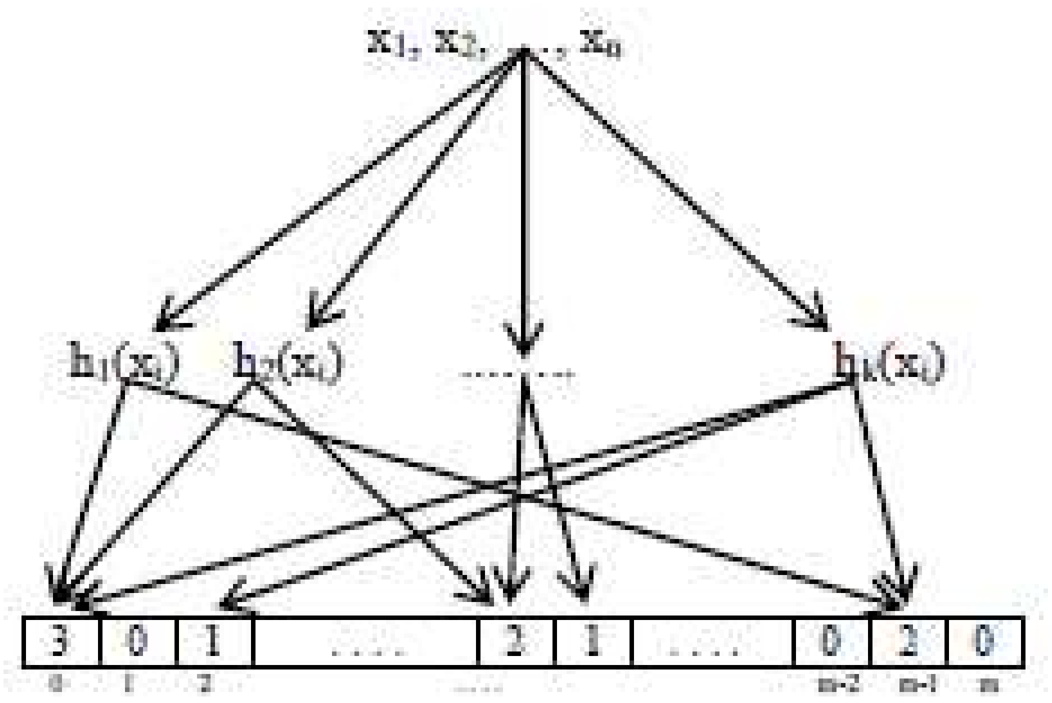
https://hur.st/bloomfilter/

Types of Bloom Filter

- **Counting Bloom Filters**
- In a counting Bloom filter, each entry in the Bloom filter is not a single bit but rather a small counter.
- When an item is inserted, the corresponding counters are incremented; when an item is deleted, the corresponding counters are decremented.

Types of Bloom Filter

•**Counting Bloom Filters**



Types of Bloom Filter

- **Compressed Bloom Filters**
- Suppose that a server is sending a Bloom filter to several other servers over a network.
- Can we gain anything by compressing the resulting Bloom filter?
- Using a larger but sparser Bloom Filter can yield the same false positive rate with a smaller number of transmitted bits.
- The resulting bloom filter is called a Compressed Bloom Filter .

Types of Bloom Filter

- **Compressed Bloom Filters**
- By using Compressed Bloom Filters networking protocols reduce the number of bits broadcast and the amount of computation per look up.
- Costs involved are larger computation time for compression and decompression.
- ***Reference paper 3***

Types of Bloom Filter

- **Variable Increment Bloom Filter**
- The Variable Increment Counting Bloom Filter is a generalization of the Counting Bloom Filter that uses variable increments to update each entry.
- In this structure, a set of possible variable increments are defined.
- For each counter update by an element we hash the element into the variable increment set and use it to increment the counter
- Similarly, to delete an element we decrement by its hashed value in the variable increment set.

Types of Bloom Filter

- **Scalable Bloom Filter**
- A Scalable Bloom Filters consist of two or more Standard Bloom Filters, allowing arbitrary growth of the set being represented. When one Bloom Filter gets filled due to the limit on the fill ratio, a new filter is added.

Types of Bloom Filter

- **Bloomier Filters**
- Bloomier Filters associate a value with each element that had been inserted thereby implementing an associative array.

Types of Bloom Filter

- **Stable Bloom Filter**
- This variant of Bloom Filter is particularly useful in data streaming applications. In these applications when more and more elements arrive, the number of 1's in the array of bloom filter will increase significantly, finally reaching the limit where every distinct element is reported as duplicate indicating that bloom filter can no longer be used.

Bloom filter-More Applications

- To warn the user for weak passwords.
- Instead of making a query to an SQL database to check if a user with a certain email exists, you could first use a bloom filter for an inexpensive lookup check. If the email doesn’t exist, great! If it does exist, you might have to make an extra query to the database.

Bloom filter-More Applications

- Spell-checker by using bloom filter to track the dictionary words.
- Reduce expensive disk (or network) lookups for non-existent keys
- You can keep a bloom filter based on the IP address of the visitors to your website to check if a user to your website is a ‘returning user’ or a ‘new user’. Some false positive value for ‘returning user’ won’t hurt you, right?

Bloom filter-More Applications

- Medium uses bloom filters for recommending post to users by filtering post which have been seen by user.
- Quora implemented a shared bloom filter in the feed backend to filter out stories that people have seen before.
- The Google Chrome web browser used to use a Bloom filter to identify malicious URLs
- Google BigTable, Apache HBase and Apache Cassandra use Bloom filters to reduce the disk lookups for non-existent rows or columns

References

1. https://www.eecs.harvard.edu/~michaelm/postscripts/im2005b.pdf
2. https://arxiv.org/pdf/1101.2245.pdf
3. http://www.eecs.harvard.edu/~michaelm/NEWWORK/postscripts/cbf2.pdf
4. https://pdfs.semanticscholar.org/d899/05bdf1ff791bdddc7c471070f34f4da18844.pdf