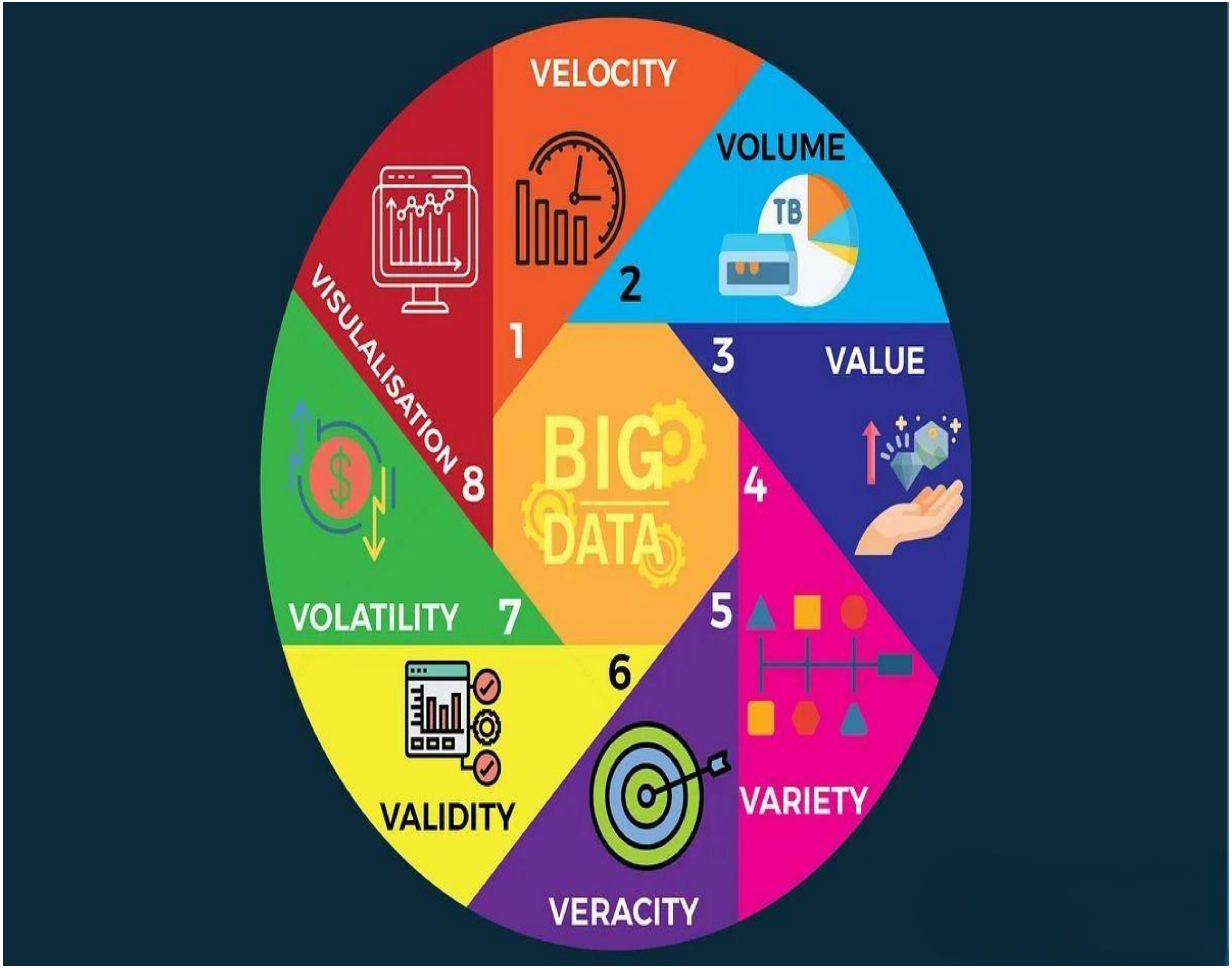


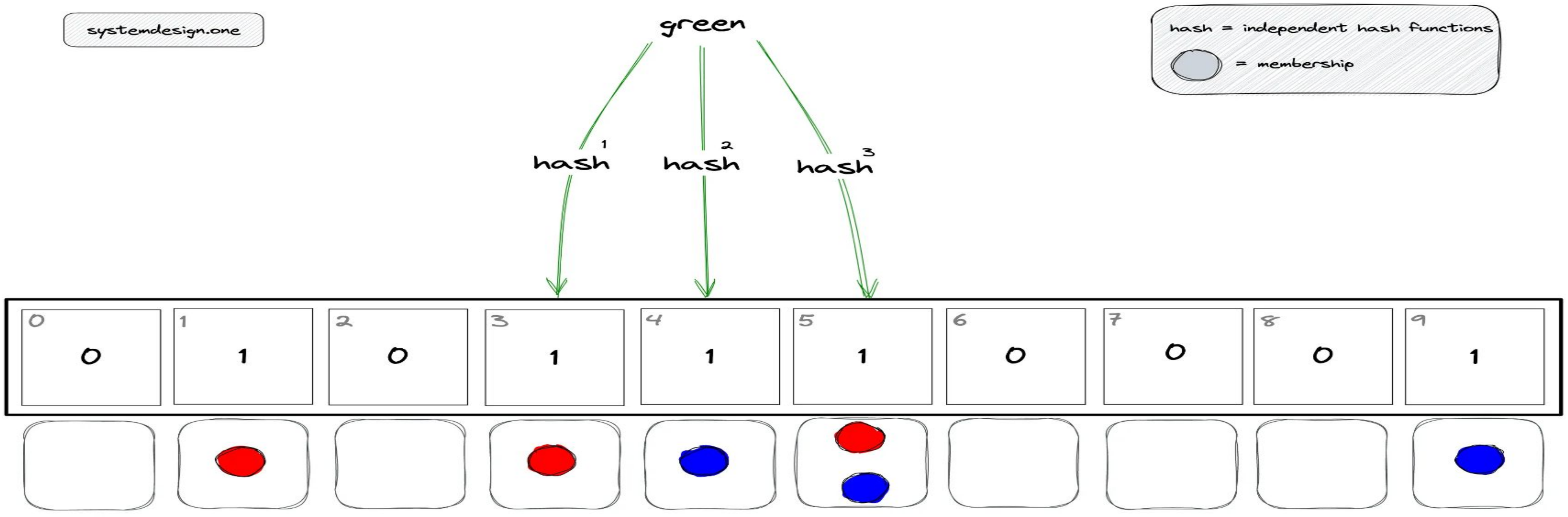
CS 3032: Big Data

Lec-11

In this Discussion . . .

- Data Streams
  - Bloom Filter
    - Bloom filter false positive
    - Asymptotic Complexity
    - Performance
    - Calculating probability of False Positives
    - Counting distinct elements in a stream
    - Use cases
    - Extensions

Bloom filter false positive



The bloom filter is queried to check the membership of item green, which is not a member of the bloom filter.	h1(green) mod 10 = 3 h2(green) mod 10 = 4 h3(green) mod 10 = 5 **The bloom filter will say yes although the item green is not a member of the bloom filter as the bits were set to one by items blue and red.**

Bloom Filter False Positives

- 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 to decrease the 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.

Asymptotic Complexity

- **The performance of the bloom filter depends on the hash functions used.**
  - The faster the computation of the hash function, the quicker the overall time of each operation on the bloom filter. If ***k*** **is the number of hash functions**, then:
- **Time Complexity**

Operation	**Time Complexity**
add item	O(k) or constant
membership query	O(k) or constant

Asymptotic Complexity

- The time complexity of the bloom filter is independent of the number of items already in the bloom filter.
- The k lookups in the bloom filter are independent and can be parallelized.

- **Time Complexity**

Operation	Time Complexity
add item	O(k) or constant
membership query	O(k) or constant

Asymptotic Complexity

- **Space Complexity**

  - **Regardless of the number of items in the bloom filter, the bloom filter requires a constant number of bits by reserving a few bits per item.** The bloom filter does not store the data items yielding a constant space complexity of O(1)

Bloom Filter Calculator

- The accuracy of the bloom filter depends on the following:
    - number of hash functions (k)
    - quality of hash functions
    - length of the bit array (n)
    - number of items stored in the bloom filter
- The properties of an optimal hash function for the bloom filter are the following:
    - fast
    - independent
    - uniformly-distributed

Bloom Filter Numerical Example

- Given, an empty bloom filter of size 11 with 4 hash functions namely:
  - $h_1(x) = (3x+ 3) \bmod 6$
  - $h_2(x) = (2x+ 9) \bmod 2$
  - $h_3(x) = (3x+ 7) \bmod 8$
  - $h_4(x) = (2x+ 3) \bmod 5$
  - Illustrate bloom filter insertion with 7 and then 8.
  - Perform bloom filter lookup/membership test with 10 and 48

Bloom Filter Numerical Example

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

K=4 (i.e., number of Hash Function)

x_1 = 7, then INSERT(x_1)	x_2 = 8, then INSERT(x_2)
● For $h_1(x_1) = ((3x_1 + 3) \bmod 6) \bmod 11$ $= ((3 * 7 + 3) \bmod 6) \bmod 11 = 0$ ● For $h_2(x_1) = ((2x_1 + 9) \bmod 2) \bmod 11$ $= ((2 * 7 + 9) \bmod 2) \bmod 11 = 1$ ● For $h_3(x_1) = ((3x_1 + 7) \bmod 8) \bmod 11$ $= ((3 * 7 + 7) \bmod 8) \bmod 11 = 4$ ● For $h_4(x_1) = ((2x_1 + 3) \bmod 5) \bmod 11$	● For $h_1(x_2) = ((3x_2 + 3) \bmod 6) \bmod 11$ $= ((3 * 8 + 3) \bmod 6) \bmod 11 = 3$ ● For $h_2(x_2) = ((2x_2 + 9) \bmod 2) \bmod 11$ $= ((2 * 8 + 9) \bmod 2) \bmod 11 = 1$ ● For $h_3(x_2) = ((3x_2 + 7) \bmod 8) \bmod 11$ $= ((3 * 8 + 7) \bmod 8) \bmod 11 = 7$ ● For $h_4(x_2) = ((2x_2 + 3) \bmod 5) \bmod 11$

Bloom Filter Numerical Example (Contd.)

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

K=4 (i.e., number of Hash Function)

$x_1 = 7$, then INSERT(x_1)

- For $h_1(x_1) = ((3x_1 + 3) \bmod 6) \bmod 11$
  $= ((3 * 7 + 3) \bmod 6) \bmod 11 = 0$
- For $h_2(x_1) = ((2x_1 + 9) \bmod 2) \bmod 11$
  $= ((2 * 7 + 9) \bmod 2) \bmod 11 = 1$
- For $h_3(x_1) = ((3x_1 + 7) \bmod 8) \bmod 11$
  $= ((3 * 7 + 7) \bmod 8) \bmod 11 = 4$
- For $h_4(x_1) = ((2x_1 + 3) \bmod 5) \bmod 11$
  $= ((2 * 7 + 3) \bmod 5) \bmod 11 = 2$

$x_2 = 8$, then INSERT(x_2)

- For $h_1(x_2) = ((3x_2 + 3) \bmod 6) \bmod 11$
  $= ((3 * 8 + 3) \bmod 6) \bmod 11 = 3$
- For $h_2(x_2) = ((2x_2 + 9) \bmod 2) \bmod 11$
  $= ((2 * 8 + 9) \bmod 2) \bmod 11 = 1$
- For $h_3(x_2) = ((3x_2 + 7) \bmod 8) \bmod 11$
  $= ((3 * 8 + 7) \bmod 8) \bmod 11 = 7$
- For $h_4(x_2) = ((2x_2 + 3) \bmod 5) \bmod 11$
  $= ((2 * 8 + 3) \bmod 5) \bmod 11 = 4$

State of Hashtable post to the insertion of x_1 and x_2

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

Bloom Filter Numerical Example (Contd.)

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

K=4 (i.e., number of Hash Function)

x_3 = 10, then INSERT(x_3)

- For $h_1(x_3) = ((3x_3 + 3) \bmod 6) \bmod 11$
  $= ((3 * 10 + 3) \bmod 6) \bmod 11 = 3$
- For $h_2(x_3) = ((2x_3 + 9) \bmod 2) \bmod 11$
  $= ((2 * 10 + 9) \bmod 2) \bmod 11 = 1$
- For $h_3(x_3) = ((3x_3 + 7) \bmod 8) \bmod 11$
  $= ((3 * 10 + 7) \bmod 8) \bmod 11 = 5$
- For $h_4(x_3) = ((2x_3 + 3) \bmod 5) \bmod 11$
  $= ((2 * 10 + 3) \bmod 5) \bmod 11 = 3$

x_4 = 48, then INSERT(x_4)

- For $h_1(x_4) = ((3x_4 + 3) \bmod 6) \bmod 11$
  $= ((3 * 48 + 3) \bmod 6) \bmod 11 = 3$
- For $h_2(x_4) = ((2x_4 + 9) \bmod 2) \bmod 11$
  $= ((2 * 48 + 9) \bmod 2) \bmod 11 = 1$
- For $h_3(x_4) = ((3x_4 + 7) \bmod 8) \bmod 11$
  $= ((3 * 48 + 7) \bmod 8) \bmod 11 = 7$
- For $h_4(x_4) = ((2x_4 + 3) \bmod 5) \bmod 11$
  $= ((2 * 48 + 3) \bmod 5) \bmod 11 = 4$

x_3 doesn't exist

x_4 => Case of False Positive

Optimum Number of Hash Functions

- The number of hash functions k must be a positive integer. If n is size of bit array and m is number of elements to be inserted, then k can be calculated as :

$$k = \frac{n}{m} \ln(2)$$

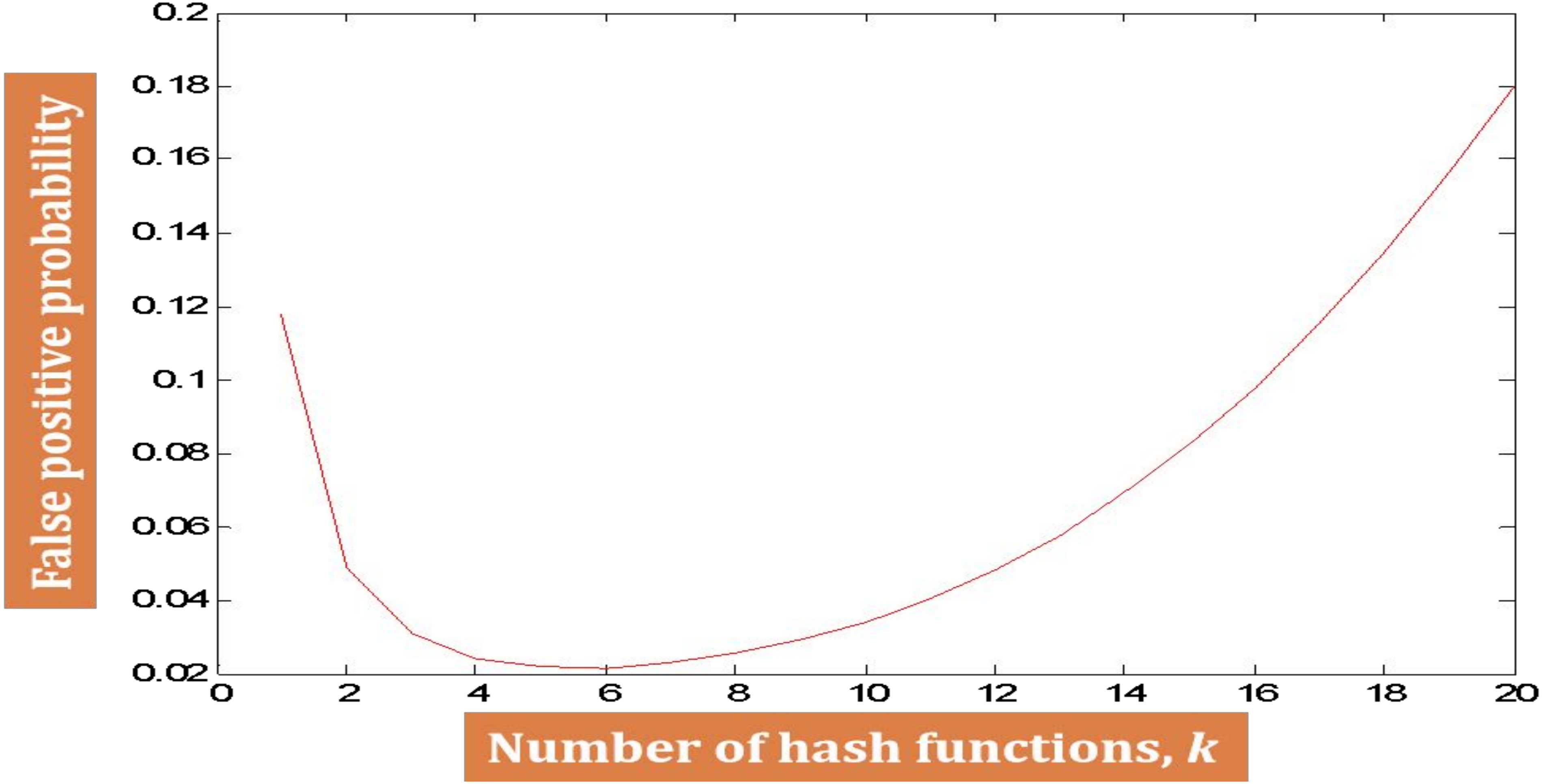
Optimum Number of Hash Functions Example

- Calculate the optimal number of hash functions for 10 bit length bloom filter having 3 numbers of input elements.

  Here $n = 10$, $m = 3$

  then $k = (n / m)\ln 2 = (10/3)\ln 2 = \lceil 2.310490601871 \rceil \cong 3$

What Happens when Increasing the Number of Hash Functions



Calculating probability of False Positives

- Probability that a slot is hashed = 1/n
- Probability that a slot is not hashed = 1 – (1/n)
- Probability that a slot is not hashed after insertion of an element for all the k hash function is:

$$\left(1 - \frac{1}{n}\right)^{k}$$

Calculating probability of False Positives (Contd.)

- Probability that a slot is not set to 1 after insertion of m element is:

$$\left(1 - \frac{1}{n}\right)^{km}$$

- Probability that a slot is set to 1 after insertion of m element is:

$$1 - \left(1 - \frac{1}{n}\right)^{km}$$

Calculating probability of False Positives (Contd.)

- Let n be the size of bit array, k be the number of hash functions and m be the number of expected elements to be inserted in the filter, then the probability of false positive p can be calculated as:

$$\left(1-\left(\frac{1}{e}\right)^{\frac{km}{n}}\right)^{k}$$

Question: Calculate the probability of False Positives with table size 10 and no. of items to be inserted are 3.

Counting distinct elements in a stream

$$(1, 2, 2, 1, 3, 1, 5, 1, 3, 3, 3, 2, 2)$$

Number of distinct elements = 4

How to Calculate?

Approach - I

1. Initialize the hashtable (large binary array) of size n with all zeros.
2. Choose the hash function $h_i : i \in \{1, ..., k\}$
3. For each flow label $f \in \{1, ..., m\}$, compute h(f) and mark that position in the hashtable with 1.
4. Count the number of positions in the hashtable with 1 and call it c.
5. The number of distinct items is m* ln (m / (m-c))

Counting distinct elements in a stream Exercise

Count the distinct elements in a data stream of elements {1, 2, 2, 1, 3, 1, 5, 1, 3, 3, 3, 2, 2} with the hash function $h(x) = (5x+1) \bmod 6$ of size 11.

Counting distinct elements in a stream : Flajolet-Martin algorithm - Approach II

- Count the distinct elements in a data stream of elements {6,8,4,6,3,4} with the hash function h(x) = (5x+1) mod 6 of size 11.

How to Calculate?

If there are m distinct elements in a set comprising of n elements, the algorithm runs in *O(n)* time and *O(log(m))* space complexity

1. **Apply Hash function(s) to the data stream and compute the slots.**

   $x_1 = 6$, $h(x_1) = ((5x_1 + 1) \bmod 6) = ((5 * 6 + 1) \bmod 6) = 1$
   $x_2 = 8$, $h(x_2) = ((5x_2 + 1) \bmod 6) = ((5 * 8 + 1) \bmod 6) = 5$
   $x_3 = 4$, $h(x_3) = ((5x_3 + 1) \bmod 6) = ((5 * 4 + 1) \bmod 6) = 3$
   $x_4 = 6$, $h(x_4) = ((5x_4 + 1) \bmod 6) = ((5 * 6 + 1) \bmod 6) = 1$
   $x_5 = 3$, $h(x_5) = ((5x_5 + 1) \bmod 6) = ((5 * 3 + 1) \bmod 6) = 4$
   $x_6 = 4$, $h(x_6) = ((5x_6 + 1) \bmod 6) = ((5 * 4 + 1) \bmod 6) = 3$

The slot numbers obtained are: {1, 5, 3, 1, 4, 3}

Counting distinct elements in a stream : Flajolet-Martin algorithm - Approach II (Contd.)

- Count the distinct elements in a data stream of elements {6,8,4,6,3,4} with the hash function h(x) = (5x+1) mod 6 of size 11.

How to Calculate?

2. Convert the slot numbers to binary

$x_1 = 6, (h(x_1)) = 1 = 001$
$x_2 = 8, (h(x_2)) = 5 = 101$
$x_3 = 4, (h(x_3)) = 3 = 011$
$x_4 = 6, (h(x_4)) = 1 = 001$
$x_5 = 3, (h(x_5)) = 4 = 100$
$x_6 = 4, (h(x_6)) = 3 = 011$

The slot numbers obtained are: {1, 5, 3, 1, 4, 3}

Counting distinct elements in a stream : Flajolet-Martin algorithm - Approach II (Contd.)

- Count the distinct elements in a data stream of elements {6,8,4,6,3,4} with the hash function h(x) = (5x+1) mod 6 of size 11.

How to Calculate?

3. Calculate the maximum trailing zeros

$x_1 = 6, (h(x_1)) = 1 = 001$
$x_2 = 8, (h(x_2)) = 5 = 101$
$x_3 = 4, (h(x_3)) = 3 = 011$
$x_4 = 6, (h(x_4)) = 1 = 001$
$x_5 = 3, (h(x_5)) = 4 = 100$
$x_6 = 4, (h(x_6)) = 3 = 011$

TZ ={0, 0, 0, 0, 2, 0}
/* TZ stands for Trailing Zeros */
R = MAX(TZ) = MAX(0, 0, 0, 0, 2, 0) = 2

Counting distinct elements in a stream : Flajolet-Martin algorithm - Approach II (Contd.)

- Count the distinct elements in a data stream of elements {6,8,4,6,3,4} with the hash function h(x) = (5x+1) mod 6 of size 11.

How to Calculate?

4. Estimate the distinct elements with the formula 2^R

$x_1 = 6, (h(x_1)) = 1 = 001$
$x_2 = 8, (h(x_2)) = 5 = 101$
$x_3 = 4, (h(x_3)) = 3 = 011$
$x_4 = 6, (h(x_4)) = 1 = 001$
$x_5 = 3, (h(x_5)) = 4 = 100$
$x_6 = 4, (h(x_6)) = 3 = 011$

TZ ={0, 0, 0, 0, 2, 0}
/* TZ stands for Trailing Zeros */
R = MAX(TZ) = MAX(0, 0, 0, 0, 2, 0) = 2
Number of distinct elements = $2^R = 2^2 = 4$

Practice Problems

1. Develop Flajolet-Martin algorithm and using it, count the distinct elements in a data stream of elements {6, 8, 4, 6, 3, 4, 7, 6, 9} with the hash function $h(x) = (5x+1) \bmod 6$ of size 11.
2. A empty bloom filter is of size 11 with 4 hash functions namely:

   $h_1(x) = (3x+ 3) \bmod 6$

   $h_2(x) = (2x+ 9) \bmod 2$

   $h_3(x) = (3x+ 7) \bmod 8$

   $h_4(x) = (2x+ 3) \bmod 5$

   Illustrate bloom filter insertion with 17, 81 and 37.

   Perform bloom filter lookup/membership test with 10 and 81

Practice Problems-I

3. A empty bloom filter is of size 11 with 2 hash functions namely:

   $h_1(x) = (3x+ 3) \bmod 18$

   $h_2(x) = (2x+ 9) \bmod 22$

   Illustrate bloom filter insertion with 7, 8 and 77.

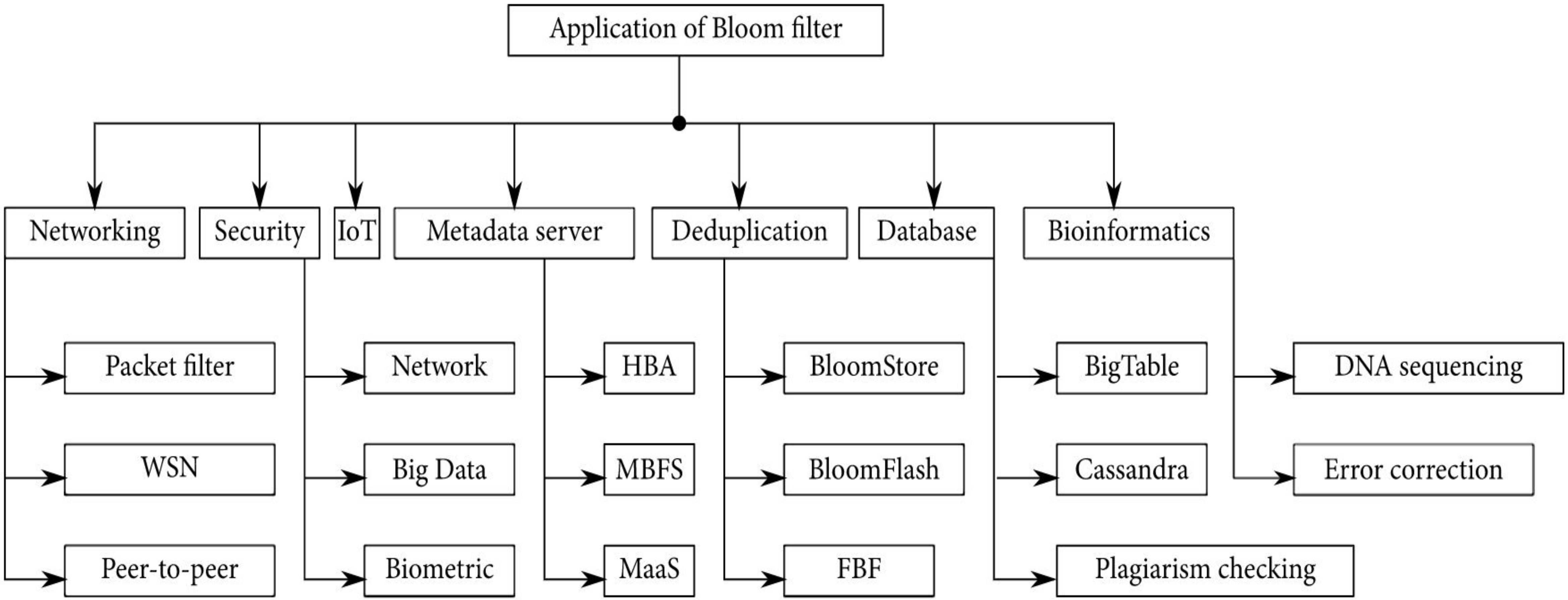
   Perform bloom filter lookup/membership test with 7, 10 and 88

4. Develop an algorithm to i) insert an item, and to ii) test the membership (or lookup) in Bloom Filter. Draw a step-by-step process in the insertion of element 25, and then 40 into the Bloom Filter of size 10. Then, draw a step-by-step process for lookup/membership test with the elements 10 and 48. The hash functions are: $h_1(x) = (3x+41) \bmod 6$, and $h_2(x) = (7x+5)$. Identify whether any lookup element (i.e. either 10 or 48) is resulting into the case of FALSE POSITIVE?

Practice Problems-II

5. Let, Facebook wants to count "How many unique users visited the Facebook this month?" What will be the stream elements in this case?

Applications of Bloom Filter



Bloom Filter Use Cases

- Bitcoin uses Bloom filters to speed up wallet synchronization and also to improve Bitcoin wallet security
- Google Chrome uses the Bloom filter to identify malicious URLs - it keeps a local Bloom filter as the first check for Spam URL
- Google BigTable and Apache Cassandra use Bloom filters to reduce disk lookups for non-existent rows or columns

Bloom Filter Use Cases (Contd.)

- The Squid Web Proxy Cache uses Bloom filters for cache digests - proxies periodically exchange Bloom filters for avoiding ICP messages
- Genomics community uses Bloom filter for classification of DNA sequences and efficient counting of k-mers in DNA sequences
- Used for preventing weak password choices using a dictionary of easily guessable passwords as bloom filter
- Used to implement spell checker using a predefined dictionary with large number of words

Extensions of Bloom Filter / Other Types of Bloom Filter

- **Compressed Bloom Filter** - Using a larger but sparser Bloom Filter can yield the same false positive rate with a smaller number of transmitted bits.
- **Scalable Bloom Filter** - A Scalable Bloom Filters consist of two or more Standard Bloom Filters, allowing arbitrary growth of the set being represented.
- **Generalized Bloom Filter** - Generalized Bloom Filter uses hash functions that can set as well as reset bits.
- **Stable Bloom Filter** - This variant of Bloom Filter is particularly useful in data streaming applications.

Data Warehouse Vs. Hadoop Vs. Stream Computing

Charactristics	Data warehouse	Hadoop	Stream computing
Type of Data Stored	Structured	Structured & Unstructured	No storage
Storage purpose	Reporting & dash board	Long running computations	Real time analytics
Age of data	Old	Past	Current/new data
Size of data	Terra/Peta bytes	Giga Bytes	Kilo Bytes
Speed of processing	Peta bytes /day	Kbps	Mbps
Implementation cost	High	Medium	Low
Volume	High	High	Low
Velocity	Nil	Nil	High
Variety	Nil	High	High

References

1. https://systemdesign.one/bloom-filters-explained/#introduction
2. https://maneesh-chaturvedi.medium.com/streaming-algorithms-ii-counting-distinct-elements-6eb03baed30e
3.