SOLUTIONS

Exercises on Block2:

Finding Frequent Item Sets
Finding Similar Items
Searching in Data Streams

Advanced Search Techniques for Large Scale Data Analytics

Pavel Zezula and Jan Sedmidubsky
Masaryk University
http://disa.fi.muni.cz

Frequent Item Sets (1) – Assignment

- Suppose 100 items (numbered 1 to 100) and 100 baskets (numbered 1 to 100)
  - Item *i* is in basket *b* if and only if *i* divides *b* with no remainder, i.e., item 1 is in all the baskets, item 2 is in all fifty of the even-numbered baskets, etc.
- Tasks:
  1) Identify the frequent items when the support threshold is set to 5
  2) Compute the confidence of these association rules
     a) {5, 7} → 2
     b) {2, 3, 4} → 5

Frequent Item Sets (1) – Recap

- **Simplest question:** Find sets of items that appear together "frequently" in baskets
- ***Support*** for item set I: Number of baskets containing all items in I
  - (Often expressed as a fraction of the total number of baskets)
- Given a ***support threshold*** s, then sets of items that appear in at least s baskets are called ***frequent itemsets***

Support of {Beer, Bread} = 2

Frequent Item Sets (1) – Recap

- **Association Rules:**
  If-then rules about the contents of baskets
- $\{i_1, i_2,\ldots,i_k\} \rightarrow j$ means: "if a basket contains all of $i_1,\ldots,i_k$ then it is ***likely*** to contain j"
- **In practice there are many rules, want to find significant/interesting ones!**
- ***Confidence*** of this association rule is the probability of j given $I = \{i_1,\ldots,i_k\}$

Frequent Item Sets (1) – Solution

1) 20 frequent items: **1–20**
2) Association rules:
   a) The baskets containing both items 5 and 7 are baskets 35 and 70, in which only basket 70 also contains item 2. Hence, the confidence of the rule {5, 7} → 2 is **1/2**.
   b) The baskets whose numbers are the multiples of 12 contain item set {2, 3, 4} as a subset – there are 8 such baskets. The baskets whose numbers are the multiples of 60 contain item set {2, 3, 4, 5} as a subset – there is 1 such basket. Hence, the confidence of the rule {2, 3, 4} → 5 is **1/8**.

Frequent Item Sets (2) – Assignment

- Consider the following twelve baskets, each of them contains 3 of 6 items (1 through 6):
  - {1, 2, 3} {2, 3, 4} {3, 4, 5} {4, 5, 6}
  - {1, 3, 5} {2, 4, 6} {1, 3, 4} {2, 4, 5}
  - {3, 5, 6} {1, 2, 4} {2, 3, 5} {3, 4, 6}
- Suppose the support threshold is 4. On the first pass of the PCY algorithm, a hash table with 11 buckets is used, and the set {*i*, *j*} is hashed to bucket *i* *j* mod 11:
  1) Compute the support for each item and each pair of items
  2) Which pairs hash to which buckets?
  3) Which buckets are frequent?
  4) Which pairs are counted on the second pass?

Frequent Item Sets (2) – Recap

PCY Algorithm – First Pass

```
FOR (each basket) :
    FOR (each item in the basket) :
        add 1 to item's count;
    FOR (each pair of items) :
        hash the pair to a bucket;
        add 1 to the count for that bucket;
```

New in PCY (applies to the FOR (each pair of items) block)

- **Few things to note:**
  - Pairs of items need to be generated from the input file; they are not present in the file
  - We are not just interested in the presence of a pair, but we need to see whether it is present at least ***s*** (support) times

Frequent Item Sets (2) – Recap

- **Observation: If a bucket contains a frequent pair, then the bucket is surely frequent**
- However, even without any frequent pair, a bucket can still be frequent ☹
  - So, we cannot use the hash to eliminate any member (pair) of a "frequent" bucket
- **But, for a bucket with total count less than *s*, none of its pairs can be frequent** ☺
  - Pairs that hash to this bucket can be eliminated as candidates (even if the pair consists of 2 frequent items)

- **Pass 2:**
  Only count pairs that hash to frequent buckets

Frequent Item Sets (2) – Solution 1/4

1) Compute the support for each item and each pair of items

- Support for each item:

item	1	2	3	4	5	6
support	4	6	8	8	6	4

- Support for each pair of items:

pair	{1, 2}	{1, 3}	{1, 4}	{1, 5}	{1, 6}	{2, 3}	{2, 4}	{2, 5}
support	2	3	2	1	0	3	4	2

pair	{2, 6}	{3, 4}	{3, 5}	{3, 6}	{4, 5}	{4, 6}	{5, 6}
support	1	4	4	2	3	3	2

Frequent Item Sets (2) – Solution 2/4

2) Which pairs hash to which buckets?

- The set {*i*, *j*} is hashed to bucket no.: *i* *j* mod 11

pair	{1, 2}	{1, 3}	{1, 4}	{1, 5}	{1, 6}	{2, 3}	{2, 4}	{2, 5}
bucket	2	3	4	5	6	6	8	10

pair	{2, 6}	{3, 4}	{3, 5}	{3, 6}	{4, 5}	{4, 6}	{5, 6}
bucket	1	1	4	7	9	2	8

Frequent Item Sets (2) – Solution 3/4

3) Which buckets are frequent?

- Bucket support – sum of supports of pairs belonging to the given bucket:

bucket	0	1	2	3	4	5	6	7
support	0	5	5	3	6	1	3	2

bucket	8	9	10
support	6	3	2

- The frequent buckets are those with support above 4, i.e., buckets: 1, 2, 4, 8

Frequent Item Sets (2) – Solution 4/4

4) Which pairs are counted on the second pass
   - As only pairs in frequent buckets will be counted on the second pass of PCY, they are:

     {1, 2}, {1, 4}, {2, 4}, {2, 6}, {3, 4}, {3, 5}, {4, 6}, {5, 6}

Finding Similar Items (1) – Assignment

- Compute the Jaccard similarities of each pair of the following three sets:
  - A = {1, 2, 3, 4}
  - B = {2, 3, 5, 7}
  - C = {2, 4, 6}

Finding Similar Items (1) – Solution

- *sim*(A, B) = 2/6 = 1/3
- *sim*(A, C) = 2/5
- *sim*(B, C) = 1/6

Finding Similar Items (2) – Assignment

- Consider two documents A and B
  - If their 3-shingle resemblance is 1 (using Jaccard similarity), does that mean that A and B are identical?
    - If so, prove it. If not, give a counterexample.

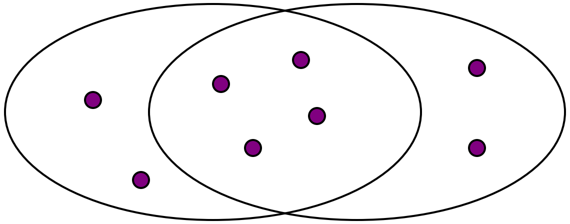
Finding Similar Items (2) – Recap

- A *k*-shingle (or *k*-gram) for a document is a sequence of *k* tokens that appears in the doc
  - Tokens can be characters, words or something else, depending on the application
  - Assume tokens = characters for examples
- **Example: k=2**; document $\mathbf{D_1}$ = abcab
  Set of 2-shingles: $\mathbf{S(D_1)}$ = {ab, bc, ca}
  - **Option:** Shingles as a bag (multiset), count ab twice: $\mathbf{S'(D_1)}$ **=** {ab, bc, ca, ab}

Finding Similar Items (2) – Recap

- **Document D_1 is a set of its k-shingles $C_1=S(D_1)$**
- Equivalently, each document is a 0/1 vector in the space of *k*-shingles
  - Each unique shingle is a dimension
  - Vectors are very sparse
- **A natural similarity measure is the Jaccard similarity:**

$$sim(D_1, D_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$



Finding Similar Items (2) – Solution

- No, the documents A and B need not be identical
  - Counterexample:
    - A: **abab**
      - 3-shingles: S(A) = {aba, bab}
    - B: **baba**
      - 3-shingles: S(B) = {bab, aba}

      - $sim(A, B) = | S(A) \cap S(B) | / | S(A) \cup S(B) | = 1$

Finding Similar Items (3) – Assignment

- For the matrix

Element	D1	D2	D3	D4
0	0	1	0	1
1	0	1	0	0
2	1	0	0	1
3	0	0	1	0
4	0	0	1	1
5	1	0	0	0

1) Compute the minhash signature for each column (document) using the following hash functions:
   - $h_1(x) = 2x + 1 \bmod 6$
   - $h_2(x) = 3x + 2 \bmod 6$
   - $h_3(x) = 5x + 2 \bmod 6$
2) Which of these hash functions are true permutations?
3) How close are the estimated Jaccard similarities for the six pairs of columns to the true Jaccard similarities?

Finding Similar Items (3) – Recap

- **Rows** = elements (e.g., shingles)
- **Columns** = sets (e.g., documents)
  - 1 in row ***e*** (shingle) and column ***s*** (document) if and only if ***e*** is a member of ***s***
  - Column similarity is the Jaccard similarity of the corresponding sets (rows with value 1)
  - **Typical matrix is sparse!**
- **Each document is a column:**
  - **Example: sim(C_1 ,C_2) = ?**
    - Size of intersection = 3; size of union = 6, Jaccard similarity (not distance) = 3/6
    - **d(C_1,C_2) = 1 – (Jaccard similarity) = 3/6**

Documents

Shingles

1	1	1	0
1	1	0	1
0	1	0	1
0	0	0	1
1	0	0	1
1	1	1	0
1	0	1	0

Finding Similar Items (3) – Recap

Min-Hashing Example

2nd element of the permutation is the first to map to a 1

Permutation π

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Input matrix (Shingles x Documents)

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix *M*

2	1	2	1
2	1	4	1
1	2	1	2

4th element of the permutation is the first to map to a 1

Finding Similar Items (3) – Solution 1+2/3

1) Compute the minhash signature for each column using the following hash functions:
   - $h_1(x) = 2x + 1 \bmod 6$
   - $h_2(x) = 3x + 2 \bmod 6$
   - $h_3(x) = 5x + 2 \bmod 6$

   Hashes are computed on element IDs:

Element	D1	D2	D3	D4	$h_1(x)$	$h_2(x)$	$h_3(x)$
0	0	1	0	1	**1**	**2**	**2**
1	0	1	0	0	**3**	**5**	**1**
2	1	0	0	1	**5**	**2**	**0**
3	0	0	1	0	**1**	**5**	**5**
4	0	0	1	1	**3**	**2**	**4**
5	1	0	0	0	**5**	**5**	**3**

Minhash signature:

D1	D2	D3	D4
5	1	1	1
2	2	2	2
0	1	4	0

(rows correspond to hash functions)

2) Which of these hash functions are true permutations: $\boldsymbol{h_3}$ only

Finding Similar Items (3) – Solution 3/3

3) How close are the estimated Jaccard similarities for the six pairs of columns (documents) to the true Jaccard similarities?

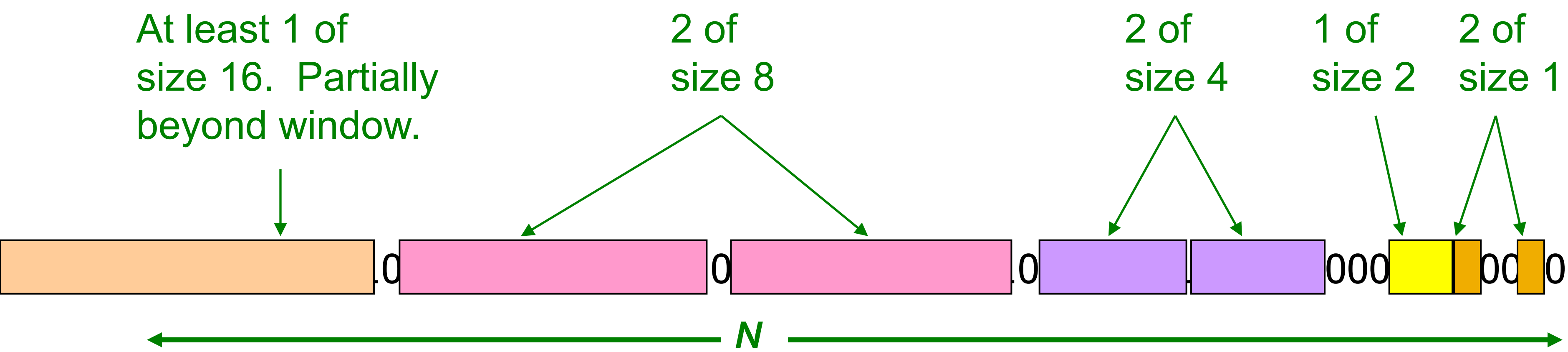
Jaccard similarities on	D1 / D2	D1 / D3	D1 / D4	D2 / D3	D2 / D4	D3 / D4
Original documents	0	0	0.25	0	0.25	0.25
Minhash signatures	0.33	0.33	0.67	0.67	0.67	0.67

- => the estimated Jaccard similarities are not close to the true ones at all
  - To make the estimated similarity closer to the true one, there is a need of more and better (i.e., resulting in true permutations) hash functions

Data Streams (1) – Assignment

- Suppose we are maintaining a count of 1s using the DGIM method
  - Each bucket is represented by (*i*, *t*)
    - *i* – the number of 1s in the bucket
    - *t* – the bucket timestamp (time of the most recent 1)
- Consider the following properties:
  - Current time is 200
  - Window size is 60
  - Current buckets are:
    - (16, 148) (8, 162) (8, 177) (4, 183) (2, 192) (1, 197) (1, 200)
  - At the next ten clocks (201 through 210), the stream has 0101010101
- What will the sequence of buckets be at the end of these ten inputs?

Data Streams (1) – Recap



Each stream bit has a ***timestamp*** (starting **1, 2,** …), recorded by modulo ***N***

A ***bucket*** is a record consisting of:

(A) The timestamp of its end

(B) The number of 1s between its beginning and end

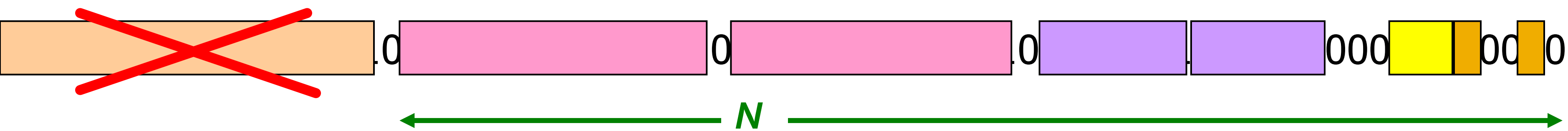
Three properties of buckets that are maintained:

- Either **one** or **two** buckets with the same **power-of-2** number of **1s**
- Buckets do not overlap in timestamps
- Buckets are sorted by size

Buckets disappear when their end-time is **>** ***N*** time units in the past

Data Streams (1) – Recap

- When a new bit comes in, drop the last (oldest) bucket if its end-time is prior to ***N*** time units before the current time



- **2 cases:** Current bit is **0** or **1**

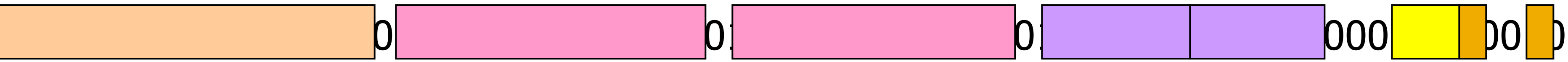
- **If the current bit is 0:**
  no other changes are needed

Data Streams (1) – Recap

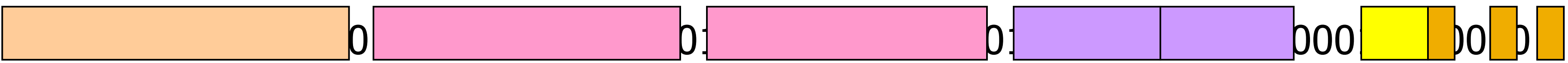
- **If the current bit is 1:**
  - **(1)** Create a new bucket of size **1**, for just this bit
    - **End timestamp = current time**
  - **(2)** If there are now **three buckets of size 1**, **combine the oldest two into a bucket of size 2**
  - **(3)** If there are now **three buckets of size 2**, **combine the oldest two into a bucket of size 4**
  - **(4) And so on …**

Updating buckets (example):

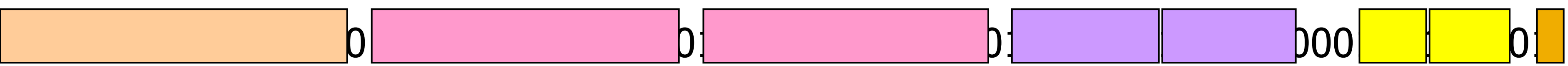
Current state of the stream:



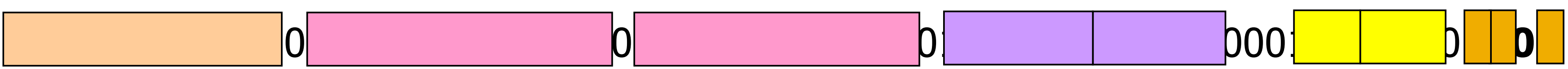
Bit of value 1 arrives



Two orange buckets get merged into a yellow bucket



Next bit 1 arrives, new orange bucket is created, then 0 comes, then 1:



Buckets get merged…

State of the buckets after merging

Data Streams (1) – Solution

- There are 5 occurrences of 1s in the upcoming stream 0**1**0**1**0**1**0**1**0**1**. Each one updates the buckets to be:
  - (1) Combine the oldest two buckets of size 1

    (16, 148) (8, 162) (8, 177) (4, 183) (2, 192) (1, 197) (1, 200) (1, 202)

    => (16, 148) (8, 162) (8, 177) (4, 183) (2, 192) (2, 200) (1, 202)
  - (2) No combination needed

    (16, 148) (8, 162) (8, 177) (4, 183) (2, 192) (2, 200) (1, 202) (1, 204)
  - (3) Combine the oldest two buckets of size 1, and then oldest two buckets of size 2

    (16, 148) (8, 162) (8, 177) (4, 183) (2, 192) (2, 200) (1, 202) (1, 204) (1, 206)

    => (16, 148) (8, 162) (8, 177) (4, 183) (2, 192) (2, 200) (2, 204) (1, 206)

    => (16, 148) (8, 162) (8, 177) (4, 183) (4, 200) (2, 204) (1, 206)
  - (4) No combination needed; window size is 60, so (16, 148) should be dropped

    (16, 148) (8, 162) (8, 177) (4, 183) (4, 200) (2, 204) (1, 206) (1, 208)

    => (8, 162) (8, 177) (4, 183) (4, 200) (2, 204) (1, 206) (1, 208)
  - (5) Combine the oldest two buckets of size 1

    (8, 162) (8, 177) (4, 183) (4, 200) (2, 204) (1, 206) (1, 208) (1, 210)

    => **(8, 162) (8, 177) (4, 183) (4, 200) (2, 204) (2, 208) (1, 210)**

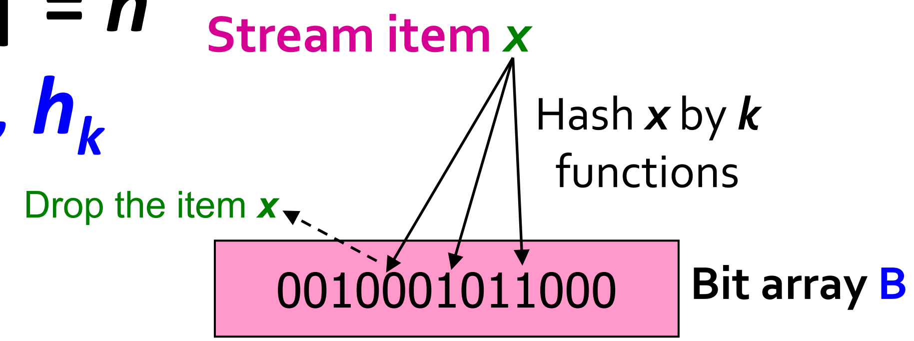
Data Streams (2) – Assignment

- Assume the Bloom Filter technique, *B* as a single array of 8 bits, and the following two hash functions:
  - $h_1(x) = x \bmod 3$,
  - $h_2(x) = x \bmod 7$.
- For the set $S = \{3, 5\}$ of two keys and the stream 7, 12, ... of integer values:

1) Determine the content of the *B* bit array;
2) Apply the Bloom Filter technique to the first two stream values (i.e., 7 and 12) and decide whether they pass through the filter, or not;
3) What should be the best number of hash functions for this scenario with $|S| = 2$ keys and $|B| = 8$ bits?

Data Streams (2) – Recap

- Consider: **|S| = *m*, |B| = *n***
- ***k* hash functions $h_1, \ldots, h_k$**
- **Initialization:**
  - Set **B** to all **0s**
  - Hash each element $s \in S$ using each hash function h_i, set $B[h_i(s)] = 1$ (for each $i = 1, \ldots, k$)

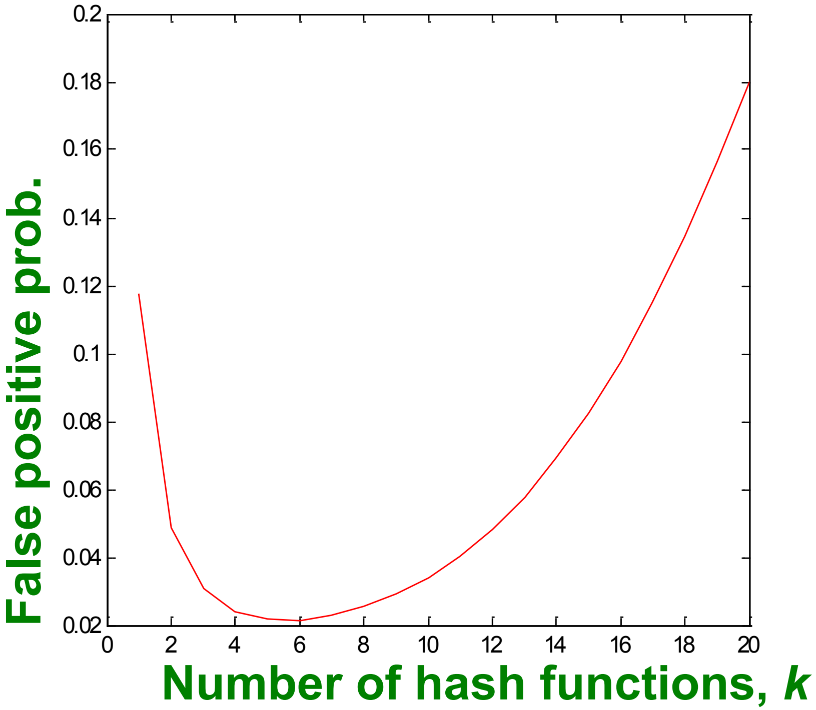


(**note:** we have a single array B!)

- **Run-time:**
  - When a stream element with key ***x*** arrives
    - If $B[h_i(x)] = 1$ <u>for all</u> $i = 1, \ldots, k$ then declare that ***x*** is in ***S***
      - That is, ***x*** hashes to a bucket set to **1** for every hash function $h_i(x)$
    - Otherwise discard the element ***x***

Data Streams (2) – Recap

- ***m* = 1 billion, *n* = 8 billion**
  - **k = 1**: $(1 - e^{-1/8})$ = **0.1175**
  - **k = 2**: $(1 - e^{-1/4})^2$ = **0.0493**



- **What happens as we keep increasing *k*?**
- "Optimal" value of ***k***: ***n/m*** **ln(2)**
  - **In our case:** Optimal **k = 8 ln(2) = 5.54 ≈ 6**

Data Streams (2) – Solution

- Keys: $S = \{3, 5\}$ Stream: 7, 12, ...
- Hash functions: $h_1(x) = x \bmod 3$ $h_2(x) = x \bmod 7$

1) The *B* bit array of size 8 contains 1s at the positions to which the keys are hashed by all the hash functions
   - $h_1(3) = 3 \bmod 3 = 0$ $h_1(5) = 5 \bmod 3 = 2$
   - $h_2(3) = 3 \bmod 7 = 3$ $h_2(5) = 5 \bmod 7 = 5$

B:

1	0	1	1	0	1	0	0

2) Stream values 7 and 12:
   - $h_1(7) = 7 \bmod 3 = \mathbf{1}$ $h_1(12) = 12 \bmod 3 = \mathbf{0}$
   - $h_2(7) = 7 \bmod 7 = \mathbf{0}$ $h_2(12) = 12 \bmod 7 = \mathbf{5}$

➔ 7 **does not pass**;12 **passes**

3) The best number of hash functions for $|S| = 2$ keys and $|B| = 8$ bits:

$$n / m \cdot ln(2) = |B| / |S| \cdot ln(2) = 8 / 2 \cdot ln(2) = 2.77 \sim \mathbf{3}$$