UNIVERSITY OF WATERLOO

Big Data Infrastructure

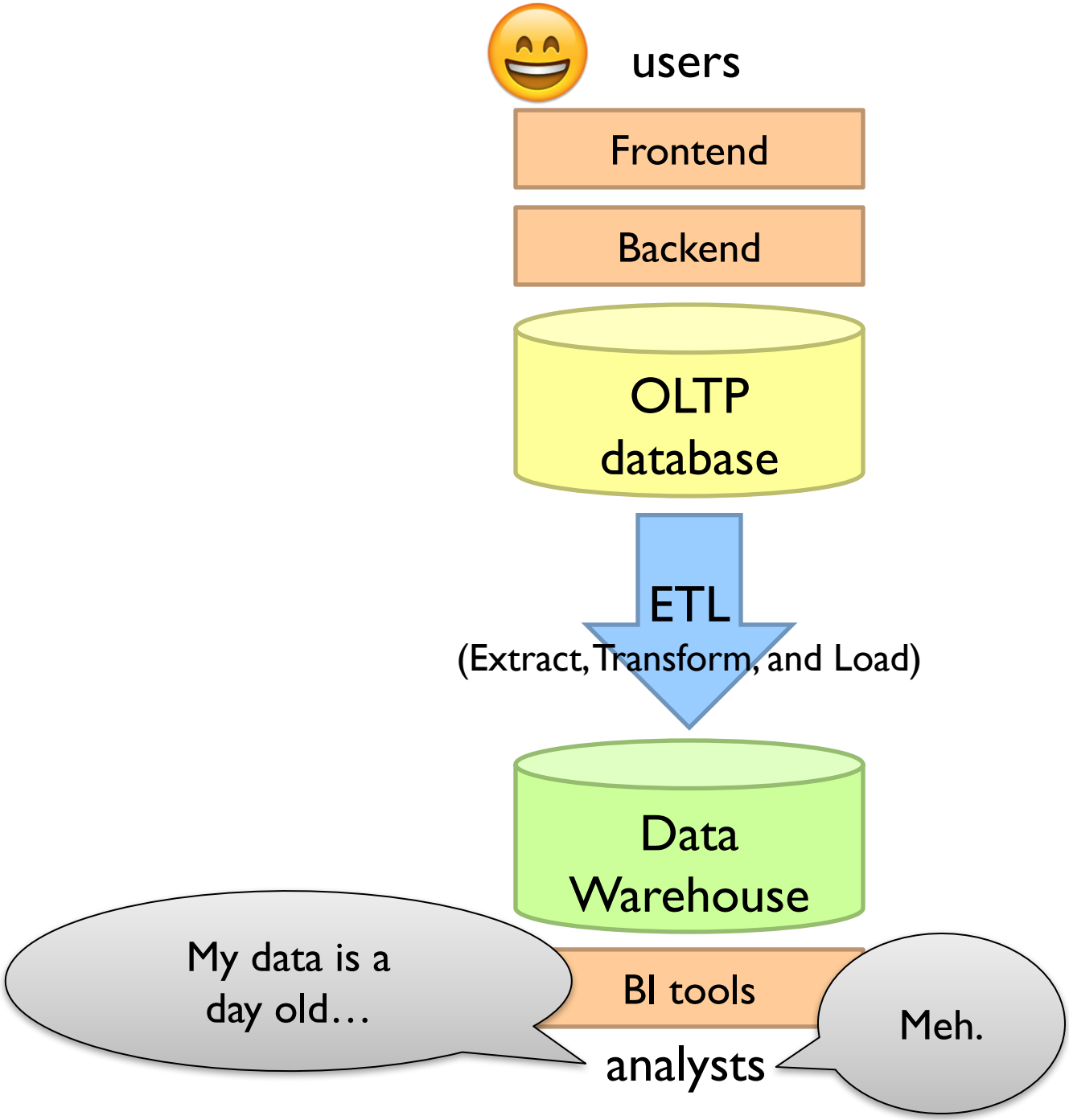
CS 489/698 Big Data Infrastructure (Winter 2017)

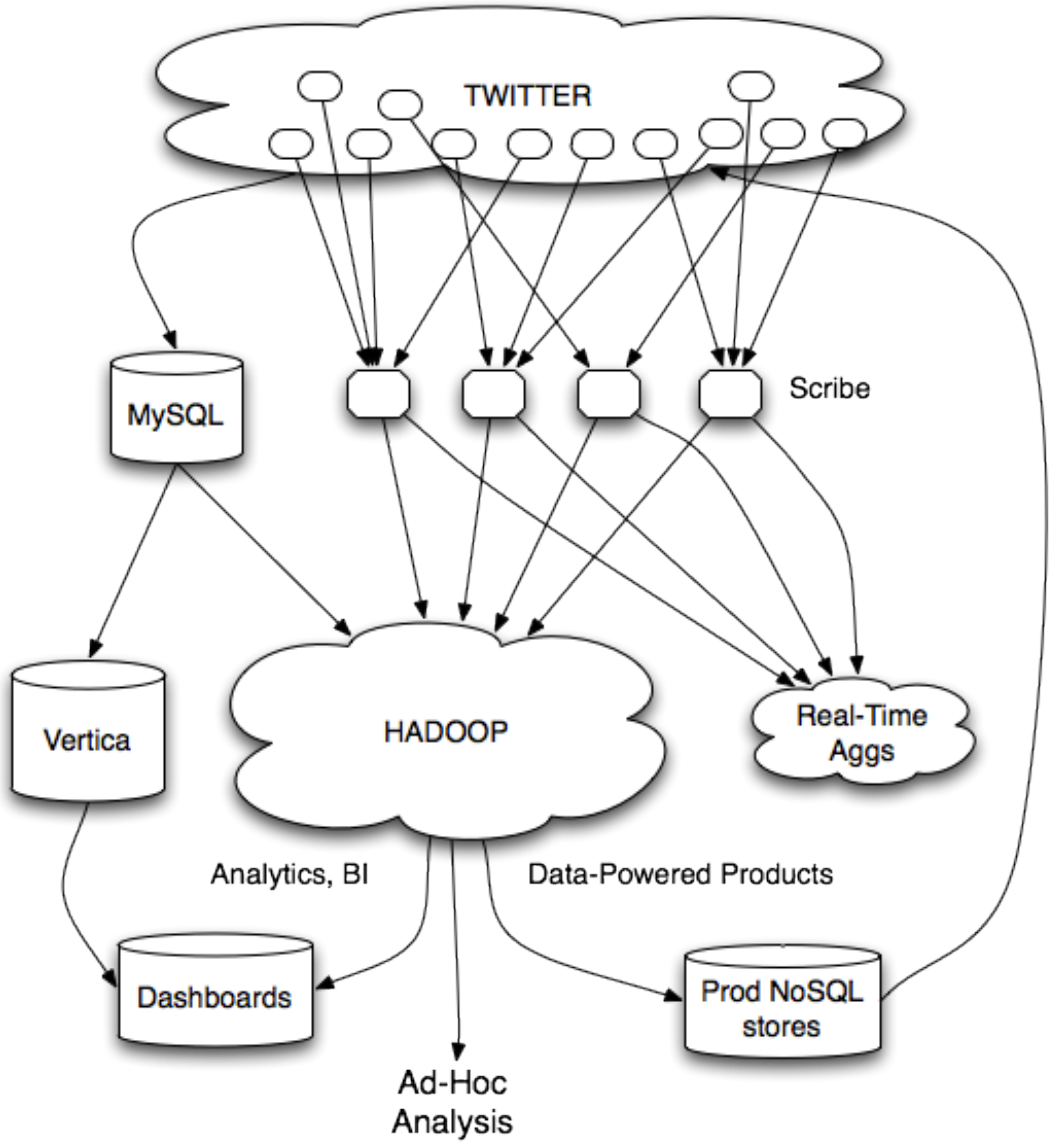
Week 12: Real-Time Data Analytics (1/2)

March 28, 2017

Jimmy Lin
David R. Cheriton School of Computer Science
University of Waterloo

These slides are available at http://lintool.github.io/bigdata-2017w/

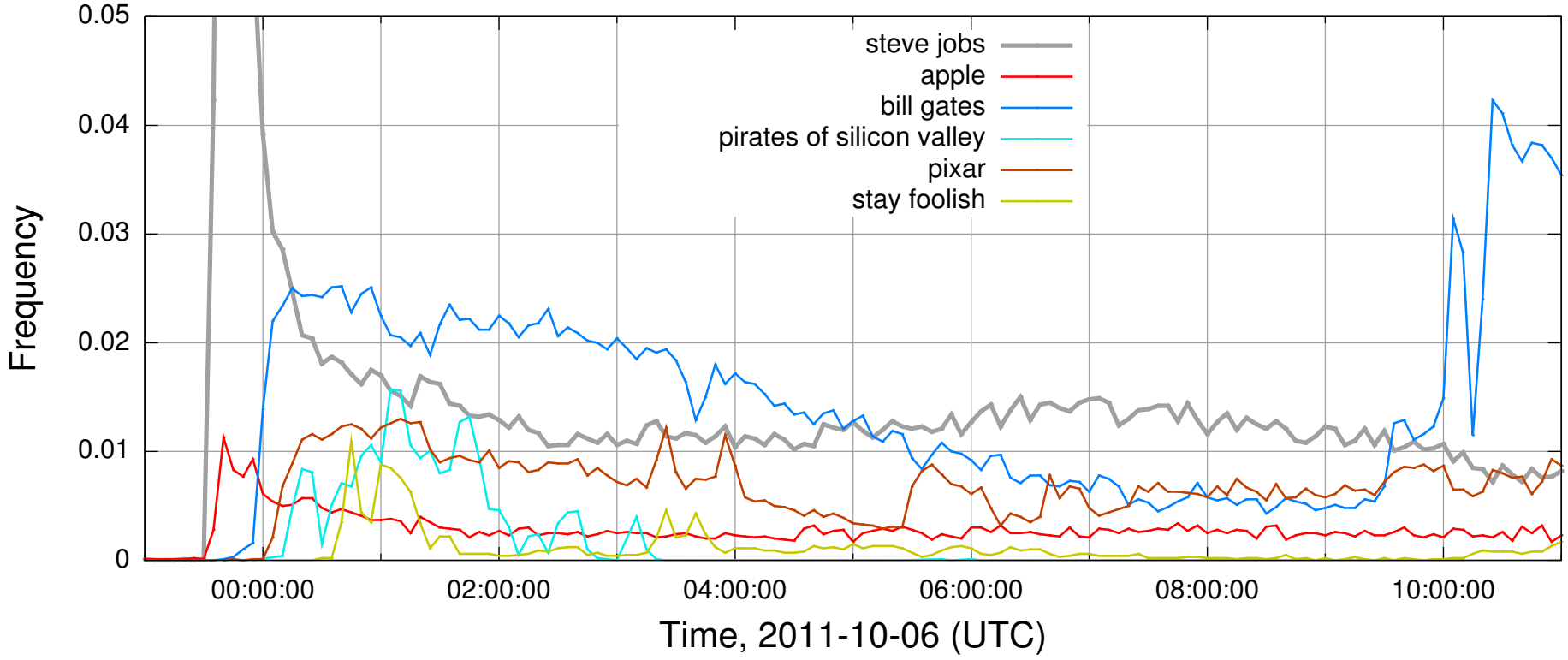

users
Frontend
Backend
OLTP database
ETL
(Extract, Transform, and Load)
Data Warehouse
BI tools
analysts
My data is a day old…
Meh.



Twitter's data warehousing architecture


Home
Connect
Discover
Me
sochi
#Sochi2014
#SochiProblems
Sochi
#SochiFail
Mishne et al. Fast Data in the Era of Big Data: Twitter's Real-Time Related Query Suggestion Architecture. SIGMOD 2013.
@lintool
TWEETS 1,647
FOLLOWING 253
FOLLOWERS 6,565
Compose new Tweet...
Who to follow · Refresh · View all
plotly @plotlygraphs
Follow
Promoted
Brad Anderson @boorad
Followed by Florian Leibert ...
Follow
Sheila Morrissey @sheilaMorr
Follow
Popular accounts · Find friends
Trends · Change
#Olympics
Promoted
Ukraine
#ConfessYourUnpopularOpinion
Venny
#PremioLoNuestro
Tweets
Promoted by Cloudera
Expand
Retweeted by Nitin Madnani
Clinton Paquin @clintonpaqui
View conversation
The Hill @thehill · 1h
View summary
Retweeted by Alex Feinberg
Popehat @Popehat · 10h
Expand
Reply Retweet Favorite More
The Hill @thehill · 1h
Boehner: I'd rather kill myself than raise the minimum wage trib.al/jZIkEus by @mollyhooper and @BobCusack
View summary
Reply Retweet Favorite More
CNN Breaking News @cnnbrk · 1h
Ukrainian Pres. says he has begun work on 3 key opposition demands: New elections, return to old constitution, formation of a unity gov's.
Expand
Reply Retweet Favorite More
Sochi 2014 @Sochi2014
Sochi Olympics 2014 @2014Sochi
Игры Сочи 2014 @sochi2014_ru
Sochi Problems @SochiProblem
NYT Olympics @SochiNYT
Sochi Problems @SochiProblems
Search all people for sochi

Case Studies: Steve Jobs passes away



Initial Implementation

Algorithm: Co-occurrences within query sessions
Implementation: Pig scripts over query logs on HDFS

Problem: Query suggestions were several hours old!

Why?
Log collection lag
Hadoop scheduling lag
Hadoop job latencies

We need real-time processing!
(stay tuned next time for actual solution)

real-time

vs.

online

vs.

streaming

What is a data stream?

Sequence of items:

Structured (e.g., tuples)

Ordered (implicitly or timestamped)

Arriving continuously at high volumes

Sometimes not possible to store entirely

Sometimes not possible to even examine all items

Applications

Network traffic monitoring
Datacenter telemetry monitoring
Sensor networks monitoring
Credit card fraud detection
Stock market analysis
Online mining of click streams
Monitoring social media streams

What exactly do you do?

"Standard" relational operations:

Select

Project

Transform (i.e., apply custom UDF)

Group by

Join

Aggregations

What else do you need to make this "work"?

Issues of Semantics

Group by… aggregate

When do you stop grouping and start aggregating?

Joining a stream and a static source

Simple lookup

Joining two streams

How long do you wait for the join key in the other stream?

Joining two streams, group by and aggregation

When do you stop joining?

What's the solution?

Windows

Windows restrict processing scope:

Windows based on ordering attributes (e.g., time)

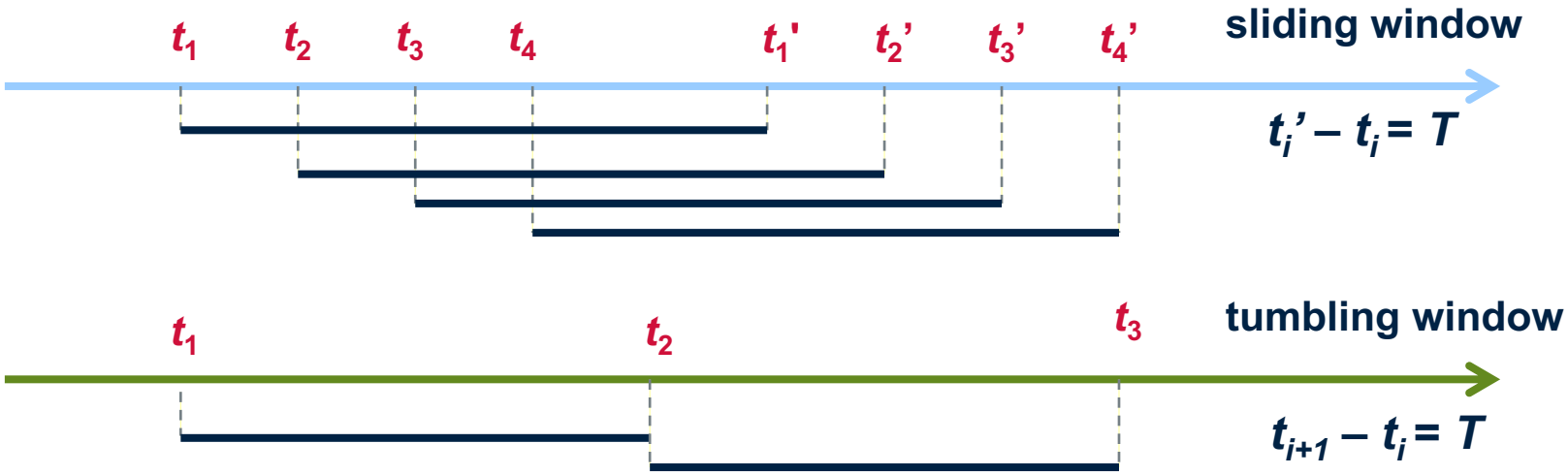
Windows based on item (record) counts

Windows based on explicit markers (e.g., punctuations)

Windows on Ordering Attributes

Assumes the existence of an attribute that defines the order of stream elements (e.g., time)

Let *T* be the window size in units of the ordering attribute



Windows on Counts

Window of size N elements (sliding, tumbling) over the stream

Windows from “Punctuations”

Application-inserted “end-of-processing”

Example: stream of actions… “end of user session”

Properties

Advantage: application-controlled semantics

Disadvantage: unpredictable window size (too large or too small)

Streams Processing Challenges

Inherent challenges

Latency requirements

Space bounds

System challenges

Load balancing

Unreliable and out-of-order message delivery

Fault-tolerance

Consistency semantics (at most once, exactly once, at least once)

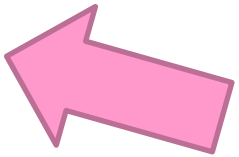
Source: Wikipedia (Forge)

Streams Processing Challenges

Inherent challenges

Latency requirements

Space bounds



System challenges

Load balancing

Unreliable and out-of-order message delivery

Fault-tolerance

Consistency semantics (at most once, exactly once, at least once)

Space: The Final Frontier

Data streams are potentially infinite

Unbounded space!

How do we get around this?

Throw old data away

Accept some approximation

General techniques

Sampling

Hashing

Reservoir Sampling

Task: select *s* elements from a
stream of size *N* with uniform probability

N can be very very large

We might not even know what *N* is! (infinite stream)

Solution: Reservoir sampling

Store first *s* elements

For the *k*-th element thereafter, keep with probability *s*/*k*
(randomly discard an existing element)

Example: *s* = 10

Keep first 10 elements

11th element: keep with 10/11

12th element: keep with 10/12

…

Reservoir Sampling: How does it work?

Example: $s = 10$

Keep first 10 elements

11th element: keep with 10/11

If we decide to keep it: sampled uniformly by definition

probability existing item is discarded: $10/11 \times 1/10 = 1/11$

probability existing item survives: 10/11

General case: at the $(k + 1)$th element

Probability of selecting each item up until now is s/k

Probability existing item is discarded: $s/(k+1) \times 1/s = 1/(k + 1)$

Probability existing item survives: $k/(k + 1)$

Probability each item survives to $(k + 1)$th round:

$(s/k) \times k/(k + 1) = s/(k + 1)$

Hashing for Three Common Tasks

Cardinality estimation

What's the cardinality of set *S*?

How many unique visitors to this page?

HashSet HLL counter

Set membership

Is *x* a member of set *S*?

Has this user seen this ad before?

HashSet Bloom Filter

Frequency estimation

How many times have we observed *x*?

How many queries has this user issued?

HashMap CMS

HyperLogLog Counter

Task: cardinality estimation of set

size() → number of unique elements in the set

Observation: hash each item and examine the hash code

On expectation, 1/2 of the hash codes will start with 1

On expectation, 1/4 of the hash codes will start with 01

On expectation, 1/8 of the hash codes will start with 001

On expectation, 1/16 of the hash codes will start with 0001

…

How do we take advantage of this observation?

Bloom Filters

Task: keep track of set membership

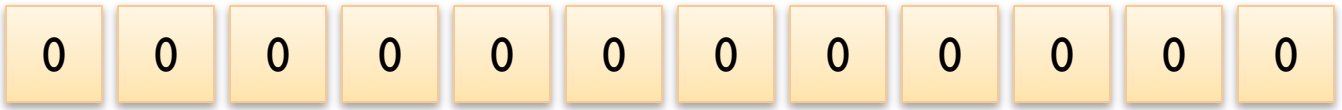
put(x) → insert x into the set

contains(x) → yes if x is a member of the set

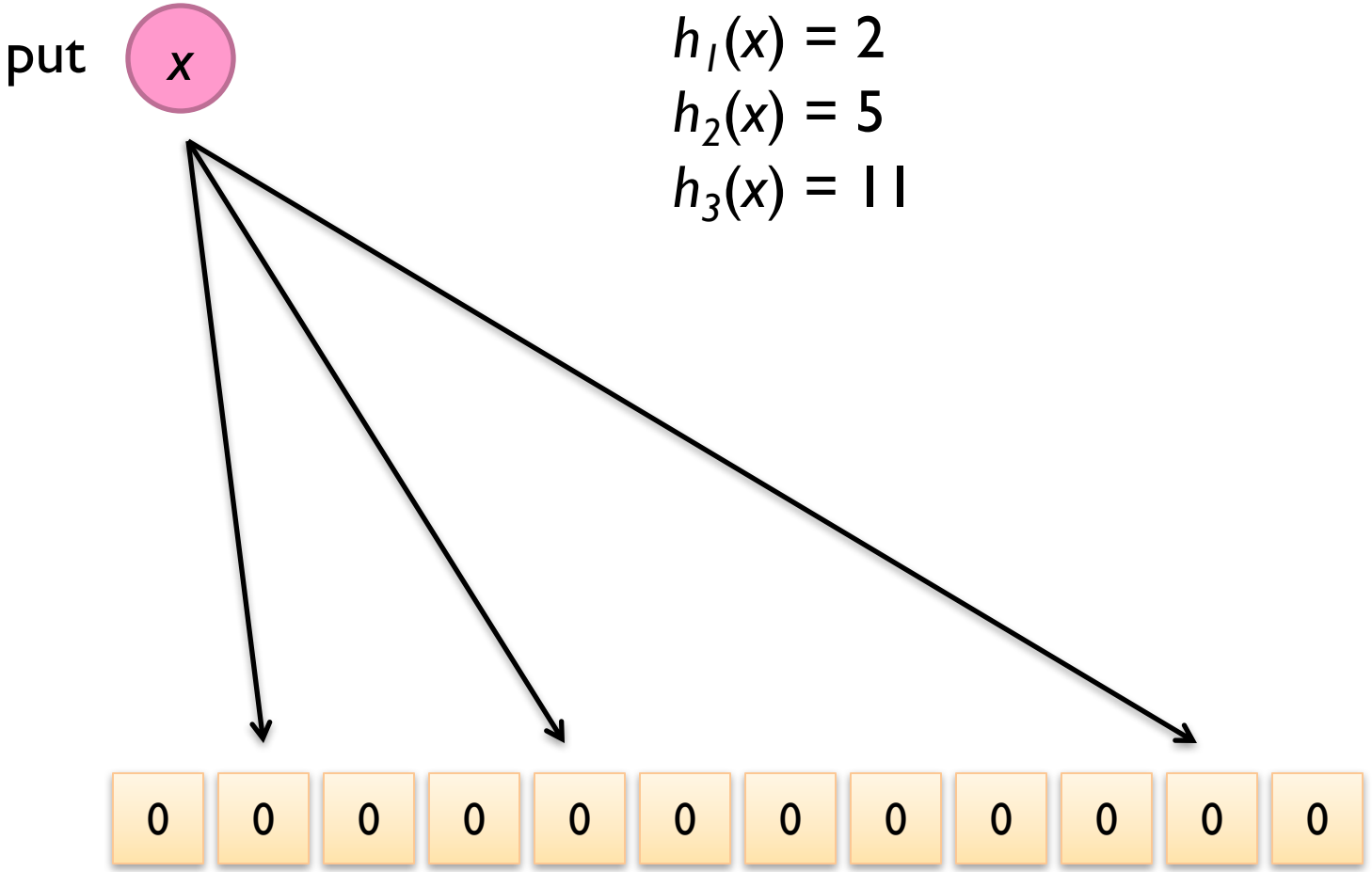
Components

m-bit bit vector

k hash functions: h_1 … h_k

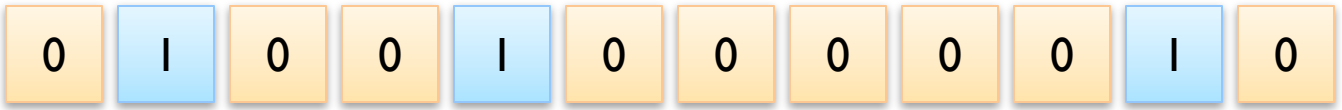


Bloom Filters: put

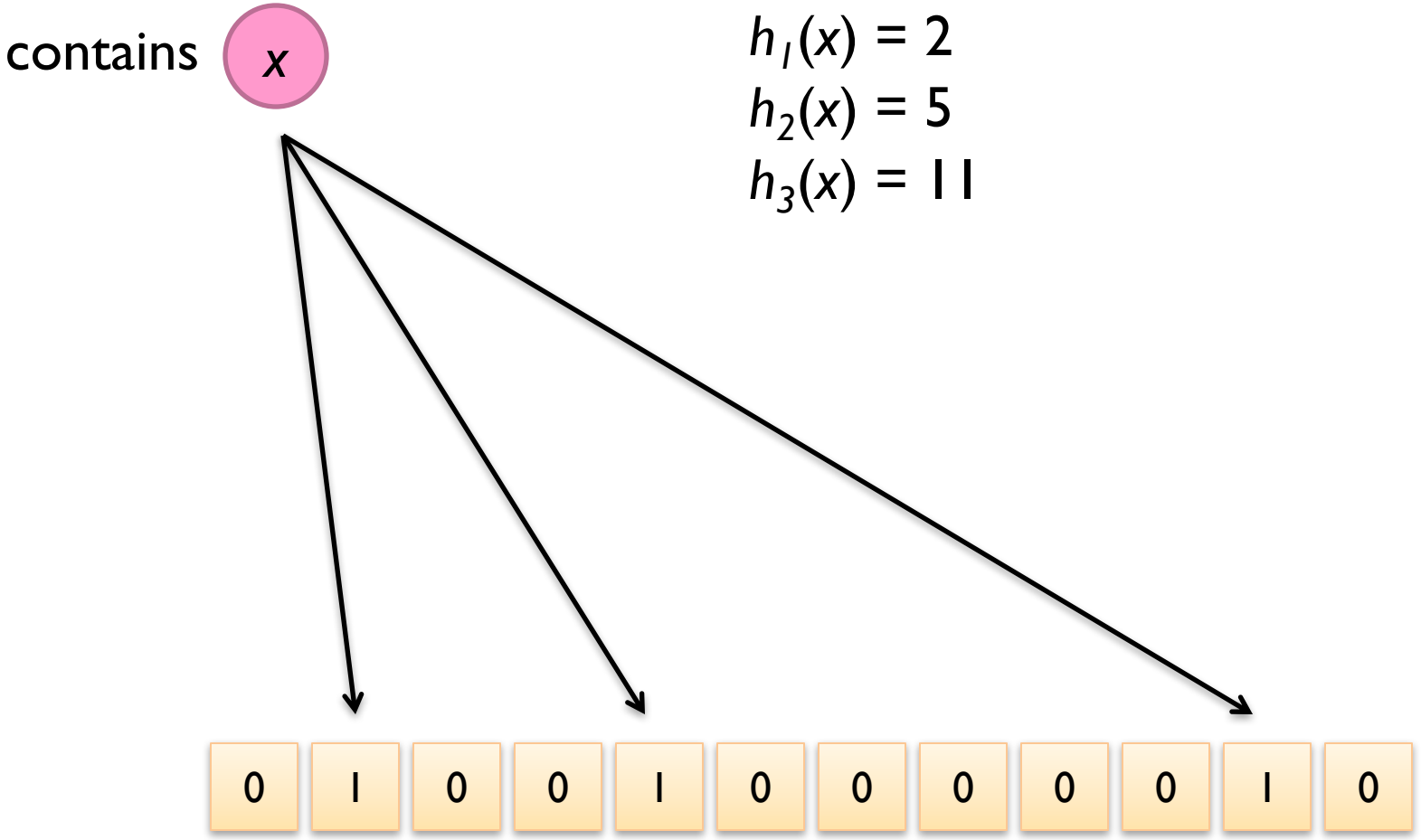


Bloom Filters: put

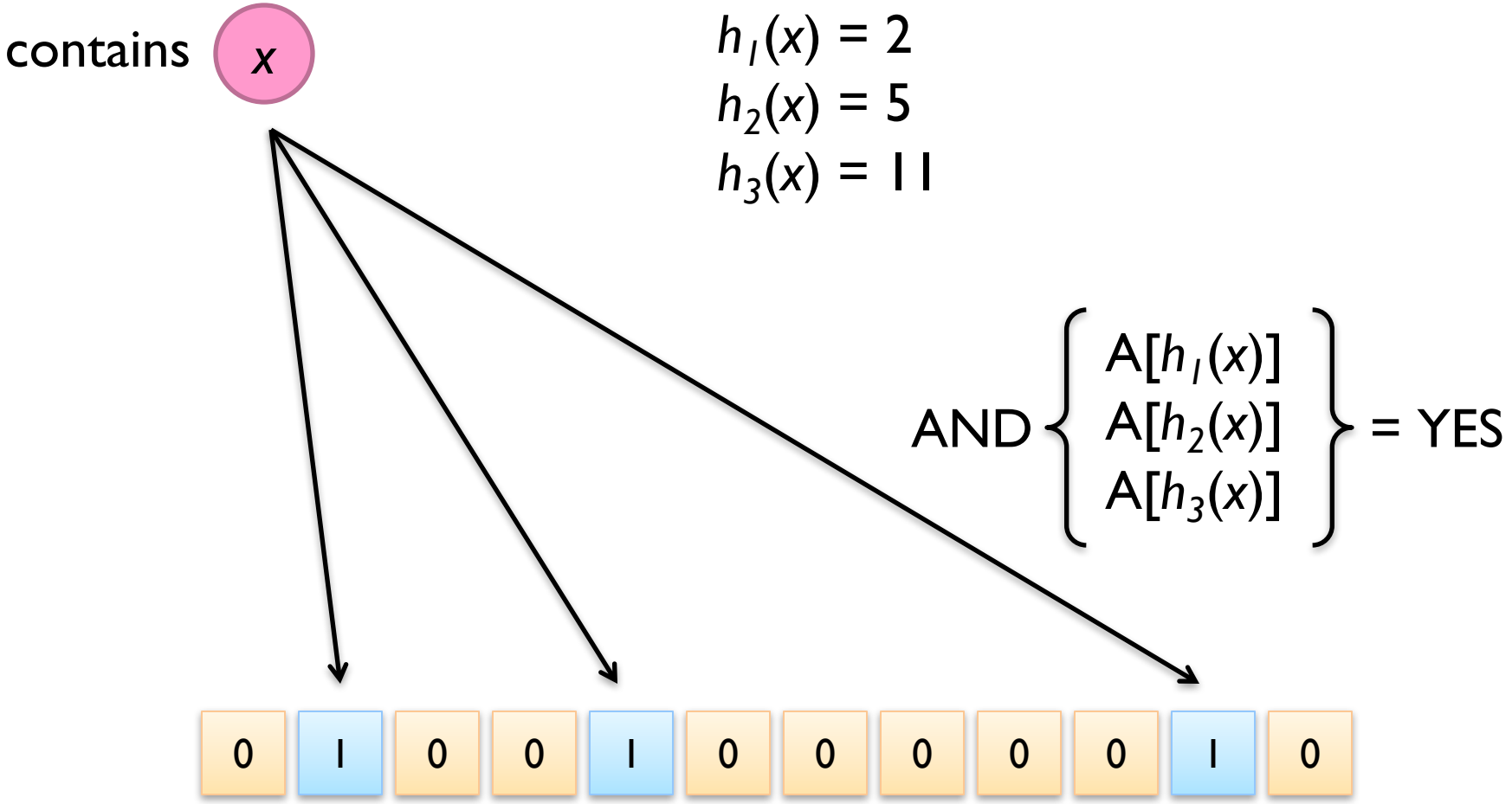
put *x*



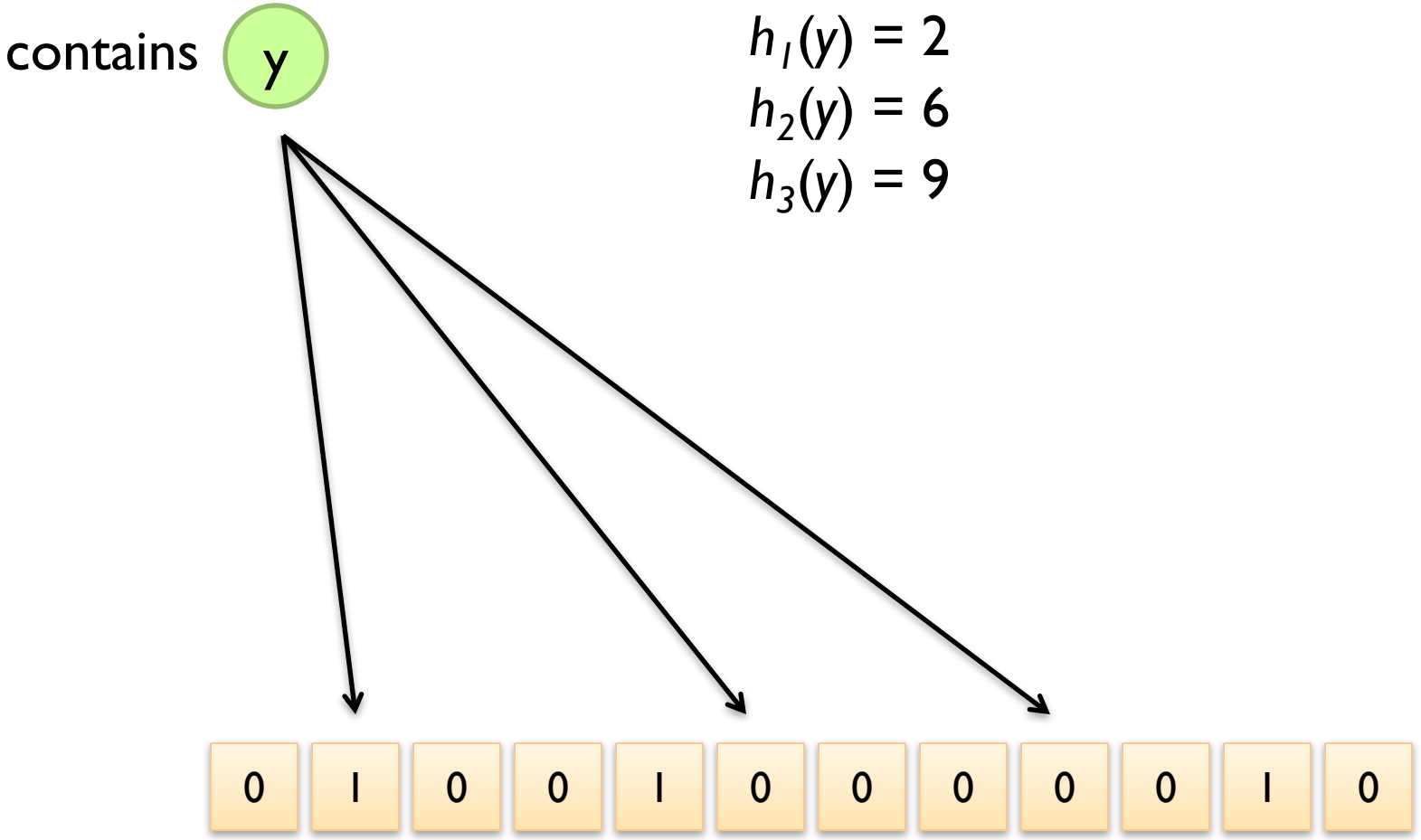
Bloom Filters: contains



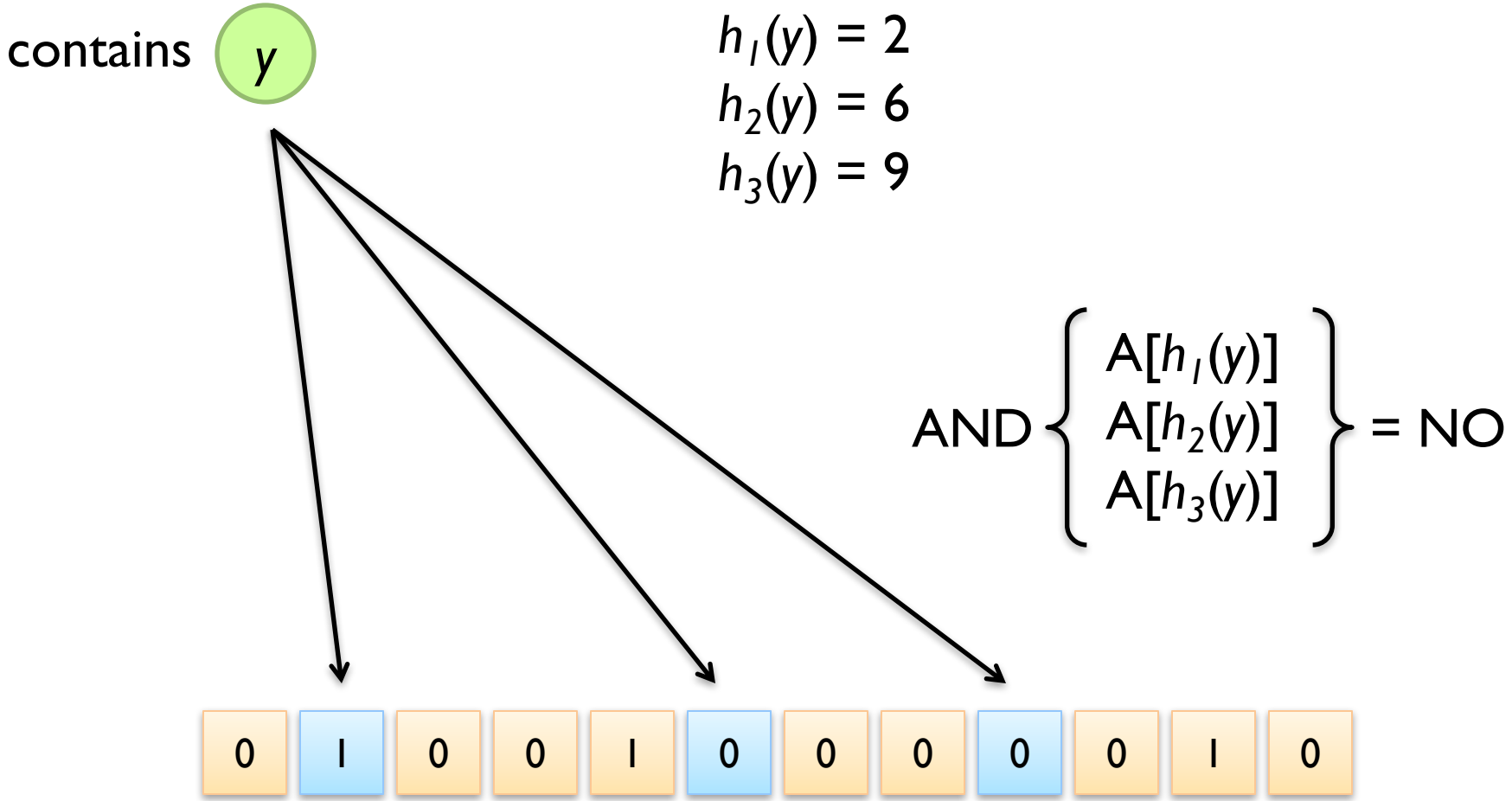
Bloom Filters: contains



Bloom Filters: contains



Bloom Filters: contains



What's going on here?

Bloom Filters

Error properties: contains(*x*)

False positives possible

No false negatives

Usage

Constraints: capacity, error probability

Tunable parameters: size of bit vector *m*, number of hash functions *k*

Count-Min Sketches

Task: frequency estimation

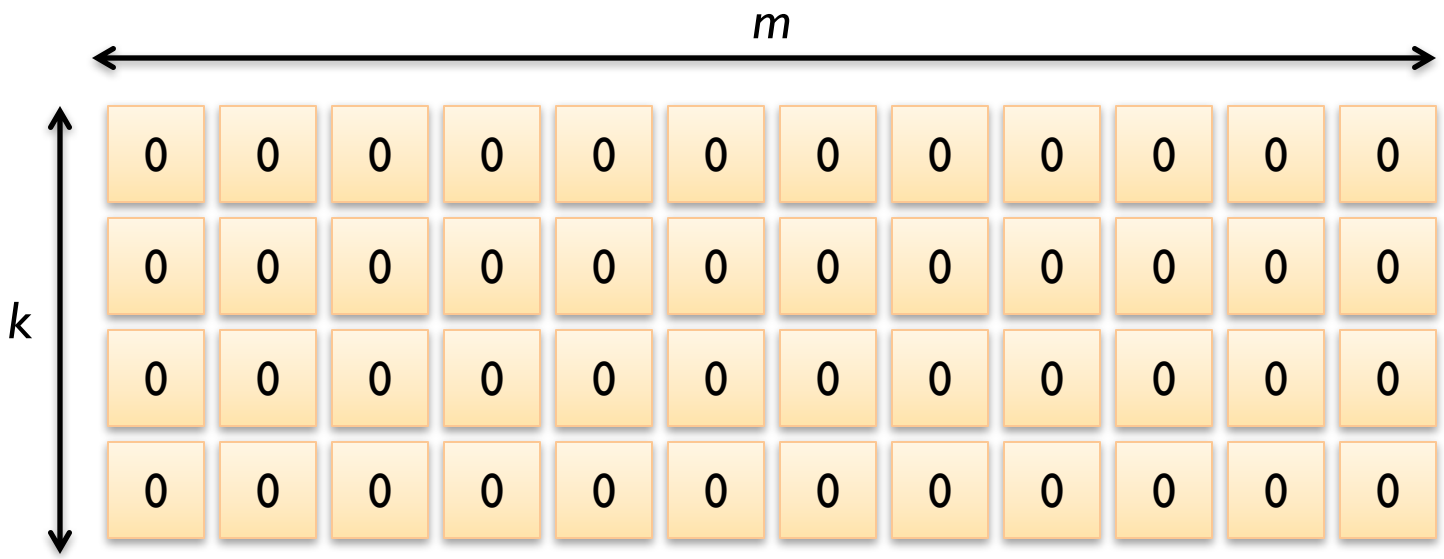
put(x) → increment count of x by one

get(x) → returns the frequency of x

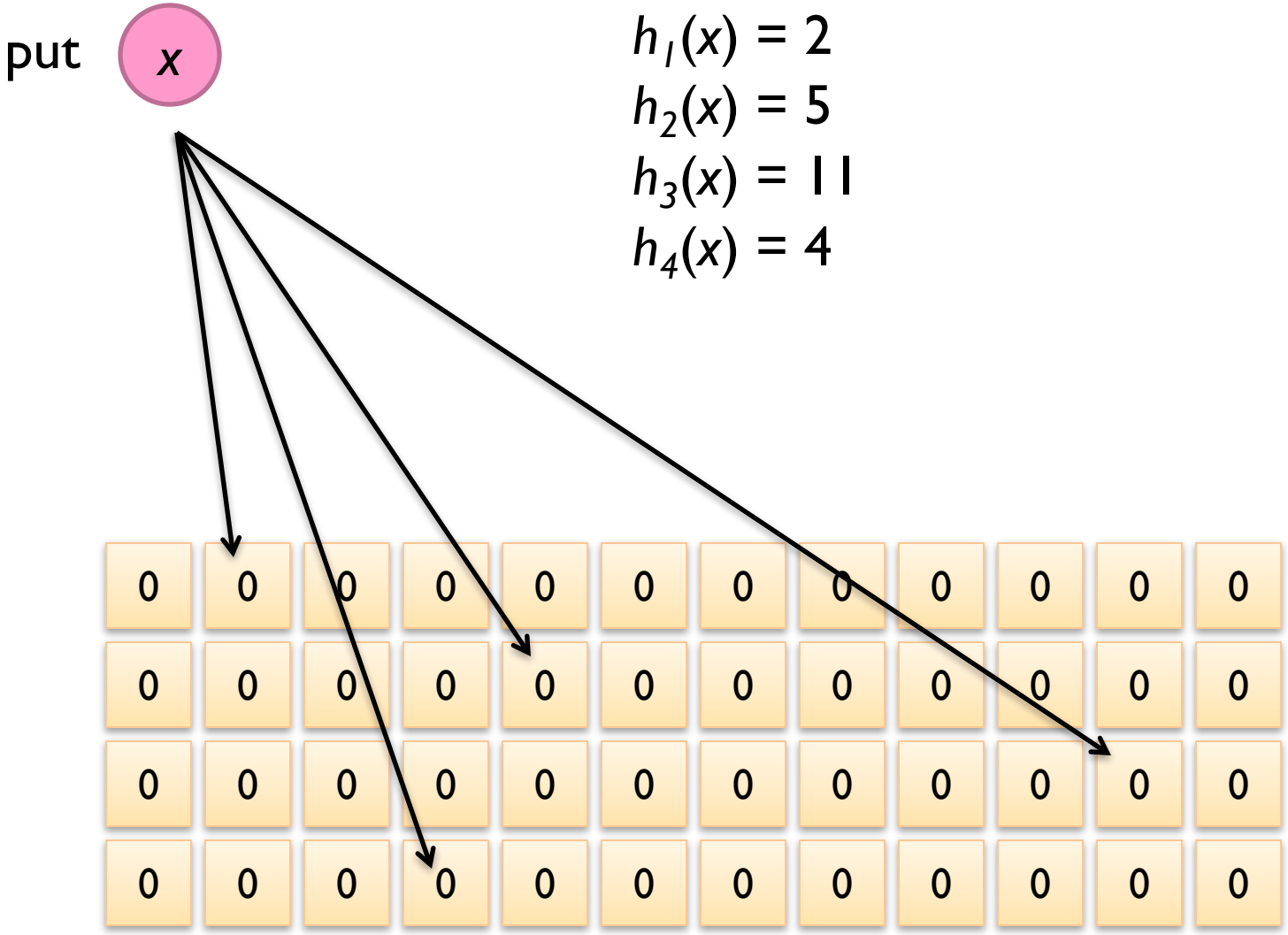
Components

m by k array of counters

k hash functions: h_1 … h_k

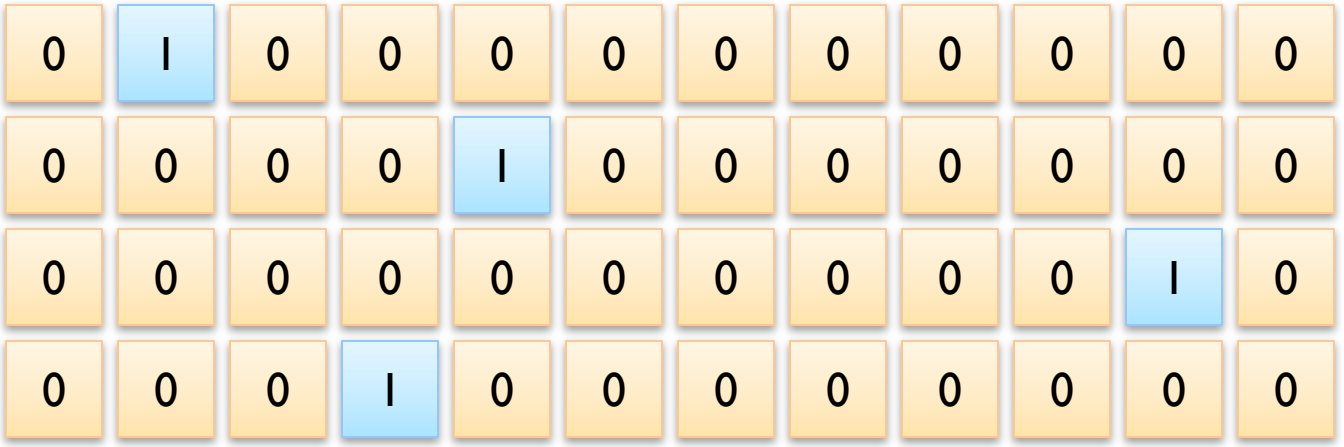


Count-Min Sketches: put

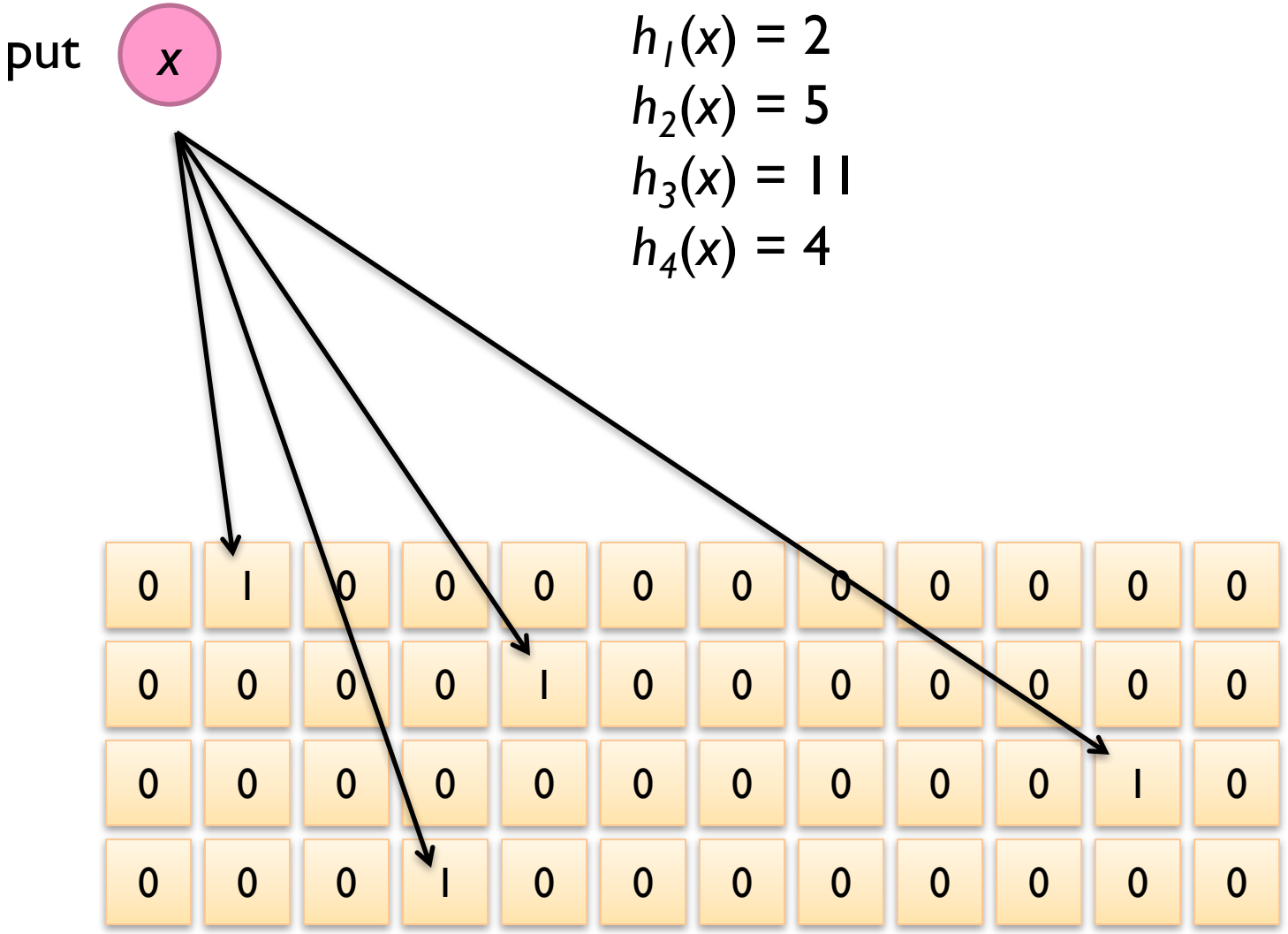


Count-Min Sketches: put

put *x*

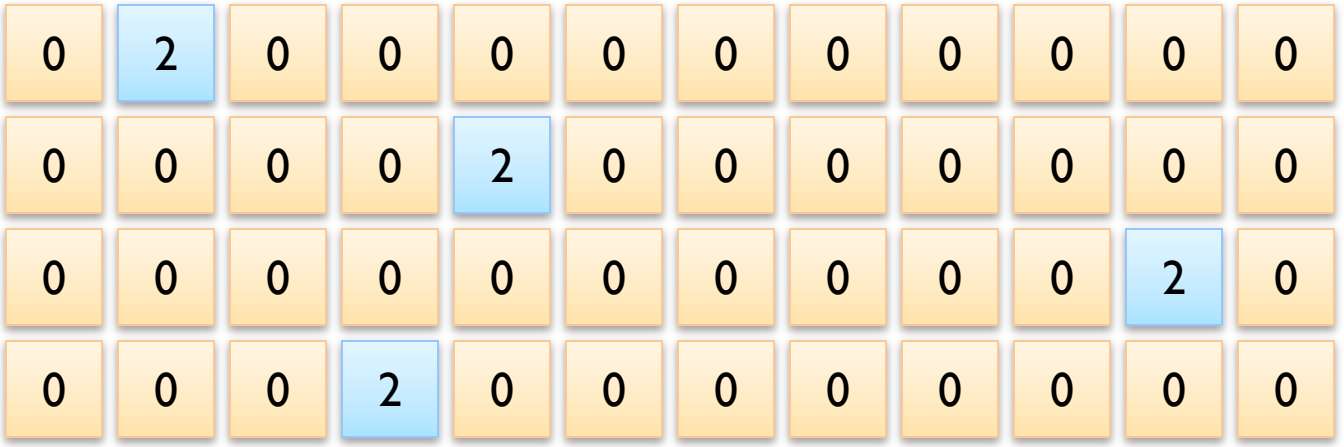


Count-Min Sketches: put

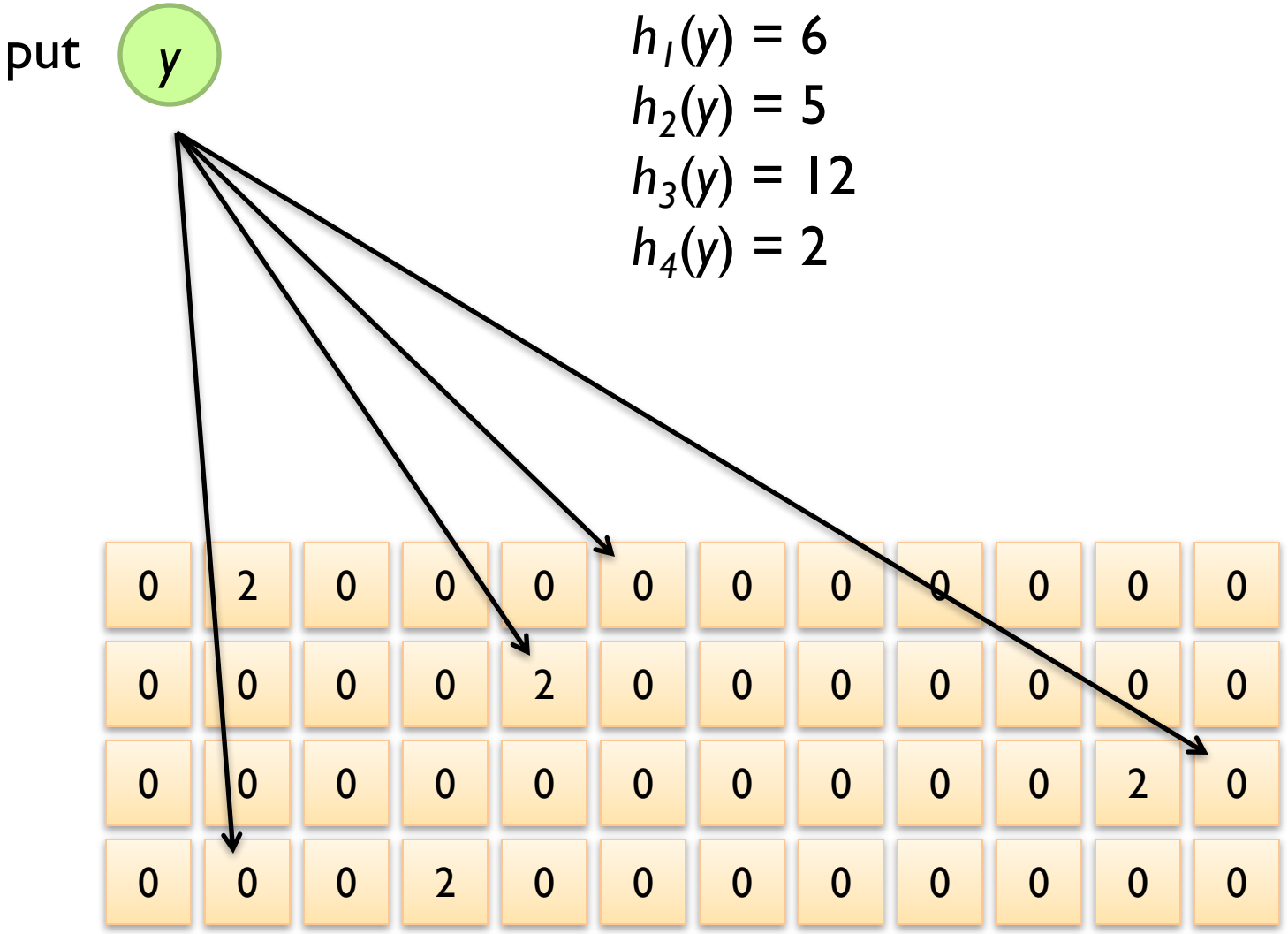


Count-Min Sketches: put

put *x*

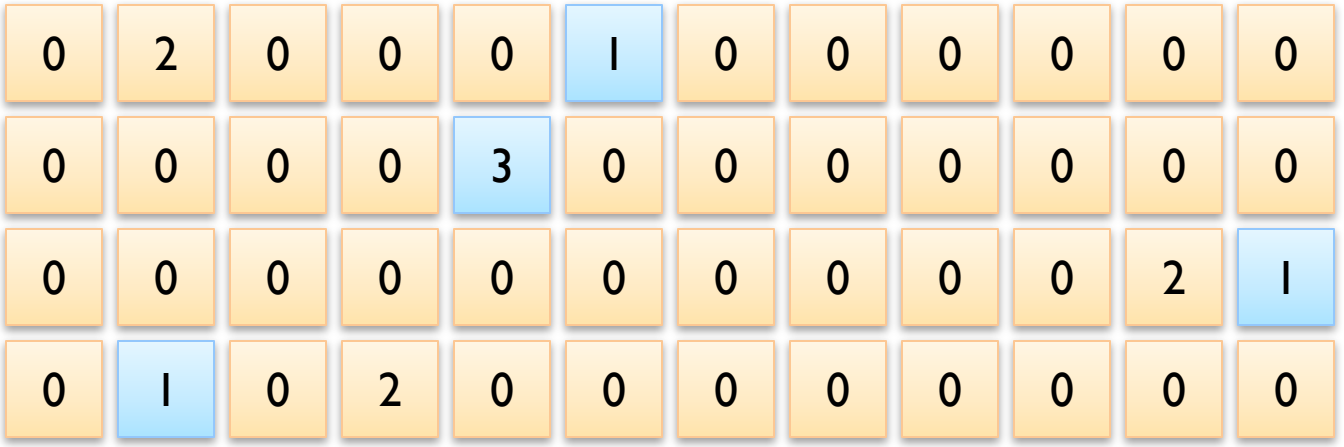


Count-Min Sketches: put

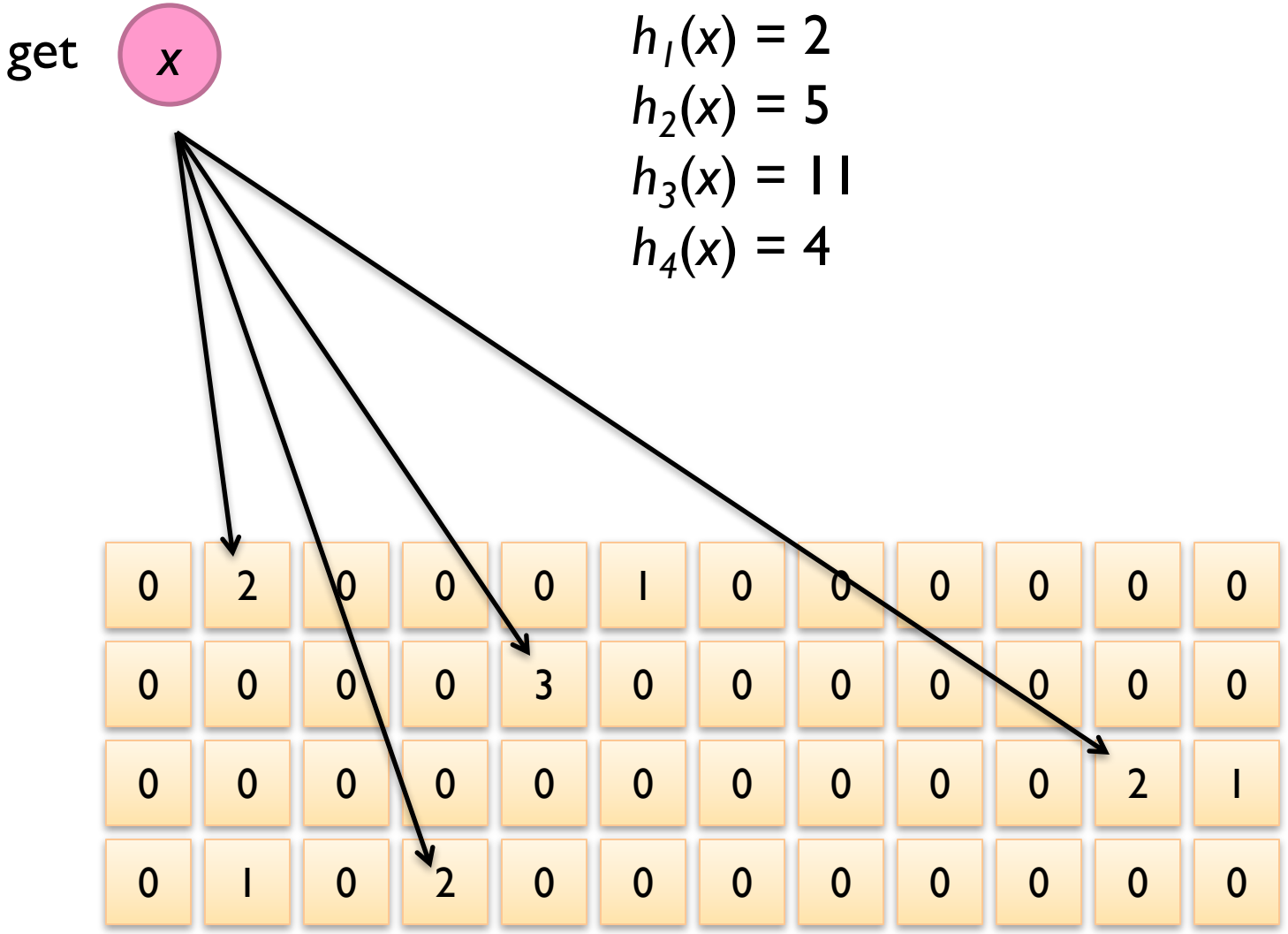


Count-Min Sketches: put

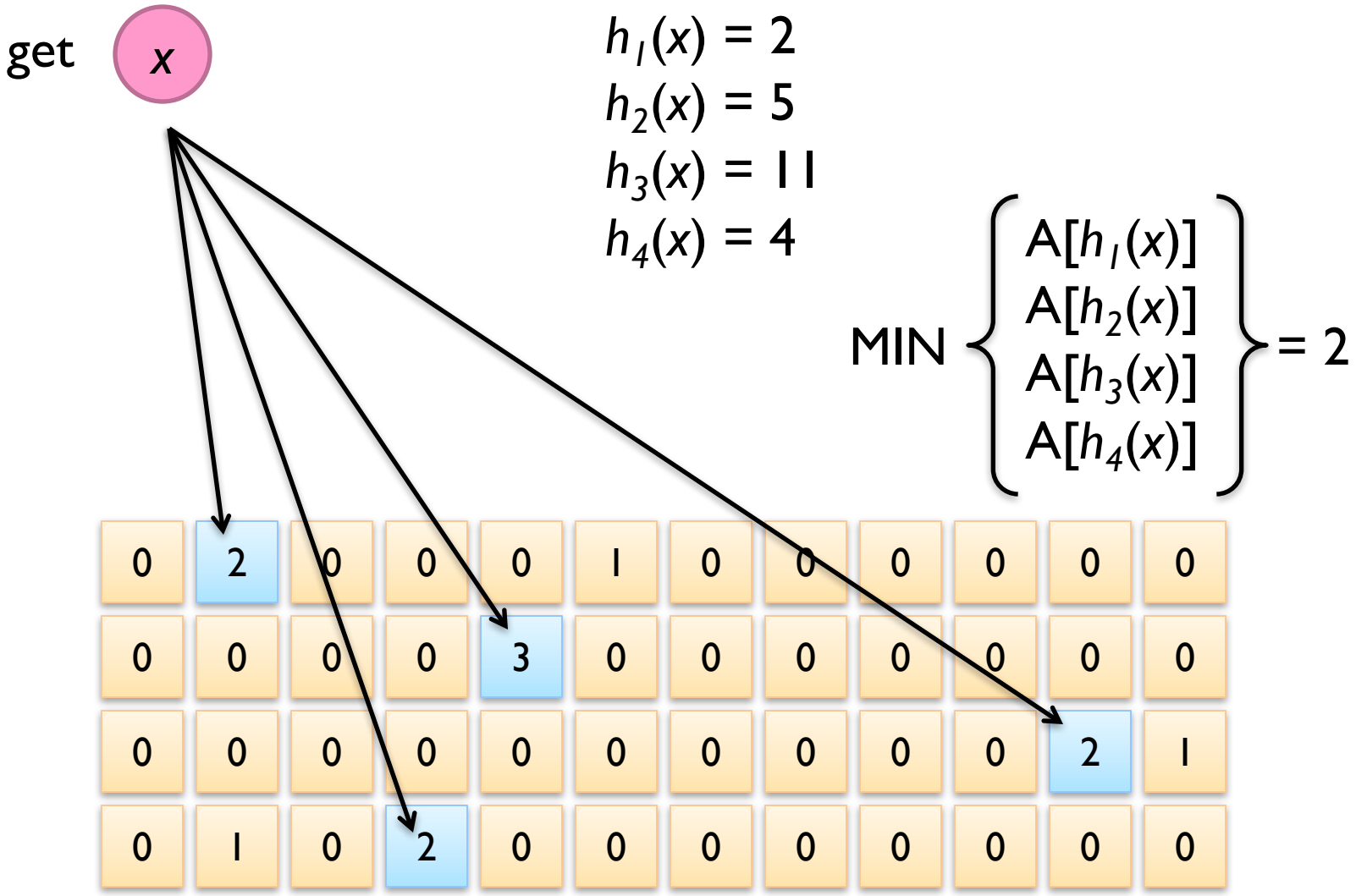
put y



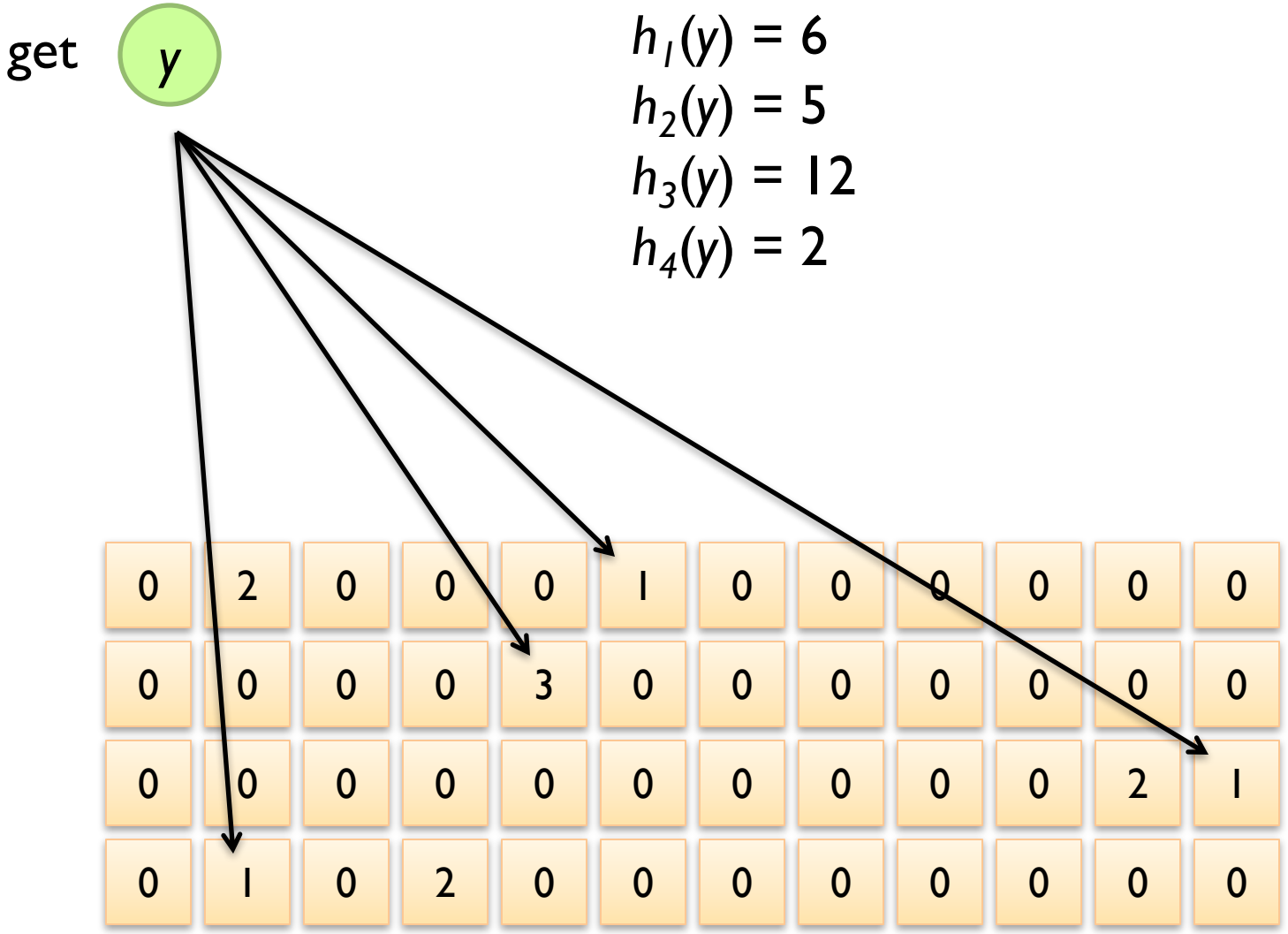
Count-Min Sketches: get



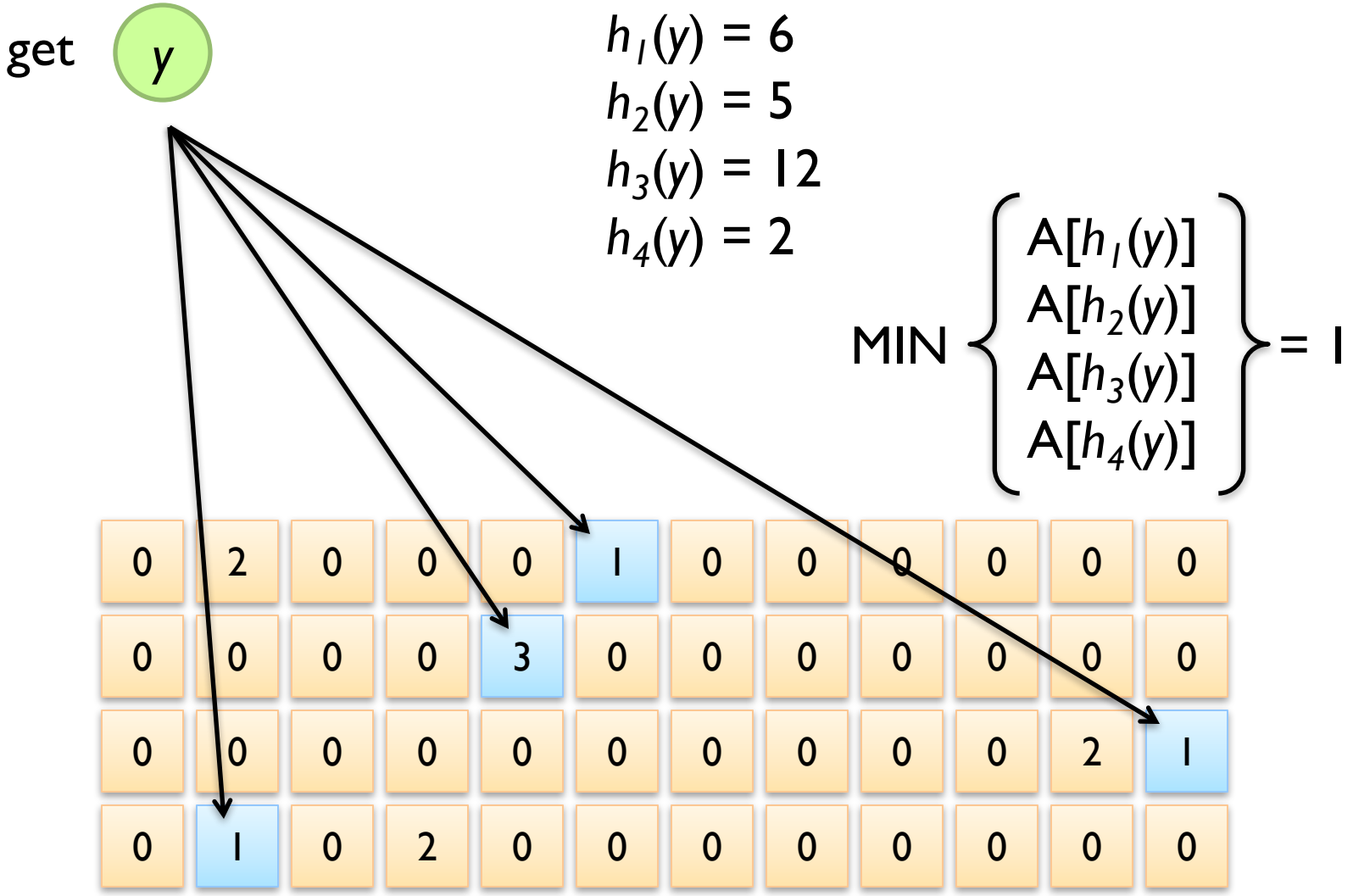
Count-Min Sketches: get



Count-Min Sketches: get



Count-Min Sketches: get



Count-Min Sketches

Error properties: get(*x*)

Reasonable estimation of heavy-hitters

Frequent over-estimation of tail

Usage

Constraints: number of distinct events, distribution of events, error bounds

Tunable parameters: number of counters *m* and hash functions *k*, size of counters

Hashing for Three Common Tasks

Cardinality estimation

What's the cardinality of set *S*?

How many unique visitors to this page?

HashSet HLL counter

Set membership

Is *x* a member of set *S*?

Has this user seen this ad before?

HashSet Bloom Filter

Frequency estimation

How many times have we observed *x*?

How many queries has this user issued?

HashMap CMS

Source: Wikipedia (River)

Source: Wikipedia (Japanese rock garden)