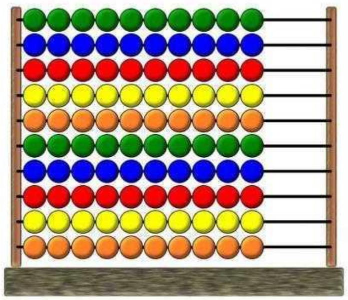
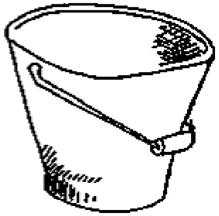
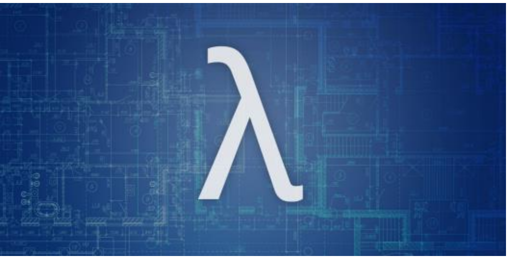


www.micetf.fr



Lambda architectures
Probabilistic Data Structures



λ vs ϰ

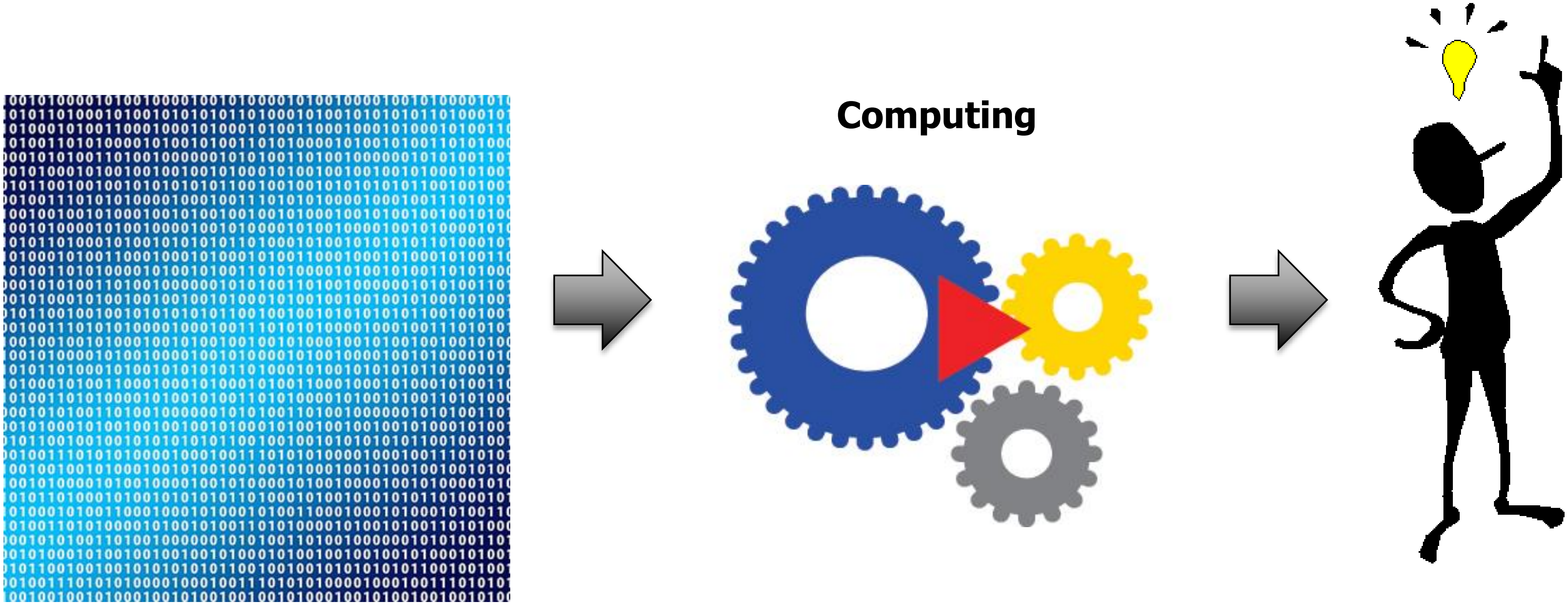
Big Data Definition

- Big Data is a collection of large datasets that cannot be processed using traditional computing techniques.

- Any data that can challenge our current technology

Big Data

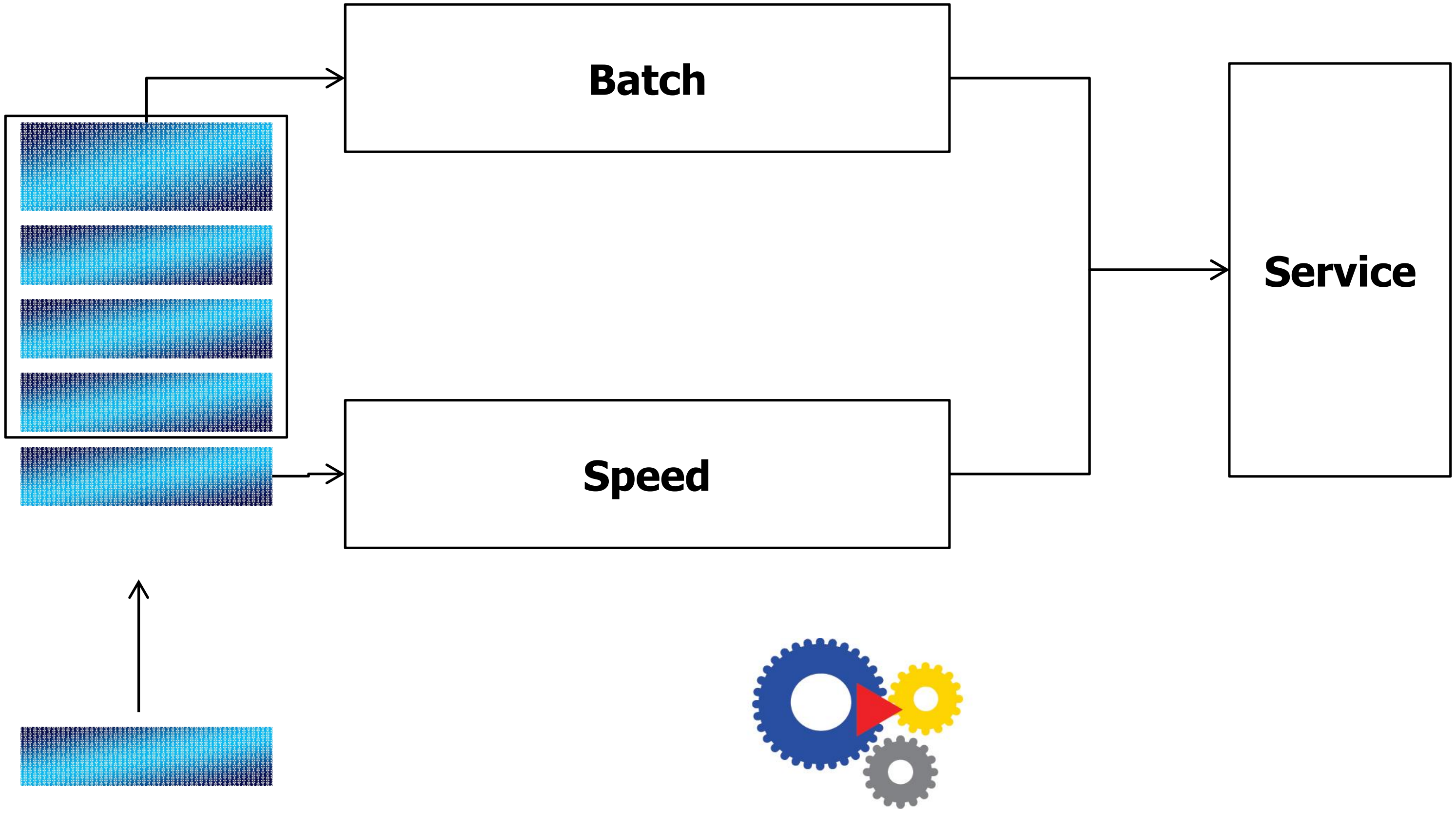
Big data is about the Insight that we want to extract from information.



Data

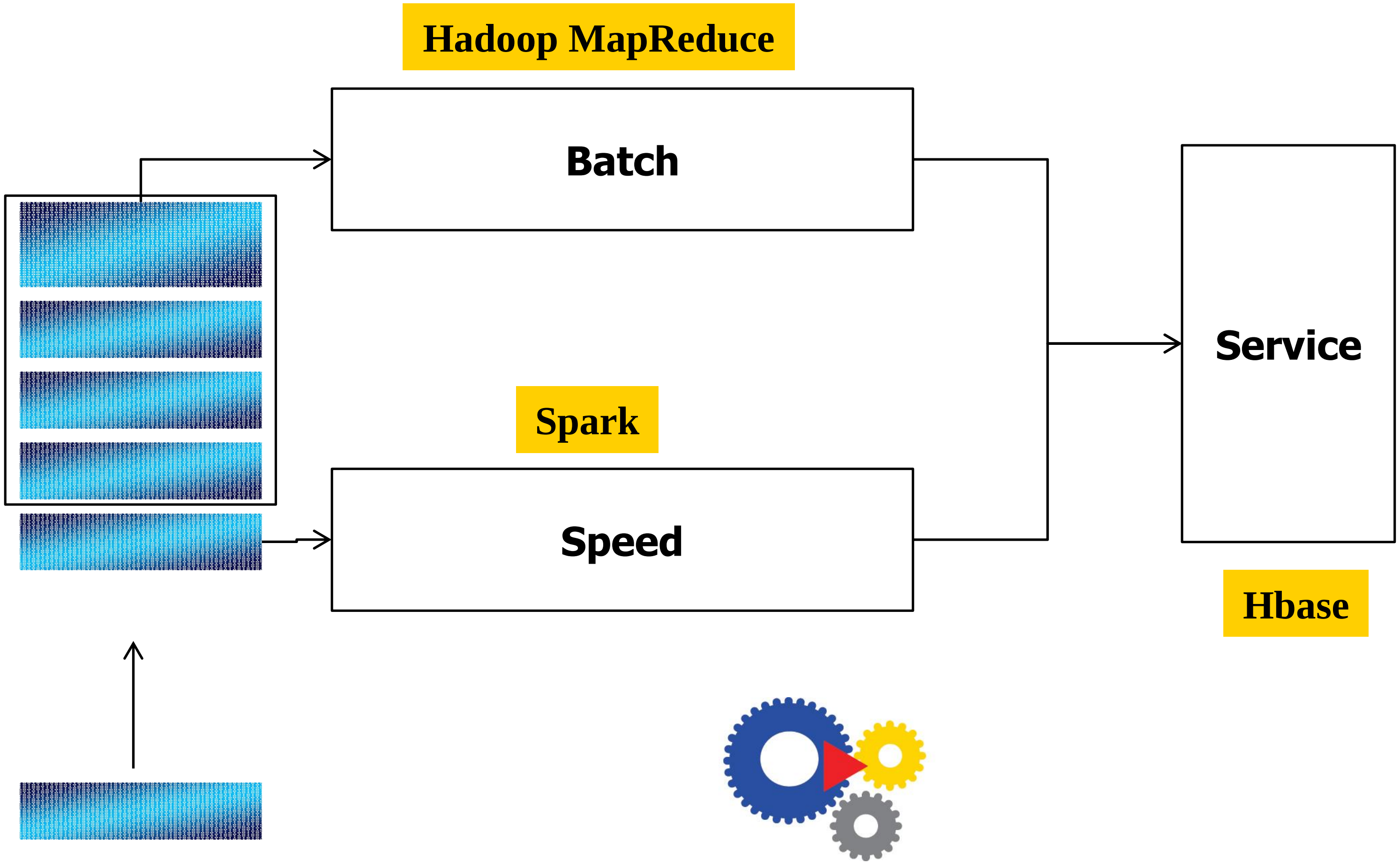
Computing

Lambda Architecture



Computing

Lambda Architecture



Hadoop MapReduce

Batch

Spark

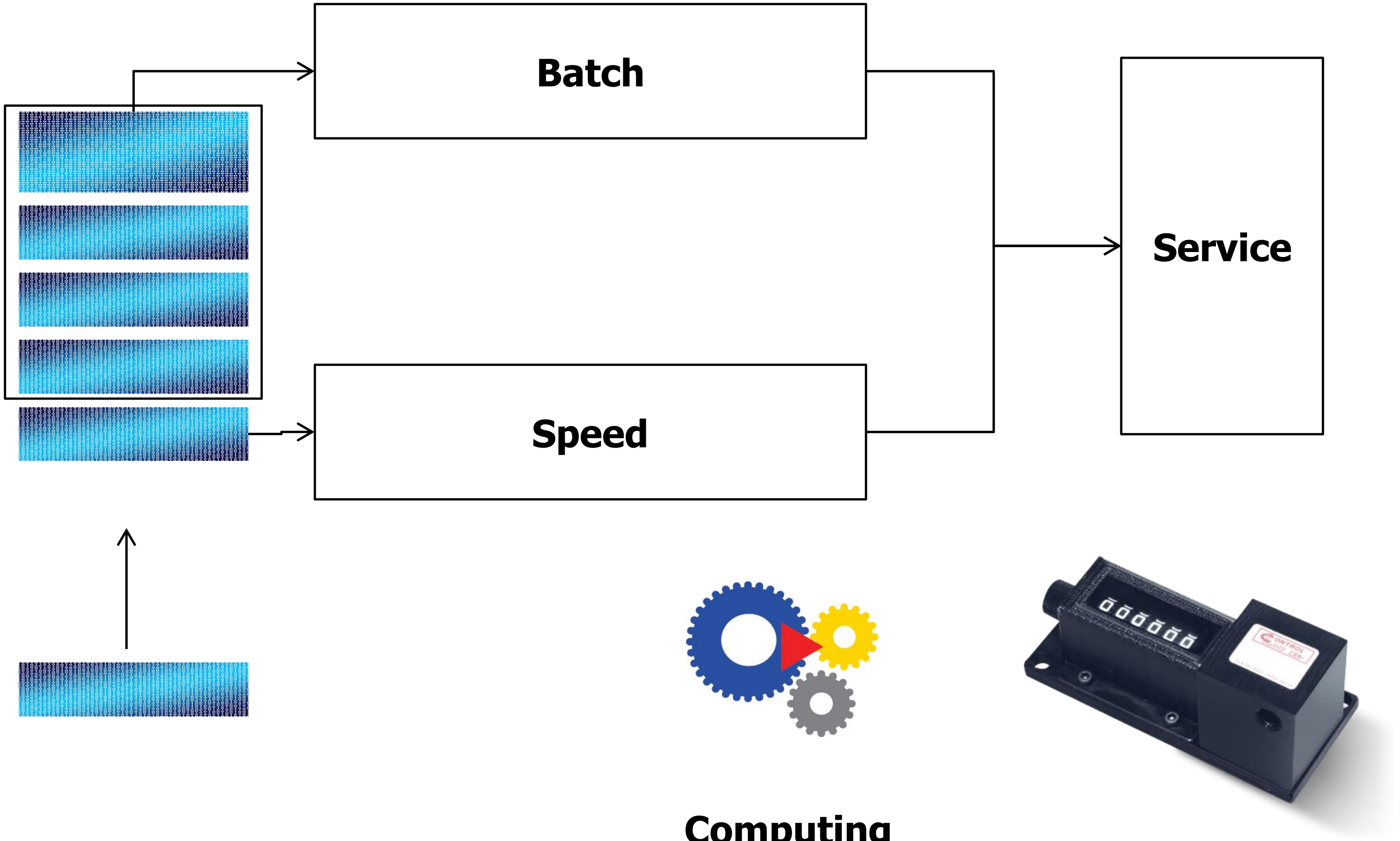
Speed

Service

Hbase

Computing

Lambda Architecture



Batch

Speed

Service

Computing

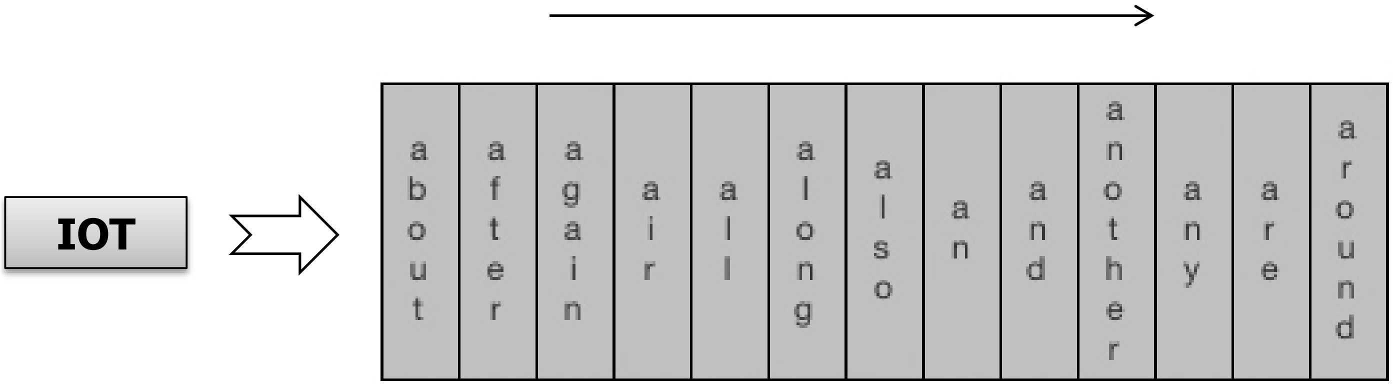
Lambda architecture

- 1) Speed/event layer :
  - Stream processing architecture.
  - The data are appended to the master data periodically over a time window.
  - Real time views
- 2) Batch layer :
  - Manages the master dataset:
    - Immutable, append-only raw data and pre-computing batch views.
  - Batch views
- 3) Serving layer:
  - Merges batch and real-time views.

Lambda architecture

- Offline or batch processing : insight
  - No time constraint : no data structures
  - Data are stored on disk
  - Data Lakes
- Streaming data : near real-time processing
  - Collecting analytics
  - Monitoring
  - Alerting,
  - Updating search indexes
  - Caches.

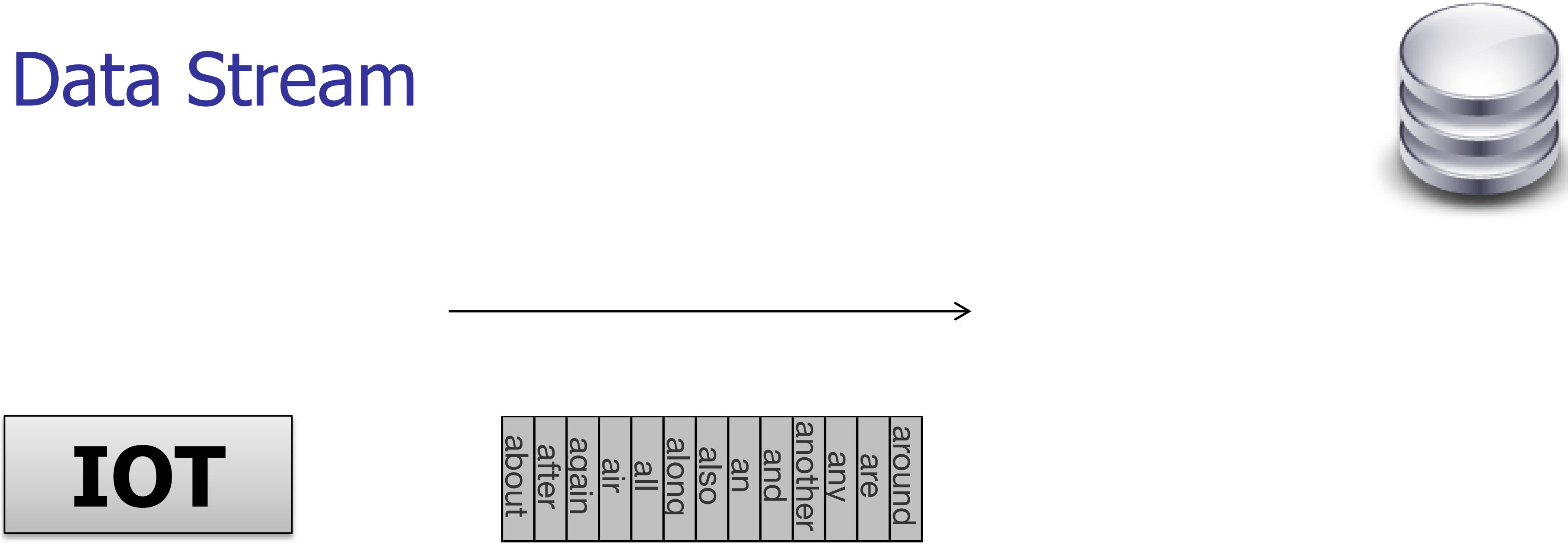
Data Stream



IOT

about after again air all along also an and another any are around

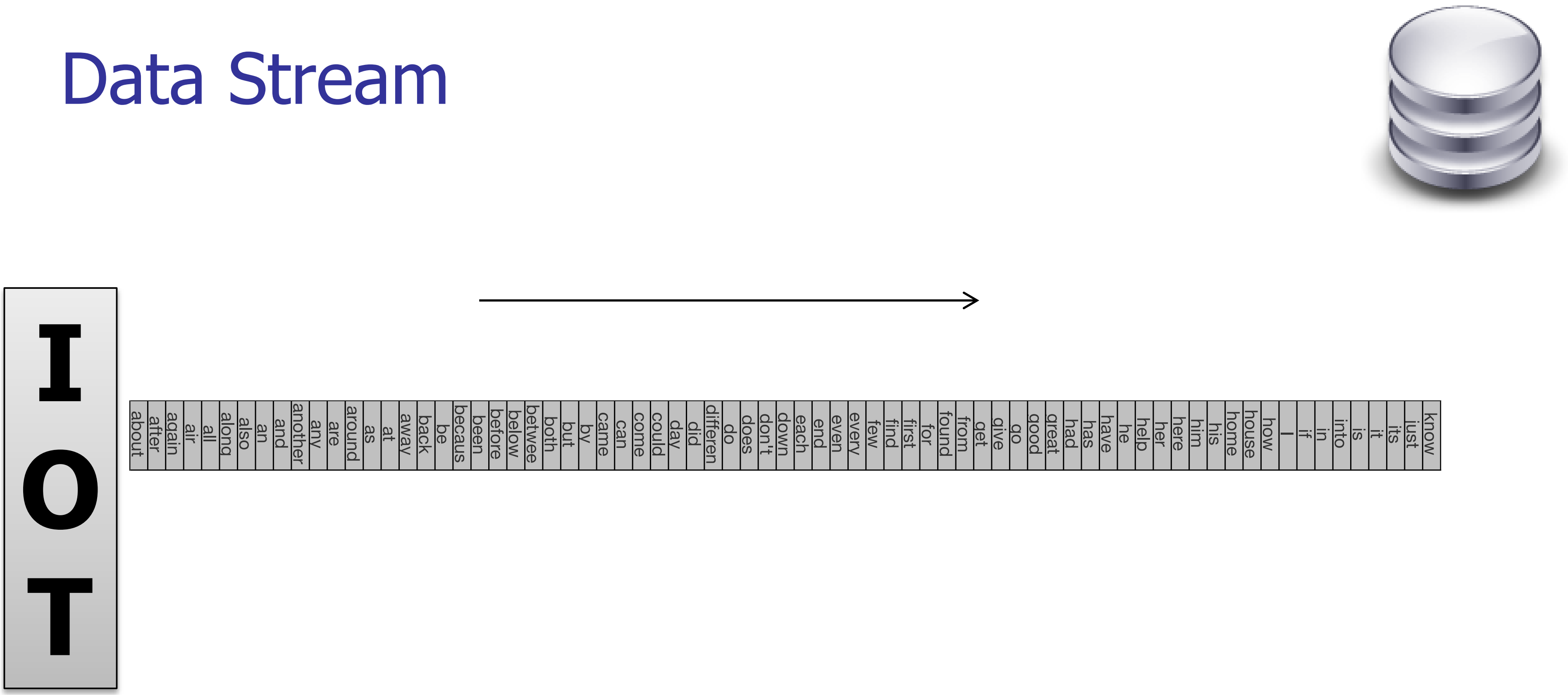
Data Stream



IOT

about after again air all along also an and another any are around

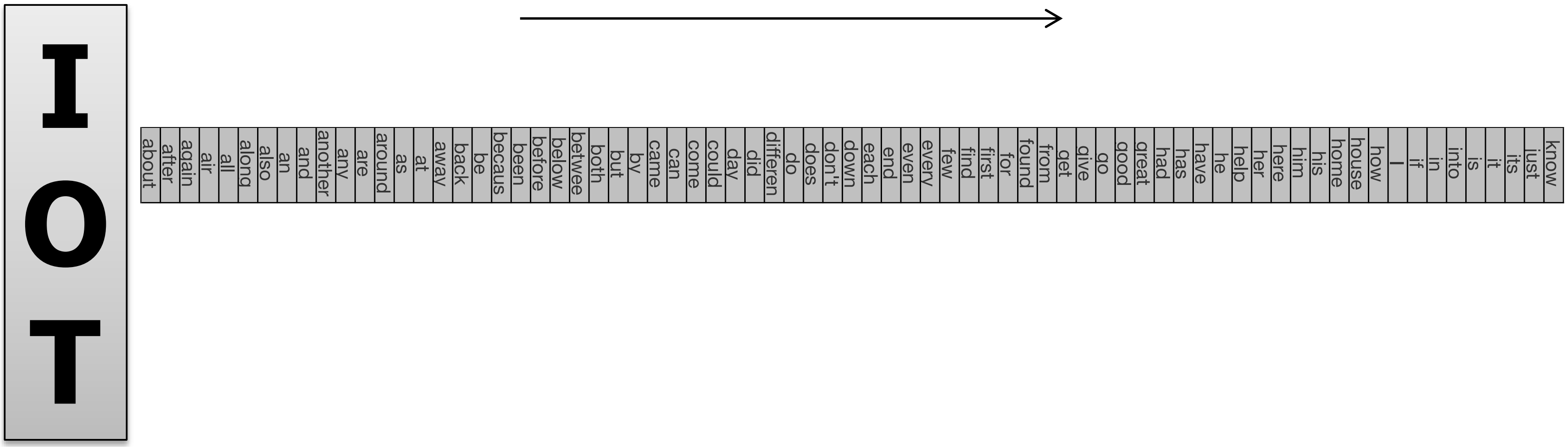
Data Stream



IOT

about after again air all along also an and another any are around as at away back be becaus been before below betwee both but by came can come could day did differen do does don't down each end even every few find first for found from get give go good great had has have he help her here him his home house how I if in into is it its just know

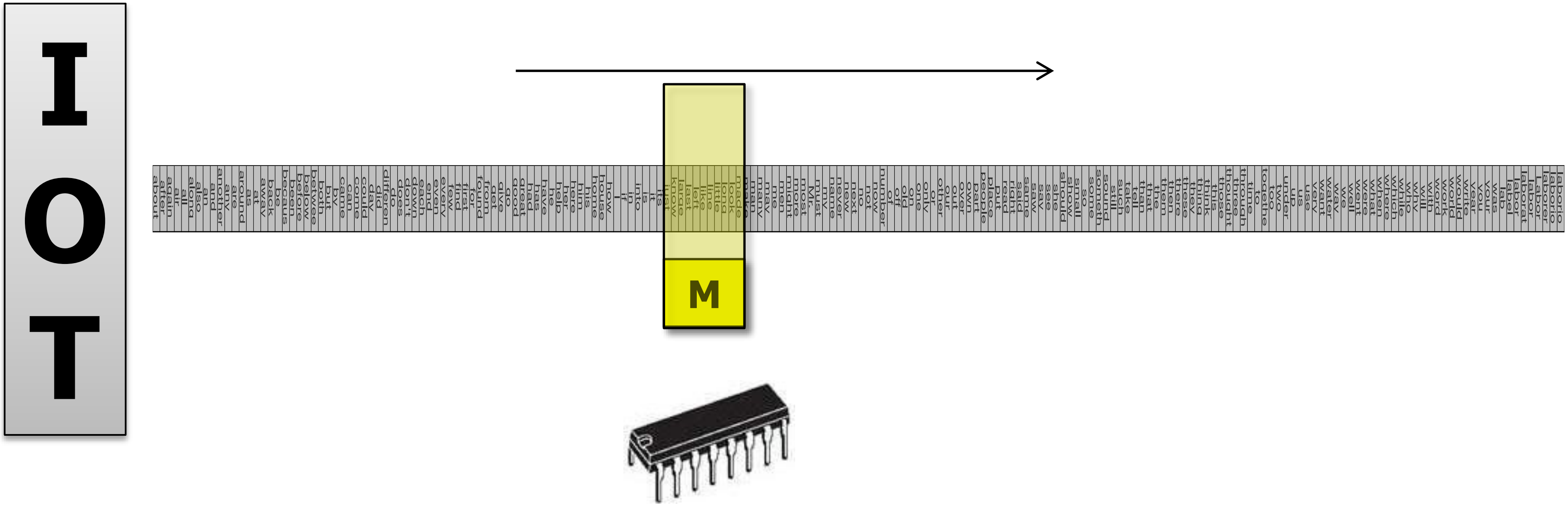
Data Stream



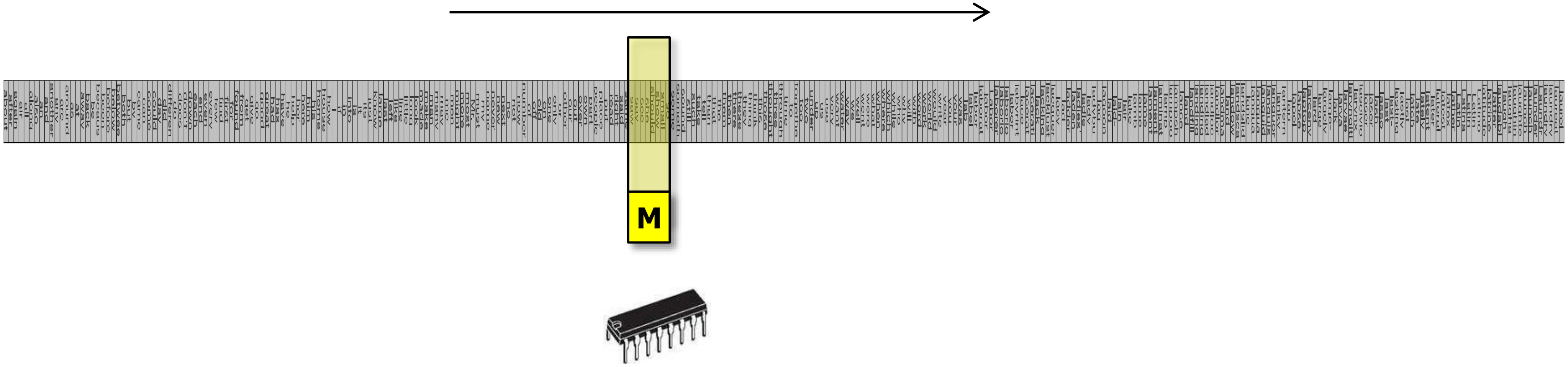
IOT

about after again air all along also an and another any are around as at away back be becaus been before below betwee both but by came can come could day did differen do does don't down each end even every few find first for found from get give go good great had has have he help her here him his home house how I if in into is it its just know

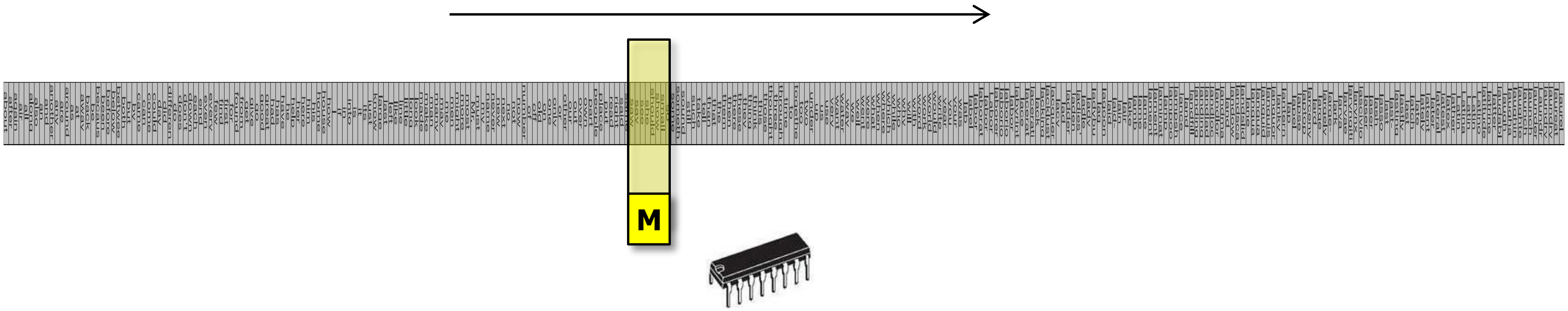
Data Stream


IOT
M
about
after
again
air
all
along
also
an
and
another
any
are
around
as
at
away
back
be
becaus
been
before
below
betwee
both
but
by
came
can
come
could
day
did
differen
do
does
don't
down
each
end
even
every
few
find
first
for
found
from
get
give
go
good
great
had
has
have
he
help
her
here
him
his
home
house
how
I
if
in
into
is
it
its
just
know
large
last
left
like
line
little
long
look
made
make
man
many
may
me
men
might
more
most
Mr.
must
my
name
never
new
next
no
not
now
number
of
off
old
on
one
only
or
other
our
out
over
own
part
people
place
put
read
right
said
same
saw
say
see
she
should
show
small
so
some
someth
sound
still
such
take
tell
than
that
the
them
then
there
these
they
thing
think
this
those
thought
three
through
time
to
togethe
too
two
under
up
us
use
very
want
water
way
we
well
went
were
what
when
where
which
while
who
why
will
with
word
work
world
would
write
year
you
your
was
lab
label
labor
laborat
labor
Labor
laborer
laborio
laborio

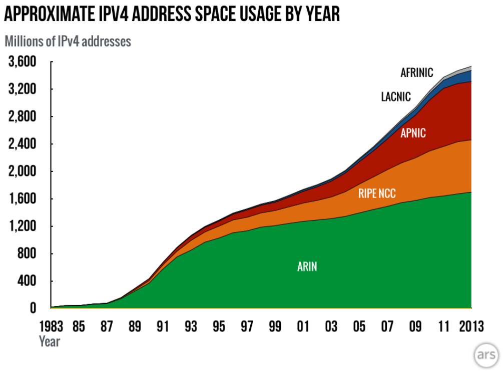
Data Stream



Data Stream



Who ?
How Many ?
What ?
Together ?
How many of them ?
Why ?



Speed/event layer

- Online, real-time processing
- Collecting real-time analytics
- In stream processing
- Large (Big) data streams :
  - Impractical to store that data in memory
  - Trade off to get real-time insight

- Stream Processing with Probabilistic counting
  - In-stream counters

Advantages Probabilistic Data Structures

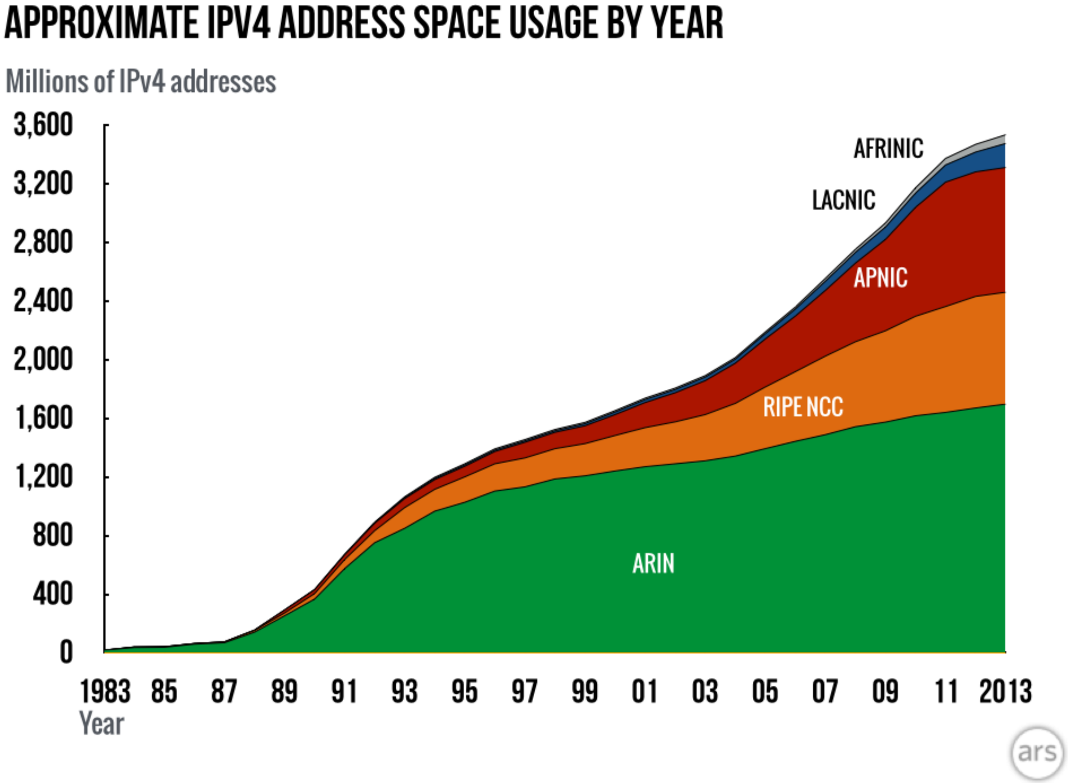
- Use small amount of memory
  - Can be controled
- Can be easily parallelizable
  - Hashes are independent
- Have constant query time
  - Not even amortized constant like in dictionary

Usage

- Computation Of Advanced Metrics :
  - Number Of Unique Visitor
  - Most Frequent Items
- Web Crawlers:
  - Process A Stream Of Urls And Documents Tp Produce Indexed Content.
- Websites:
  - Process A Stream Of Page Views And Update Various Counters And Gauges.
- Email:
  - Process A Stream Of Text And Produce A Filtered, Spam-free Inbox.

Exemple

- How to count the cardinality of IP addresses which access our website ?
- There are over four billion possible distinct IP V4 addresses.



Probabilistic data structures

- Probabilistic data structures can not provide definite answer
- Instead they provide a reasonable approximation of the answer and a way to approximate this estimation.
- Useful for big data and streaming application because they allow to decrease the amount of memory needed with comparison to data structures that give you exact answers.

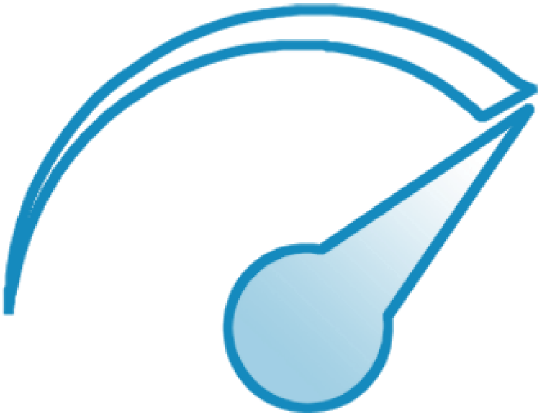
Membership

Is this guy on board ?

How many of them are there ?

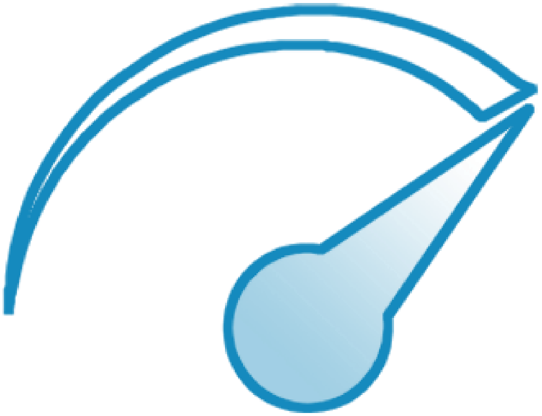
Presence evaluation

- How to evaluate the presence of the element in a stream ?



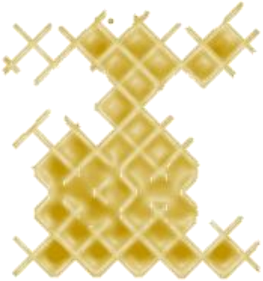
Frequency evaluation

- How to store a numerical value associated with each element ?
- How to evaluate the number of occurrences of the element in a stream ?



Hash Functions

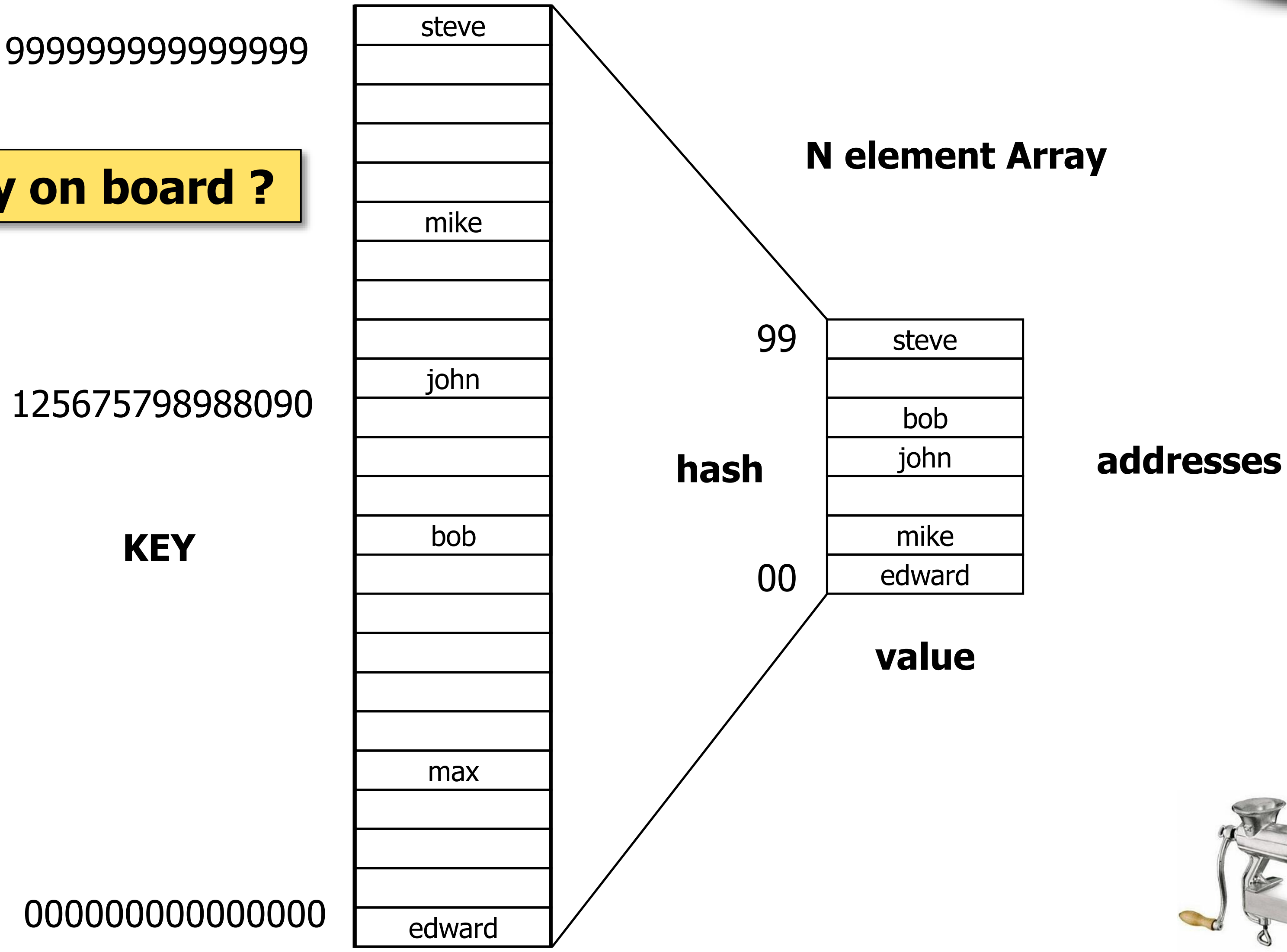
- Probabilistic data structures use hash functions to randomize items.
- They ignore collisions to keep the size constant
- But this is a reason why they can not give exact values.



Hash Usage : Set Membership, Counting

SSN : Social Security Number

Is this guy on board ?



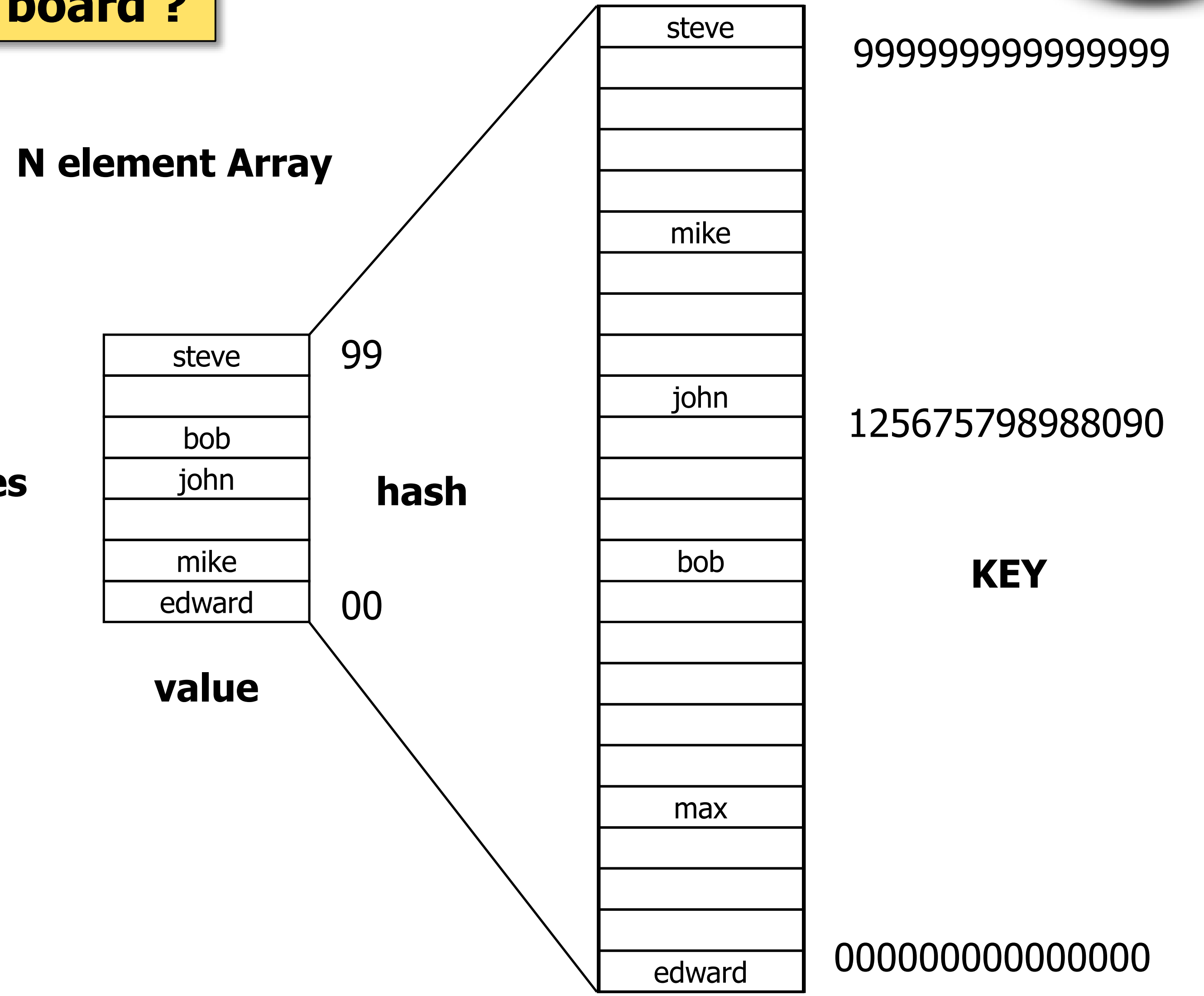
Hash Usage : Set Membership, Counting

SSN : Social Security Number

Is this guy on board ?

Probe O(1)

addresses



N element Array

value
steve
bob
john
mike
edward

99

hash

00

value

KEY
steve
mike
john
bob
max
edward

9999999999999999

12567579898809O

KEY

0000000000000000

Hash Usage : Set Membership, Counting

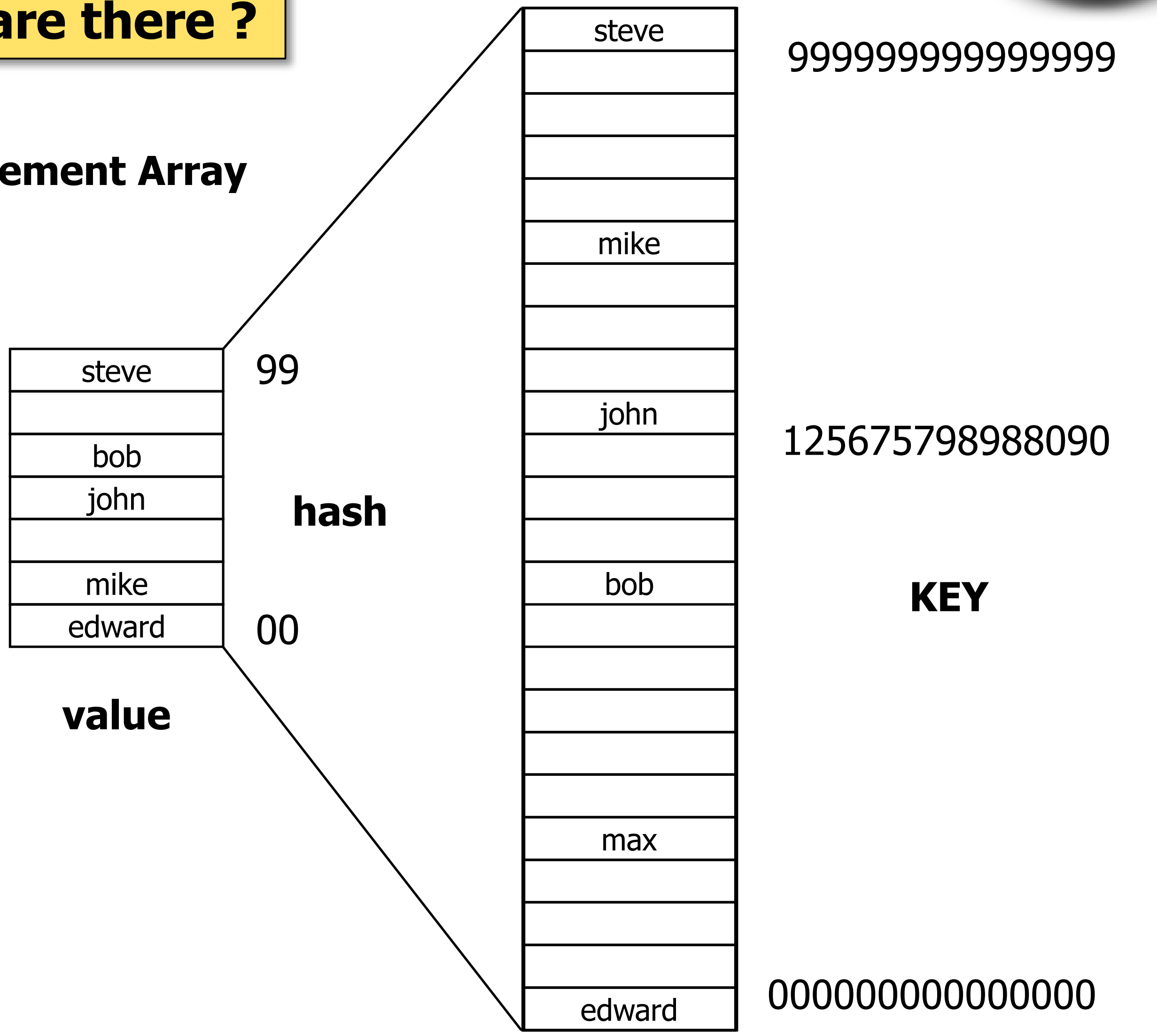
SSN : Social Security Number

How many of them are there ?

N element Array

Probe O(1)

addresses



steve

bob

john

mike

edward

99

hash

00

steve

mike

john

bob

max

edward

999999999999999

125675798988090

KEY

000000000000000

value

Hash Usage : Set Membership, Counting

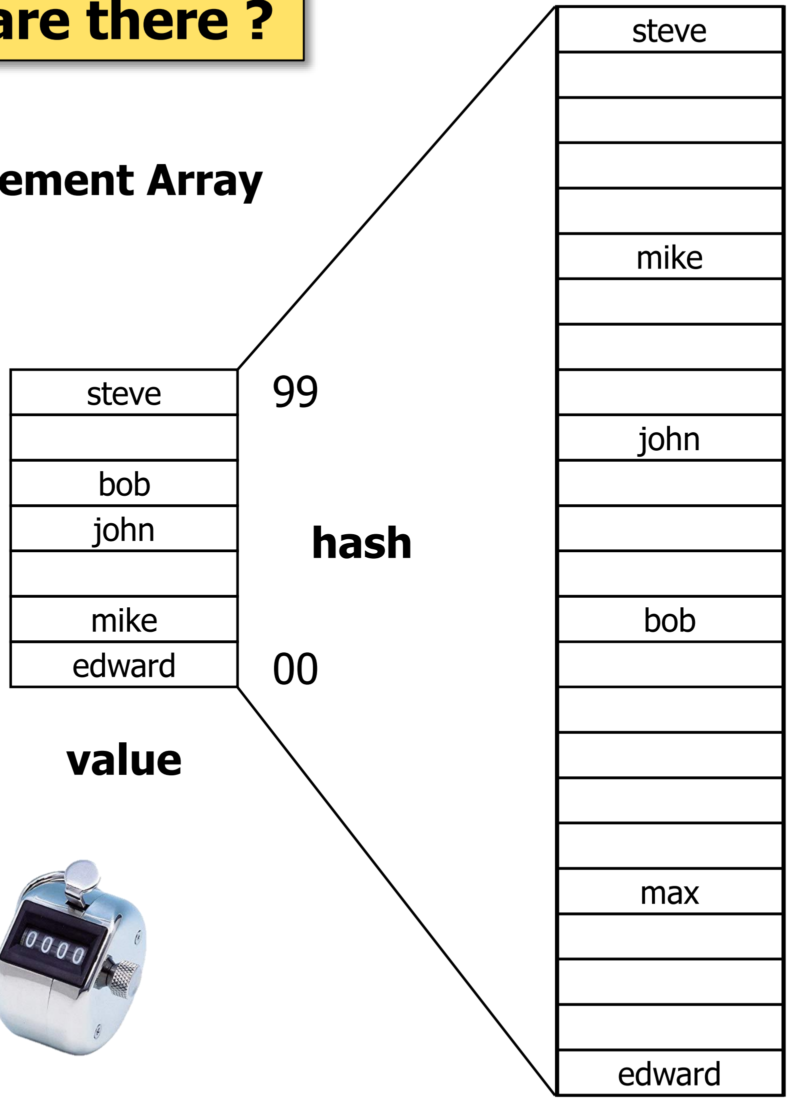
SSN : Social Security Number

How many of them are there ?

N element Array

Probe O(1)

addresses



hash

value

999999999999999

125675798988090

KEY

000000000000000

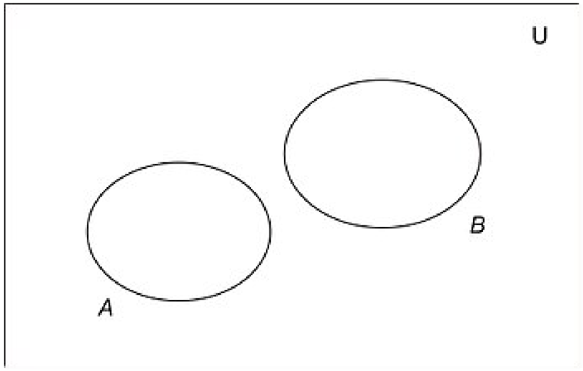
Emmanuel fuchs Architectures des Systèmes de Bases de Données

Hash Usage : Set Membership, Counting

SSN : Social Security Number

Is this guy on board ?

0	
1	
2	1
3	
4	
5	
6	1
7	
8	
9	
10	
11	1
12	
13	1
14	
15	

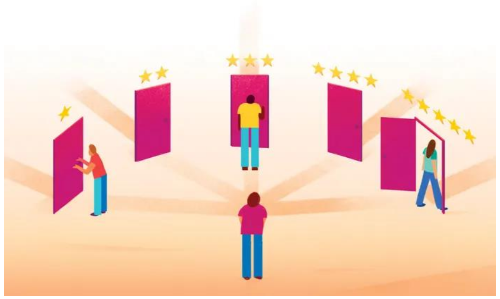


Hash Usage : Set Membership, Counting

SSN : Social Security Number

How many of them are there ?

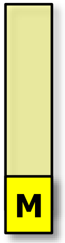
0	3
1	
2	2
3	
4	5
5	
6	3
7	
8	
9	4
10	
11	1
12	
13	1
14	
15	2

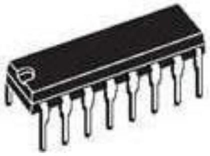


Hash Usage : Set Membership, Counting

SSN : Social Security Number

0	3
1	
2	2
3	
4	5
5	
6	3
7	
8	
9	4
10	
11	1
12	
13	1
14	
15	2





What about memory size ?

Exemple with alphabet letters

The set

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Exemple with alphabet letters

The set

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Bit Map : Probalistic Data Structure

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Exemple with alphabet letters

The set

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Size = 27 * Short = 432 bytes

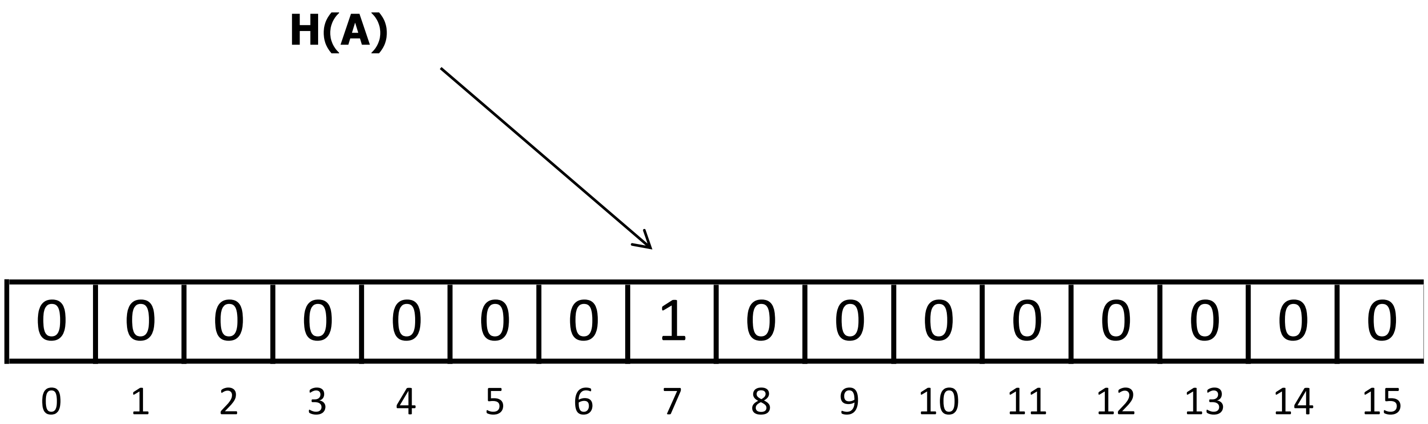
Probalistic Data Structure

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Size = 2 bytes

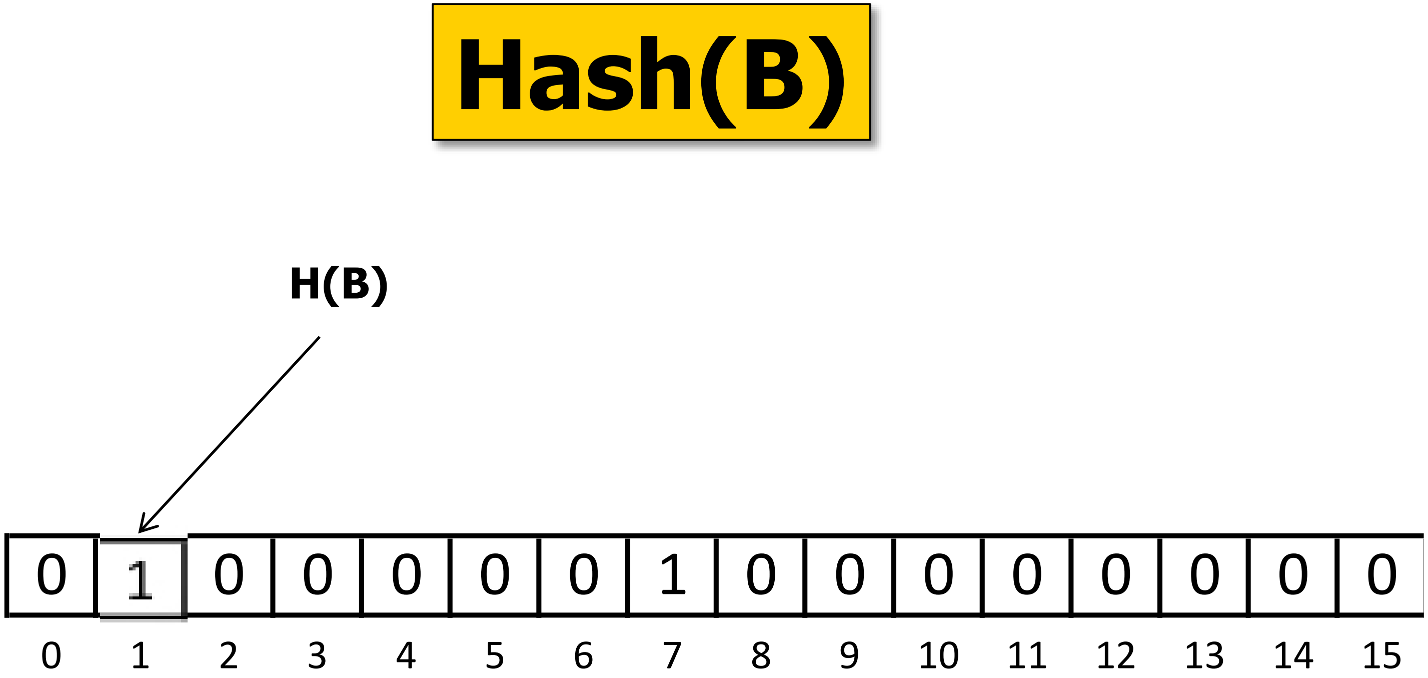
Hash set

Hash(A)



0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Hash set

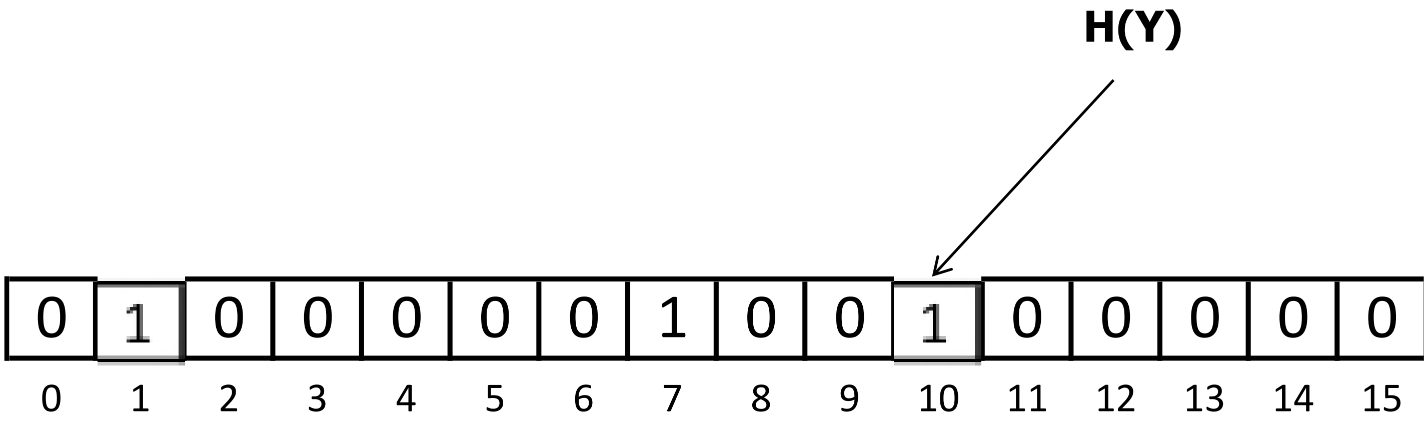


Hash(B)

0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Hash set

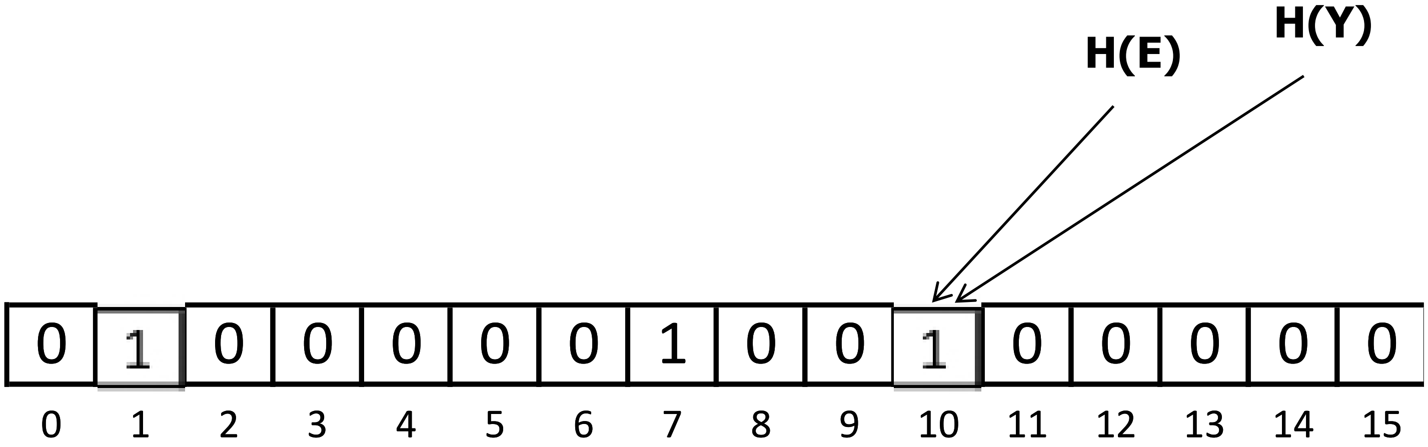
Hash(Y)



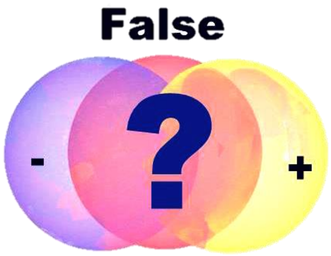
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0

Hash set

Hash(E)= Hash(Y)

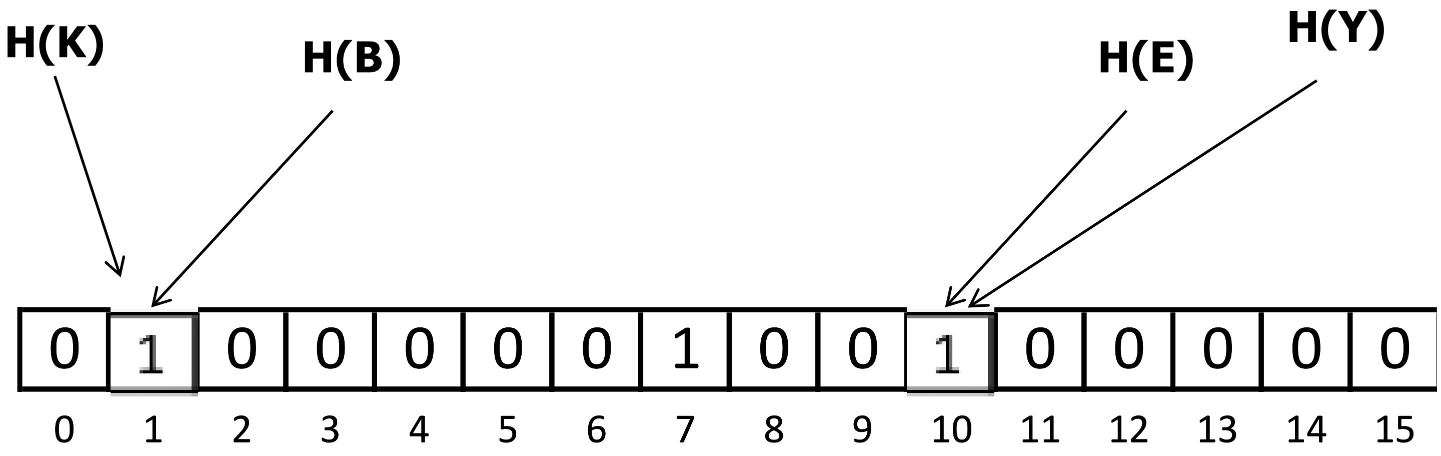


0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

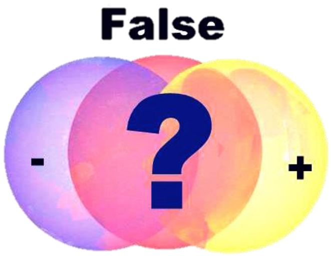


Hash set

Hash (K) = Hash (B)



0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



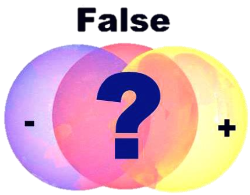
Fault Positive Prevention

- Two Hash functions : H1(), H2()
- Two Hash sets

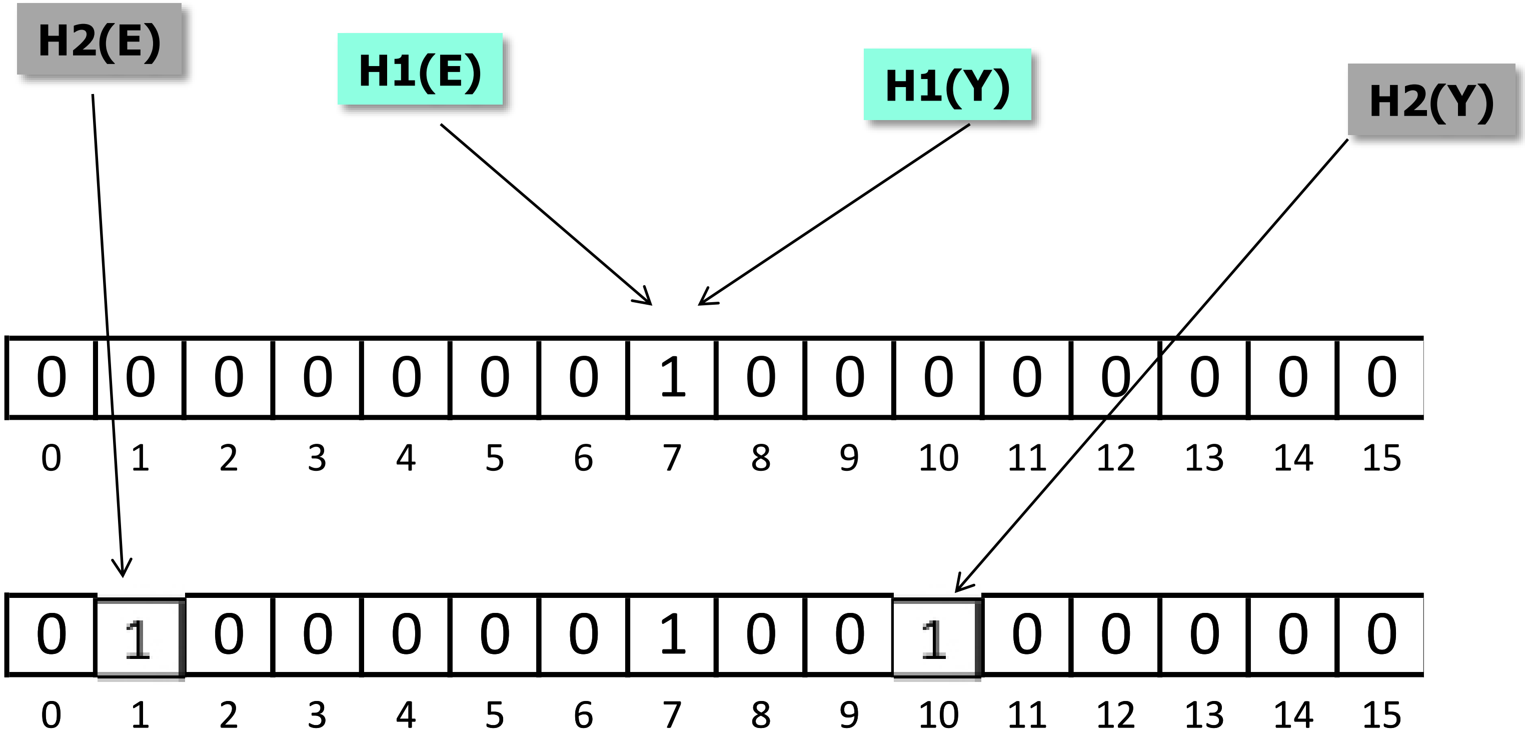
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Fault Positive



Collision



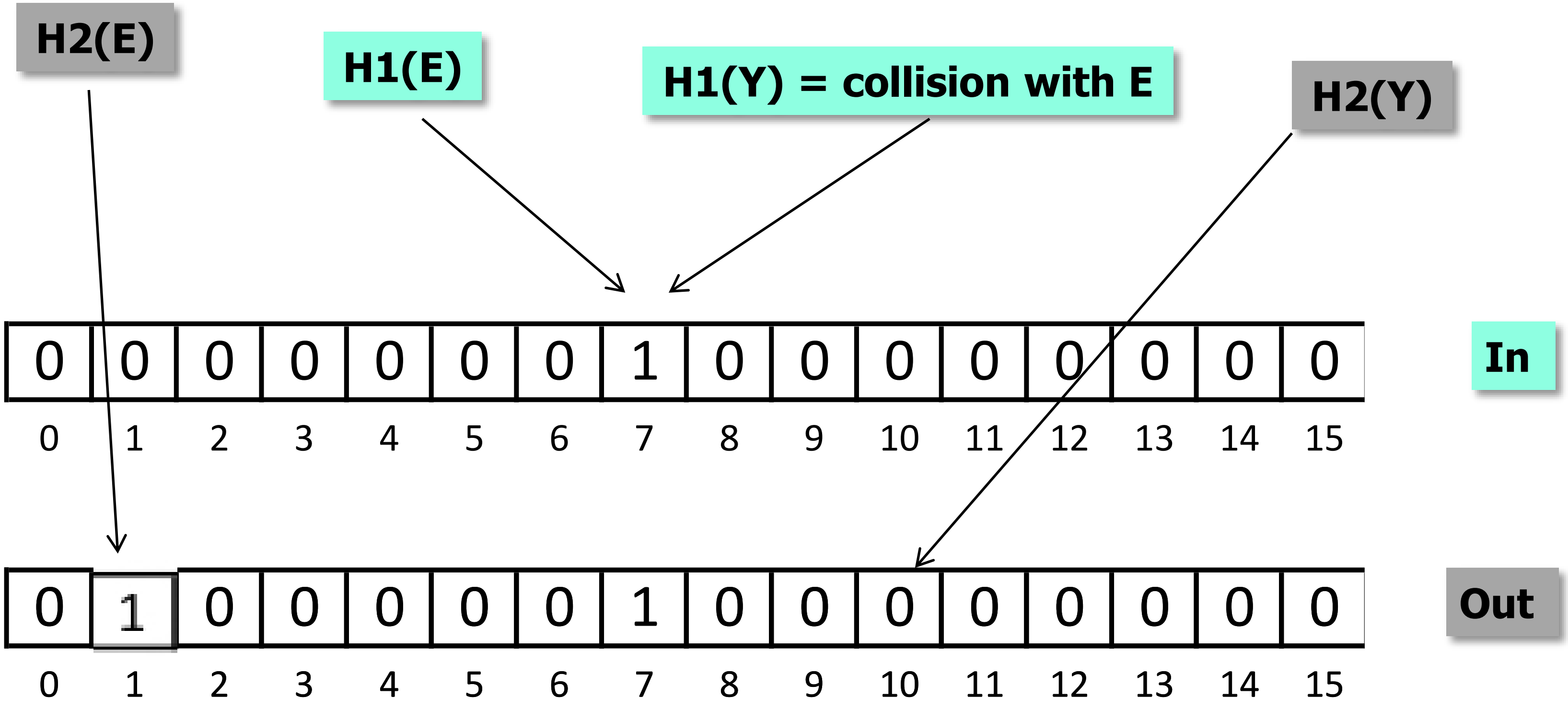
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Fault Positive

Collision

If Y not In



H2(E)

H1(E)

H1(Y) = collision with E

H2(Y)

0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	**In**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	**Out**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

Y is definitiveley Out

Usual probabilistic data structures

- Bloom Filters
- Count-min Sketch
- Count-Mean-Min Sketch
- Hyperloglog