CSC4524: Streaming algorithms

Bloom filter

Mini-project 1

- **Be able to know if a given Wikipedia page was visited the 01-08-2016**
  - Input: domain code + page title
  - Output: Yes | No
  - Work due to 17-11- 2019 23:59

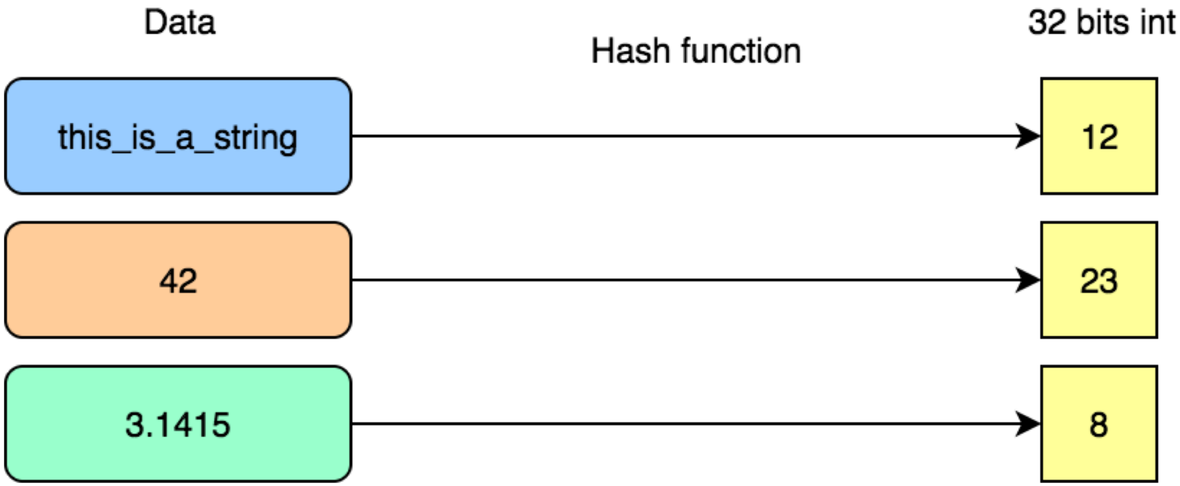
Solution

Bloom filter

Hash function

- A **hash function** is any function that can be used to map data (string, integer, float …) of arbitrary size to fixed-size values.

Example:

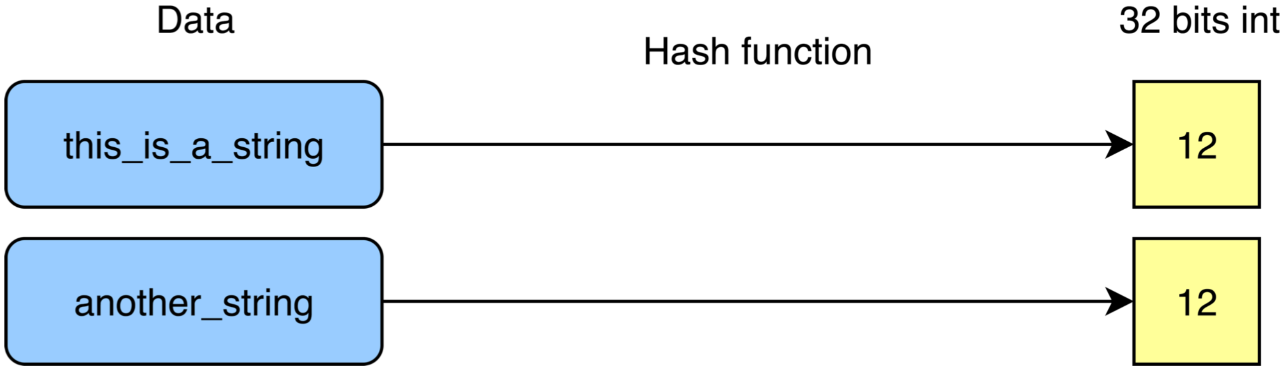


Hash function

- **Uniformity**

A good hash function should map the expected inputs as evenly as possible over its output range. That is, every hash value in the output range should be generated with roughly the same probability. This is an important criteria to reduce collisions.

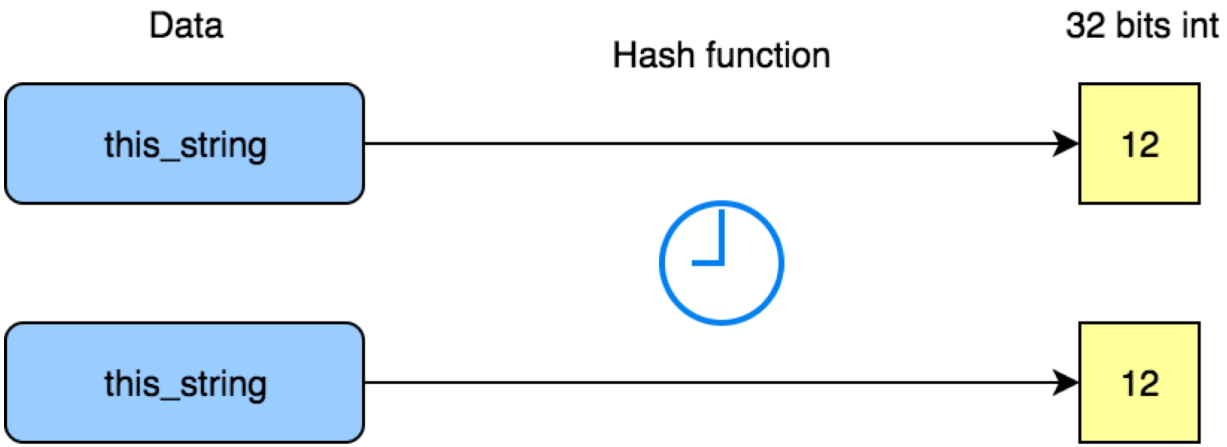
- **Collisions**



Hash function

- **Deterministic**

A hash procedure must be **deterministic**, meaning that for a given input value it must always generate the same hash value.



Hash function

- **Applications**

  - Hash tables
  - Bloom filter
  - Pseudonymization
  - A/B test engine

Bloom filter

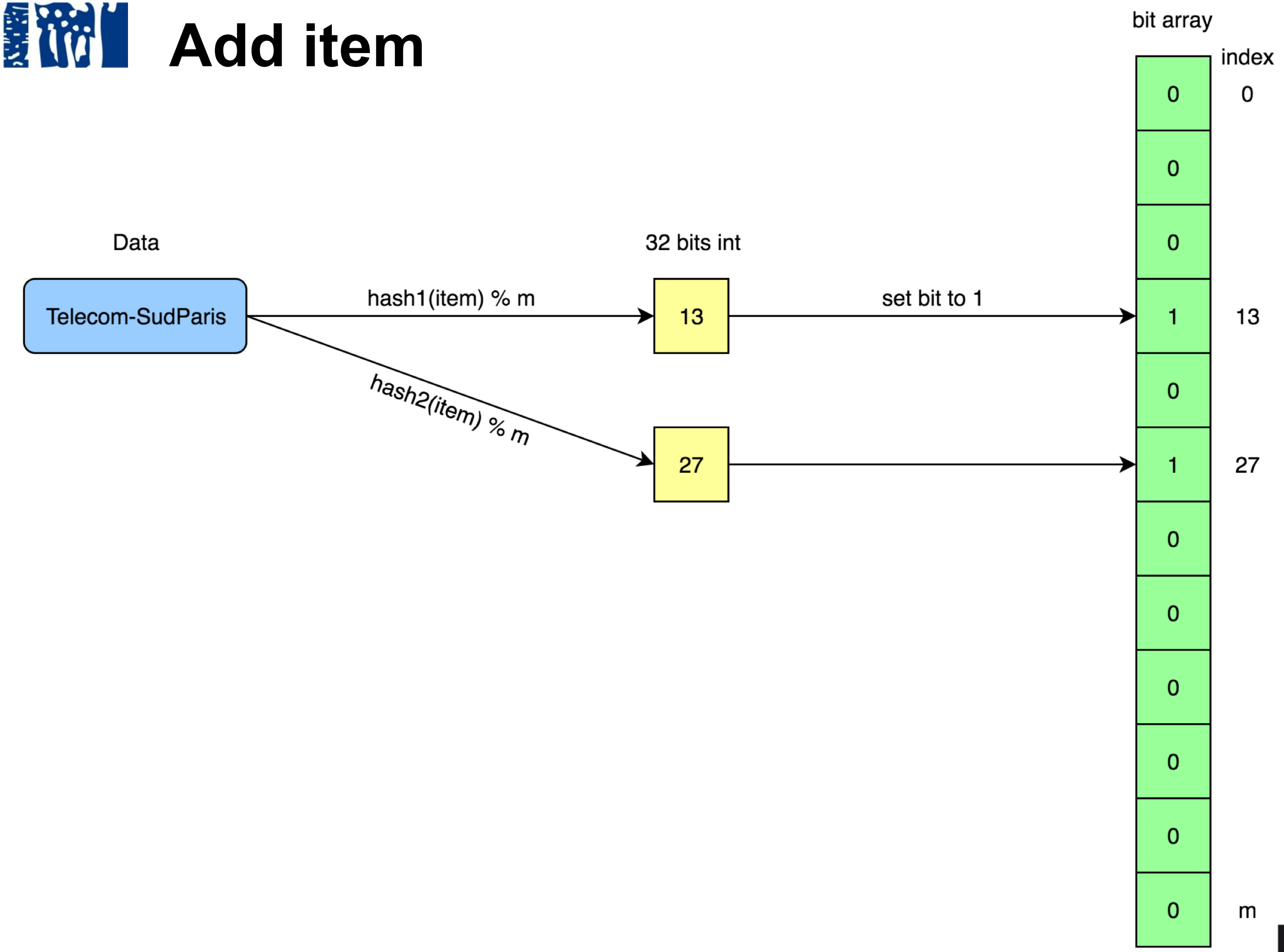
- A **Bloom filter** is a space-efficient probabilistic data structure used to test whether an element is a member of a set. False positive matches are possible, but false negatives are not. In other words, a query returns either "possibly in set" or "definitely not in set". The more elements that are added to the set, the larger the probability of false positives.

Bloom filter

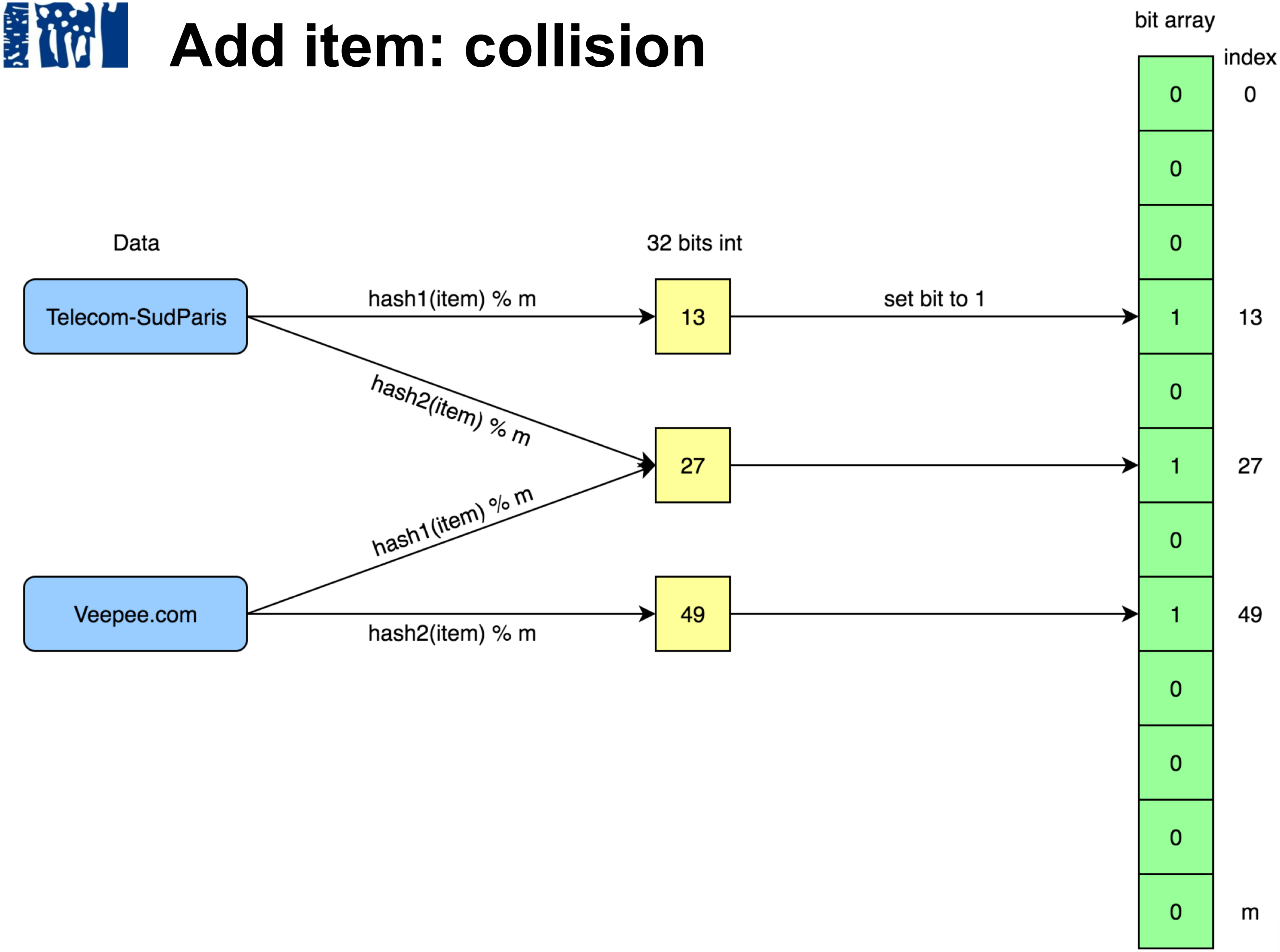
- Structure: **bit array** of ***m*** bits, all initialized to 0

- ***k*** different **hash functions** defined, each of which maps or hashes some set element to one of the *m* array positions. Example: f(item) = hash(item) % m

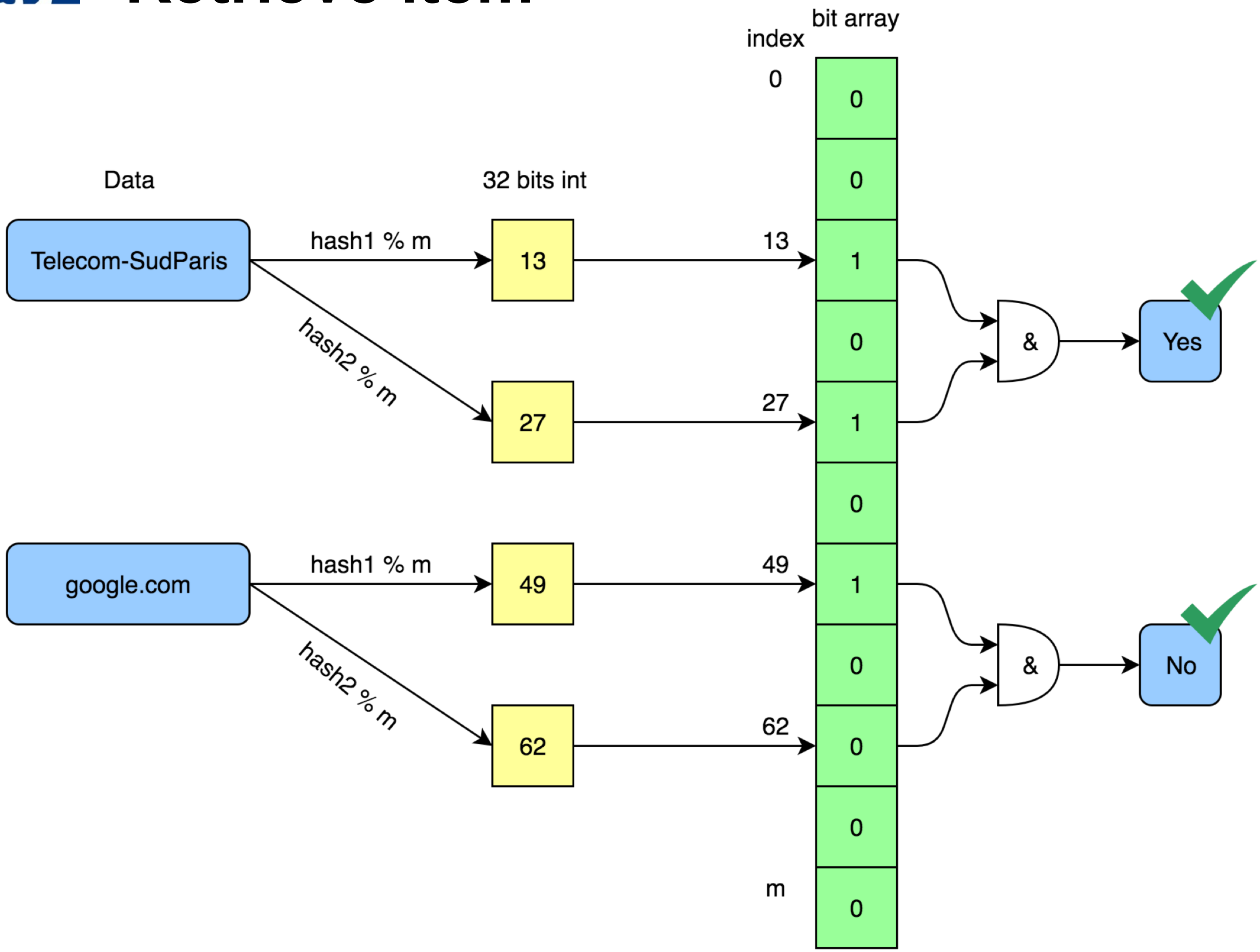
Add item


bit array
index
Data
32 bits int
Telecom-SudParis
hash1(item) % m
hash2(item) % m
13
27
set bit to 1
0
0
0
1
0
1
0
0
0
0
0
0
0
13
27
m

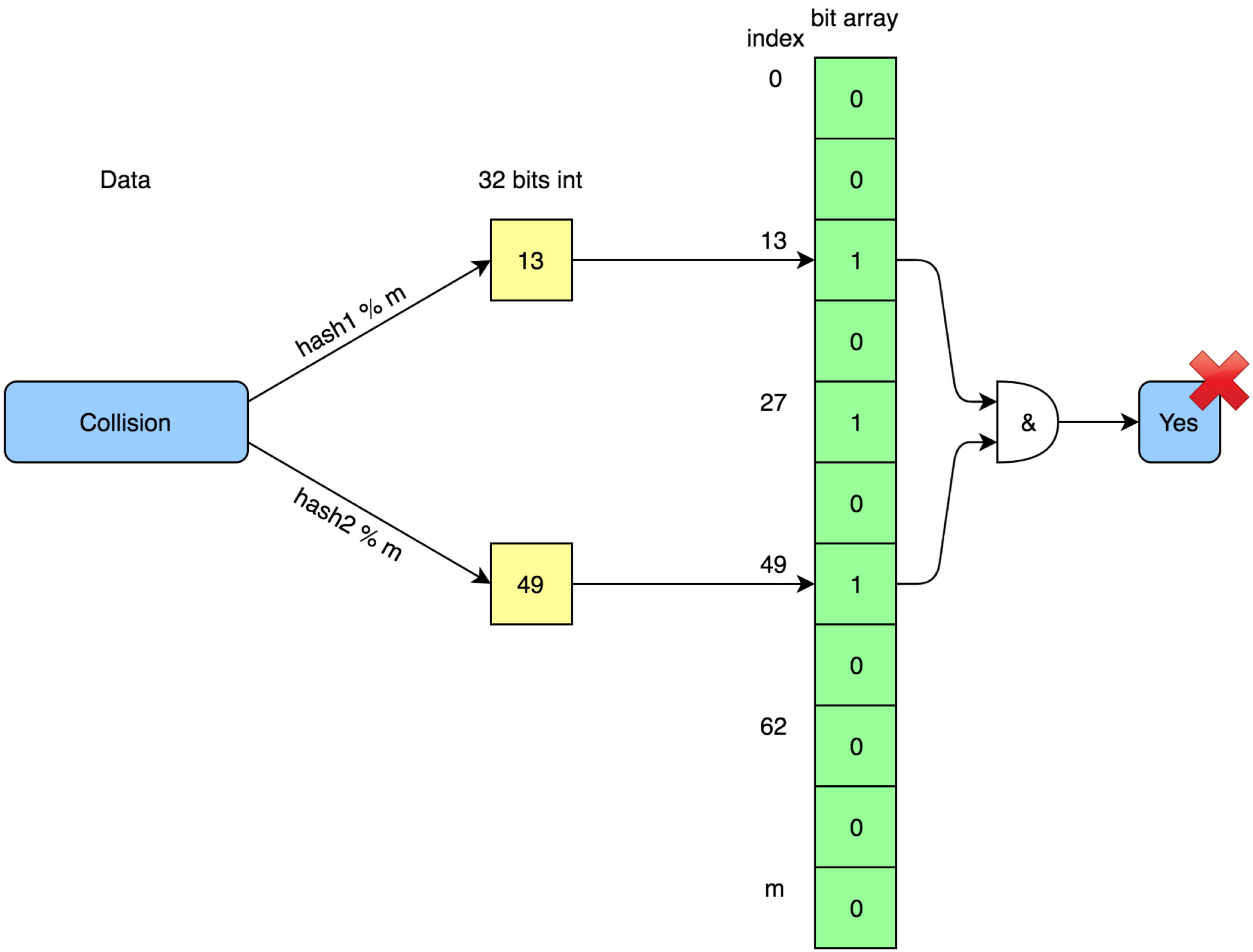
Add item: collision



Retrieve item



Retrieve item: collision



Error rate

- **False negatives rate** **0**

- **False positives rate**

$$\left(1-\left[1-\frac{1}{m}\right]^{kn}\right)^k \approx \left(1-e^{-kn/m}\right)^k.$$

m: bit array size
k: number of hash functions
n: input cardinality

Complexity

- Time

$O(n)$

- Memory

$O(m)$

Advantages

- 1 unique pass through the data

- Fast processing

- Drastic dimension reduction

Applications

- Cache filtering (solve the "one-hit-wonders » issue)

- Malicious sites listing

- Data base item existence

Mini-project 2

- **What is the number of unique (domain name, page title) couples visited the 01-08-2016 ?**
  - Output: Input cardinality
  - Work due to 25-11- 2019 23:59