Hashing - LSH - Bloom Filters

Maria Despoina Siampou (sdi1600151), George Michas (sdi1400109)

HELLENIC REPUBLIC
National and Kapodistrian University of Athens

May 17, 2019

Contents

Contents

The Symbol Table ADT

A symbol table T is an abstract storage that contains table entries that are either empty or are pairs of (K, I) where K is a key and I is some information associated with the key.

Distinct table entries have distinct keys.

Introducing Hashing by Example

Hashing differs from the representations based on searching by key comparisons because we are trying to refer directly to elements of the table by transforming keys into addresses in the table.

- We will use as keys letters of the alphabet having as subscripts their order in the alphabet. For example, A1, C3, R18.
- We will use a small table T of 7 positions as storage. We call it hash table.
- We will find the location to store a key K by using the following hash function: h(Ln) = n % 7.

Table T after Inserting keys $B_2, J_{10}, S_{19}, N_{14}$

	Table T
0	N_{14}
1	
2	B_2
3	J_{10}
4	
5	S_{19}
6	

Introducing Hashing by Example

Insert X_{24}

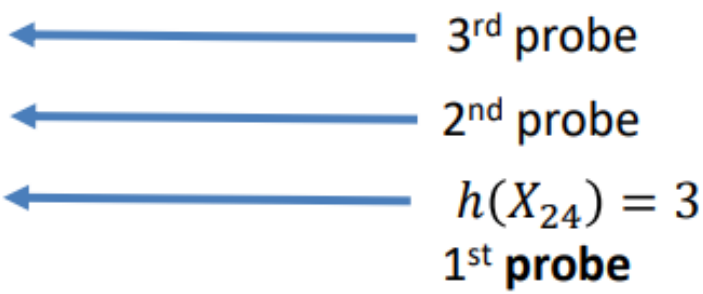
	Table T
0	N_{14}
1	
2	B_2
3	J_{10}
4	
5	S_{19}
6	

$h(X_{24}) = 3$

Introducing Hashing by Example

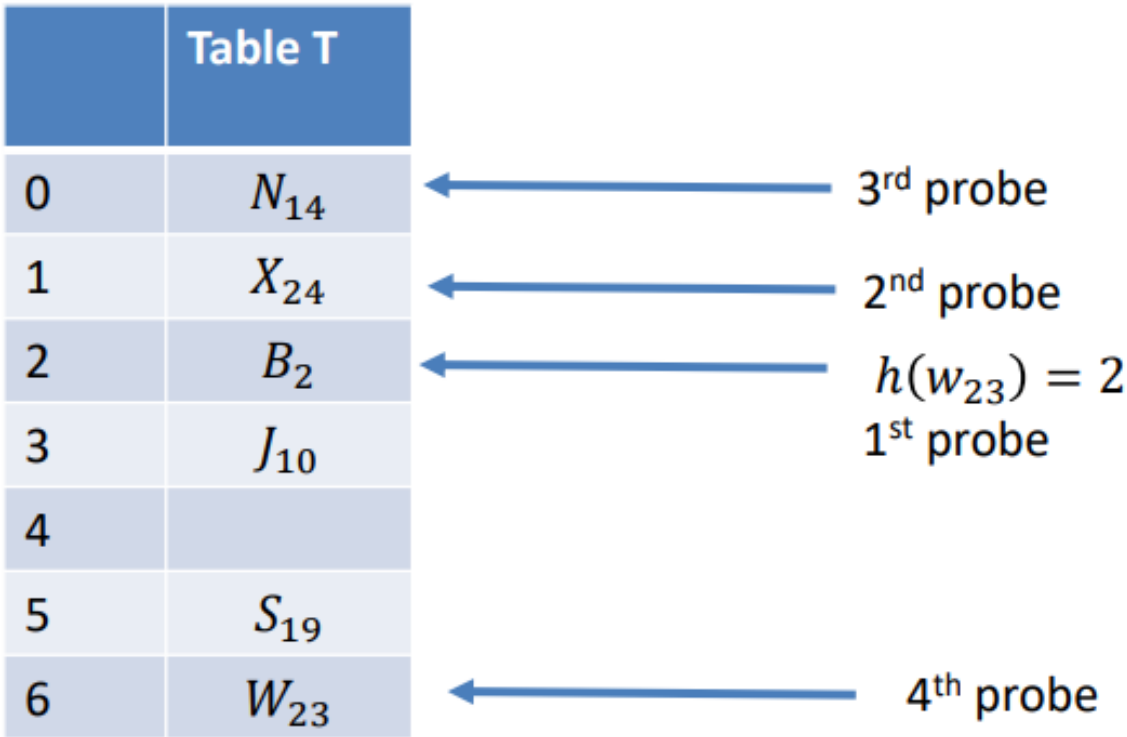
Insert X_{24}

	Table T
0	N_{14}
1	X_{24}
2	B_2
3	J_{10}
4	
5	S_{19}
6	



Introducing Hashing by Example

Insert W_{23}



Open Addressing

Open addressing, or closed hashing, is a method of collision resolution in hash tables. With this method a hash collision is resolved by probing, or searching through alternate locations in the array (the probe sequence) until either the target record is found, or an unused array slot is found, which indicates that there is no such key in the table.

Well-known probe sequences include:

- **Linear probing:** in which the interval between probes is fixed - often set to 1.
- **Quadratic probing:** in which the interval between probes increases quadratically (hence, the indices are described by a quadratic function).
- **Double hashing:** in which the interval between probes is fixed for each record but is computed by another hash function.

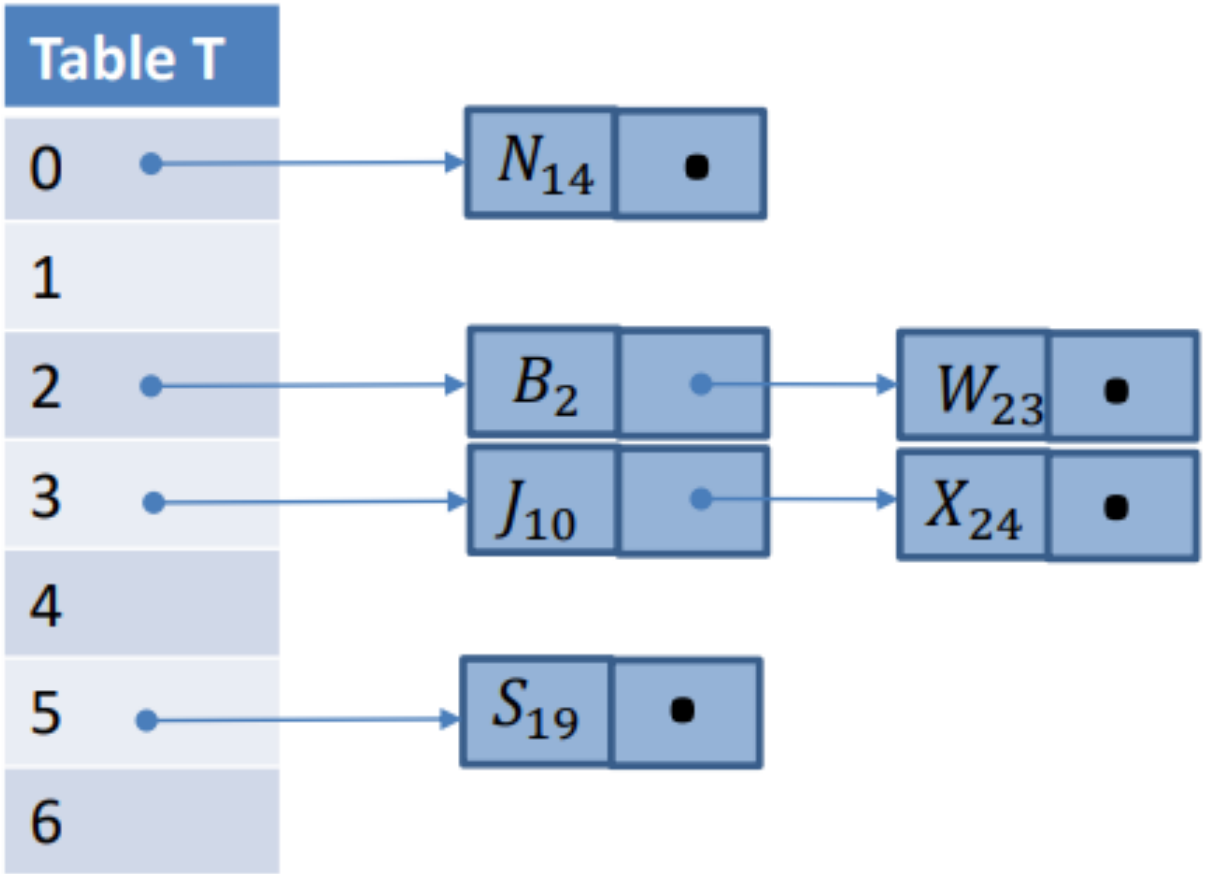
Collision Resolution by Separate Chaining

The method of collision resolution by **separate chaining** uses a linked list to store keys at each table entry.

For separate-chaining, the **worst-case scenario** is when all entries are inserted into the same bucket, in which case the hash table is ineffective and the cost is that of searching the bucket data structure.

Chained hash tables also inherit the **disadvantages** of linked lists: When storing small keys and values, the space overhead of the next pointer in each entry record can be significant. An additional disadvantage is that traversing a linked list has poor cache performance, making the processor cache ineffective.

Separate Chaining - Example



Good Hash Functions

Suppose T is a hash table having M entries whose addresses lie in the range 0 to M - 1.

- For any arbitrarily chosen key, any of the possible table addresses is equally likely to be chosen.
- The computation of a hash function should be very fast.
- Use of prime numbers to minimize collisions when the data exhibits some particular patterns.

Complexity of Hashing

- To enumerate the entries of a hash table, we must first sort the entries into ascending order of their keys. This requires time O(nlogn) using a good sorting algorithm.
- Insertion takes the same number of comparisons as an unsuccessful search, so it has complexity O(1) as well.
- Retrieving and updating also take O(1) time

Contents

Introduction

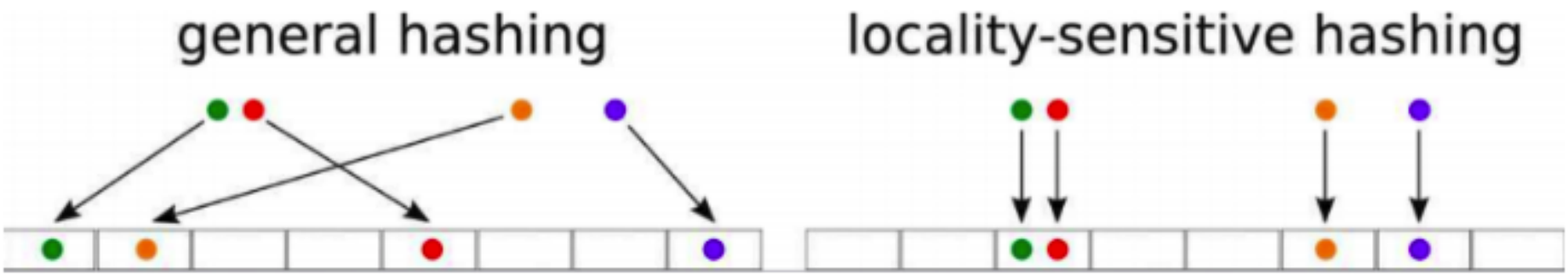
LSH refers to a family of functions (known as LSH families) to hash data points into buckets so that data points near each other are located in the same buckets with high probability, while data points far from each other are likely to be in different buckets. This makes it easier to identify observations with various degrees of similarity.

LSH Family

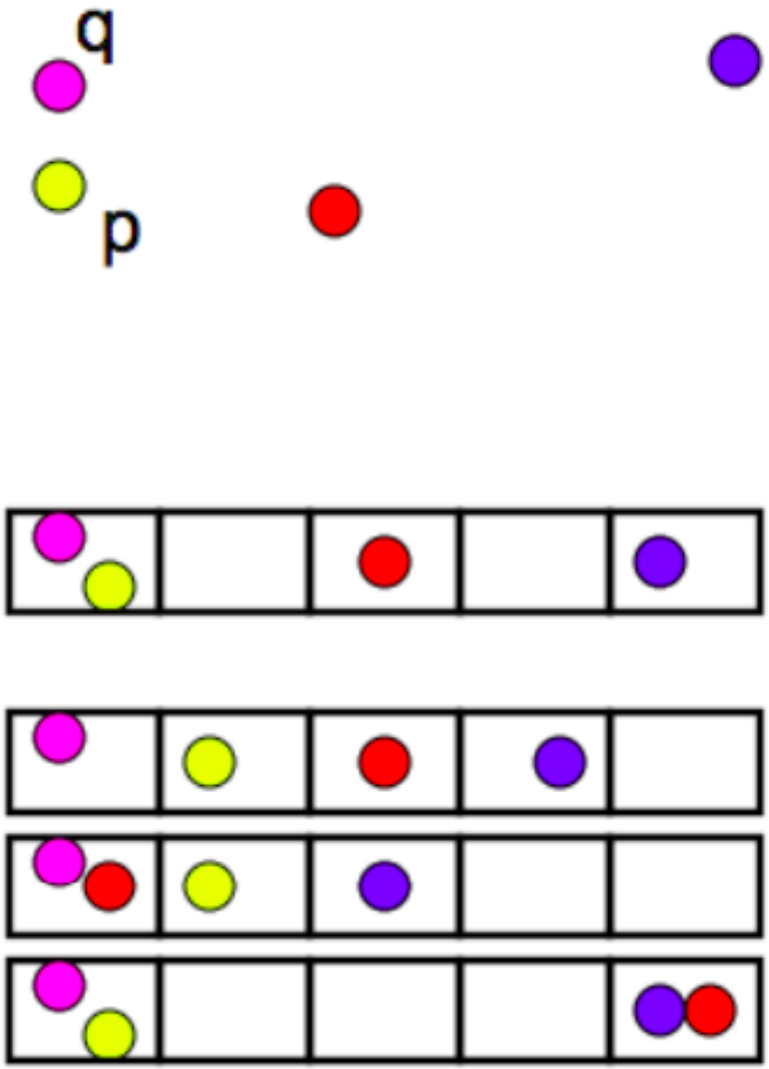
Let $r_1 < r_2$, probabilities $P_1 > P_2$. A family H of functions is (r_1, r_2, P_1, P_2) -sensitive if, for any points $p \neq q$ and any randomly selected function $h \in_R H$,

- if dist(p, q) $\leq$ r1, then prob [h(q) = h(p)] $\geq$ P1,
- if dist(p, q) $\geq$ r2, then prob [h(q) = h(p)] $\leq$ P2.

Introduction



Introduction



Introduction

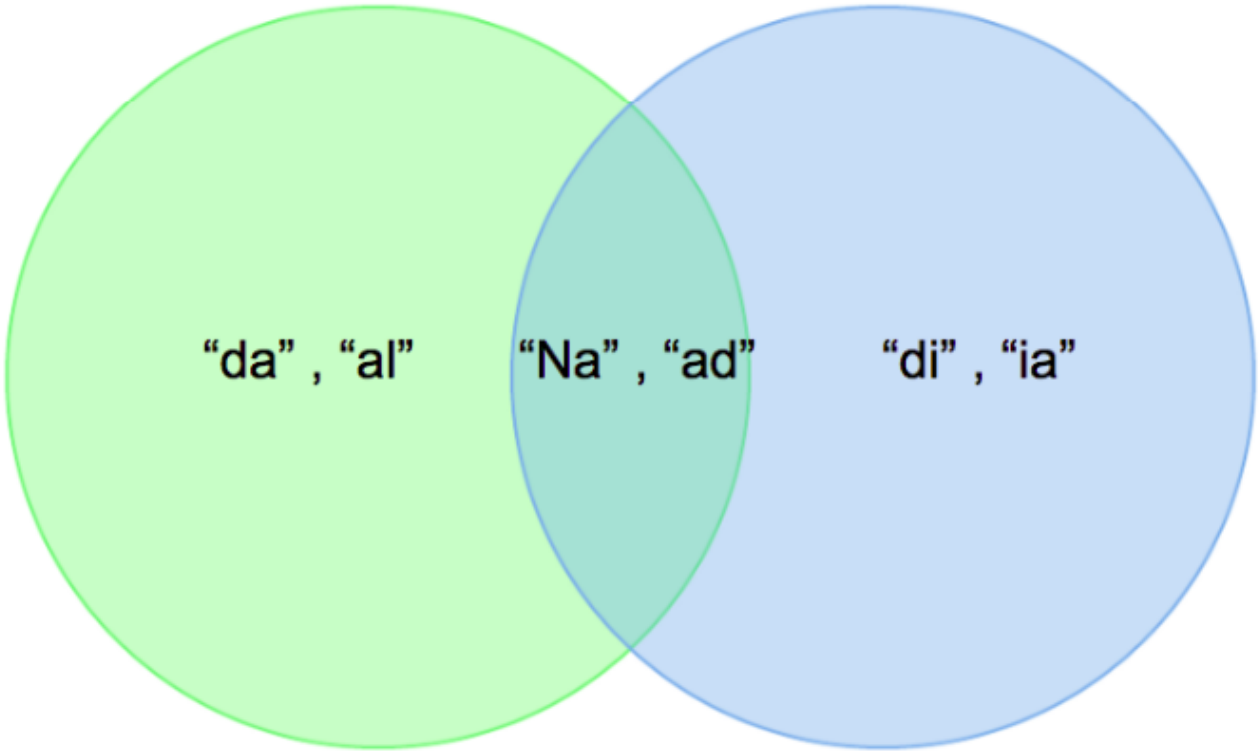
LSH creates every hash-table using an amplified hash function g by combining k functions $h_i \in_R H$, chosen uniformly at random (with repetition) from H. This implies some h_i may be chosen more than once for a given g, or for different g's (LSH builds several hash-tables).

Typically g is defined by concatenation: g(p) = [h1(p)|h2(p)| ...|hk(p)]

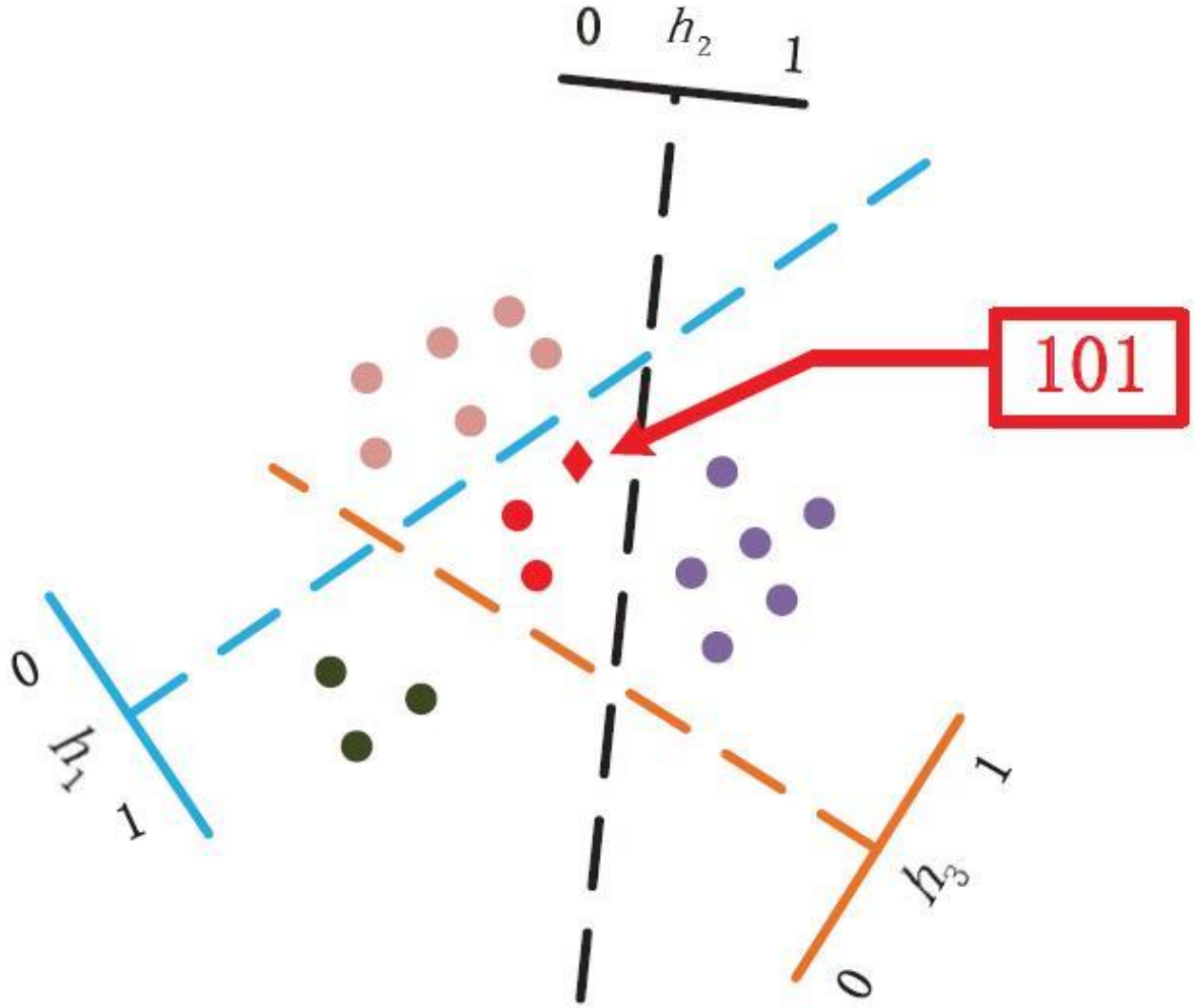
Intersection Over Union

Suppose A: “Nadal” and B: “Nadia”, then 2-shingles representation will be:

A: {Na, ad, da, al} and B: {Na, ad, di, ia}.



Example



Construction

Preprocess

Having defined H and hash function g:

- Randomly select L amplified hash functions g1, ..., gL.
- Initialize L hash-tables, hash all points to all tables using g.

Large k $\Rightarrow$ larger gap between P1, P2. Practical choices are k $=$ 4 to 6, L $=$ 5 (or 6), and HashTable size n/4 (alternatively n/2 or n/8).

Range Search

Range (r, c)-Neighbor search

Input: r, c, query q
for i from 1 to L **do**
  for each item p in bucket $g_i(q)$ **do**
    if dist$(q, p) < cr$ **then output** p
    end if
  end for
end for

In practice, if c not given assume $c = 1$.

Decision problem: "**return** p" instead of "**output** p".
At end "**return** FAIL"; may also FAIL if many examined points.

NN Search

Approximate NN

Input: query q
Let $b \leftarrow$Null; $d_b \leftarrow \infty$
for i from 1 to L **do**
  for each item p in bucket $g_i(q)$ **do**
    if large number of retrieved items (e.g. $> 3L$) **then return** b // trick
    end if
    if $\text{dist}(q, p) < d_b$ **then** $b \leftarrow p$; $d_b \leftarrow \text{dist}(q, p)$
    end if
  end for
  return b
end for

Theoretical bounds for $c(1+\epsilon)$-NN by reduction to $((1+\epsilon)^i, c)$-Neighbor decision problems, $i = 1, 2, \ldots, \log_{1+\epsilon} d$.

Euclidean Space

For simplicity we use Euclidean Distance : $dist_{l_2}(x, y)^2 = \sum_{i=1}^{d}(x_i - y_i)^2$

Definition

Let d-vector $v \sim N(0, 1)^d$ have coordinates identically independently distributed (i.i.d.) by the standard normal. Set "window" w $\in \mathbb{N}^*$ for the entire algorithm, pick single-precision real t uniformly $\in_R$ [0, w).

For point p $\in R^d$, define: h(p) $= \left\lfloor \frac{p * v + t}{w} \right\rfloor \in \mathbb{Z}$

- Essentially project p on the line of v, shift by t, partition into cells of length w.
- In general, w $= 4$ is good but should increase for range queries with large r.
- Also k $= 4$ (but can go up to 10), and L may be 5 (up to 30).

Hash-table

Amplified function g(p) = [h1(p)|h2(p)|...|hk (p)] would yield a k-dimensional hashtable with many empty buckets. So we define φ as random combination of the h_i's as follows:

Classic hash-function

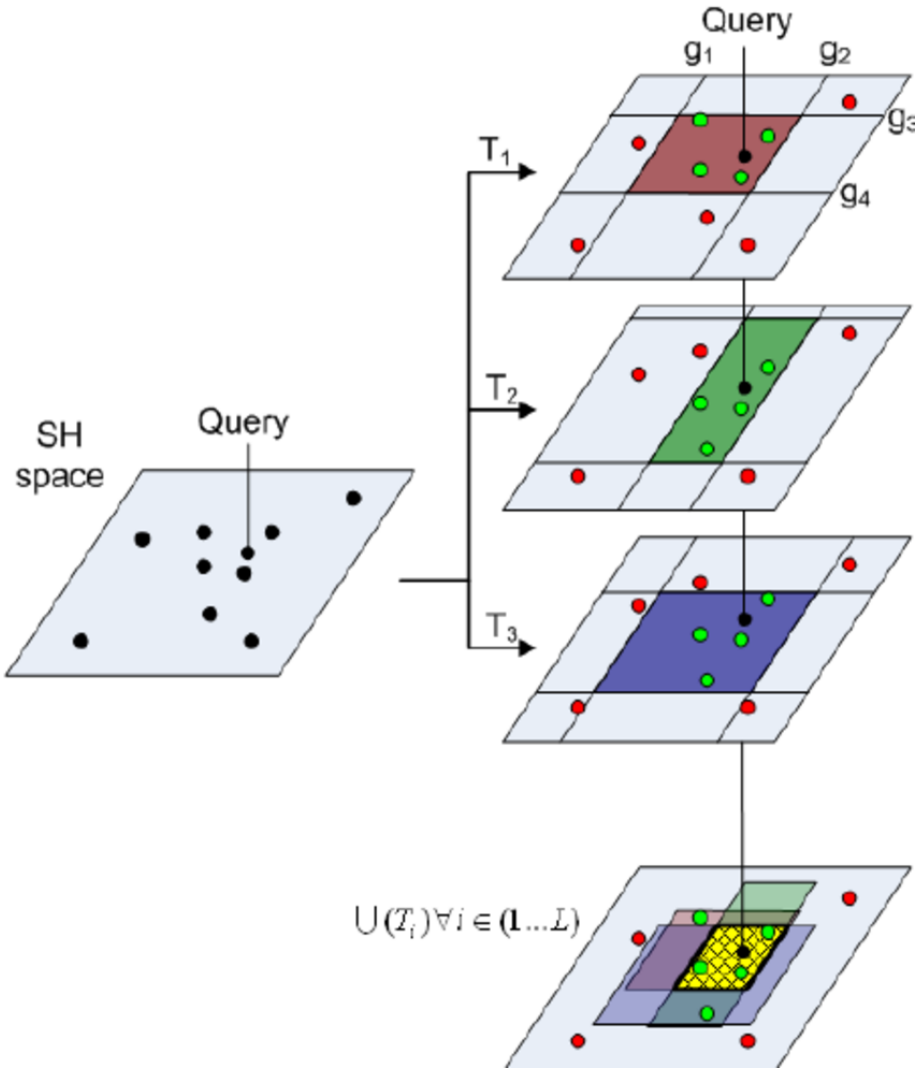
We build a 1-dim hash-table with classic index:

$$\varphi(\mathrm{p}) = [(r_1 h_1(\mathrm{p}) + r_2 h_2(\mathrm{p}) + \ldots + r_k h_k\ (\mathrm{p})) \bmod \mathrm{M}] \bmod \mathrm{TableSize},$$

s.t. int $r_i \in_R \mathbb{Z}$, prime M = 2^{32}-5 if h_i(p) are int, TableSize= n/2 (or n/4)

Notice φ computed in int arithmetic, if all h_i(p), r_i are int ($\leq$ 32 bits).
Can have smaller TableSize=n/8 or n/16 (heuristic choice).
Recall (ab) mod m = ((a mod m)(b mod m)) mod m.

Example


Query
g1
g2
g3
g4
T1
T2
T3
SH
space
Query
$\bigcup(T_i)\forall i \in (1...L)$

Contents

Bloom Filters

Existence Set Problem

Assume we have a set of elements , $S = \{s_1, s_2, \ldots, s_n\}$ *with* $n \in \mathbb{Z}$.Given an element s' we have to determine if $s' \in S$.

Bloom Filter Definition

Bloom Filter is a space-efficient probabilistic data structure, conceived by Burton Howard Bloom in 1970, that is used to test whether an **element is a member of a set**.

- **False Positive (indicates presence of a condition, when in reality it is not present)** matches are possible.
- **False Negatives (indicates no presence of a condition, when in reality it is present)** matches are not possible.
- So a query returns either **"possibly in set" or "definitely not in set"**.

Elements can be added to the set, but not removed (though this can be addressed with a "counting" filter); the more elements that are added to the set, the larger the probability of false positives.

Bloom Filter Parameters and Insertion

Parameters

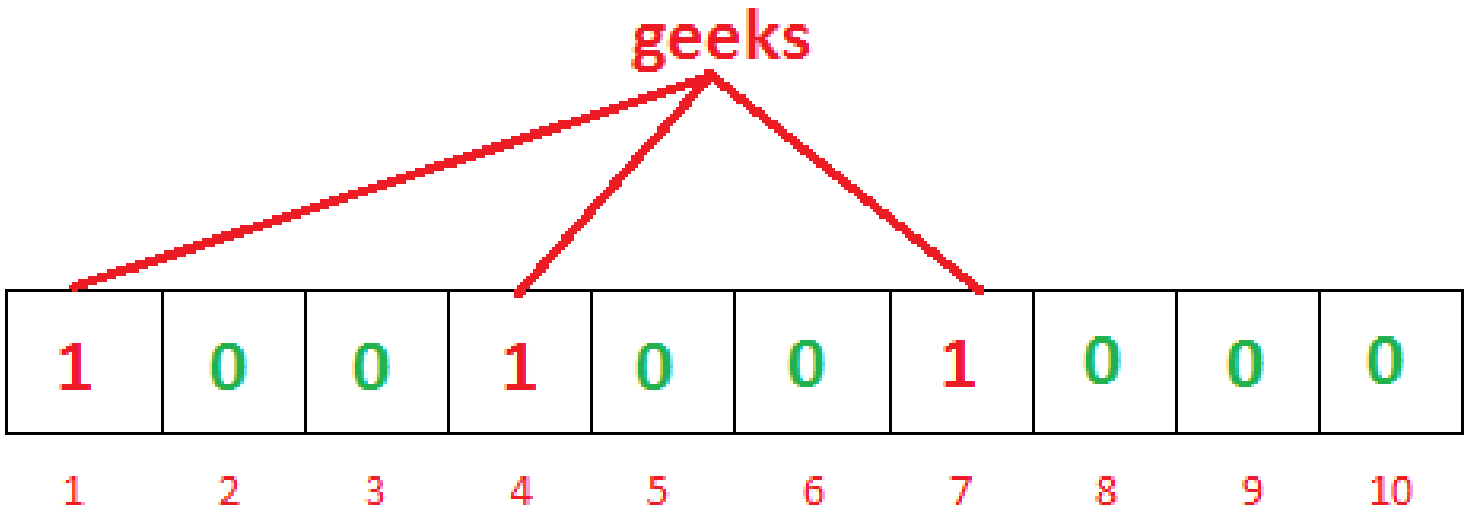
- **m** : Number of bits for the bit vector array (size fo array).
- **k** : Number of hash functions. ($k = \ln(2) * \frac{m}{n}$ (n is the number of insertions(assumption))), hash value of each function must be in the range of $\{0, \ldots, m-1\}$.

Bloom Filter Insertion

- Initialize all bits of bit vector array to 0.
- Hash each one input element to k hash functions and get a value in range of $\{0, \ldots, m-1\}$ for each one. (k array indexes).
- Set for each index we found from the k hash functions the corresponding bit values to 1.

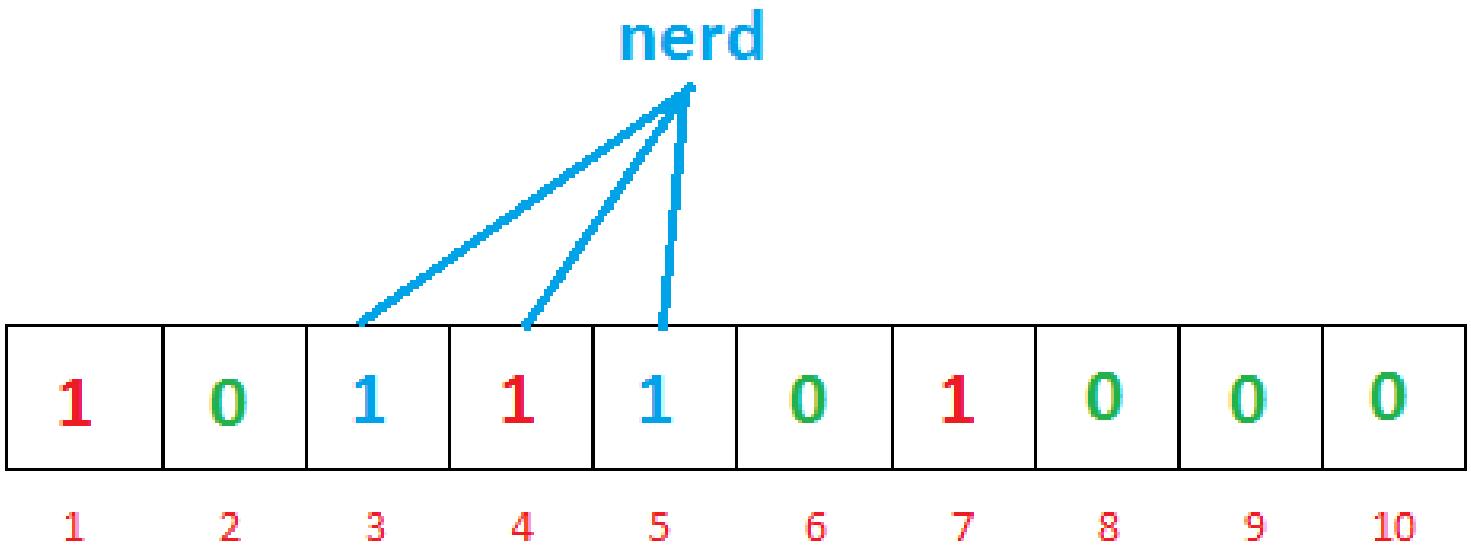
Bloom Filter Insertion Example (1)

- h1("geeks") mod 10 = 1
- h2("geeks") mod 10 = 4
- h3("geeks") mod 10 = 7



Bloom Filter Insertion Example (2)

- h1(“nerd”) mod 10 = 3
- h2(“nerd”) mod 10 = 4
- h3(“nerd”) mod 10 = 5



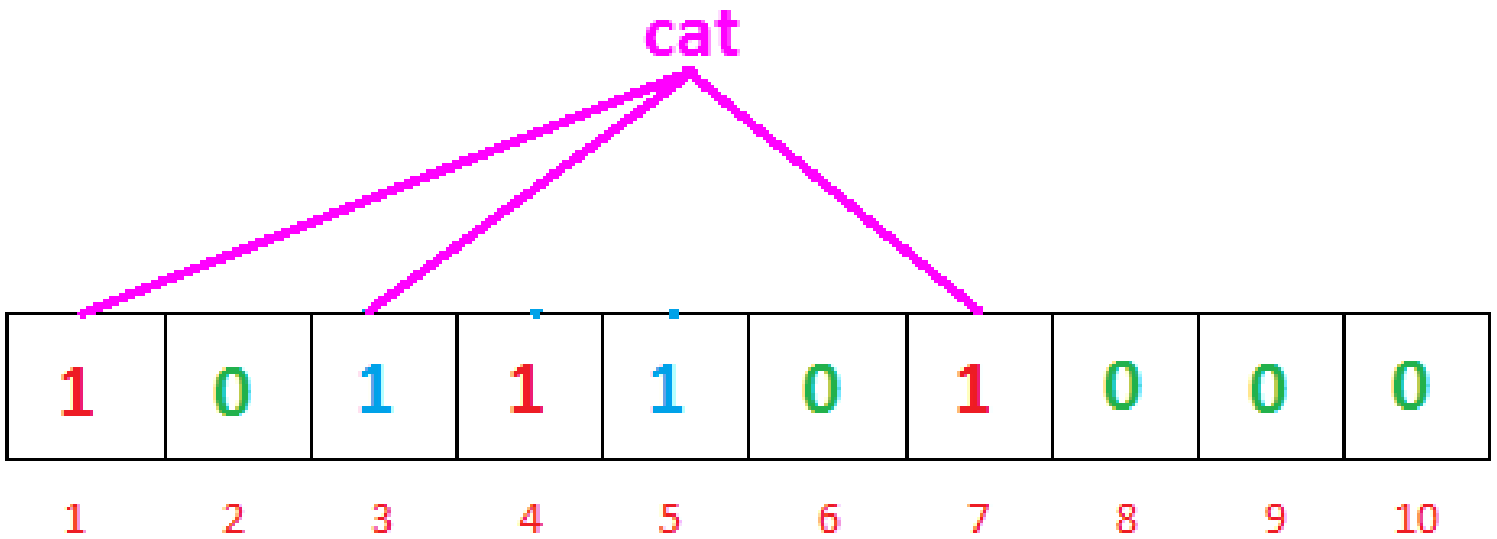
Bloom Filter Search

Bloom Filter Search

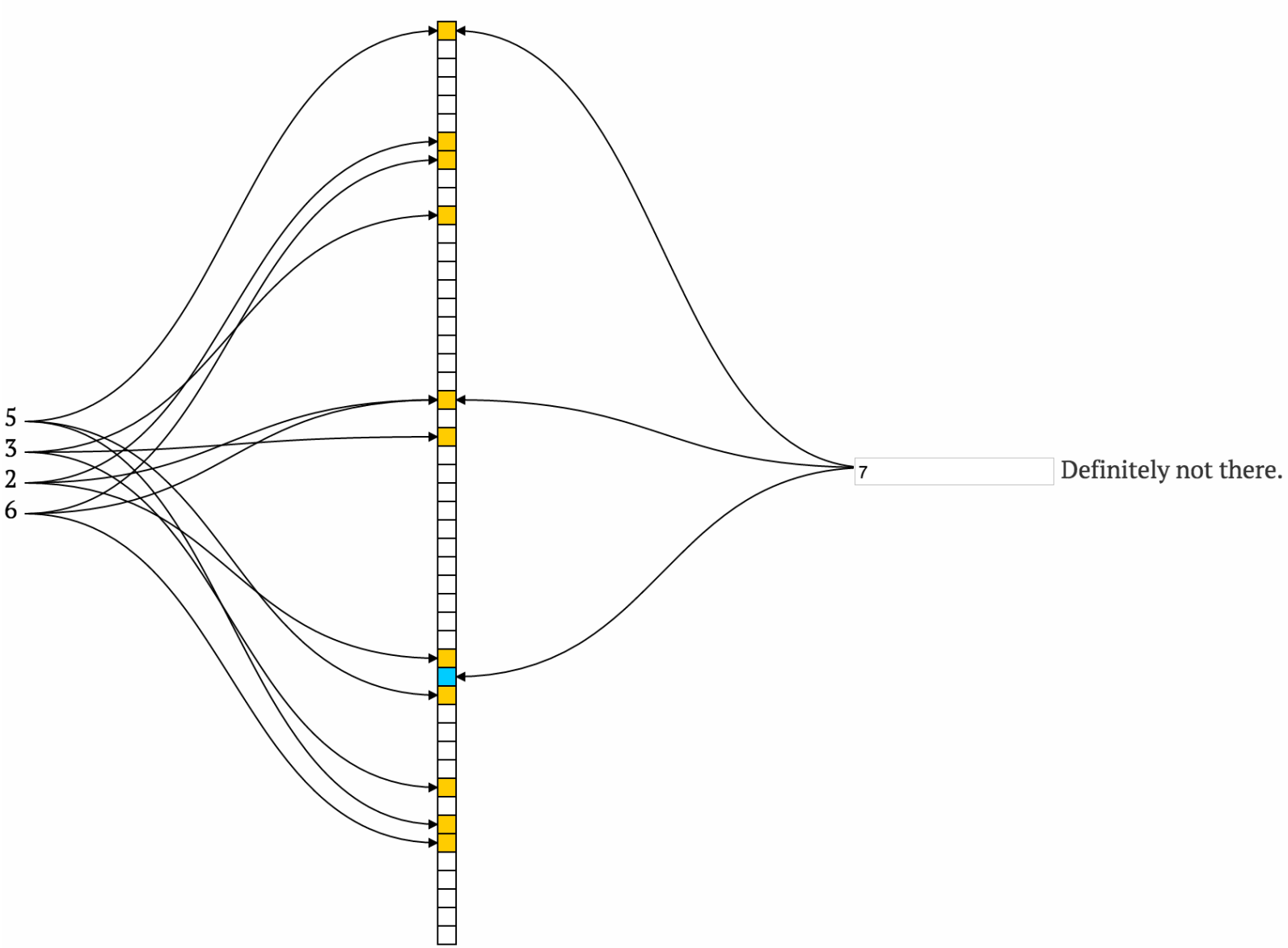
- Hash the input element to k hash functions and get a value in range of $\{0, \ldots, m-1\}$ for each one. (k array indexes).
- If we found an index with corresponding bit value 0 then the element is **definitely not present**.
- If all corresponding bit values are 1 then the element is **probably present**.

Bloom Filter Search Example Probably Present

- h1("cat") mod 10 = 1
- h2("cat") mod 10 = 3
- h3("cat") mod 10 = 7



Bloom Filter Search Example Not Present



Bloom Filter

- **False Positive Probability** $= (1 - \exp(\frac{-k*n}{m}))^k$
- **Insertion Complexity** : O(k) (k = number of hash functions)
- **Search Complexity** : O(k) (k = number of hash functions)
- **Space Complexity** : O(m) (m = number of bits)