

VE281

Data Structures and Algorithms

Bloom Filter

Learning Objectives:

- Know what Bloom filter is and how it works
- Know the advantages and disadvantages of Bloom filter

Bloom Filter

- Invented by Burton Bloom in 1970
- Supports **fast insert** and **find**
- Comparison to hash tables:
 - Pros: more space efficient
 - Cons:
 - 不能存 (key,value) pair
 - 1. Can't store an associated object
 - 2. **No deletion** (There are variations support deletion, but this operation is complicated)
 - 3. Small **false positive** probability: may say x has been inserted even if it hasn't been
 - But no false negative (x is inserted, but says not inserted)

Bloom Filter Applications

- When to use bloom filter?
 - If the false positive is not a concern, no deletion, and you look for space efficiency
- Original application: spell checker
 - 40 years ago, space is a big concern, it's OK to tolerate some error
- Canonical application: list of forbidden passwords
 - Don't care about the false positive issue
- Modern applications: network routers
 - Limited memory, need to be fast
 - Applications include keeping track of blocked IP address, keeping track of contents of caches, etc.

Bloom Filter Implementation: Components

- An array of n **bits**. Each bit 0 or 1
 - $n = b|S|$, where b is small real number. For example, $b \approx 8$ for 32-bit IP address (That's why it is space efficient)
- k hash functions h_1, \dots, h_k , each mapping inside $\{0, 1, \dots, n - 1\}$.
 - k usually small.

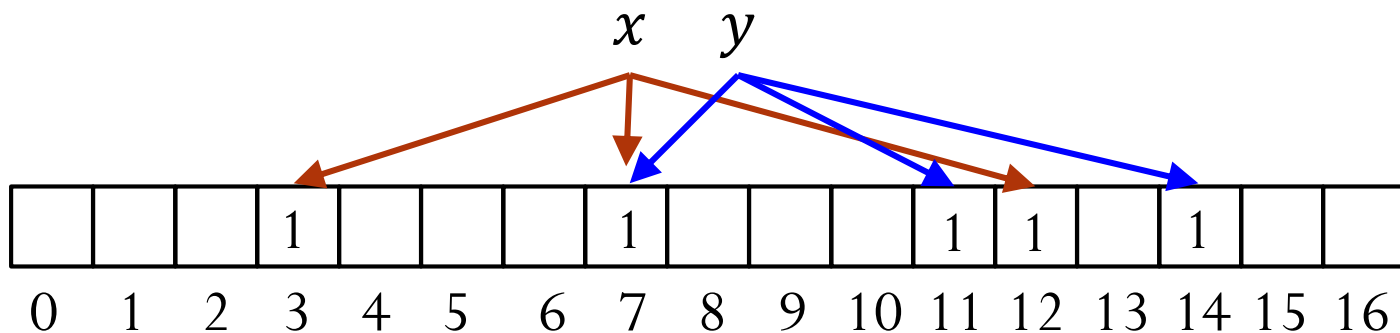
Bloom Filter Insert

- Initially, the array is all-zero.
- Insert x : For $i = 1, 2, \dots, k$, set $A[h_i(x)] = 1$
 - No matter whether the bit is 0 or 1 before

Example: $n = 17$, 3 hash functions

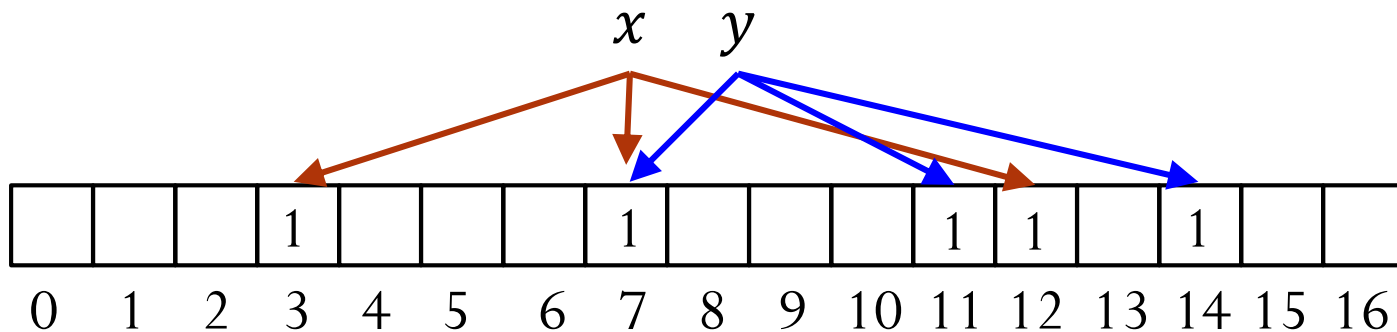
$$h_1(x) = 7, h_2(x) = 3, h_3(x) = 12$$

$$h_1(y) = 11, h_2(y) = 14, h_3(y) = 7$$



Bloom Filter Find

- Find x : return true if and only if $A[h_i(x)] = 1, \forall i = 1, \dots, k$



Suppose $h_1(x) = 7, h_2(x) = 3, h_3(x) = 12$. Find x ? Yes!

Suppose $h_1(z) = 3, h_2(z) = 11, h_3(z) = 5$. Find z ? No!

- No false negative: if x was inserted, $\text{find}(x)$ guaranteed to return true
- False positive possible: consider $h_1(w) = 11, h_2(w) = 12, h_3(w) = 7$ in the above example

Heuristic Analysis of Error Probability

- Intuition: should be a trade-off between space (array size) and false positive probability
 - Array size decreases, more reuse of bits, false positive probability increases
- Goal: analyze the false positive probability
- Setup: Insert data set S into the Bloom filter, use k hash functions, array has n bits
- Assumption: All k hash functions map keys uniformly random and these hash functions are independent

Probability of a Slot Being 1

- For an arbitrary slot j in the array, what's the probability that the slot is 1?
- Consider when slot j is 0
 - Happens when $h_i(x) \neq j$ for all $i = 1, \dots, k$ and $x \in S$
 - $\Pr(h_i(x) \neq j) = 1 - \frac{1}{n}$
 - $\Pr(A[j] = 0) = \left(1 - \frac{1}{n}\right)^{k|S|} \approx e^{-\frac{k|S|}{n}} = e^{-\frac{k}{b}}$
 - $b = \frac{n}{|S|}$ denotes # of bits per object
- $\Pr(A[j] = 1) \approx 1 - e^{-\frac{k}{b}}$

False Positive Probability

- For x not in S , the false positive probability happens when all $A[h_i(x)] = 1$ for all $i = 1, \dots, k$
 - The probability is $\epsilon \approx \left(1 - e^{-\frac{k}{b}}\right)^k$
- For a fixed b , ϵ is minimized when $k = (\ln 2) \cdot b$
- The minimal error probability is $\epsilon \approx \left(\frac{1}{2}\right)^{\ln 2 \cdot b} \approx 0.6185^b$
 - Error probability decreases exponentially with b
- Example: $b = 8$, could choose k as 5 or 6. Min error probability $\approx 2\%$