CSC263 - Week 5, Lecture 1

Cristyn Howard

Monday, February 5, 2018

Q: A webserver needs to reference a large list of blacklisted websites. The entire list cannot be stored in main memory, because it is too large. How can we store it?

A: In a BLOOM FILTER

Bloom Filters

- probablistic (i.e. approximate) dictionary that stores F_S : a summary/fingerprint of a dynamic set S

  BF$[0, ..., m-1]$: *array of* m *bits, initialized to* 0.

  $\{h_1, ..., h_t\}$: *set of* $t < m$ *hash fuctions where* $h_i : U \to \{0, ..., m-1\}$

  $m,\ t$: *parameters defining* $\#$ *of bits* $\&$ $\#$ *of hash functions, respectively*

- Operations:

  $\Theta(n)$ INSERT(F_S , x) : *for* i *from* 1 *to* t, $BF[h_i(x)] = 1;$

  * element passed to all hash functions, produces a t-tuple of indices $\in \{0, ..., m-1\}$ that form fingerprint of x
  * bits at all indices in fingerprint of x permanently set to 1

  $\Theta(n)$ SEARCH(F_S , x) : *for* i *from* 1 *to* t, *if (BF*$[h_i(x)] == 0$*) return FALSE, else return TRUE*

  * Returns: {FALSE ≡ x definitely not in F_S}||{TRUE ≡ x **probably** in F_S}
  * If any index in the hash of x is 0, then x has definitely not been placed in S.
  * However, if x is not in S, and the union of all indices in the hashes of all elements stored in S contains all indices in the hash of x, then SEARCH will return a **false positive**.

- Benefits: very space and time efficient way to store and search large sets.
- Limitations:
  - cannot delete elements from S
  - SEARCH may return *false positives* - may say that x is likely in S when it is not.
- Use when: space constraints exist, false positives are acceptable, and when deleting is not necessary.

False Positive Probability Analysis

Assume n elements inserted. Compute probability that SEARCH(BF, x) returns a false positive, i.e. a positive when x has not been inserted.

Recall that hash functions have uniform, independent probability distributions.