Lecture 9, Oct 11, 2023

Bloom Filters

- Bloom filters are like space-efficient "probabilistic dictionaries"
- Instead of storing the entire key, we will store only the hashes of a key in a set S
- The following operations are supported:
  - $\textsc{Insert}(S, x)$: insert x into S
  - $\textsc{Search}(S, x)$: search for x in S; this can have two results: "false" (in which case $x \notin S$ for sure) or "likely true" (in which case it is likely $x \in S$, but it could be a false positive)
- A bloom filter consists of an array $BF[0..m-1]$ of m bits, initially all set to 0, and t independent hash functions $h_1, \dots, h_t$ that all map to the range $[0, m-1]$
  - We will assume that these hash functions are all SUHA, i.e. any key is equally likely to be hashed into any slot
- On $\textsc{Insert}(S, x)$, we hash x using all t functions, resulting in t indices; the bits at all these indices are set to 1
- On $\textsc{Search}(S, x)$, we hash x using all t functions, and check that all the bits at those indices are 1; if this is true, then x is likely in S, otherwise it is definitely not in S
- Suppose we insert n keys into an empty Bloom filter with m bits and t independent hash functions all satisfying SUHA; what is the probability that searching for a key not in the filter will return a positive result?
  - Consider an arbitrary index i in the filter; the probability that a key hashes to i for each hash function is $\dfrac{1}{m}$, or $1 - \dfrac{1}{m}$ to miss i
  - Therefore with t has functions, the probability of i remaining zero is $\left(1 - \dfrac{1}{m}\right)^t$, since all hash functions are independent
  - After n keys are inserted, the probability of i remaining zero is now $\left(1 - \dfrac{1}{m}\right)^{nt}$
    - Assuming $\dfrac{1}{m}$ is small, then we can approximate this as $\left(e^{-\frac{1}{m}}\right)^{nt} = e^{-\frac{nt}{m}}$
  - For a false positive we require that all t indices that x hashes to are 1; however the probability that each individual index is 1 is technically not independent
  - In practice, we can assume that these events are independent to get a (pretty good) approximation that the probability of a false positive is $\left(1 - e^{-\frac{nt}{m}}\right)^t$
- How do we find the optimal size of t?
  - Fix the ratio $\dfrac{m}{n}$, and minimize $\left(1 - e^{-\frac{nt}{m}}\right)^t$ with respect to t
  - The optimal t turns out to be $\ln(2)\dfrac{m}{n} \approx 0.69\dfrac{m}{n}$
  - Substituting this back gives us $0.62^{\frac{m}{n}}$ as the chance of a false positive
  - e.g. allocating 8 bits per element gives us an optimal t of 5.52 (which we round to 6 has functions), giving us about 2% chance of false positives