Lecture 24: Compression

- Intro
- Lossless Compression
  - Some "buzzwords" & high-level ideas
    RLE, Huffman, LZ
  - Graphs
- Lossy Compression
  - Images, movies, music ...
  - Maintaining a set
    Bloom Filters
Maintaining a set

Given $S = \{X_1, \ldots, X_n\}$ \( X_i \in D \)

Goal a way of storing $S'$ so that membership queries:

- is $y \in S$?
- are supported.

Desire:
- small space
- fast query time

A solution:
- array $A$ of length $|D|$
  \[ A_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{o.w.} \end{cases} \]

- Space = $|D|$
- query time = 1 \( \leftarrow \text{best possible} \)
- can be huge!
Another Solution:

- Write $X_i$'s in sorted order

\[ X_{i_1} \leq X_{i_2} \leq X_{i_3} \leq \ldots \]

Space = $n \log |D|$ \quad (< best possible (see below))

Query time = $O(\log n)$ \quad (binary search)

↑ not terrible, but not great.

Can you do better?

- If $S$ can be any subset of $D$:

2^{101} \quad \text{subsets of } D

each needs separate encoding

\[ \Rightarrow \log (2^{101}) = 101 \text{ bits} \quad \text{required} \]

- If $S$ can be any subset of size $n$:

\[ \binom{101}{n} \quad \text{subsets of size } n \]

\[ \left( \frac{101}{n} \right)^n \leq \binom{101}{n} \]

\[ \Rightarrow \Theta(n \log 101) \text{ bits required} \]
But:

What if small false positive rate is ok?

i.e., if yes, always answer "yes"

if y & S, answer "no" for mostly

but might answer "yes"

(say for 10% of y's)

Why would this be reasonable?

Example: Dictionaries (Bloom)

- hyphenation of words

  easy words: 90%

  hard words: 10%

  Store S=hard words in compressed way

Given word w:

  Is w ∈ S?

  if yes do table lookup

  no apply simple rules

Tradeoff: cost of extra table lookups

vs. savings on storing S.
**Bloom Filters**

**Goal:** maintain a set $S = \{x_1, \ldots, x_n\}$ under inserts and queries

**Bloom Filter:** uses $k$ hash functions $h_1, \ldots, h_k$ mapping to $\{1, \ldots, m\}$
- array of $m$ bits
  - initially all 0

**Algorithm:**

**Insert $x$:**
- For each $1 \leq i \leq k$,
  - Set $h_i(x)$ to 1

**Query $x \in S$?**
- Output "yes" if for all $1 \leq i \leq k$, $h_i(x) = 1$
- else output "no"

**Behavior:**
- if $x \in S$, then always outputs "yes"
- if $x \notin S$, then might output "yes"
  - "False positive" might be ok if not too many
Example

\[ h_1(x) \rightarrow h_2(x) \rightarrow h_3(x) \]

\[ X_1 \rightarrow h_3(x_2) \rightarrow h_2(x_2) \rightarrow X_2 \]

might be in S (false positive)

Can't be in S since \( h_3(y_3) = 0 \)

might be in S (and is!)

\[ n = \# \text{ elements} \]

\[ m = \text{range of hash fn.} \]

\[ \equiv \text{size of Bloom Filter} \]

\[ k = \# \text{ of hash fn} \]
What is false positive rate?

Assume 1) $k/n \leq m$

- good upper bound on total # of bits set to 1 for $n$ elements with $k$ hash functs

2) hash functs are perfectly random (+ independent)

Some calculations:

1) After all $S$ hashed into $B.F$,:

$$p' = \text{prob } i^{th} \text{ bit of } BF \text{ is still 0}$$

$$p' = \left[ (1 - \frac{k}{m}) \right]^k \sum_{\text{prob} \text{ all } x \in S \text{ doesn't hash to } i \text{ under } h_j} \text{prob no } x \in S \text{ hashes to } i \text{ under any } h_j$$

$$\approx e^{-kn/m} \equiv p$$

(1) note: $p$ is a good approx to $p'$
2) What is \( p \) = fraction of 0 bits in BF after \( S \) inserted?

\[
p = \frac{1}{n} \sum \delta I_x \quad \text{where} \quad I_x = \begin{cases} 1 & \text{if } i^{th} \text{ bit of BF} \text{ still } 0 \\ 0 & \text{o.w.} \end{cases}
\]

\[
E[p] = \frac{1}{n} \sum I_x
\]

\[
= E[I_x]
\]

\[
= \text{Pr}[I_x = 1] = p'
\]

\( p + p' \) are close with high probability

3) \( \text{Prob [ } y \text{ is false positive] } \)

\[
= \text{Pr}(\text{all } h_x(y)'s \text{ land on a } "1")
\]

\[
= (1-p)^k
\]

\( \approx (1-p')^k \)

\( \approx (1-p)^k \)

\[
= (1-e^{-kn/m})^k
\]
How should we pick $k$?

- More hash functions gives a tradeoff
  - Helps find a 0 for $y \& S$
  - Decreases number of orbits in array
    - More likely to get false positive

- Find $k$ to minimize prob of false positive via derivatives

\[ k = (\ln 2) \left( \frac{m}{n} \right) \approx 0.7 \left( \frac{m}{n} \right) \]

\[ p = \frac{1}{2} \]

\[ \text{prob [false positive]} \approx (0.62)^m \]

Note: if need false positive rate < some constant, then $m = O(n)$ is sufficient!

\[ k = O(1) \]

⇒ Space = $O(|S|)$

Lookup time = $O(1)$

Can we beat Bloom Filters?

Any scheme with \( \leq \varepsilon \) false positive rate needs

\[ m \geq n \log \frac{1}{\varepsilon} \]

B.F. uses $m \geq (1.44) n \log \frac{1}{\varepsilon}$