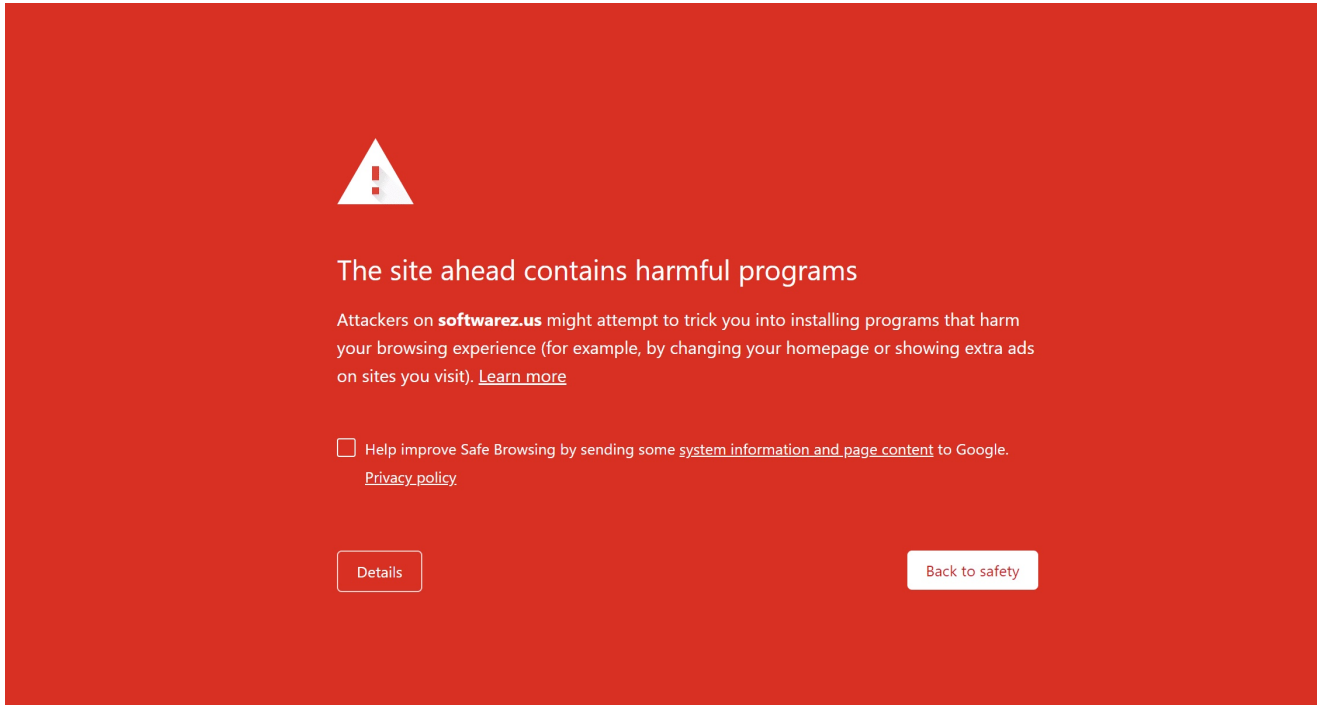
5112

11-28

# [Bloom] Filters

A common problem: have a set and want to know if a particular item is in the set.

For example web browsers store lists of malicious URLs.



So you know how to solve this using:
Hash tables, BSTs, linked list

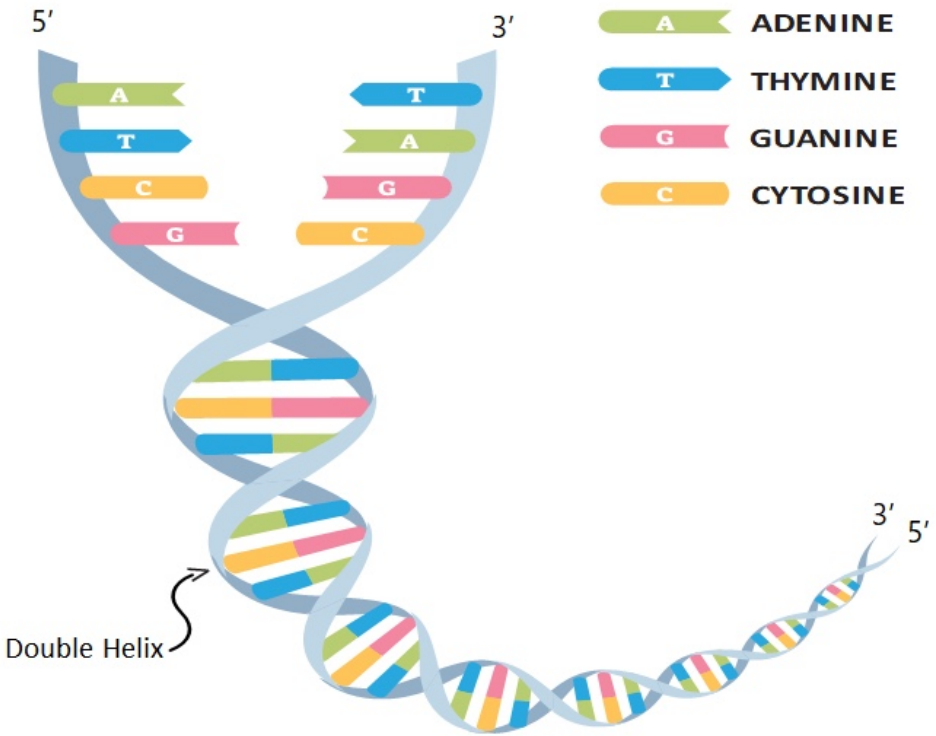
All of these store all the URLs → lots of space.

Query: Is x in my dataset?

Obvious answer: go look on the disk.


What if we could keep a compact representation in memory?

In genome sequencing, storing an entire genome is a huge amount of memory.




So how can I tell if a sequence likely belongs to a given species?

One technique is to store the k-mers of the reference species and query them for the k-mers found in the sample.

# Filters

## Randomized

- hash tables
- treaps
- skip lists
- quick select / sort

} always gen the right answer

## Approximate

- Misra-Gries
- Boyer-Moore

} deterministic only sometimes or partially correct.

A filter is a data structure that approximately stores a set $S$.

No false negatives: If $x \in S$, then Query$(x) \to$ YES.

Rare false positives: If $x \notin S$, then Query$(x) \to$ NO $1-\varepsilon$ of the time.

# A first idea

$S = \{A, L, G, O\}$

$h(L) = 3$

$h(A) = 10$

$h(G) = 17$

$h(O) = 12$

a bit array with $m$ bits to store $n$ items

$$\text{Query}(z) = \begin{cases} \text{YES} & \leq n/m \\ \text{NO} & \geq 1 - n/m \end{cases}$$

Insert: Hash to a location, and set the bit to 1.

Query: Hash to a location, and return YES if 1. NO if 0.

How large does $m$ need to be to guarantee 1% FPR?

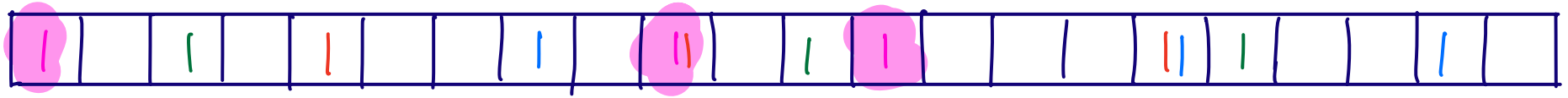
False positive rate is $\approx n/m$, so $m = 100n$. ∴

How can we do better?

# Bloom Filter

$S = \{A, G, L, O\}$

$h_1(A) = 10,\ h_2(A) = 1,\ h_3(A) = 13$



a bit array with m bits
to store n items

Insert: Use k hash functions to set locations to 1.

Query: Use k hash functions to determine locations.
Return NO if any one is 0.
YES otherwise.

Let's focus on bit $i$.

If $x \in S$ and some hash function $h$,

what's the prob. that $h(x) \neq i$? $1 - \frac{1}{m}$.

What's the prob. that across all $x \in S$ and $k$ choices of hash functions that $i$ is not set? $\left(1 - \frac{1}{m}\right)^{kn}$

Lemma: $\left(1 - \frac{1}{m}\right)^m \approx e^{-1}$ if $m$ is large.

$$\left(1 - \frac{1}{m}\right)^{m \cdot \frac{kn}{m}}$$

$$= e^{-kn/m}.$$

Set $\alpha = n/m$.

Prob ith bit is unset is $\approx e^{-\alpha k}$.

Prob of a false positive is prob $k$ randomly chosen bits are all set:

$$(1 - e^{-\alpha k})^k$$

Claim: This is minimized when $k = \alpha^{-1} \ln 2$.

Proof. Calculus.

Corollary: FPR is $2^{-\alpha^{-1} \ln 2}$

If we want an FPR of $\varepsilon$, what should $\alpha$ be?

$$2^{-\alpha^{-1}\ln 2} = \varepsilon.$$

$$-\alpha^{-1}\ln 2 = \log_2 \varepsilon$$

$$\alpha^{-1} = \frac{-\log_2 \varepsilon}{\ln 2} = \frac{\log_2 \frac{1}{\varepsilon}}{\ln(2)} \approx 1.44 \cdot \log_2\left(\frac{1}{\varepsilon}\right).$$

$$\frac{m}{n}$$

For 1% FPR: $\log_2(100) \approx 6.6$

$\Rightarrow \frac{\text{bits}}{\text{item}} \approx 9.6$

If we use 2B/item, FPR is 0.00046