- Given a set $\mathcal{S} = \{s_1, \ldots, s_m\}$ of strings from a universe U, preprocess $\mathcal{S}$ into a data structure so that to answer queries of the following form: given a string s, determine whether $s \in \mathcal{S}$.

  The objective is to preprocess $\mathcal{S}$ to build a data structure $\mathcal{D}$ of size $O(m)$ bits, and using $\mathcal{D}$, for any given query string s, decision algorithm answers correctly when $s \in \mathcal{S}$ and answers with a constant probability of error when $s \notin \mathcal{S}$.

- Let A be an array of size n bits, called a *fingerprint* of $\mathcal{S}$. Let $\mathcal{H} = \{h_1, h_2, \ldots, h_k\}$ be a collection of hash functions, where $h_i : \mathcal{S} \rightarrow A$ for $1 \leq i \leq k$. Each h_i is assumed to hash any input string independently and uniformly at random. It is also assumed any hash function $\mathcal{H}$ is encoded using a constant number of bits.

  Below, we preprocess $\mathcal{S}$ to build a fingerprint A of $\mathcal{S}$ by applying every hash function in $\mathcal{H}$ on every string in $\mathcal{S}$. The query algorithm probes a subset of entries of A to decide whether the input string $s \in \mathcal{S}$. Again, every entry of A that is probed is determined by applying a unique hash function in $\mathcal{H}$ on s, and exactly k entries of A are probed.

- Preprocessing algorithm to build a fingerprint A of strings in $\mathcal{S}$:

  1. for every i from 1 to n
  2. $\quad A[i] \leftarrow 0$ (denotes 0 is being stored in i^{th}-bit of A)
  3. for every $s_i \in \mathcal{S}$
  4. $\quad$ for every $h \in \mathcal{H}$
  5. $\quad\quad A[h(s_i)] \leftarrow 1$

- The preprocessing algorithm takes $O(n + mkt)$ time, where t is the maximum time any hash function in $\mathcal{H}$ takes to hash any s_i.

- The data structure A constructed in the preprocessing phase has $O(n)$ bits. The beauty of this data structure lies in fingerprint A of $\mathcal{S}$ being independent of length of any string in $\mathcal{S}$.

- Since each $h \in \mathcal{H}$ hashes independently and uniformly at random, for any $s_i \in \mathcal{S}$, after hashing s_i with h, $pr(A[j] = 1) = \frac{1}{n}$ for any $j \in [1, n]$.

  Specifically, after preprocessing algorithm exits, $pr(A[j] = 0) = (1 - \frac{1}{n})^{km}$ for any $j \in [1, n]$. ——- (1)

- Algorithm to query whether $s \in \mathcal{S}$:

  1. for every i from 1 to k
  2. $\quad$ if $A[h_i(s)]$ is 0 return "$s \notin \mathcal{S}$"
  3. return "$s \in \mathcal{S}$"

- Since hashing any string with any $h \in \mathcal{H}$ takes $O(t)$ time in the worst case, the query algorithm takes $O(kt)$ time in the worst case.

- The probability query algorithm outputs $s \in \mathcal{S}$ given $s \notin \mathcal{S}$ is equal to probability all the k locations of A probed by the query algorithm on behalf of s and hash functions in $\mathcal{H}$ are 1 while $s \notin \mathcal{S}$. From (1), the latter is equal to $(1-(1-\frac{1}{n})^{km})^k$, which is approximately $(1-e^{-km/n})^k$, since $(1-x) \leq e^{-x}$ for any $x \in \mathbb{R}$. ——- (2)

  Since the preprocessing algorithm sets $A[h(s_i)]$ to 1 for every $h \in \mathcal{H}$ and every $s_i \in \mathcal{S}$, all entries probed by the query algorithm for input s are guaranteed to be 1 for any $s \in \mathcal{S}$. Hence, the probability query algorithm outputs $s \notin \mathcal{S}$ given $s \in \mathcal{S}$ is 0.

- To minimize the error probability by choosing the right k in (2), we equate $\frac{d}{dk}(k \ln{(1-e^{-km/n})})$ to 0. This leads to $k = (\ln 2)(\frac{n}{m})$.

  Substituting k in (2), the probability of error when query string $s \in \mathcal{S}$ is, $(1-e^{-\ln 2})^{(\ln 2)\frac{n}{m}} \approx (0.6185)^{n/m}$.

  Significantly, as n/m increases, the probability of error falls exponentially. Note that n/m is the number of bits in fingerprint A per string in $\mathcal{S}$.

- When the fingerprint size n is chosen as $O(m)$,

  the number k of hash functions $\mathcal{H}$ is a constant,

  preprocessing algorithm takes $O(mt)$ time,

  fingerprint A computed after preprocessing $\mathcal{S}$ consumes $O(m)$ bits,

  the query time is a constant if t is a constant (since k is a constant), and

  for any string s input to query algorithm,

  if $s \in \mathcal{S}$ query algorithm answers correctly, and

  if $s \notin \mathcal{S}$ query algorithm errs with a constant probability (since k is a constant).