Definition: Let $H$ be a set of functions from $U$ to $T$. $H$ is 2-universal if $\forall x \neq y \in U$, $\alpha, \beta \in T$

$$
Pr_{h \in H}[h(x) = \alpha \wedge h(y) = \beta] = \frac{1}{|T|^2}
$$

Clearly, the set of all functions from $U$ to $T$ is 2-universal. However, the cardinality of this set is very large. Our goal is to construct a family of 2-universal hash functions whose size is small.

We will see some examples now.

Let $U = \{0, 1, 2, ..., p - 1\}$, and $T = \{0, 1, 2, ..., p - 1\}$, where $p$ is a prime.

$$
H = \{h_{ab} \mid a, b \in Z_p\}
$$

where

$$
h_{ab}(x) = ax + b ( \mod p)$$

Fix $x \neq y \in U$, $\alpha, \beta \in T$

$$
Pr_{a,b \in Z_p}[ax + b = \alpha \wedge ax + b = \beta] = \frac{|\{(a,b) \in Z_p^2 \mid ax + b = \alpha \wedge ay + b = \beta\}|}{|Z_p|^2} = \frac{1}{p^2} = \frac{1}{|T|^2}
$$

We have to show that $|\{(a,b) \in Z_p^2 \mid ax + b = \alpha \wedge ay + b = \beta\}| = 1$.

$$
|\{(a,b) \in Z_p^2 \mid ax + b = \alpha \wedge ay + b = \beta\}| = \text{the number of pairs } (a,b) \text{ satisfying the equations}
$$

$$
\begin{pmatrix}
  x & 1 \\
  y & 1 
  \end{pmatrix}
\begin{pmatrix}
  a \\
  b 
  \end{pmatrix}
= \begin{pmatrix}
  \alpha \\
  \beta 
  \end{pmatrix}
$$

since $x \neq y$, the inverse of $\begin{pmatrix}
  x & 1 \\
  y & 1 
  \end{pmatrix}$ exists and hence the result follows.

Now suppose $U = GF(2^m)$, and $T = GF(2^m)$.

$$
H = \{h_{ab} \mid a, b \in GF(2^m)\}
$$

where

$$
h_{ab}(x) = ax + b \text{ over } GF(2^m)
$$

It can be shown that $H$ is also 2-universal.

Now consider a more general case where $|T|$ is less than $|U|$. Let $U = GF(2^m)$, and $T = GF(2^n)$ where $m \geq n$.

$$
H = \{h_{ab} \mid a, b \in GF(2^m)\},
$$

where

$$
h_{ab}(x) = \text{first } n \text{ bits of } ax + b \text{ over } GF(2^m).
$$

We will now show that $H$ is 2-universal. Given a $n$ bit string $u$ let $S_u = u \sigma^{m-n}$.
Let \( \alpha \) and \( \beta \) be any two distinct strings from \( GF(2^m) \). Let \( u \) and \( v \) be any two strings from \( GF(2^n) \). Observe that \( h_{ab}(\alpha) = u \) if and only if \( a\alpha + b \in S_u \). Thus

\[
\Pr_{h \in \mathcal{H}}[h(\alpha) = u \wedge h(\beta) = v] = \Pr_{h \in \mathcal{H}}[h(\alpha) \in S_u \wedge h(\beta) \in S_v]
\]

\[
= \sum_{a \in S_u} \sum_{b \in S_v} \Pr_{h \in \mathcal{H}}[h(\alpha) = a \wedge h(\beta) = b]
\]

\[
= \frac{2^{2(m-n)}}{2^{2m}}
\]

\[
= \frac{1}{2^{2n}}
\]

**Static Dictionaries**

The static dictionary problem is one of the most fundamental data-structuring problems. Informally, an instance of the problem is given by a set \( S \) of keys, each associated with data, and the task is to store \( S \) in a way that allows rapid retrieval of the data of a given key.

Let \( U \) be set of possible key values and a \( S \) is a subset of \( U \). The goal is to store \( S \) in a table \( T \). Suppose \( h \) is function from \( U \) to \( T \) that is one-one on \( S \). Then we can use \( h \) to store \( S \) as follows: Given an element \( k \in S \), we store it at location \( T[h(k)] \). In future, if we want to search for a key \( a \), then we compare \( a \) with \( T[h(a)] \).

Now the question is how to find such a \( h \). Observe that if we allow \( |T| = |U| \), then this is trivial. However we want \( |S| \approx |T| \). To accomplish this, we use 2-universal hash functions. Let \( \mathcal{H} \) be a family of 2-universal hash functions. Let \( |U| = M, |S| = N \), we first consider the case when \( |T| = t = N^2 \). We show that a random function from \( \mathcal{H} \) is one-one on \( S \) with good probability.

For every \( x \) and \( y \) in \( S \), let

\[
C_{xy} = \begin{cases} 1 & \text{if } h(x) = h(y) \\ 0 & \text{else} \end{cases}
\]

\[
C = \sum_{x, y \in S, x < y} C_{xy}
\]

Thus \( C \) is random variable that denotes the number of colliding pairs.

\[
E(C_{xy}) = \Pr_{h \in \mathcal{H}}(h(x) = h(y)) = \frac{1}{t}
\]

\[
E(C) = \sum_{x,y \in S, x < y} \frac{1}{t} = \binom{N}{2} \frac{1}{t}
\]

\[
if \ t = N^2 \ 
E(C) = \frac{N(N-1)}{2} \frac{1}{N^2}
\]

\[
E(C) \leq \frac{1}{2}
\]
Using Markov inequality,
\[ \Pr(C < 1) \geq \frac{1}{2} \]

Thus, if we randomly pick a hash function then probability that there is no collision is at least half. If we randomly pick 100 hash functions then at least one of them is one-one on \( S \) is probability bigger than \( 1 - 1/2^{100} \).

Thus the time required to find \( h \) is \( O(N) \), the space required to store \( S \) in the size of \( T \) which is \( N^2 \). The query time is \( O(1) \). Next we see how two reduce \( |T| = O(N) \). For this we use a two-stage hashing.

If \( t = N \),
\[ E(C) \leq \frac{N}{2} \]

By Markov inequality,
\[ \Pr(C \geq N) \leq \frac{1}{2} \Rightarrow \Pr(C < N) \geq \frac{1}{2} \]

Thus for a random \( h \), expected number of colliding pairs is less than \( N \), with probability at least half. Pick such function \( h \). This is our hash function in first stage.

For \( i \in T \), let \( N_i \) is the set of elements from \( S \) that are mapped to \( i \) via \( h \). Let \( |N_i| = n_i \). Now, store \( N_i \) is in a secondary table of size \( n_i^2 \).

The total size of the table is given by
\[ N + \sum n_i^2 + \text{ space to store all hash functions} \]

Observe that
\[ C = \sum_{i \in T} \left( \frac{n_i}{2} \right), \]
and
\[ \sum_{i \in T} \left( \frac{n_i}{2} \right) = \sum \frac{n_i^2}{2} - \sum \frac{n_i}{2} \]

Thus
\[ 2C = \sum n_i^2 - \sum n_i, \]

Since \( C = N \) and \( \sum n_i \leq N \), \( \sum n_i^2 \leq 3N \).

The number of hash functions we need at the second stage is at most \( N \), and each has function needs \( O(\log |U|) \) space. Since we assumed that the word size of \( \log U \), it follows that that total space needed to store \( S \) is \( O(N) \).

In general, we would like to answer the following question? Let \( \mathcal{H} \) be a family of hash functions from \( U \) to \( T \). Let \( S \) be any subset of \( U \). If we randomly pick a function from \( \mathcal{H} \), then “how much one-one” is \( h \) on \( S \)? We now answer this question.

Let \( |U| = M \), \( |S| = N \), \( |T| = t \). An element \( i \in S \) is said to be unique with respect to \( h \) if \( h^{-1}[h(i)] \cap S = \{i\} \).

We would like to know how many unique elements can \( S \) have?. Let
\[ X_i = \begin{cases} 1 & \text{if } i \text{ is unique} \\ 0 & \text{else} \end{cases} \]
\[ E(X_i) = \Pr(X_i = 1) = 1 - \Pr(X_i = 0) \]

\[ \Pr(X_i = 0) = \Pr_h(\exists j \neq i \text{ and } j \in S, \ h(i) = h(j)) \]

\[ \Pr(X_i = 0) \leq \sum_{j \in S, j \neq i} \Pr(h(i) = h(j)) \text{ by union bound} \]

\[ \leq \frac{N - 1}{T} \]

Therefore

\[ \Pr(X_i = 1) = 1 - \frac{N}{T} + \frac{1}{T} \]

\[ E(X_i) \geq 1 - \frac{N}{T} \]

Let \( X \) = the number of unique elements of \( S \).

i.e.

\[ X = \sum_{i \in S} X_i \]

\[ E(X) \geq N - \frac{N^2}{T} \]

In particular if we take \( T = kN \), then \( E(X) = (1 - 1/k)N \).