The FKS hashing scheme:

**Theorem**

- A hash function maps a key $x$ to position $h(x)$.
- A set $H$ of hash functions is weakly universal if for any two keys $x, y \in U$ (with $x \neq y$),
  $$\Pr(h(x) = h(y)) \leq \frac{1}{m}$$
  where $h$ is picked uniformly at random from $H$.

Using weakly universal hashing:

- For any $n$ operations, the expected run-time is $O(1)$ per operation.
- But this doesn’t tell us much about the worst-case behaviour.

### Static Perfect Hashing

A static dictionary stores (key, value) pairs and supports:
- add(key, value), lookup(key) (which returns value) and delete(key)
- Every lookup takes $\mathcal{O}(1)$ worst-case time,
- Uses $\mathcal{O}(n)$ space,
- Can be built in $\mathcal{O}(n)$ expected time.

The construction is based on weakly universal hashing (with an $\mathcal{O}(1)$ time hash function).

### Perfect hashing - a first attempt

1. Insert everything into a hash table of size $m = n$
2. Check for collisions
3. Repeat if necessary

How many collisions do we get on average?

- Number of collisions $n m / 2$
- Linearity of expectation $\frac{1}{m}$
- Definition of expectation $\frac{n^2}{2}$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

### Perfect hashing - a second attempt

- A set $H$ of hash functions is weakly universal if for any two keys $x, y \in U$ (with $x \neq y$),
  $$\Pr(h(x) = h(y)) \leq \frac{1}{m}$$
  where $h$ is picked uniformly at random from $H$

Using weakly universal hashing:

- For any $n$ operations, the expected run-time is $O(1)$ per operation.
- But this doesn’t tell us much about the worst-case behaviour.

### Dictionaries and Hashing recap

- A dynamic dictionary stores (key, value) pairs and supports:
  - add(key, value), lookup(key) (which returns value) and delete(key)

### Expected construction time

1. Insert everything into a hash table of size $m = n^2$
2. Check for collisions
3. Repeat if there was a collision

How many times do we repeat on average?

The expected number of collisions $E(C) \leq \frac{1}{2}$

The probability of at least one collision $\Pr(C \geq 1) \leq \frac{1}{2}$

- i.e. at least as good as tossing a heads on a fair coin
  $$E(1_{\text{heads}}) \leq \frac{1}{2}$$

The probability of zero collisions is at least $\frac{1}{2}$

- $E(1_{\text{heads}}) \geq \frac{1}{2}$

... and then the look-up time is always $O(1)$

(because any $h_i(x)$ can be computed in $O(1)$ time)
Perfect Hashing

**Expected construction time**

1. Insert everything into a hash table, $T$, of size $n$ using a weakly universal hash function, $h$

2. Check for collisions

3. Repeat if there are more than $n$ collisions

The expected number of collisions: $E(C) = \binom{n}{2}$

The probability of at least $n$ collisions: $P(C \geq n) = \frac{1}{2}$

The probability of at most $n$ collisions is at least $\frac{1}{2}$

...but the look-up time could be rubbish (lots of collisions)

**Perfect hashing - attempt three**

1. Compute $i = h(x)$ (the key)
2. Compute $j = h_j(x)$
3. The item is in $T_j[i]$

Let $n_j$ be the number of items in $T_j$.

**Perfect Hashing - Space usage**

- Insert everything into a hash table, $T$, of size $n$ using a weakly universal hash function, $h$.
- Immediately repeat if either
  a) $T$ has more than $n$ collisions
  b) some $T_j$ has a collision

The overall expected construction time is therefore:

$$E(\text{construction time}) = E\left(\text{construction time of } T + \sum_j \text{construction time of } T_j\right)$$

$$= E(\text{construction time of } T) + \sum_j E(\text{construction time of } T_j)$$

$$= O(n) + \sum_j O(n_j^2) = O(n) + O\left(\sum_j n_j^2\right) = O(n)$$
Perfect Hashing - Summary

Step 1: Insert everything into a hash table, $T$, of size $n$
using a weakly universal (w.u.) hash function, $h$

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$
of size $n_i^2$ using w.u hash function $h_i$

(Step 3) Immediately repeat if either
a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision

THEOREM
The FKS hashing scheme:
• Has no collisions
• Every lookup takes $O(1)$ worst-case time,
•Uses $O(n)$ space,
• Can be built in $O(n)$ expected time.

In fact this scheme can be made dynamic
with $O(1)$ expected time inserts and deletes
but occasionally the inserts take $\Theta(n)$ time.