Universal Hashing.

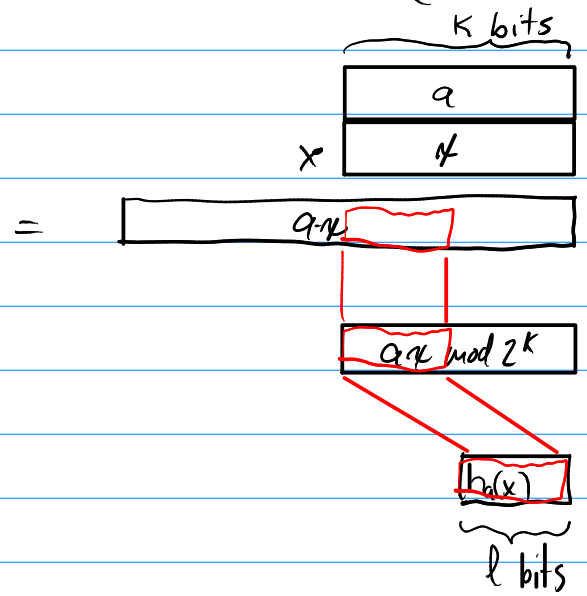
We have $n \approx 2^\ell$ integers in $U = \{0, \ldots, 2^K - 1\}$.

We want to store them in a table of size 2^ℓ, $\ell < K$.

Let $A = \{1, 3, 5, \ldots, 2^K - 1\}$.

Pick $a \in A$ uniformly at random and use the hash function

$$h_a(x) = (ax \bmod 2^K) \operatorname{div} 2^\ell$$



Similar to the "Multiplication Method" in §12.3.2 of CLRS.

In C:

```
unsigned int hash(unsigned int x) {
    return a*x >> (W-ℓ)
}
```

→ # of bits in an unsigned int.
- usually #define W 32

Skip to page 4 to see how this theorem is used.

Theorem: If $x, y \in \{0, \ldots, 2^k - 1\}$ and $x \neq y$, then $\Pr\{h_a(x) = h_a(y)\} \leq 1/2^{\ell-1}$

Suppose $x, y \in U$ with $x \neq y$.

We want to study $\Pr\{h_a(x) = h_a(y)\}$.

$$(ax \bmod 2^k) \operatorname{div} 2^\ell = (ay \bmod 2^k) \operatorname{div} 2^\ell$$

$$\Rightarrow |ax \bmod 2^k - ay \bmod 2^k| < 2^{k-\ell}$$

notice: $a(x-y) \bmod 2^k \neq 0$ since a is odd.

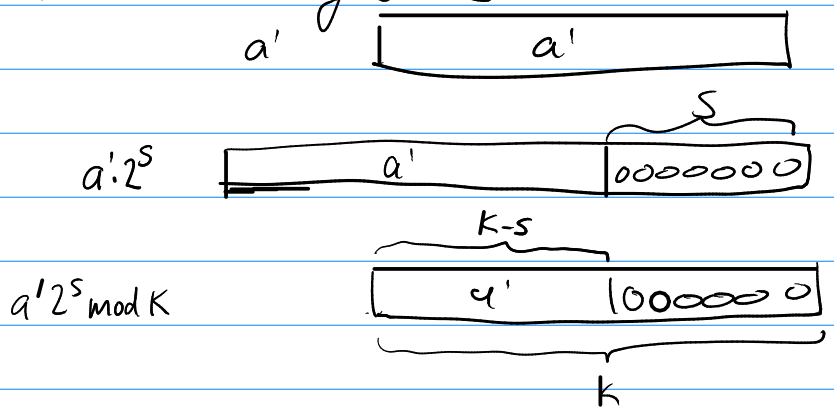
$$\Rightarrow a(x-y) \bmod 2^k \in \{1, 2, \ldots 2^{k-\ell} - 1\} \cup \{2^k - 2^{k-\ell} + 1, \ldots, 2^k - 1\}$$

Call this B

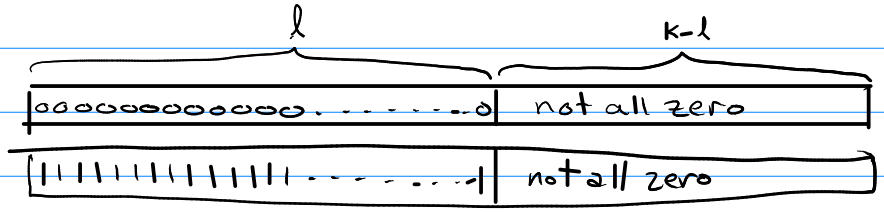
Let z be and odd integer so that $x - y = 2^s z$
Let $a' = az$ and notice that a' is uniformly distributed in A [because $a \to az$ is a permutation of A and a is chosen uniformly in A]

$$\Pr\{a(x-y) \bmod 2^k \in B\}$$
$$= \Pr\{a 2^s z \bmod 2^k \in B\}$$
$$= \Pr\{a' 2^s \bmod 2^k \in B\}$$

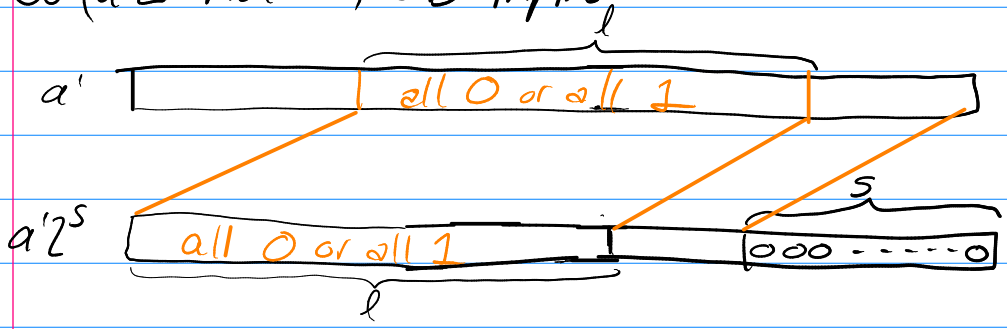
But $a' 2^s \bmod 2^k$ is just the lower-order $k-s$ bits of a' followed by s 0's.



The elements in B look like:



So $(a' 2^s \bmod 2^k) \in B$ implies



There are 2^l possible choices for these bits, so $\Pr\{(a' 2^s \bmod 2^k) \in B\} \leq 2/2^l = 1/2^{l-1}$.

QED.

How to use Theorem.

To store n elements of a set S, make an array A of size $m = 2^\ell$ where $\ell = \lceil \log_2 n \rceil$.

Note: $n \leq m \leq 2n$

$A[i]$ stores a list L_i of all elements $x \in S$ such that $h_a(x) = i$.

Time to search for x is the length of the list $L_{h_a(x)}$.

For each $y \in S \setminus \{x\}$, define

$$I_y = \begin{cases} 1 & \text{if } h_a(x) = h_a(y) \\ 0 & \text{otherwise} \end{cases}$$

By Theorem 1, $E[I_y] = 0 \cdot Pr\{h_a(x) \neq h_a(y)\} + 1 \cdot Pr\{h_a(x) = h_a(y)\}$
$\leq 1/2^{\ell-1} = 2/m \leq 2/n$

$E[\text{Length of list } L_{h_a(x)}] = 1 + E\left[\sum_{y \in S \setminus \{x\}} I_y\right]$

(1 = if x is in the list)

$$= 1 + \sum_{y \in S \setminus \{x\}} E[I_y]$$

$$\leq 1 + (n-1) \cdot 2/n \leq 3.$$

Theorem: Using universal hashing we can store a set $S \subseteq \{0, \ldots, 2^k - 1\}$, $|S| = n$, in a data structure of size $O(n)$ that can find any element $x \in S$ in $O(1)$ expected time

Later, we will see that we can also insert into and delete from S in $O(1)$ amortized expected time per operation.