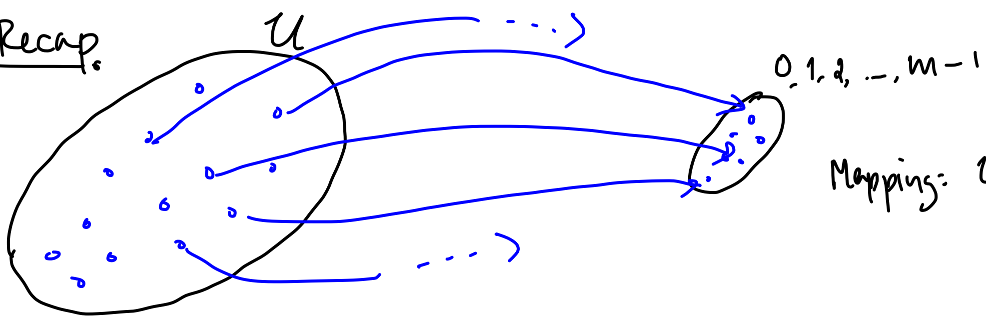
L4: Cuckoo Hashing

Hashing Recap:



Mapping: $\mathcal{U} \to [m]$

- Typically not talked about as a single function
- A Hash family: $\mathcal{H} = \{ \text{functions } \mathcal{U} \to [m] \}$
- Most commonly: Draw a hash function $h \in_R \mathcal{H}$ u.a.r.
  - → Give a sense of how hashing is "random"
- Ex: $\mathcal{H} = \{ x \mapsto (a \cdot x + b) \% p \mid a \geq 1, \text{ and } b \in 0..p-1 \}$
  - Draw a function by randomizing a & b.
  - Not a good hash func.

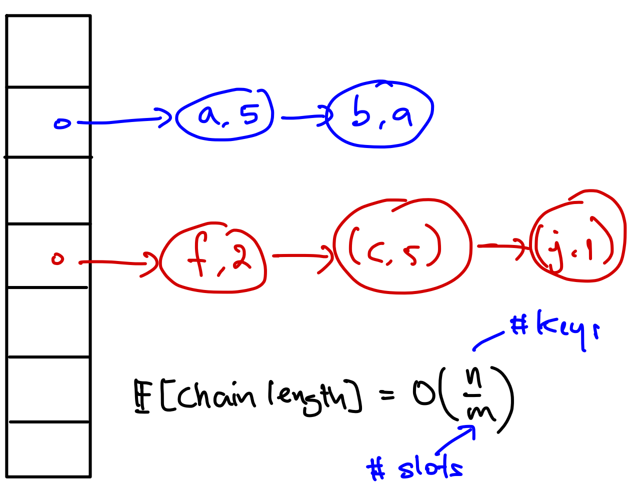
- Ideal World: Different keys all hash to different values [No collisions]
- Perfect hashing is a thing. Unclear benefits in practice esp. with inserts/del's

→ Most often: collision resolution

Collision Resolution Schemes:

→ What to do when keys k_1 & k_2 hash to the same number?

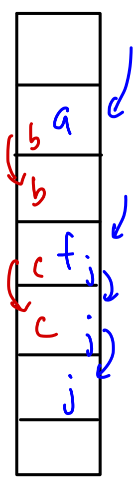
idea #1: Chaining



$\mathbb{E}[\text{Chain length}] = O\left(\frac{n}{m}\right)$ — # keys: n; # slots: m

→ $\mathbb{E}[\text{max chain len}] = O\left(\frac{\log n}{\log\log n}\right)$

idea #2 Open addressing.



$\mathbb{E}[\# \text{trials}] = O(1)$
a.k.a. probes

Summary:

① Insertion / Query: $O(1)$ expected.

② Space: $O(n)$

③ Worst-case + reality: worse than $O(1)$

[was told]

Cuckoo Hashing: → after the Cuckoo birds. Nesting behavior: eject existing egg to lay one's own

→ Search is worst-case $O(1)$ & insert is $O(1)$ expected.
(delete)

→ a Cuckoo hash table stores ① an array a of size m ($m = cn$, $c = 4$)
② Two hash functions h_1 & h_2 : $\mathcal{U} \to [m]$.

→ Invariant: Any key $x \in S$ is either at $a[h_1(x)]$ or $a[h_2(x)]$

set we're storing

→ Search & delete are (dirt) simple!

→ We'll focus on insert's.

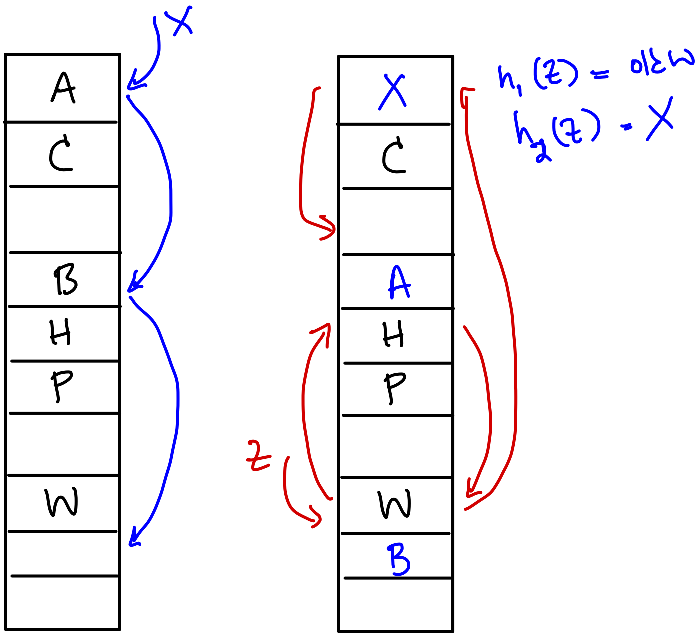
value omitted + assume: k not yet in there

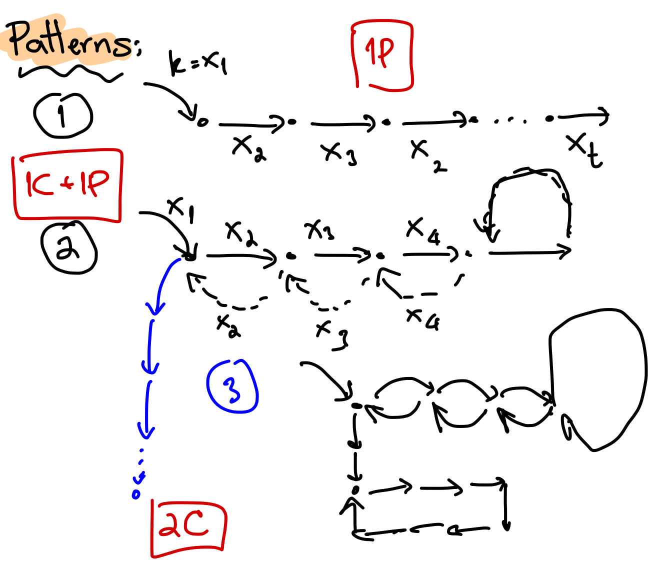
iterate

```
def insert(k):{
    pos = h_1(k)
    for _ in range(n):
        if a[pos] is empty:
            a[pos] = k; return;
        swap(&k, a[pos])  // kick out a[pos] into k && put old k in there instead.
        if pos==h_1(k): pos = h_2(k) else pos = h_1(k)
    reboot; insert(k)
}
```

pick 2 new hash functions and start over

Ex:





One simple/subtle change: □ ~~Rehash~~ Reboot more quickly

□ for _ in range(50.$\log_2 n$) { ...}

Analysis Sketch: T = time to carry out an insertion

$P_t = \mathbb{1}\{$form a path of length $\geq t\}$

$C_t = \mathbb{1}\{$ " —— " one cycle + 1p with #unique edges $\geq t\}$

$D_t = \mathbb{1}\{$ " —— " double cycle " —————— " $\}$

$$\mathbb{E}[T] \leq \mathbb{E}\left[\sum_{t=1}^{\infty} P_t + 2\sum_{t=1}^{\infty} C_t\right] + \underset{\text{reinsert } n \text{ keys}}{n} \cdot \mathbb{E}[T] \underbrace{\left[\sum_{t=1}^{\infty} \Pr(D_t = 1) + \Pr\left[\begin{smallmatrix}\text{exceeds}\\ 50\lg n\end{smallmatrix}\right]\right]}_{\alpha = \text{reboot prob.}}$$

$$\Rightarrow \mathbb{E}[T] \leq \frac{1}{\underset{\text{is } 1-o(1)}{1-\alpha \cdot n}} \underbrace{\left[\sum_{t=1}^{\infty} \mathbb{E}[P_t] + \sum_{t=1}^{\infty} \mathbb{E}[C_t]\right]}_{O(1).}$$

Claim:

→ Will do $\mathbb{E}[P_t]$ – the other RVs are similar.

→ Note: $\mathbb{E}[P_t] = \Pr[P_t = 1] = \Pr[$ 1P case with length $\geq t]$

$$\xrightarrow{x_1 = k} y_1 \longrightarrow y_2 \longrightarrow \cdots \xrightarrow{x_{k+1}} y_{k+1}$$

(A) Fix $\{x_2, \dots, x_{k+1}\}$ & $\{y_1, \dots, y_{k+1}\}$

x_1 is given

→ Prob k hashes to y_1 is $1/m$

→ Prob $y_i \to y_{i+1}$ is an edge is

$$\leq \underset{\substack{h_1(x_i) \text{ is } y_i \\ \& \ h_2(x_i) \text{ is } y_{i+1}}}{\frac{1}{m^2}} + \underset{\substack{h_1(x_i) \text{ is } y_{i+1} \\ \& \ h_2(x_i) \text{ is } y_i}}{\frac{1}{m^2}}$$

OR

→ Prob: $\frac{1}{m}\left(\frac{2}{m^2}\right)^k$

(B) How many such combinations?

edge choices $\leq n^k$

vtx choices $\leq m^{k+1}$

$\Rightarrow n^k \cdot m^{k+1}$

(C) $\mathbb{E}[P_t] \leq \frac{1}{m} \cdot \left(\frac{2}{m^2}\right)^k \cdot n^k \cdot m^{k+1}$

Using $m = 4n$.

$\hookrightarrow \leq \left(\frac{2}{4}\right)^k$

$$\Rightarrow \mathbb{E}\left[\sum_{t=1}^{\infty} P_t\right] \leq \sum_{t=1}^{\infty} 1/2^t = 1 \ \checkmark$$