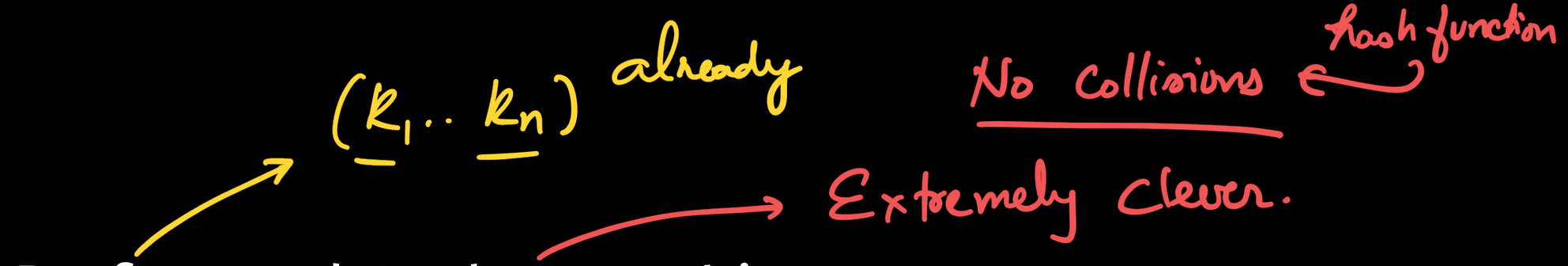


Perfect and Cuckoo Hashing

Data Structures And Algorithms

Sriram Sankaranarayanan

March 13, 2021

Department of Computer Science

Introduction

Why is Hashing not *Perfect*?

→ Hash Collisions

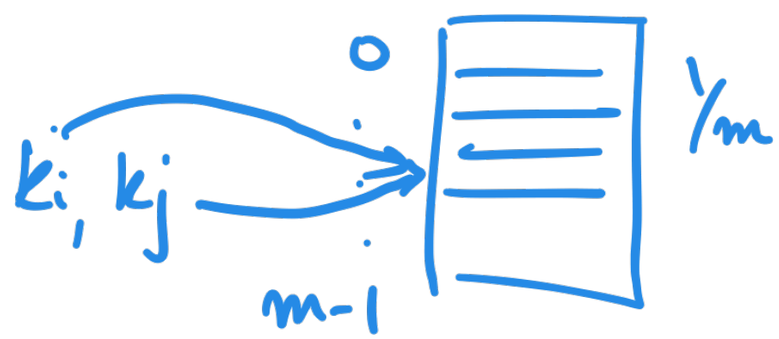
⇝ Chaining or Open Addressing

⇝ Worst Case Complexity can be bad.

→ Universal Hash Functions as a fix?

Recall: Universal Hash Function Families

Family of hash functions $\mathcal{H} : \{h_1, \ldots, h_N\}$.



$$h_i : \text{Keys} \to \{0, \ldots, m-1\}.$$

Guarantee: If we randomly choose two *different* functions h_i, h_j from this family.

- For any keys k_i, k_j where $k_i \neq k_j$
- Probability they collide for randomly chosen hash function :

$$\mathbb{P}_{h \in \mathcal{H}}(h(k_i) = h(k_j)) \leq \frac{c}{m} \sim \frac{1}{m}$$

randomly

Recall: Universal Hash Function Families

Every time a new hashtable is created:

Randomly choose a function $h \in \mathcal{H}$.

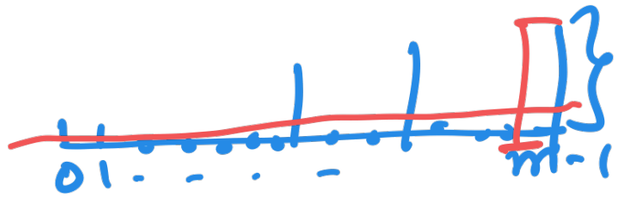
Suppose we insert n distinct keys $k_1, \ldots, k_n$.

Fix any one of the keys k_i. How many other keys will "collide" with k_i?

$$\sum_{j=1, j \neq i}^{n} \mathbb{P}(k_i \text{ collides with } k_j) \leq n \frac{c}{m}.$$

Average length of each "chain" $= c \times \left(\frac{n}{m} + 1\right) = c\alpha + 1$

“Expected Worst Case” for Chaining



n = # of keys

$\alpha < 1/2$

Lesson: Even if the average is small, there may be large out-liers.

Question: How big can the largest chain be?

Answer: We can expect the largest chain to be as large as $O(\frac{\log(n)}{\log(\log(n))})$ (Assume load factor is less than half).

1. *Deal breaker?* Say $n = 2^{32}$ (nearly 4 billion), we have $\frac{\log_2(n)}{\log_2(\log_2(n))} \approx \frac{32}{5} = 6$.
2. Nevertheless, it is not a constant.

Improvements: Perfect Hashing and Cuckoo Hashing

No collisions

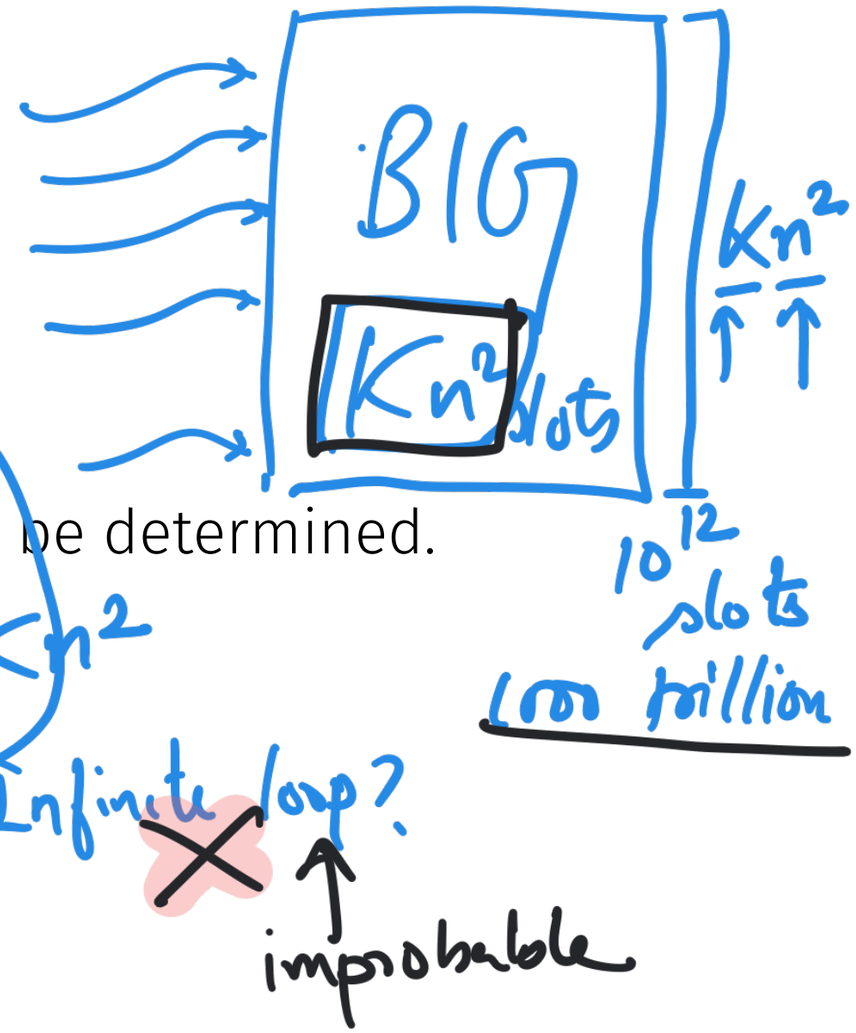
"Clever"

Perfect Hashing

Basic Idea

Note: n keys to be inserted available *upfront*.

1. Choose a random hash function: $h \in \mathcal{H}$
2. Hashtable with Kn^2 slots, where K is a parameter to be determined.
3. Insert each key into hash table.
4. If collision happens: ~~abort~~ and redo procedure.



Perfect Hashing Analysis

Pr. of collision of two keys $\leq \frac{c}{m} \leq \frac{c}{Kn^2}$

Probability that at least one collision occurs:

$$\mathbb{P}(\text{at least one collision}) \leq \frac{c}{Kn^2} \times \frac{n(n-1)}{2} \leq \frac{c}{2K}$$

Boole's ineq.

Set $K = 2c$,

$$\mathbb{P}(\text{at least one collision}) \leq \frac{1}{4}.$$

Question: What is the probability that any iteration of perfect hashing fails?

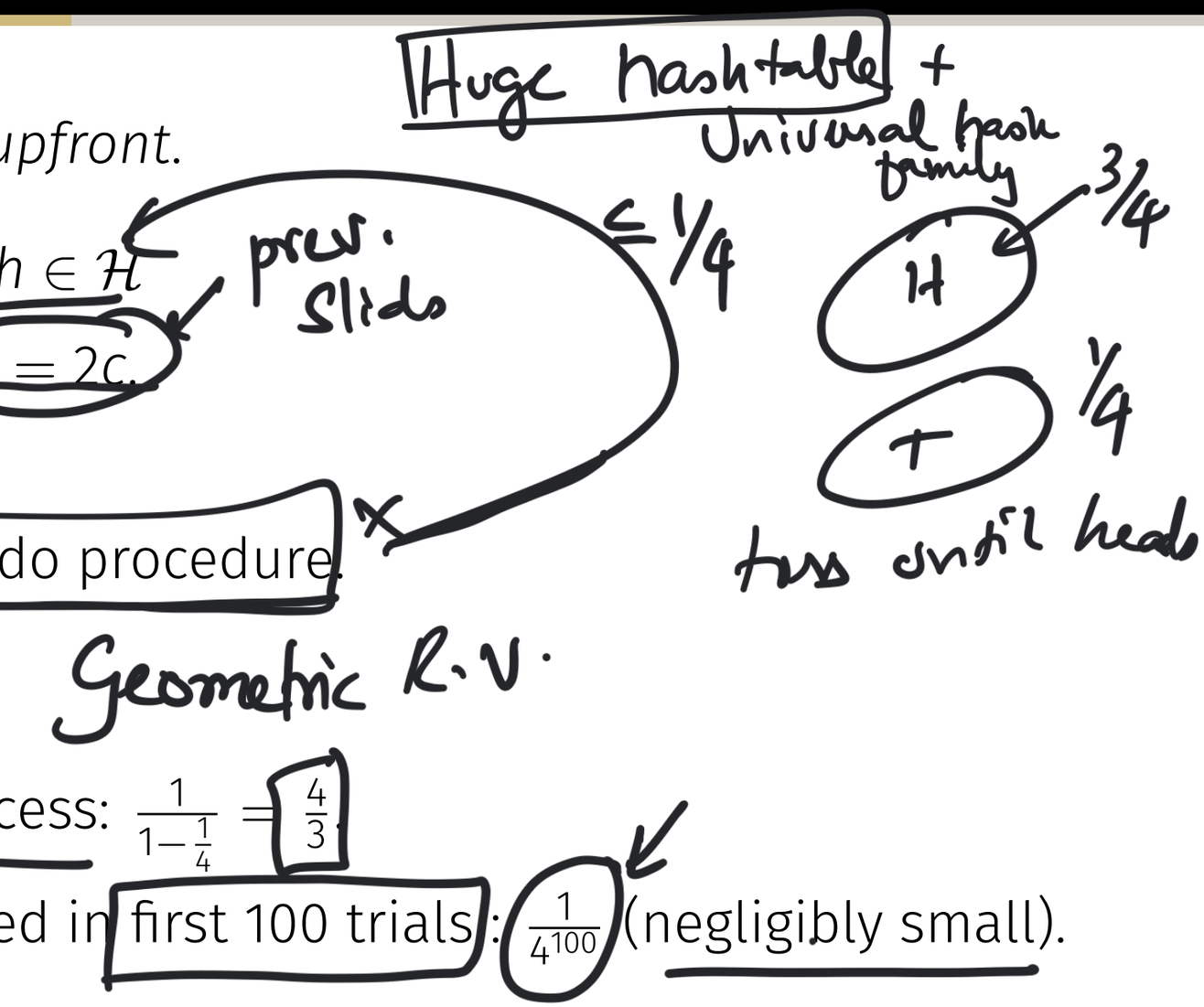
Perfect Hashing: First Try

Note: n keys to be inserted available *upfront*.

1. Choose a random hash function: $h \in \mathcal{H}$
2. Hashtable with Kn^2 slots, where $K = 2c$.
3. Insert each key into hash table.
4. If collision happens: abort and redo procedure.

Fails with probability $\frac{1}{4}$ on any trial.

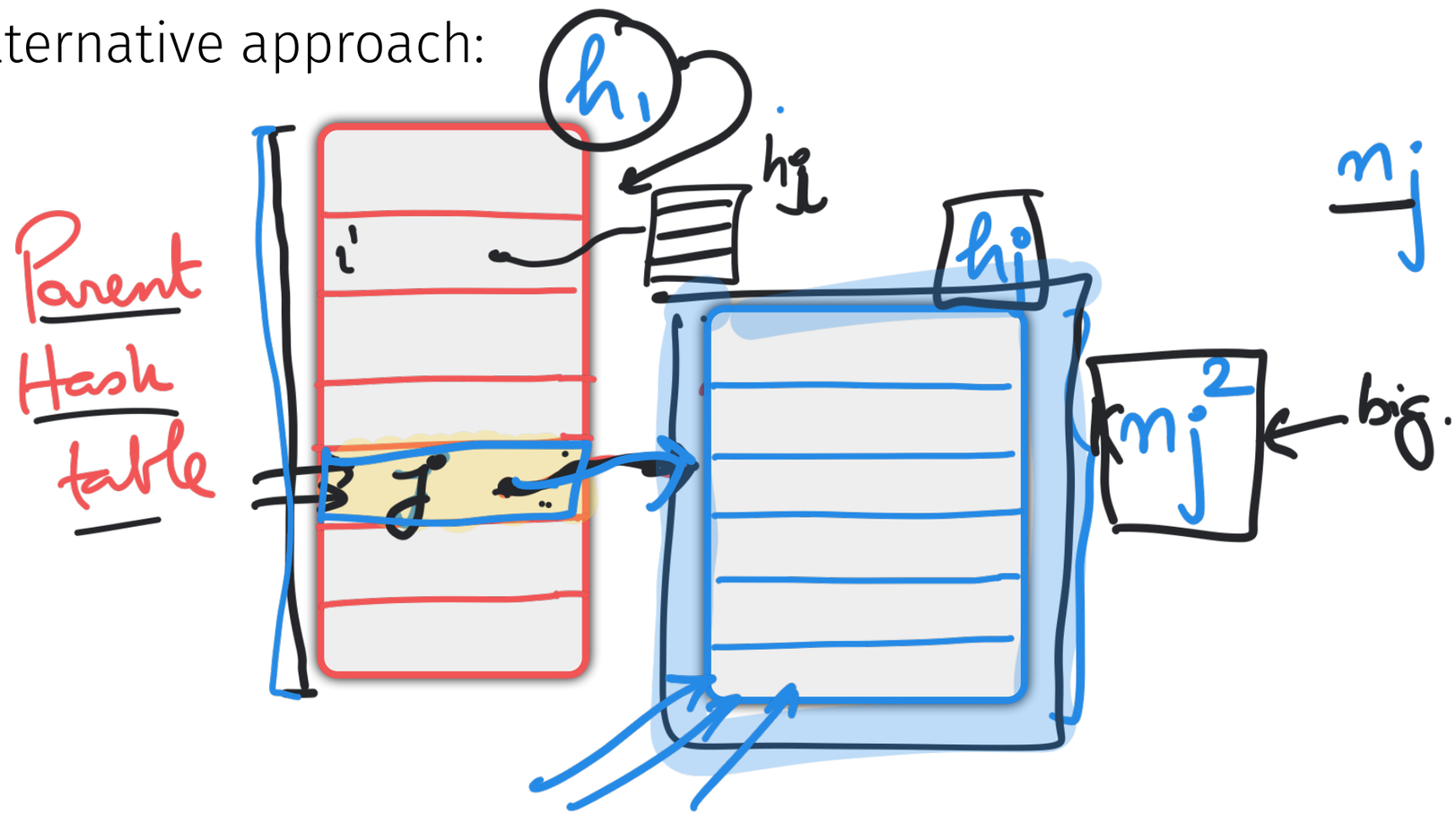
- Expected number of trials for success: $\frac{1}{1-\frac{1}{4}} = \frac{4}{3}$.
- Probability that it does not succeed in first 100 trials: $\frac{1}{4^{100}}$ (negligibly small).



Perfect Hashing: Second Try

Designing Kn^2 slots up front is quite wasteful.

Alternative approach:



n_j : # of elements that hash to i^{th} slot.

$\leq Cn$.

Perfect Hashing: Total Space Requirements

Suppose we insert n elements into a "two level" perfect hash table.

Total number of slots $\leq \sum_{j=1}^{n} n_j^2$, n_j = number of elements that collide with key k_j.

n

$\sum_{j=1}$

$E(n_j^2) \leq$ Constant for c

each j

$\therefore$ total size of all the second level hash tables $\leq cn$

$$\mathbb{E}(n_j^2) = \mathbb{E}\left(\left(\#\text{ of keys that collide with } k_j\right)^2\right)$$

$$= \left(\sum_{i=1}^{n} \left(\Pr(k_i \text{ collides with } k_j)\right)\right)^2$$

$$\leq \left(\sum_{i=1}^{n} \Pr(k_i \text{ collides with } k_j)\right)^2$$

$$+ 2 \sum_{i=1}^{n} \cdot \sum_{l=1}^{n} \Pr(k_i \text{ collides with } k_j)$$

$$\Pr(k_l \text{ collides with } k_j)$$

$$\leq 1 + (n-1)\frac{c^2}{n^2} + 2(n-1)\frac{c}{n} + 2(n-1)^2\frac{c^2}{n^2}.$$

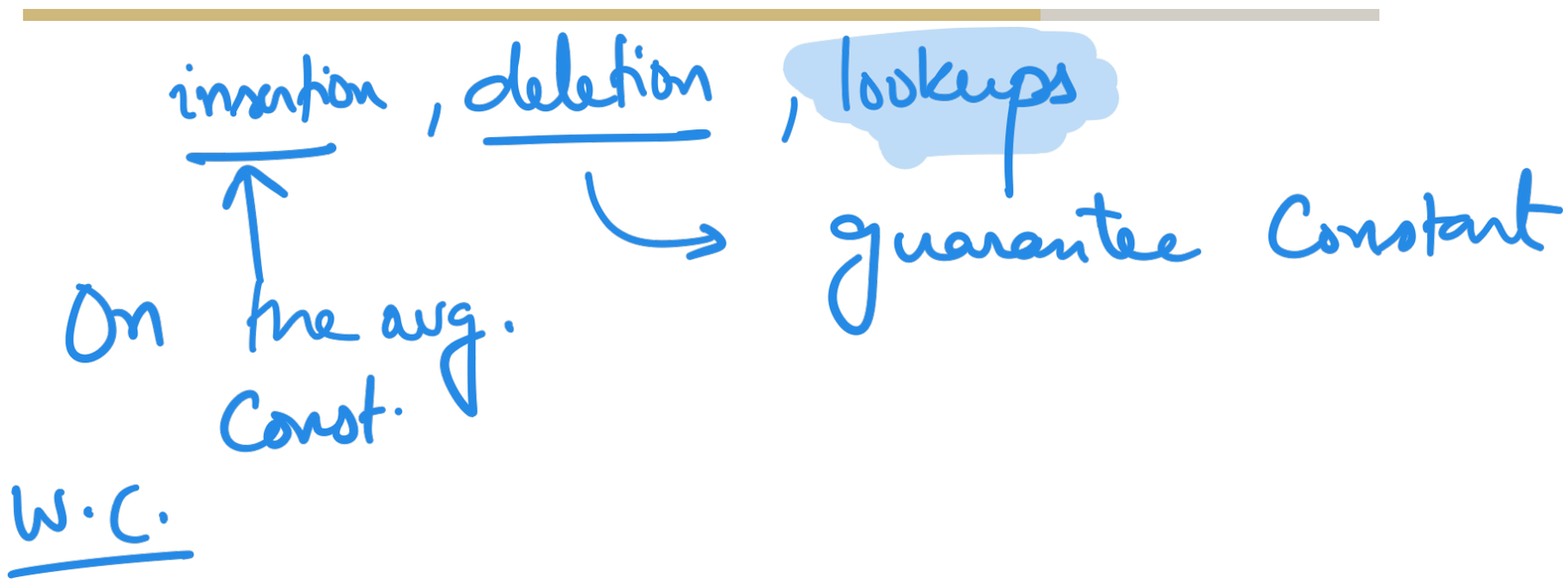
$$\leq \text{Constant}.$$

$$\sum_{j=1}^{n} \mathbb{E}(n_j^2) \leq (\text{Constant}) \times n.$$

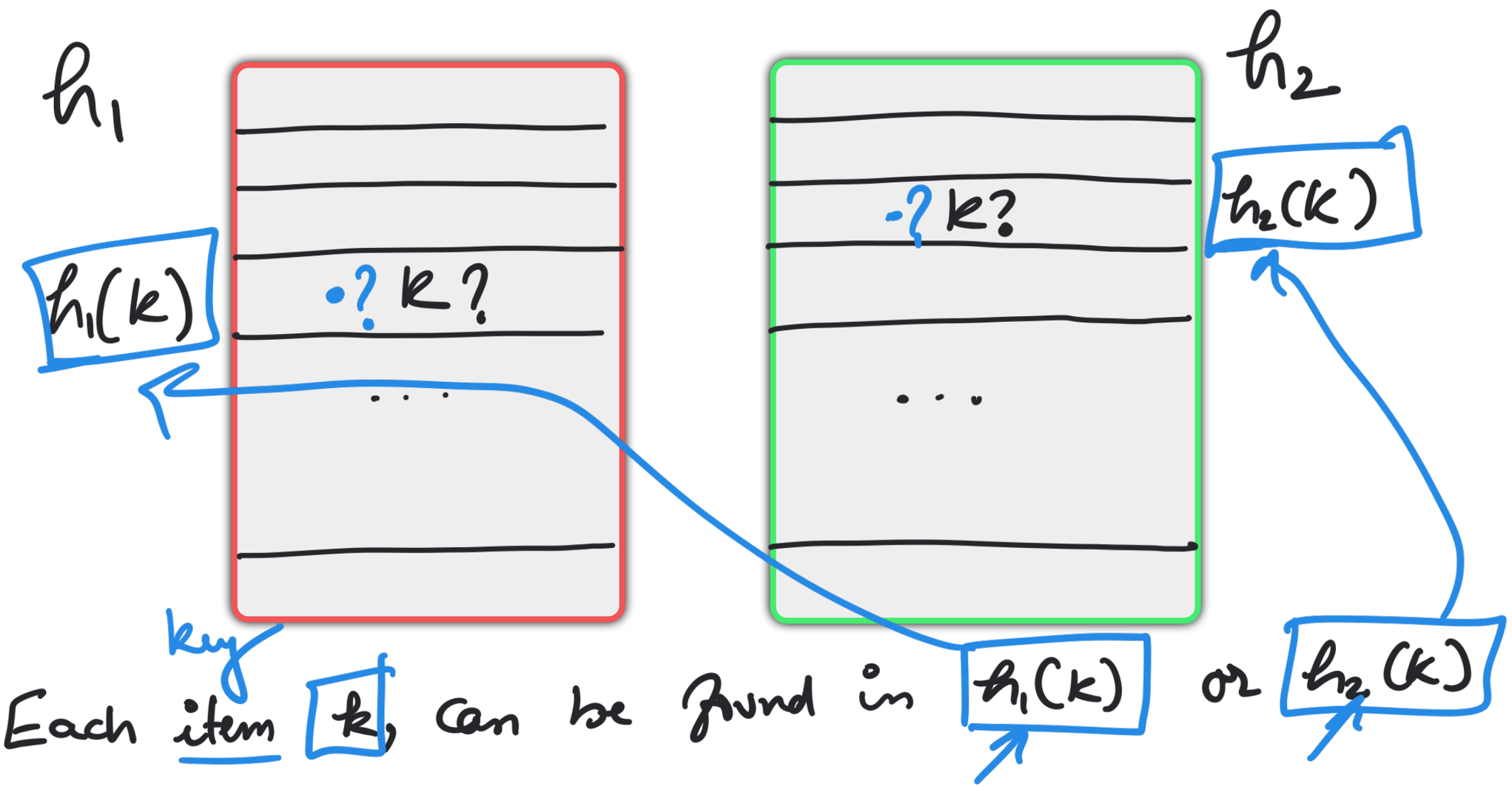
Perfect Hashing: Problems

- All keys need to be known up-front.
- Cannot insert/delete new keys.
- More complex data structure:
  - Hash table whose slots are themselves hash tables.

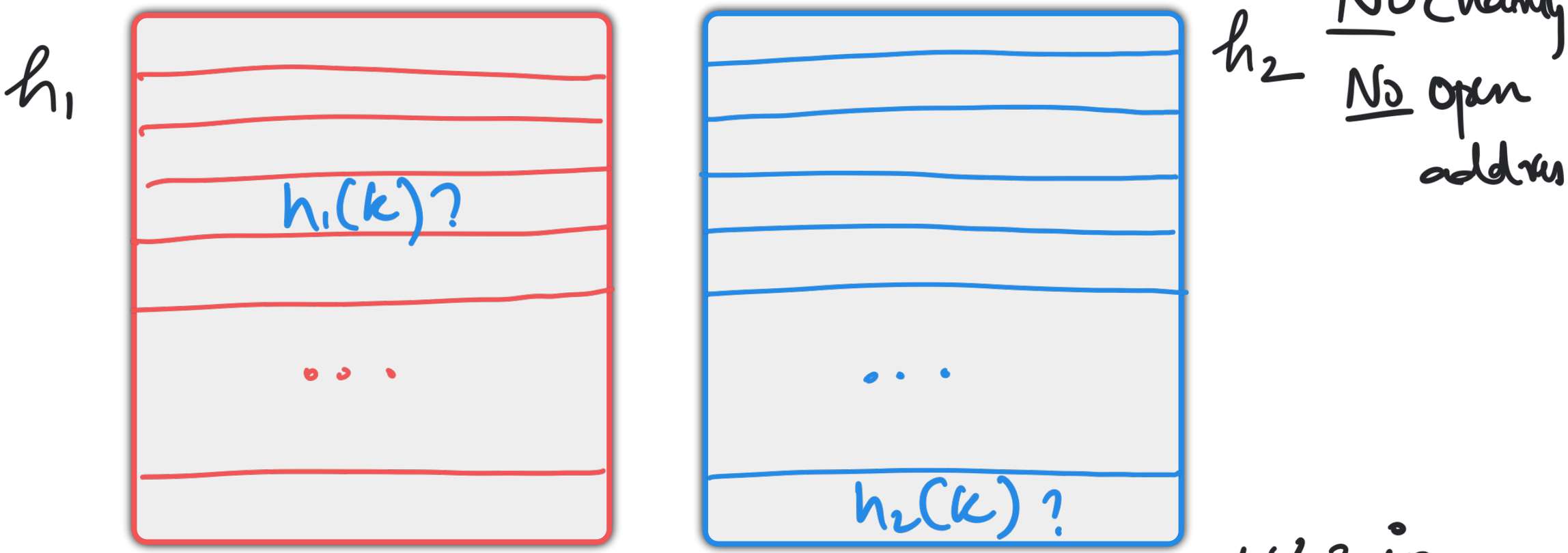
Cuckoo Hashing



Cuckoo Hashing: Basic Idea

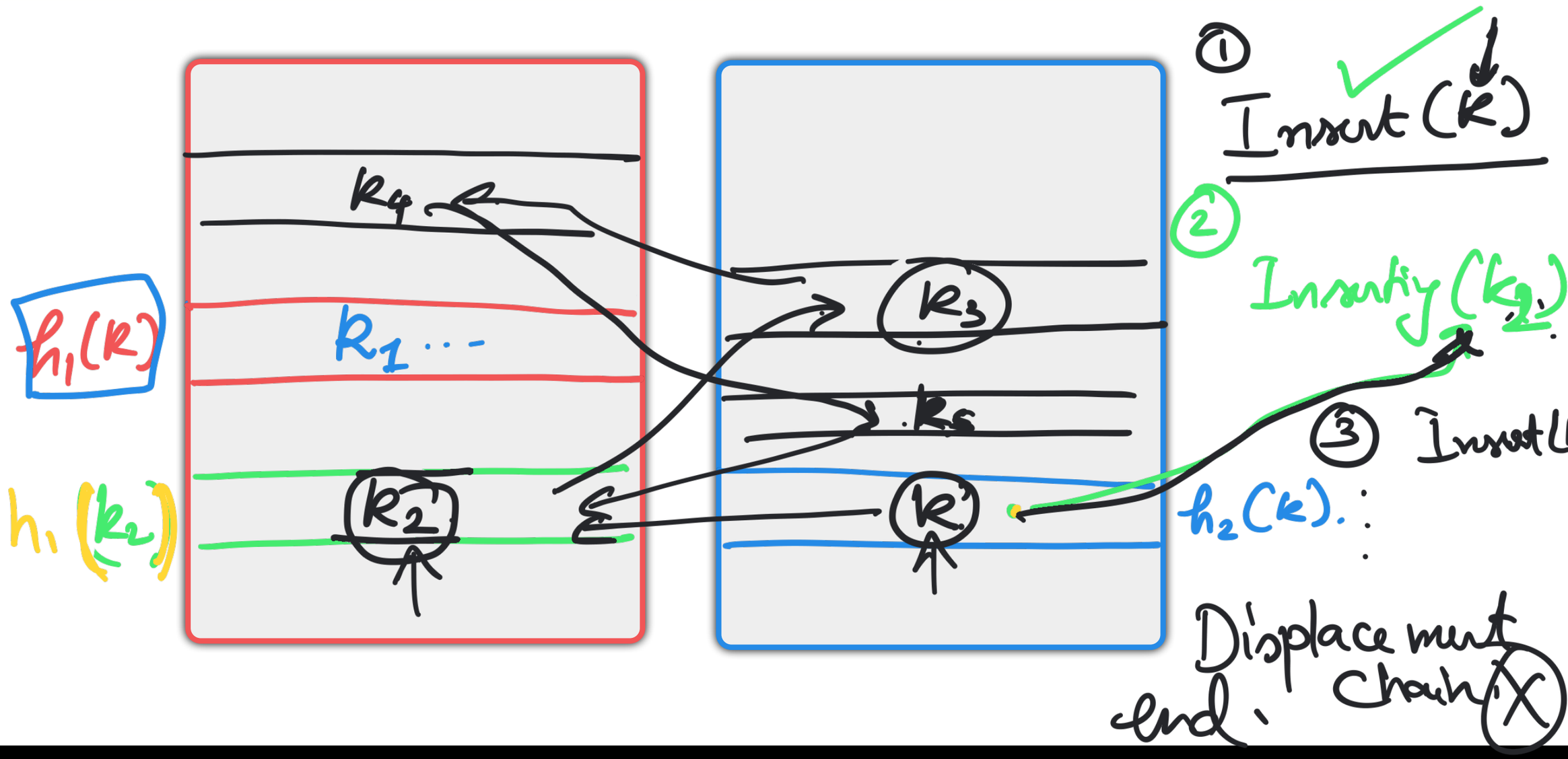


Cuckoo Hashing:Lookup



Lookup key k: Hashtbl 1 in $h_1(k)$ or Hashtbl 2 in $h_2(k)$

Cuckoo Hashing:Insertion



Cuckoo Hashing:Rehashing

→ (If "too many" displacements $\geq \log(n)$)

$n = 2^{10}$

10 displacement.

→ Build a new cuckoo hashtable and re-insert all keys.

→ Important: Choose different hash functions at random.

Cuckoo Hashing:Analysis

→ Analysis is very Complicated

→ Probability that insertion results in $\geq k$ displacements is provably small

→ Probability that insertion results in rehashing is also proved to be small.

→ Expected complexity of all operations is Constant.