Lecture 8

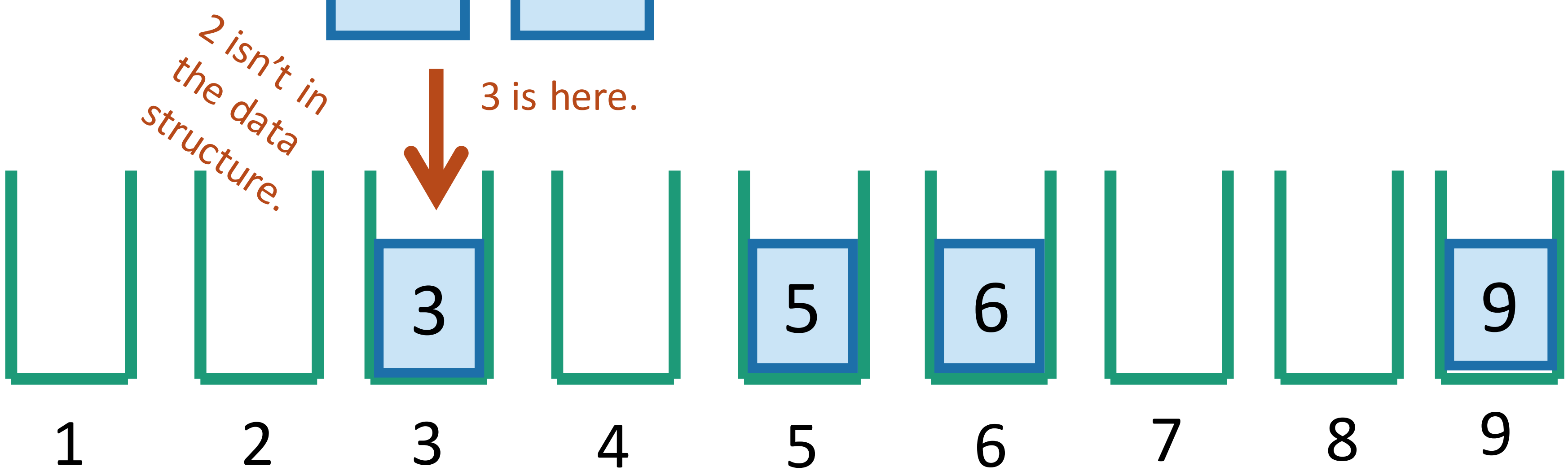
HASH TABLES

Motivation

- **Hash tables** are another sort of data structure that allows fast INSERT/DELETE/SEARCH.
  - like self-balancing binary trees
  - The difference is we can get better performance in expectation by using randomness.
    - Like QuickSort vs. MergeSort

Direct Addressing

- Say all keys are in the set {1,2,3,4,5,6,7,8,9}.
- INSERT: 9 6 3 5
- DELETE: 6
- SEARCH: 3 2

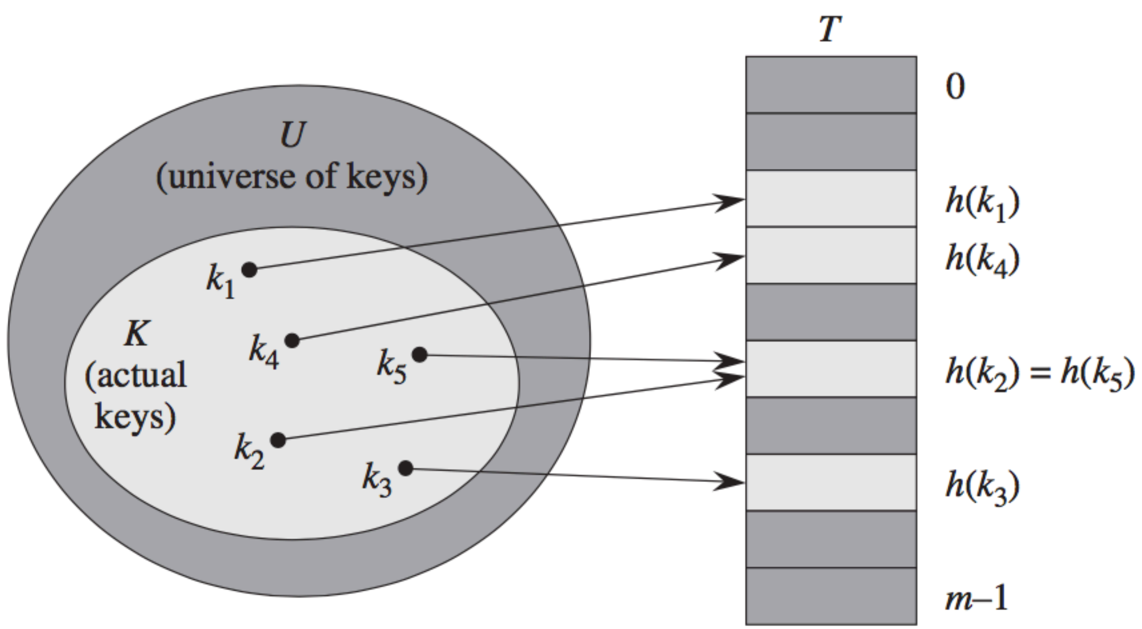


Direct Addressing

- if the universe U is large, storing a table T of size |U| may be impractical (memory)
- The set K of keys *actually stored* may be so small relative to U that most of the space allocated for T would be wasted.

Hash Functions

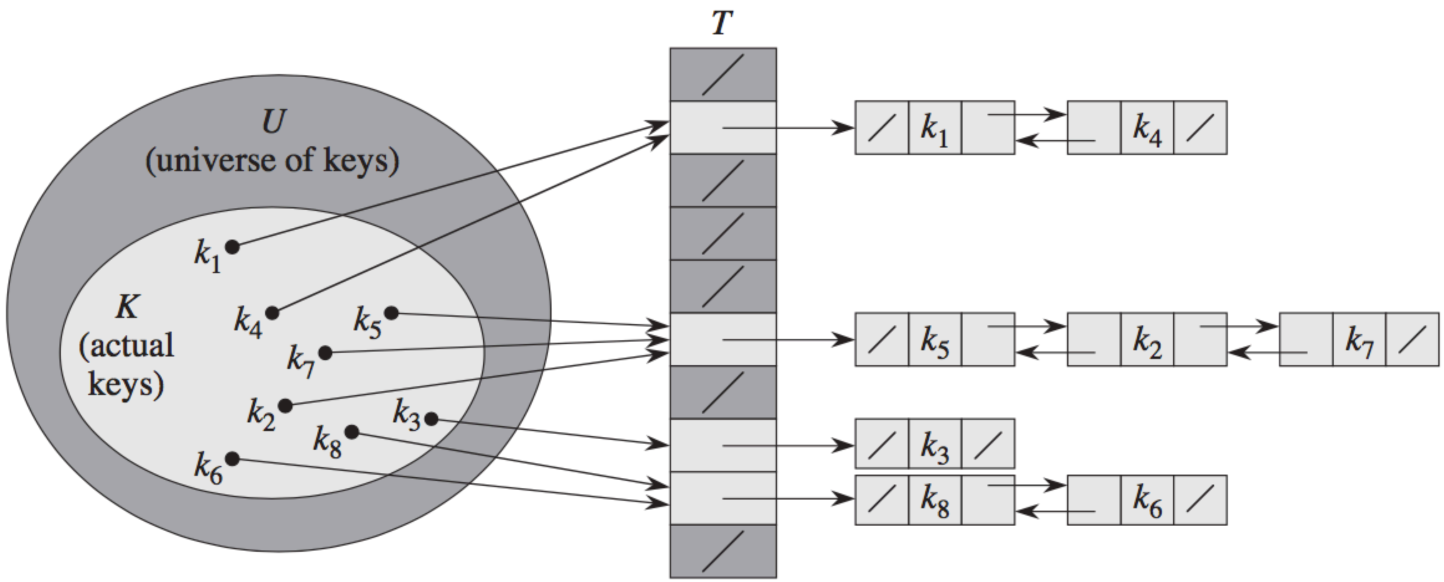
Collision !!!!



Direct addressing → an element with key k is stored in slot k.

Hashing → an element is stored in slot h(k), that is, we use a ***hash function*** h to compute the slot from the key k.

Collision resolution (chaining)



Hash Function

- The average-case performance of hashing depends on how well the hash function h distributes the set of keys to be stored among the m slots, on the average.

- Worst case
  - All n keys hash to the same slot,

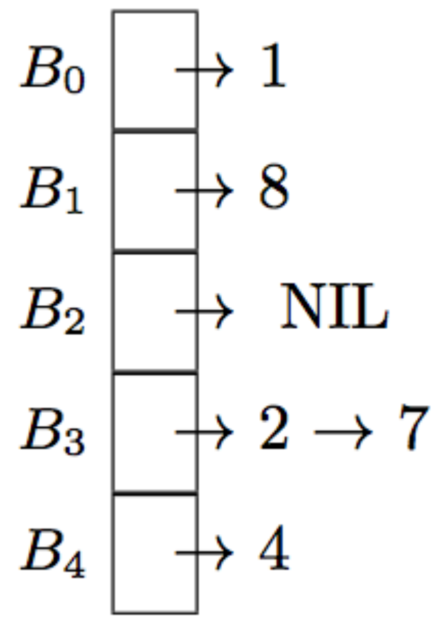
Hash Function

- assume that any given element is equally likely to hash into any of the m slots, independently of where any other element has hashed to.
- ***simple uniform hashing***.

Hash Function

$$h(x) = 13x + 2 \bmod 5$$

$$h(1)=15 \bmod 5=0$$



Hash Function

- if a malicious adversary chooses the keys to be hashed by some fixed hash function, then the adversary can choose n keys that all hash to the same slot, yielding an average retrieval time of O(n)

choose the hash function *randomly*

Expected cost of Random Hash Functions

X = number of items in u_i's bucke

Each key appears in the hash table at most once.

- $E[\ X\ ] = \sum_{j=1}^{n} P\{\ h(u_i) = h(u_j)\}$
- $= 1 + \sum_{j \neq i} P\{\ h(u_i) = h(u_j)\}$
- $= 1 + \sum_{j \neq i} 1/n$
- $= 1 + \frac{n-1}{n} \leq 2.$

The expected cost of any hashing operation is a constant.

Random Hash Functions

h is chosen uniformly and at random from amongst the set of all hash functions $h : U \rightarrow \{1, 2, . . . , n\}$.

Impractical !!!!!!

$n^{|U|}$ possible hash functions

Is it possible to construct a small, practical subset of hash functions with this property?

Carter and Wegman (1978)

Universal hash family

- Here's one:
  - Pick a prime $p \geq M$.
  - Define

$$f_{a,b}(x) = ax + b \quad mod\ p$$

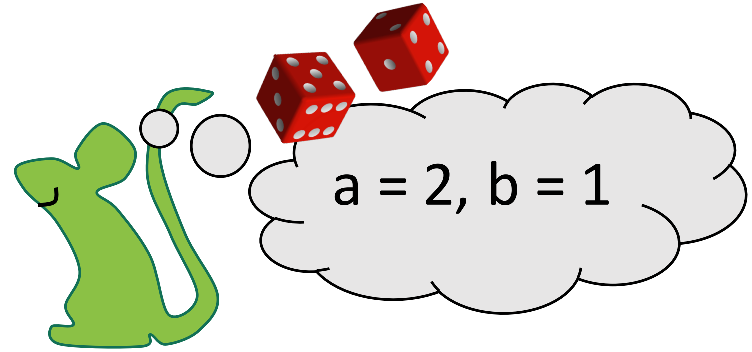
$$h_{a,b}(x) = f_{a,b}(x) \quad mod\ n$$

  - Claim:

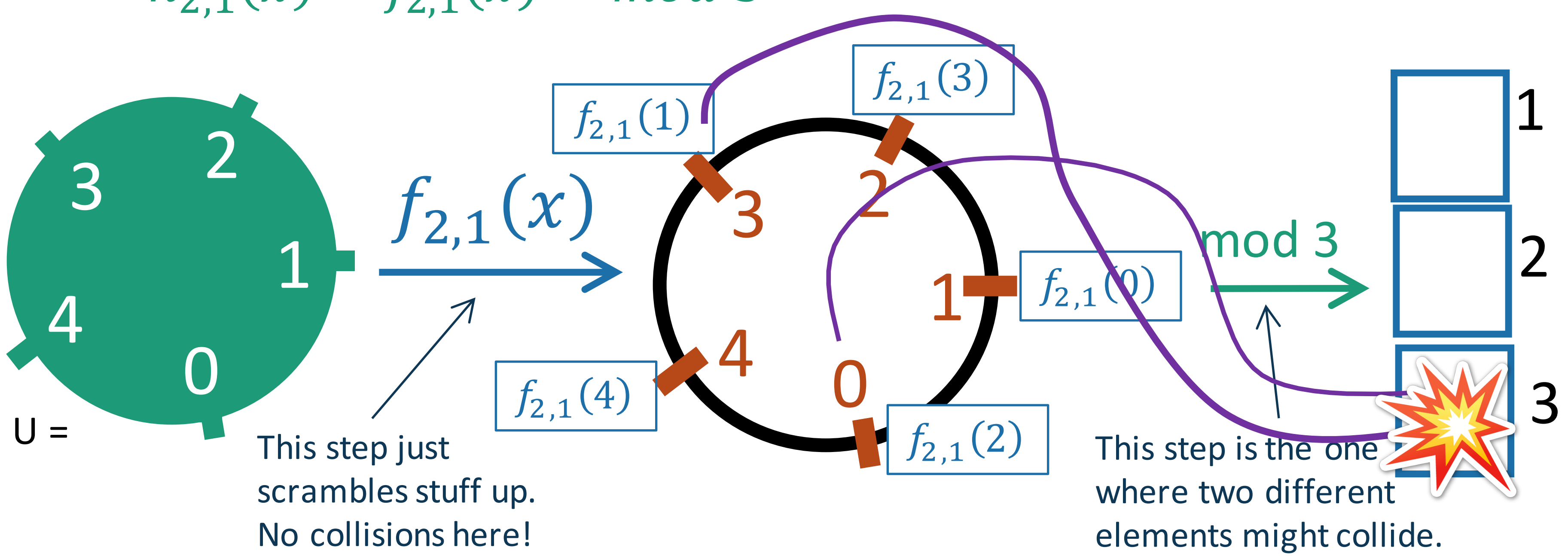
$$H = \{\, h_{a,b}(x) \;:\; a \in \{1, \ldots, p-1\}, b \in \{0, \ldots, p-1\} \,\}$$

is a universal hash family.

Universal hash family



- Example: M = p = 5, n = 3
- To draw h from H:
  - Pick a random a in {1,…,4}, b In {0,…,4}
- As per the definition:
  - $f_{2,1}(x) = 2x + 1 \quad mod\ 5$
  - $h_{2,1}(x) = f_{2,1}(x) \quad mod\ 3$



Universal hash family

- Here's one:
  - Pick a prime $p \geq M$.
  - Define

$$f_{a,b}(x) = ax + b \quad mod\ p$$

$$h_{a,b}(x) = f_{a,b}(x) \quad mod\ n$$

  - Claim:

$$H = \{\ h_{a,b}(x)\ :\ a \in \{1, \ldots, p-1\}, b \in \{0, \ldots, p-1\}\ \}$$

is a universal hash family.

Proof ????