LECTURE-13

Hashing and Chained Hash Tables

Workings of Chained Hash Tables

For Poll Ev

CS202: Data Structures (Fall 2025)

Dr Maryam Abdulghafur, Momina Khan

Department of Computer Science, SBASSE

Agenda

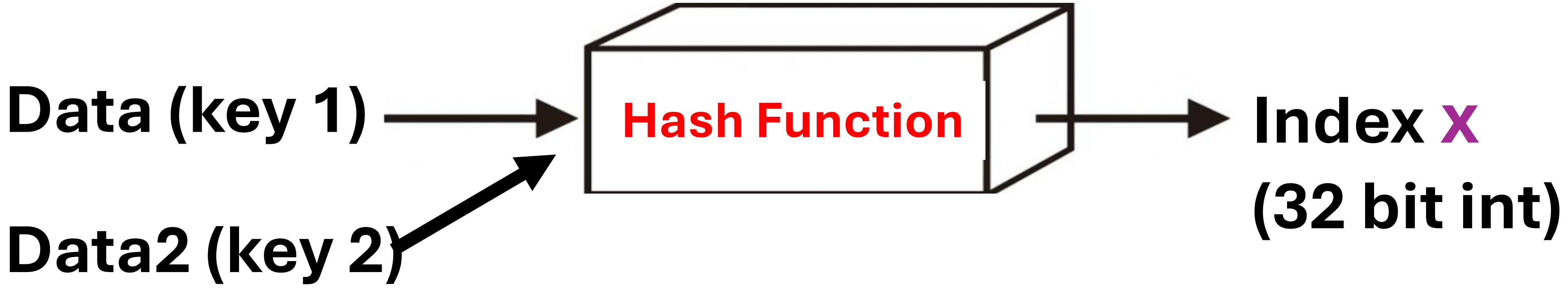
- Exploring a Hashing and Hash Tables
- Collision management through Chaining – Chained HashTables
- Hashing and its issues addressed
- Quality of a Hash Table
- Complexity of the various functions of the Hash Tables

RECAP: MOTIVATION O(1) SEARCH

What if x comes out to be much larger than the largest index?

Hashing

Collision



Hash tables

Provides a mechanism to deal with collisions

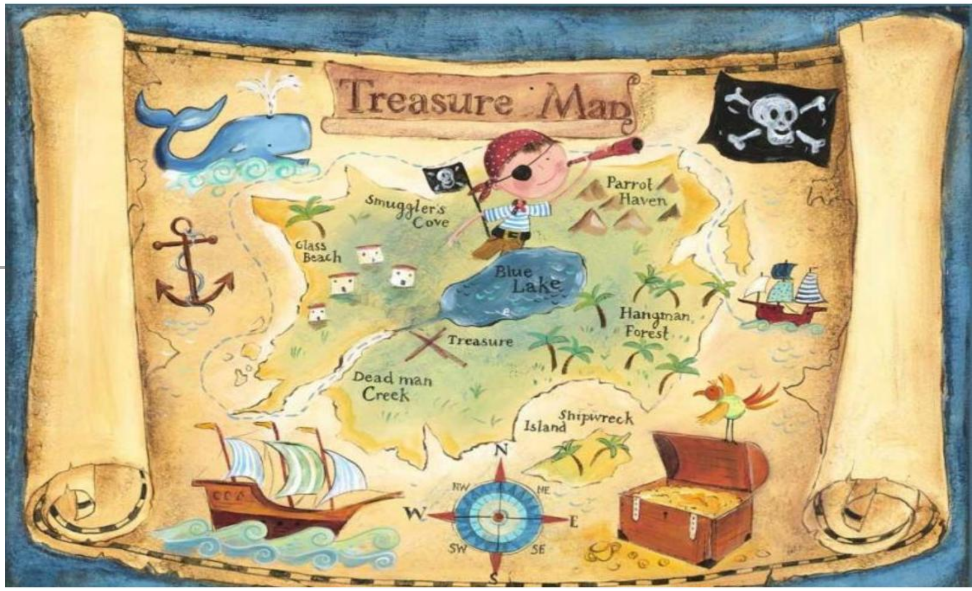
Hash Function examples

- What if the key is a string or alphanumeric (like car reg. numbers CAR 123)

- Naïve hash function:
  - Use Ascii value of 1st letter as an index.

- What is a good hash function?

Important functions:



- **Search** ....

  O(1) search that was our motivation right?!

- **Hash Function** .... Something that associates data key $\longrightarrow$ 32 bit index
  - Think of this as a treasure map

- **Insert** ... when do we draw the map?

  Think of every item as a treasure with its own map!

- **Delete** ... when do we discard the map?

  When a treasure has been retrieved or used.

Hash Table

- Must have **an independent** and **deterministic** associated hash function.

- An **array** to hold elements of the hash table.

- A mechanism for **insert, delete and search** into the hash table.

- A mechanism to **handle collisions!**

Hash Table

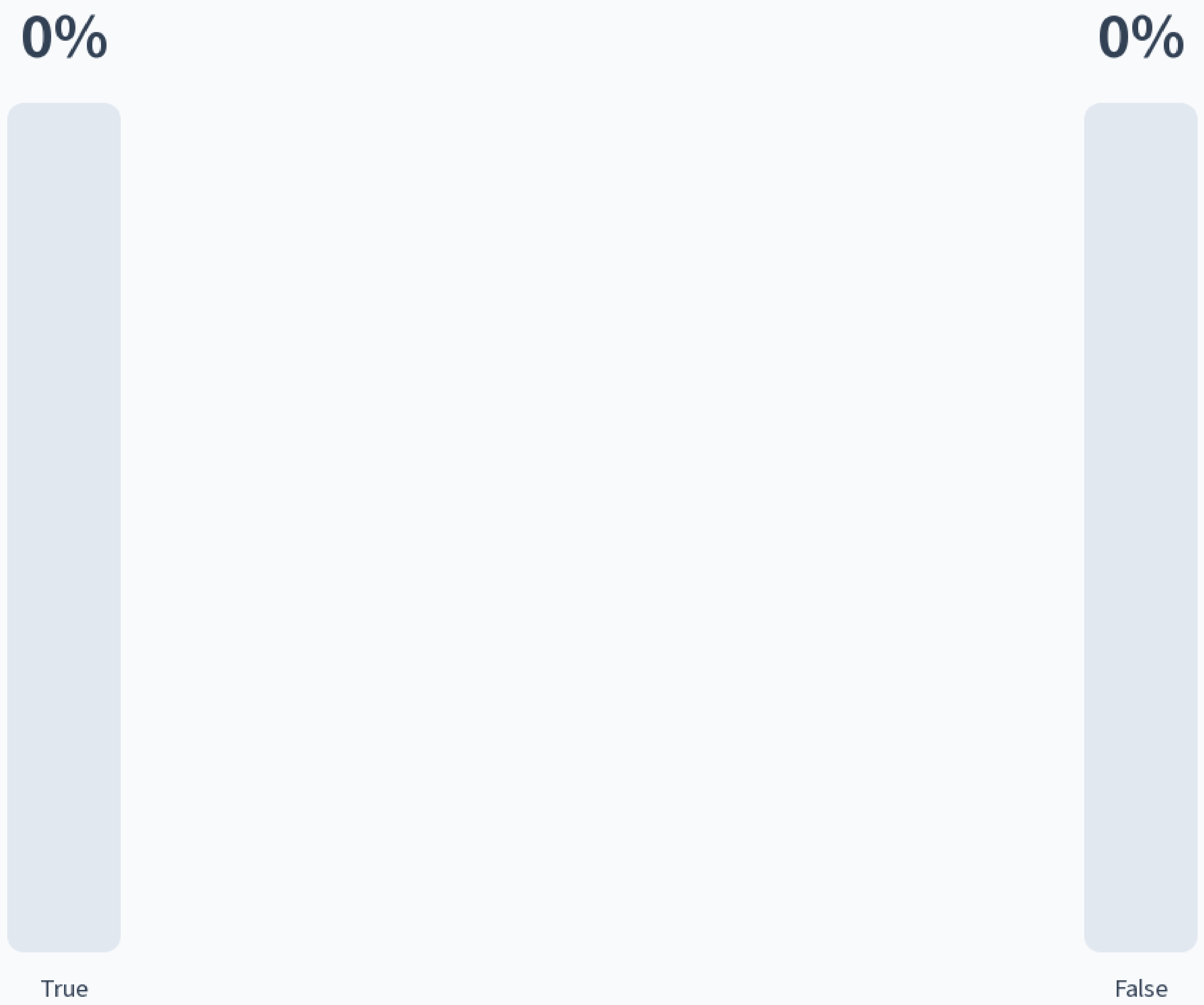
What would be class definition of the

class HashTable

its data components …

- **An array**
- **Size of array**
- **Load factor (how many slots are occupied)**

Can I run out of empty slots in a HashTable array?



Some important concerns

- How is data(key) and index it is stored at, related?
- What if I run out of empty slots in the array? What can I do?
- What contributes to collisions? Are they avoidable?
- Can I manage to keep the rate of collisions low?

Hash Table

What would be class definition of the

class HashTable

its data components …

- **Array ***
- **Size of array**
- **Load factor (how many slots are occupied)**

What if anything will change in the class HT?

HashTable Big O performance

What can be said about the Big O or worst case complexities of the HashTable?

- **search()** O(1)
- **insert()** O(1)
- **delete()** O(1)

Chained Hash Tables

- **If collisions cannot be avoided, how do we manage them?**
- **We grow lists out of every index as and when a collision happens for that index!**

- **Does the mechanism for search(), insert() and delete() change in any way??**

Let us study this with the help of an example

Chained HashTables

What can be said about the **Big O** or worst case complexities of the proposed Chained HashTable?

Long Chains growing out of each index

What can we do to keep worse complexity in check?

Keep chains short! How?

Low rate of collisions is the answer? How?