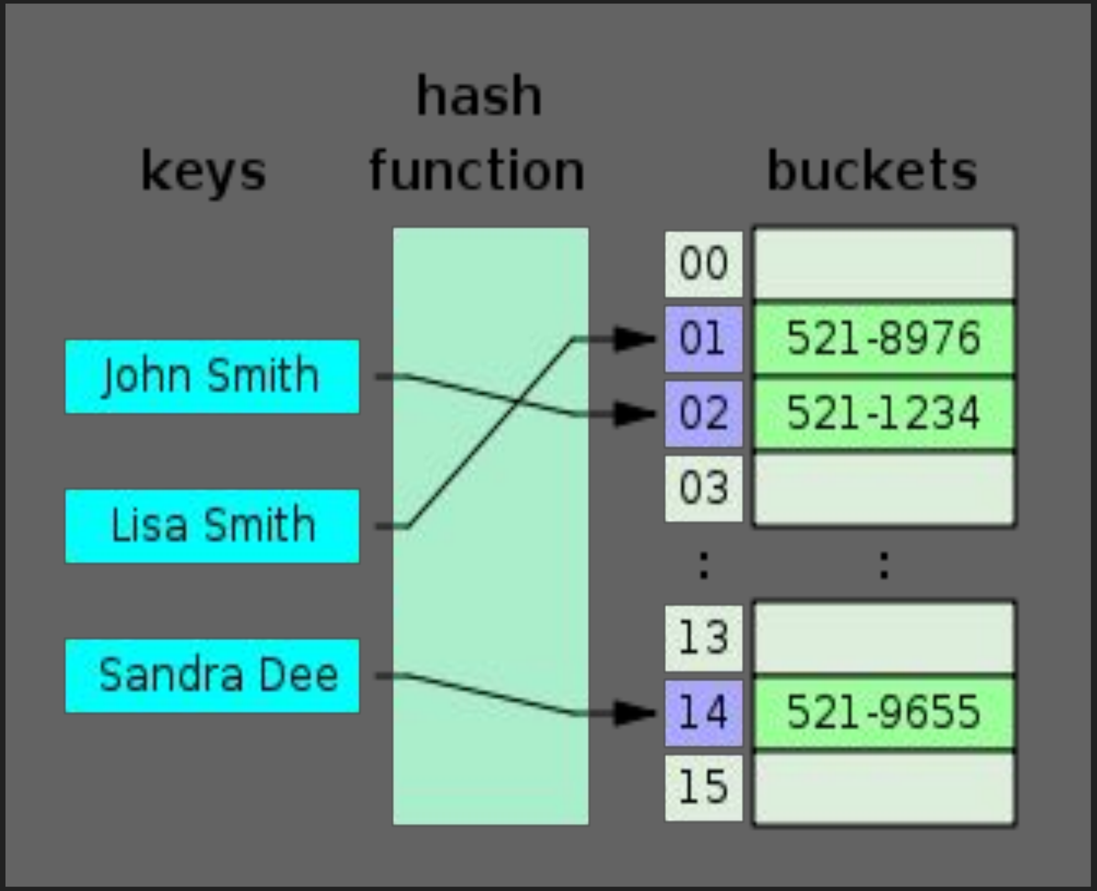
CSCI 104 Lab: Hash Tables

What is a Map?

- Recall that a map is a data structure used mostly for fast lookups or searching data.
- It stores data in the form of key, value pairs, where every key is unique. Each key maps to a value, hence the name "map."
- The look up speed and ordering of map elements depends on the data structure we use to implement our map.

Hash Tables

- Like an array, but to find the index we use a **hash function.**
- A hash function converts the input into an index location that the input is then stored into.
- Hashing is a function - O(1)
- With a good hash function that distributes keys **uniformly** around the table so to minimize **collisions,** all commands have average runtimes of O(1).



Hash Functions

- Hash function - a function that converts an object into an index location within our array.
- Goals of a hash function:
    - Easy and fast to compute
    - Uniformly distributes keys across the hash table
    - Deterministic!

Hash Functions (continued)

- A bad hash function:

```
int hash(int data) {
    return 42 % size;
}
```

  This is fast (only one operation) but always returns the same value!

- A good hash function:

```
int hash(int data){
    return 31 * 54059 ^ (data * 76963) % size;
}
```

  A few more operations (but still constant), and gives a variety of output numbers, and is deterministic!

Collisions

- So what happens if the hash function outputs the same index for multiple objects?
  - This is called a **collision**.

- Two approaches to handling collisions:
  - Open addressing
  - Closed addressing (chaining or buckets)

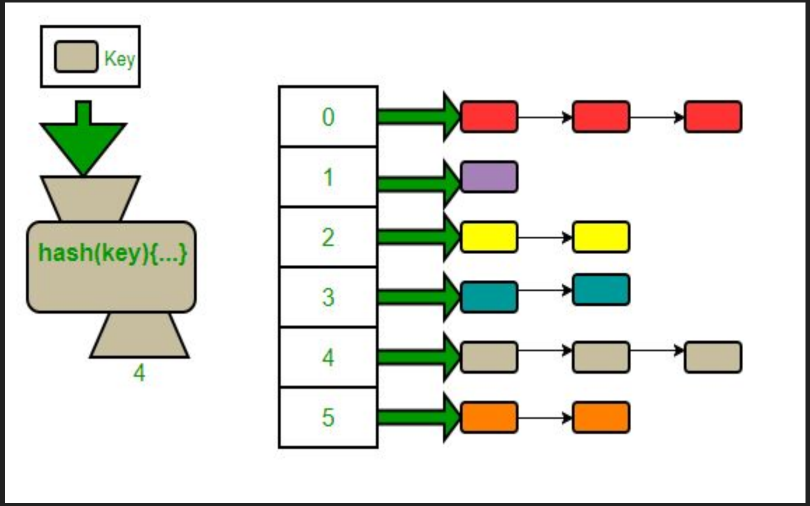
Open Addressing

- The idea with open addressing is that every location in the array can only have 1 thing in it. This means that we will have to find a free spot that we can place the object in. Linear probing is a very simple solution.

- Linear probing is where you just keep incrementing up/looking at the next index until you find a free location.
- Other examples of open addressing are:
  - Quadratic Probing
  - Double Hashing

Closed Addressing (Chaining)

- Chaining allows for multiple objects to reside within the same array location. The array is changed to be an array of lists or some other data structure, allowing us to store multiple items per index. We often use an array of linked lists, hence the name "chaining."



- Because chaining allows for buckets, n objects could all be placed within the same bucket. The worst case runtime is O(n). Ideally, this should not occur if the hash function is good and the size of the hash table is big enough.

OrderedMap

Function	Runtime
get(key)	O(log n)
push(key, value))	O(log n)
remove(key)	O(log n)

- An ordered map uses a balanced binary search tree as its underlying data structure.
- We haven't yet gone over it in lecture yet, but for now, it is sufficiently to know that it is used in the implementation of std::map (it usually uses a version of balanced binary search tree called a Red-Black Tree)

UnorderedMap

Function	Runtime
get(key)	O(1) on average
push(key, value))	O(1) on average
remove(key)	O(1) on average

- An unordered map uses a **hash table** as its underlying data structure. This means that access operations are **O(1) on average**, but because of this, **no order can be inferred**
- You must explicitly create an unordered map using **std::unordered_map.**

Lab Assignment

- Implement remove function for an open addressing hash table