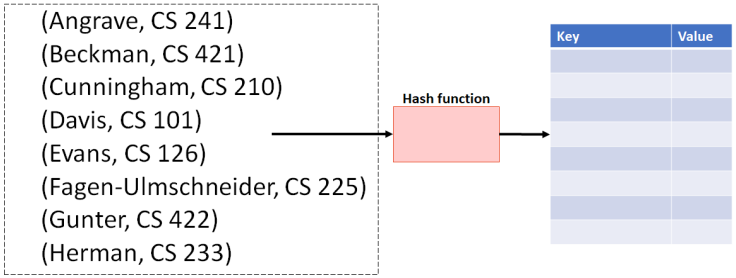
CS 225

#29: Hashing

Every hash table contains three pieces:

1. A **hash function**. The hash function transforms a key from the keyspace into an integer.
2. **A data storage structure**. (Usually an array)
3. A method of handling **hash collisions**.

A Perfect Hash Function



...characteristics of this function?

A hash function must be:

- **Deterministic**:
- **Efficient**:
- **Defined for a given size:**

In CS 225, we think hash functions as two separate parts:

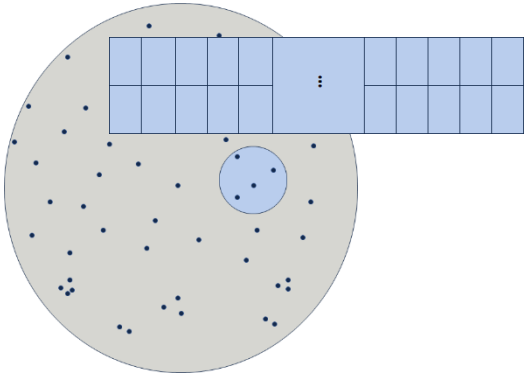
- A **hash**:
- A **compression**:

Towards a general-purpose hashing function:

It is easy to create a perfect hashing function when the keyspace is proportional to the table size:

- **Ex:** Professors at CS@Illinois
- **Ex:** Anything you can reason about every possible value

It is difficult to create a general-purpose hashing function when the keyspace is large:



For example, given a fixed collection of books what is a viable hash function that will yield no collisions?

... will those hash functions work for all *possible* books?

What is an example of bad input data on this hash function?

Reflections on Hashing
We are starting the study of general-purpose hash functions. There are many other types of hashes for specific uses (ex: cryptographic hash functions).

Even if we build a good hash function, it is not perfect. Given that our universe will usually be larger than our hash table what problem do we have to deal with?

Dealing with hashing depends on which type of storage structure you are using.

Open Hashing:

Closed Hashing:

Draw the following hash table using ***separate chaining.***

Key	Value	Hash
Bob	B+	2
Anna	A-	4
Alice	A+	4
Betty	B	2
Brett	A-	2
Greg	A	0
Sue	B	7
Ali	B+	4
Laura	A	7
Lily	B+	7

Simple Uniform Hashing Assumption (SUHA)

SUHA assumes that our hash function is uniform and independent for all keys in the keyspace (universe).

The expected length of a chain under SUHA:

This value is also known as our 'load factor'.

Running time of Separate Chaining:

	Worst Case	**SUHA**
Insert		
Remove/Find		