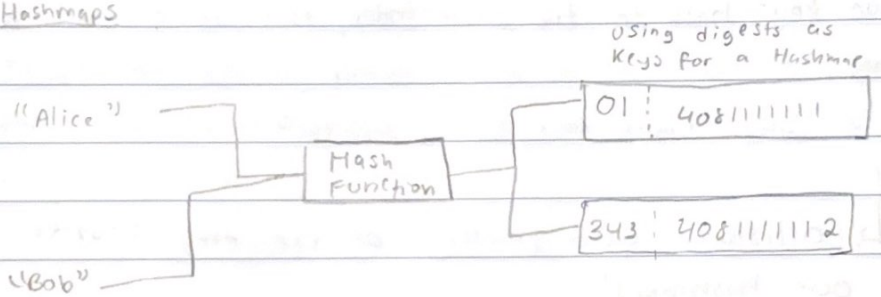
Hashing

Hashing is the process of converting data into a [usually] fixed-length output.

Data is converted into these fixed-length values, or hash values, by using a special function called a hash function.

Hashmaps



01: "408111111" - Alice phone

343: "408111112" - Bob phone

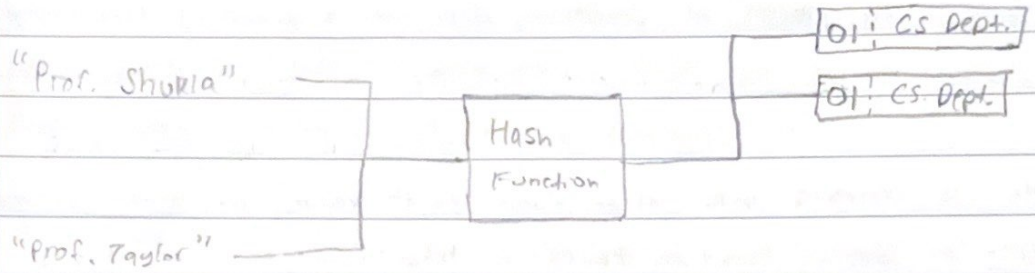
Hash Functions

Hash Functions are not all made equal.

An ideal function should be:

- Be efficiently computable
- Uniformly distribute the keys
- Should MINIMIZE collisions!

Hashmap Collisions



When two keys hash to the same index, this is called a COLLISION!

- but why would this be a problem?

↳ Collisions cause problems because they clutter our hashmaps!

ex: ID # vs. People with Name like Sam

We definetly want to avoid collisions, but sometimes they are unavoidable

Ways to deal with Collisisns:

- Linear Probing
- Quadratic probing
- Seperate Chaining
- Double Hashing

$x \bmod y \equiv \frac{x}{y} \equiv$ Remainder

$y\overline{)x}$

Linear probing

Strategy: ① Use a Hash function to find the index for a key.

② If that spot contains a value, use the next available spot "a higher index"

If you reach the end of the array, go back to the front.

ex: Insert the following numbers into a hash table of size 5 using the hash functions H(key) = key mod 5. Show the result when collisions are resolved.

Numbers: 10, 11, 12, 15

10	11	12	15	
0	1	2	3	4

Index

H(10) = 10 mod 5 = 0

H(11) = 11 mod 5 = 1

H(12) = 12 mod 5 = 2

H(15) = 15 mod 5 = 0

So since H(15) is also at a index of 0, like H(10) previously done before. So, we will go to the next avaible index, which in our case is @ index 3.