What is Hashing?

A hash is an *into* mapping of some input (number, string) into a usually small and usually integer number

- The input is the *hash key*
- The mapping is the *hash function*
- The output is the *hash value*
- The *hash value* is frequently used to index (*hash index)* a table (*hash table)*
  - *Doing a lookup in a table by referencing its index is essentially a one cycle operation- really, really fast*

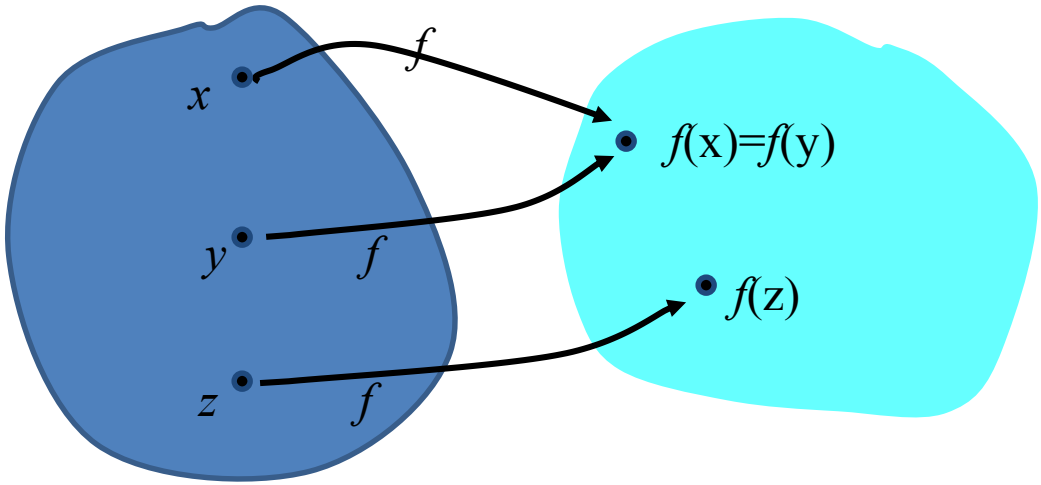
Hash Collisions

$$f(x) \neq f(y) \Rightarrow x \neq y$$

BUT

$$f(x) = f(y) \not\Rightarrow x = y$$

Restated, ***f*** is *into*, but neither necessarily *1-1* nor even necessarily *onto*. If it were *1-1* and *onto*, there would be no such thing as a collision.



Hash Tables

- ***Direct-address table***
  - If the table is big enough, every substring maps to an unique location in the table
  - Can be very sparse and very very big
- ***Hash table***
  - If the table size is made less than the number of substrings, it becomes a hash table
    - Requires an *into* function (hash function) to generate an index that maps the substring data to the correct location in the table
    - By the Pigeon-hole Theorem, there will be collisions if there are more substrings than locations

Hash Function

A good hash function

- Must map all substrings into the table (index cannot exceed table size)
- Spreads the collisions uniformly over the table, avoiding concentrations of collisions

Issues with a Hash Table

- Size of hash table
- Choice of hashing function
- Hash Collisions
  - Minimize the number of collisions
  - Avoid pockets of collisions- *i.e.,* spread collisions evenly over the table
  - Have a process to handle collisions, both on storage and lookup
    - Chaining
    - Probing
- Hash table lookup efficiency
  - Rapid generation of hash values

Modulus Arithmetic Hashing function

- Although there are other techniques, a tried and true way to hash is by choosing a hash function like x *mod* k, where x is a numerical representation of a string and k is an integer. k would set the maximum size of the required table
- **If k is a <u>prime number</u>, the collisions are most effectively spread uniformly!***

*The explanation: *magic!*