# Concept of Hashing

15-123

Systems Skills in C and Unix

# What is hashing

- Internet has grown to millions of users generating terabytes of content every day
- With such large data sets, how do we find anything?
- Two standard search techniques
  - Linear search – O(n)
  - Binary Search – O(log n)
- What if we need to find things even quicker?

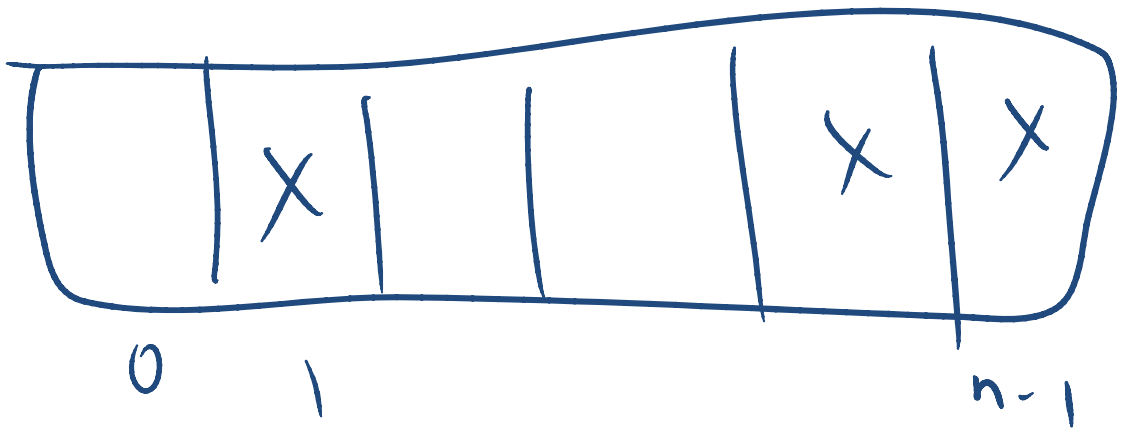
# Finding things in O(1)

- Suppose our intent is to find an item in O(1)
  - That is, constant time or time does not depend on data size n
- In most cases, we only care about
  - Finding and retrieving things quickly
  - Updating and inserting things quickly
- We do not care about
  - Order statistics of the data

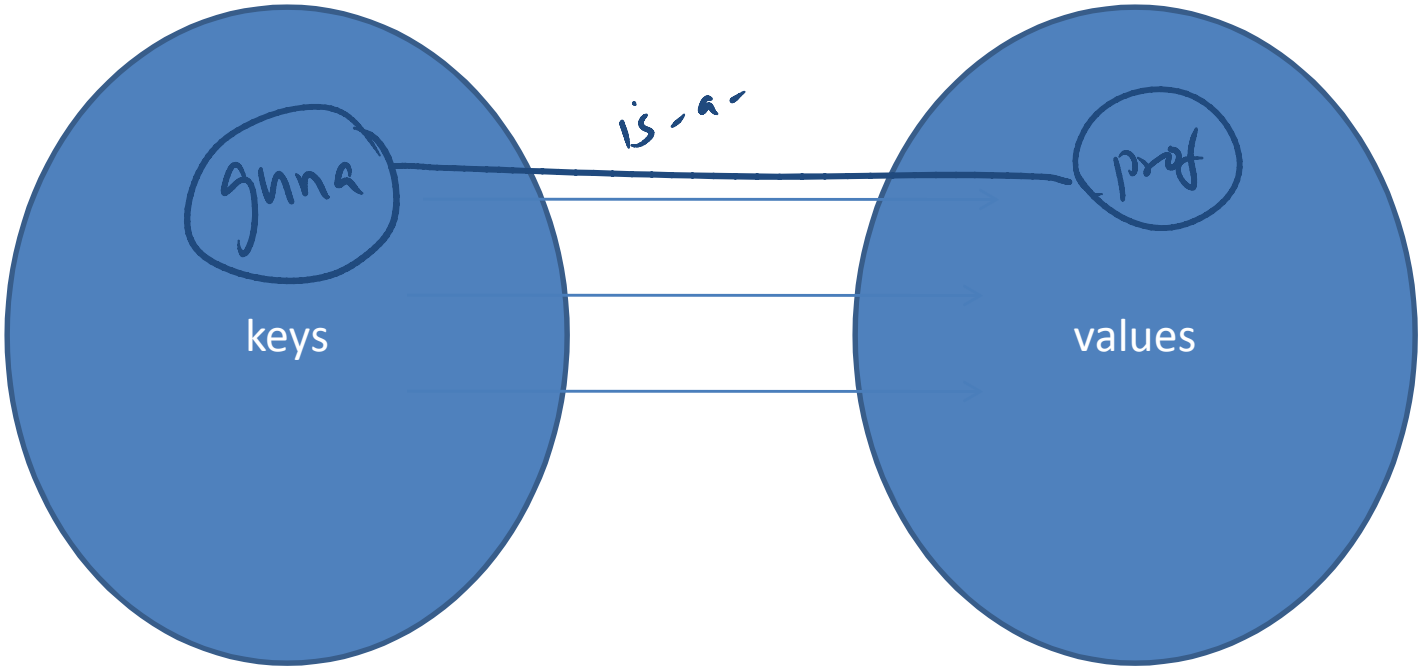
# Finding things quickly

- Strategy – hashing

- Data Structure – hash table

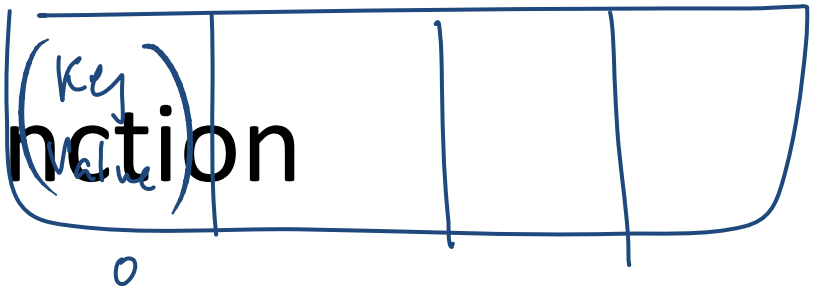


# Maps

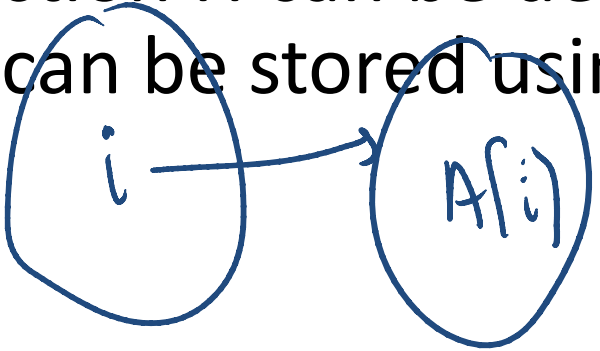



**A relation between two sets defined by a simple function**

# Hash Function




- A hash function maps a key to a value
- Simplest Form
  - A[i] - a mapping of index (an integer) to a value
- The hash table idea is much more general
  - Keys don't have to be integers
  - H("guna") = "professor"
- If a hash function H can be defined, then information can be stored using (key,value) pairs



# What makes a good hash function?

- A hash function must be
  - Easy to calculate
  - Must avoid “collisions”
- What do we mean by “easy to calculate”?
  - The cost of computing the hash value must be minimized
- What do we mean by “collisions”?
  - It is possible that two keys can map to the same value (unless you can come up with a perfect hash function)
  - Finding the perfect hash function is “hard”

# Example

- **Take a simple set of strings {"abc", "bda", "cad"}**
- **Define a hash function as follows**
  - **H("abc") = sum of the characters % 5**
  - **Where n = 5 is the table size**
- **Find H("abc"), H("bda"), H("cad")**

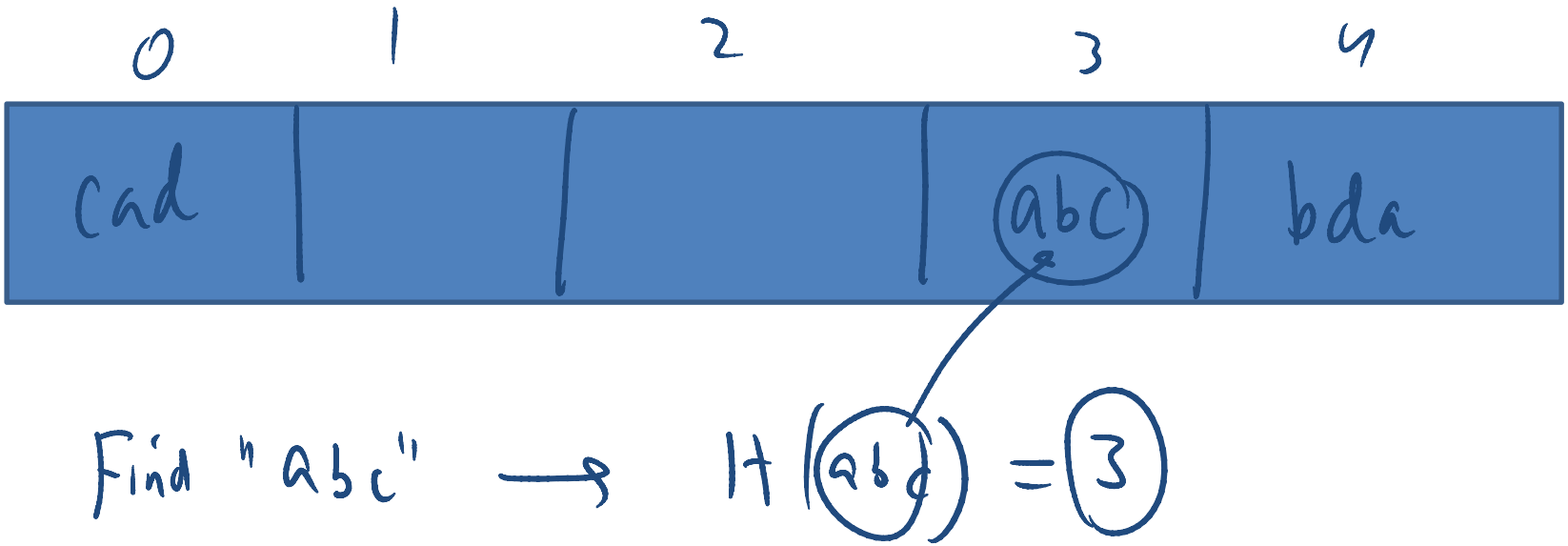
A = 97 → 0
b = 98 → 1
C = 99 → 2
D = 100 → 3

3

4

5 % 5 = 0

# Storing the values



# Questions

- What happens if "abc" and "bac" hash into the same location?

H("abc") = 3 = H("bac")

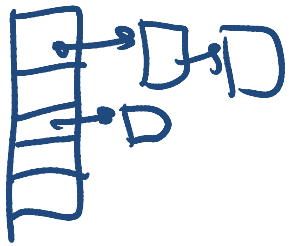
- How do we resolve it?

- Using a collision resolution strategy

→ linear probing x, x+1, x+2, ...

→ quadratic probing x, x+1, x+4, x+9, x+16

→ separate chaining



# Using a better hash function

- **H(s) = ∑ sum of characters has too many collisions**
- **Define H(s) as a polynomial representation of characters of s**

$S = a_0 a_1 .. a_{n-1} \rightarrow n!$

$H(s) = a_0 + a_1 p + a_2 p^2 + a_3 p^3 + .. + a_{n-1} p^{n-1}$

$p \simeq 101$

$$H(abcd) = a + b \cdot p + c \cdot p^2 + d p^3$$
$$= a + p(b + cp + dp^2)$$
$$= a + p(b + p(c + dp))$$

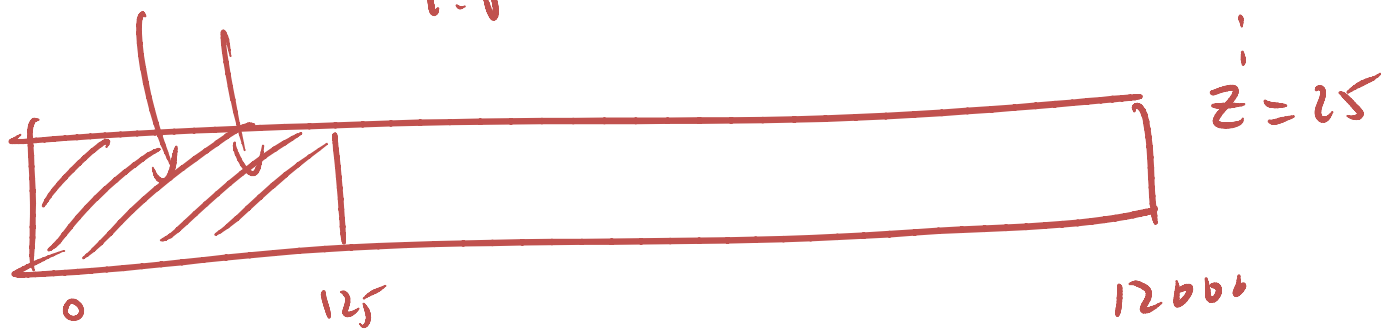
# Making things more efficient

- **How can we calculate H(s) more efficiently?**

See Notes for Code

# Questions

- Suppose we would like to hash 10000 keys, (each up to a 5 character string ) into a hash table of size 12000. We use the function
  - H(string) = ∑ sum of the characters of the string
- What would be the key distribution?



# Coding Examples