

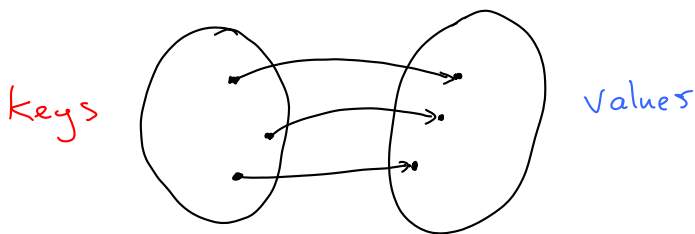
lecture 26

maps

Examples

- address book
(input a name, output an address & phone #, etc)
 - student file
(input an ID number (or name) and output a record eg. a transcript, address, etc).
- Inputs are "keys". Outputs are "values".

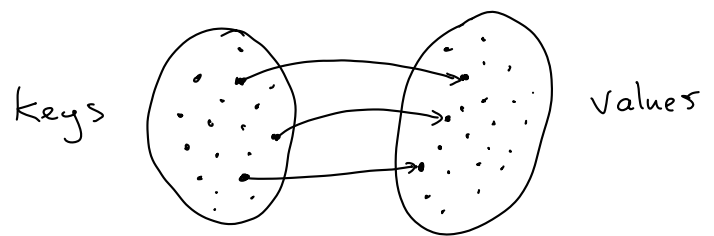
Map



A **map** is a set of (key, value) pairs. For each key, there is one value.

(But two keys could have the same value.)

Map (set notation)

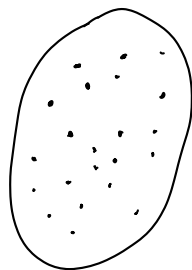
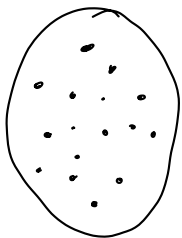


A map is a set of ordered pairs:

$$\{(k, v)\} \subseteq K \times V.$$

key K

value V



Examples

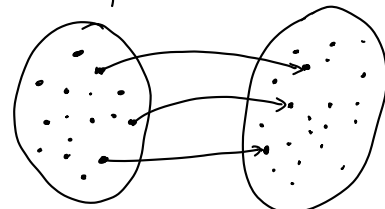
- numbers
- strings
- ...

- objects
(very general)

Example 1 : address book

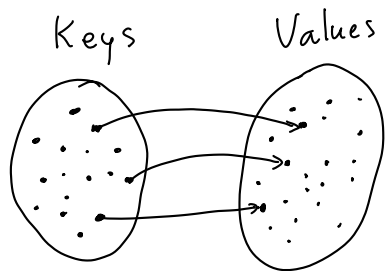
Keys

Values



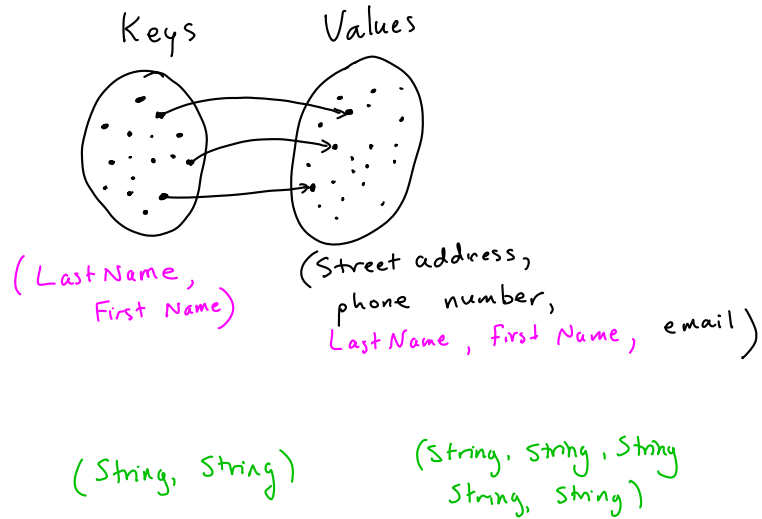
(Street address, phone number, email)

(String, String, String)

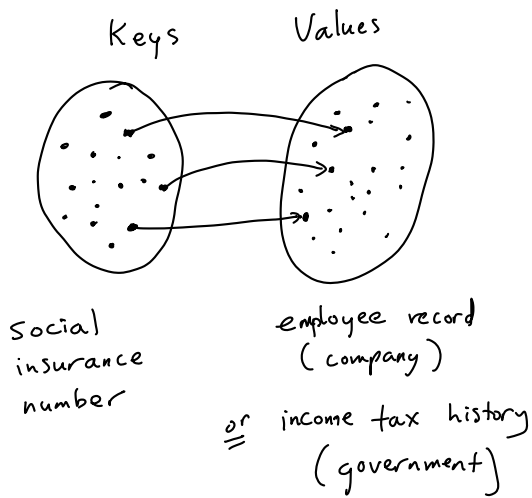


In this example, the map has only 3 entries, even though the key's value spaces are large.

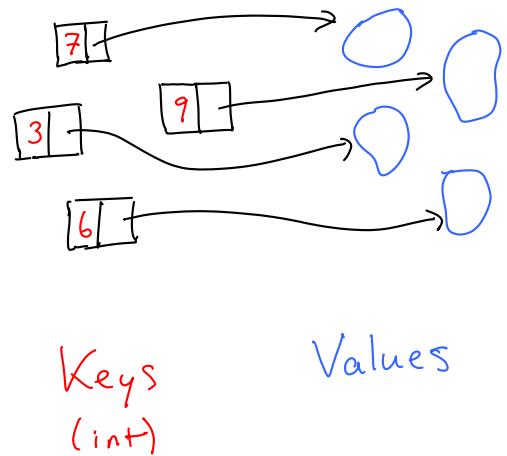
Example 1 (alternative)



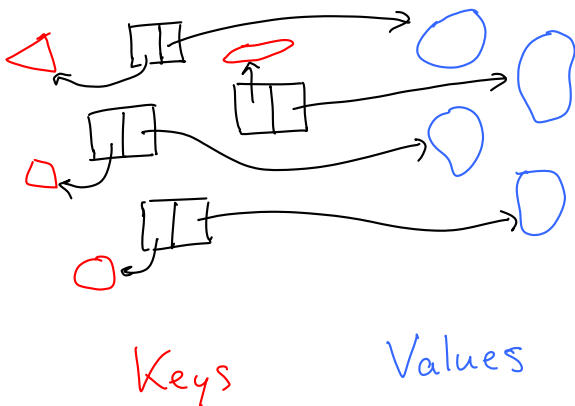
Example 2



Map: Data Structure



Map: Data Structure

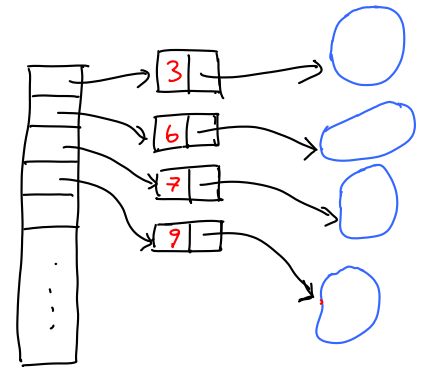


How can we organize (key, value) pairs so that we can index the value associated with a given key?

We have discussed how to organize a set of **comparable keys** so that we can efficiently **add, find, remove**

- sorted array
- BST

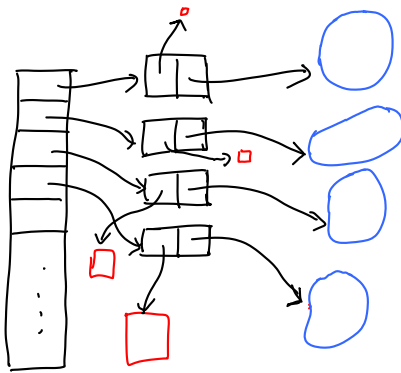
Sorted Arrays



insert, remove $O(n)$
find $O(\log n)$



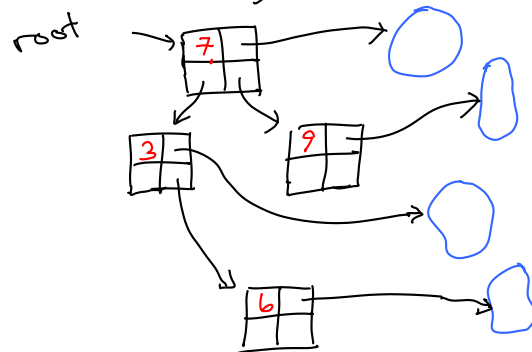
Sorted Arrays



insert, remove $O(n)$
find $O(\log n)$



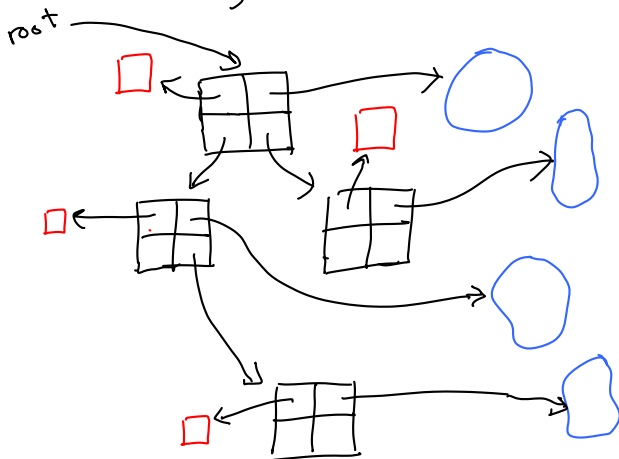
Binary search trees



- $O(n)$ access if not balanced
- $O(\log n)$ access if balanced



Binary search trees

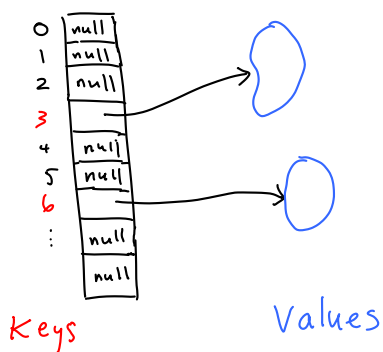


Next lecture, we will see a method called hashing which allows $O(1)$ access to the value associated with a given key.

(Before that, you need to understand the following.)

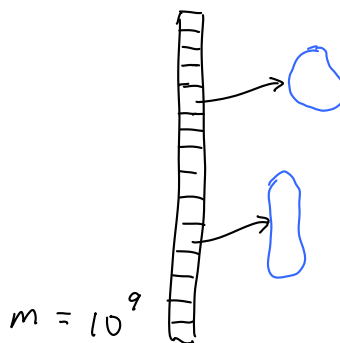
Maps: Direct Addressing

Suppose our key space K is $\{0, 1, \dots, m-1\}$.
 Our map has at most m keys.



Using an array
 would give
 $O(1)$ access!

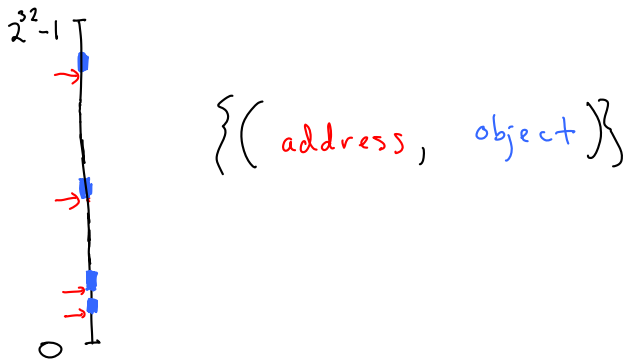
Example: Employee social insurance number
 (9 digits)



m too big
 (impractical)

More Subtle Example

- keys are (starting) memory addresses of objects
- values are objects



Example

Keys are **strings** of **unicode** chars.

We can define a map:

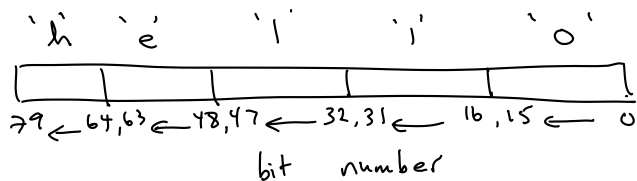
$$\{\text{strings}\} \rightarrow \{\text{positive integers}\}$$

Let s be a string

$$s[0] s[1] \dots s[\text{length}-1]$$

Define:

$$\text{value} = \sum_{i=0}^{\text{length}-1} s[i] (2^{16})^{\text{length}-1-i}$$



Each string of m chars
 would define a $16 \times m$ bit
 number.