Lecture 31

Maps

A map is a set of (key, value) pairs.

- For each key, there is one value.
- Two keys could have the same value.

Examples

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>address + phone + email name</td>
</tr>
<tr>
<td>ID #</td>
<td>student record name</td>
</tr>
<tr>
<td>word</td>
<td>ordered list of page &amp; s</td>
</tr>
</tbody>
</table>

Two keys could have the same value.

Map (set notation)

A map is a set of ordered pairs (called “entries”)

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

Keys

Values

In this example, the key and value spaces are large but the map has only three entries.

Primitive types

- key K
  - numbers
  - strings
  - ...

- value V
  - numbers
  - strings
  - ...
Java maps (Reference Types)

key $K$

value $V$

objects of type $K$

objects of type $V$

Example: address book

Keys

Values

(Last Name, First Name)

(Street address, phone number, email)

(String, String)

(String, String, String)

Example (alternative)

Keys

Values

(Last Name, First Name)

(Street address, phone number, Last Name, First Name, email)

(String, String)

(String, String, String, String)

How to organize (key, value) pairs?

eg.

- void add ($k$, $v$)
- void remove ($k$, $v$)
- void find ($k$)
- ....

Map: Data Structures

Keys (int)

Values (objects)

Map: Data Structure

Keys (objects)

Values
How to organize the keys?

- array, linked list
- sorted list (comparable?)
- binary search tree
- priority queue
- hash map/table (next lecture)

(unsorted) array

find\[O(n)\]
insert, remove \[O(n)\]

Sorted Array

find \[O(\log n)\]
insert, remove \[O(n)\]

Binary Search Tree (Nodes)

left child right child

left child right child

Binary Search Tree

root

- \[O(\log n)\] access if balanced
- \[O(n)\] access if not balanced

Maps: Direct Addressing

Suppose our key space \( K \) is \( \{0, 1, \ldots, m-1\} \) and our map has at most \( m \) keys.

Using an array would give \[O(1)\] access!
Example 1

Employee social insurance number
(9 digits)

m too big
(impractical)

m = 10^9

Example 2

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

\{ (\text{address}, \text{object}) \} \rightarrow

Example 3

Keys are strings of unicode characters.

We can define a map:

\{ \text{strings} \} \rightarrow \{ \text{integers} \}

Let \( s \) be a string

\( s[0], s[1], \ldots, s[\text{length-1}] \)

Define:

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

Value ("hello") > Value ("prime")

Value ("two") < Value ("prime")