Lecture 6: Dictionary ADT

CSE 332: Data Structures & Parallelism

Winston Jodjana

Summer 2023

Take Handouts!

(Raise your hand if you need one)

Announcements

- P1 Final Due
  - Due **Tomorrow**
  - Late **Thursday**
- EX03 Heaps + EX04 D-rithmetic
  - Released!
  - Due **Friday**
- P2
  - Released **Tomorrow**
- Midterm
  - Next **Friday**

Today

- Asymptotic Analysis: Recursive
  - Writing a Recurrence Relation
  - Solving a Recurrence Relation 1: Unrolling
  - Solving a Recurrence Relation 2: Tree Method
- Dictionary ADT
- Review: Binary Search Trees
  - Trees
  - Basics, Properties, Operations

Today

- Asymptotic Analysis: Recursive
  - Writing a Recurrence Relation
  - Solving a Recurrence Relation 1: Unrolling
  - Solving a Recurrence Relation 2: Tree Method
- Dictionary ADT
- Review: Binary Search Trees
  - Trees
  - Basics, Properties, Operations

Where we are

ADTs so far:

1. Stack: `push`, `pop`, `isEmpty`, etc.
2. Queue: `enqueue`, `dequeue`, `isEmpty`, etc.
3. PriorityQueue: `insert`, `deleteMin`, etc.

Next:

4. Dictionary (a.k.a. Map): Associating `keys` with `values` (`k-v` pairs)
   - ONE OF THE MOST IMPORTANT ADTs
   - Also Set

The Dictionary (a.k.a. Map) ADT

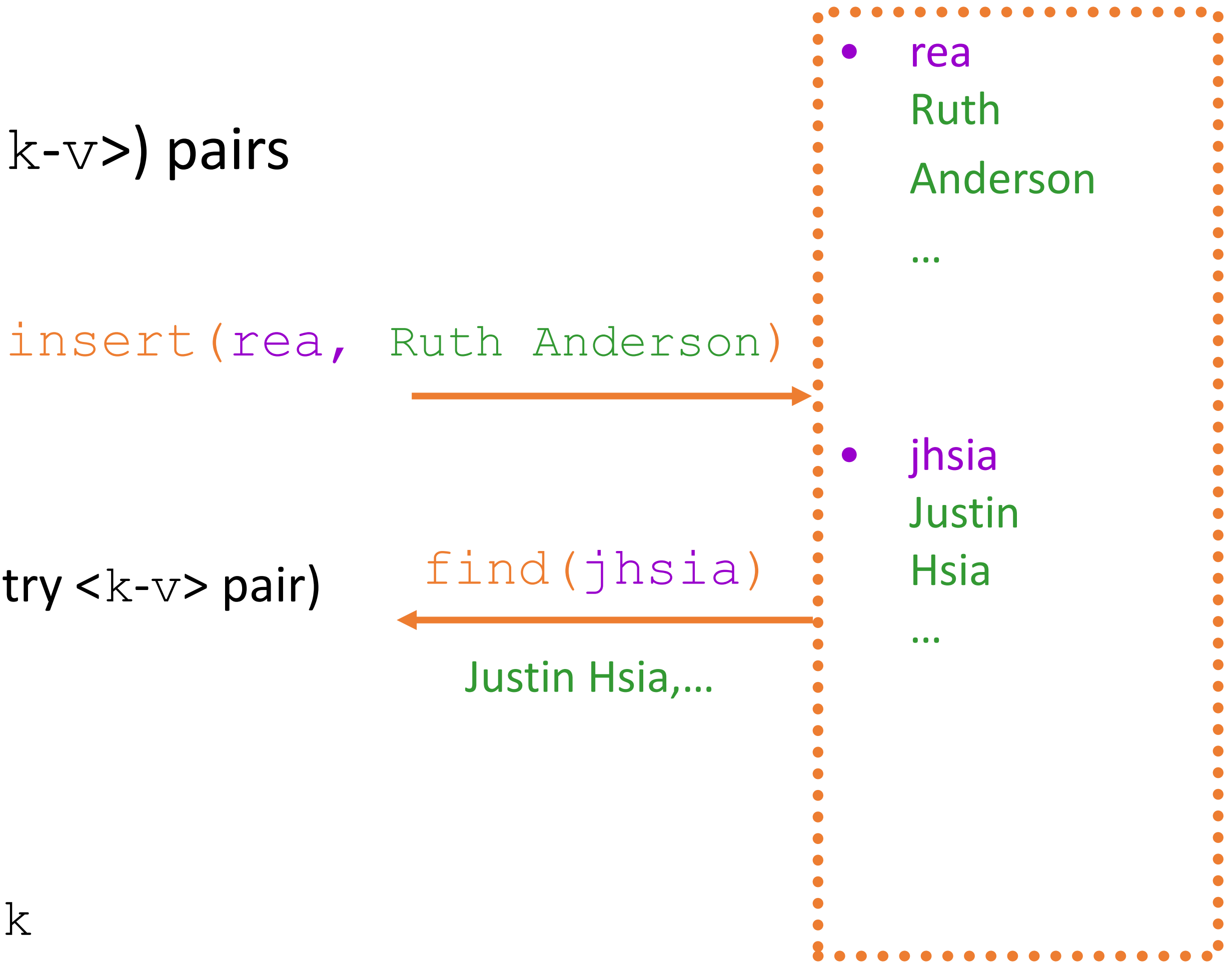
Data:

- Set of <u>unique</u> <`key-value`> (i.e., <`k-v`>) pairs

Operations:

- `insert(k,v)`:
  - places <`k-v`> in map

  (if `k` already used, overwrites existing entry <`k-v`> pair)
- `find(k)`:
  - returns `v` associated with `k`
- `delete(k)`:
  - returns and deletes `v` associated with `k`



`insert(rea, Ruth Anderson)`

`find(jhsia)`

Justin Hsia,…

- rea
  Ruth
  Anderson
  …
- jhsia
  Justin
  Hsia
  …

We will tend to emphasize the `keys`, but don't forget about the stored `values`!

Comparison: Set ADT vs. Dictionary ADT

The Set ADT is similar to a Dictionary ADT without any values

- Set: A `key` exists or not (no duplicates)
- Dictionary: A `key` has a `value` or not (no duplicates)

For `find`, `insert`, `delete`, there is little difference

- In Dictionary, values are "just along for the ride"
- So same data structure ideas work for Dictionaries and Sets
  - Java `HashSet` implemented using a `HashMap`, for instance

Set ADT may have other important operations

- `union`, `intersection`, `isSubset`, etc.
- Notice these are binary operators on sets
- We will want different data structures to implement these operators

Dictionary: Applications

Any time you want to store information according to some key and be able to retrieve it efficiently - Dictionary is the ADT to use!

- Lots of programs do that!

Networks:	router tables
Operating systems:	page tables
Compilers:	symbol tables
Databases:	dictionaries with other nice properties
Search:	inverted indexes, phone directories, …
Biology:	genome maps
etc…	

Dictionary: Primitive Data Structures

For Dictionary with n unique `k-v` pairs, worst case,

	`insert`	`find`	`delete`
Unsorted Linked List	$\Theta(\ \ )$	$\Theta(\ \ )$	$\Theta(\ \ )$
Unsorted Array	$\Theta(\ \ )$	$\Theta(\ \ )$	$\Theta(\ \ )$
Sorted Linked List	$\Theta(\ \ )$	$\Theta(\ \ )$	$\Theta(\ \ )$
Sorted Array	$\Theta(\ \ )$	$\Theta(\ \ )$	$\Theta(\ \ )$

Dictionary: Primitive Data Structures (Soln.)

For Dictionary with n unique `k-v` pairs, worst case,

	`insert`	`find`	`delete`
Unsorted Linked List	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
Unsorted Array	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
Sorted Linked List	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
Sorted Array	$\Theta(n)$	$\Theta(\log n)$	$\Theta(n)$

Timeline

- Dictionary ADT
- Review: Binary Search Trees
  - Trees
  - Basics, Properties, Operations
- Balanced BSTs?
- AVL Tree
  - Basics, Properties, Operations
- AVL Tree `insert`
  - Single Rotation
  - Double Rotation
- AVL Tree Conclusions