Lecture 27

Dictionaries

Terminology
Implementation
structure Package

TABLES
Driver's Ed' Full Sketch - I Think You Should Leave Season 2 aka TABLES
Dictionaries

TABLES
Dictionaries

An English dictionary is a map from words to their definitions. However, it also has some additional features.

For example, we can ask what the “next” word in a dictionary is. This is not possible in an arbitrary map.

We model this in data structures by defining a *dictionary* to be a map with an additional property:

The keys are comparable (and hence orderable).

Note: The textbook and structure package use the term *table* instead of dictionary.
Terminology
Note that the “map” and “dictionary” terminology is not standard across computer science.

- In Python, a `dict` is a mapping with hashable keys, and `map` applies a function to an iterable. Hashable implies orderable, so this aligns closely with the our use of dictionary in these slides.

- In the C++ standard library, a `map` has ordered keys (i.e., a dictionary here), and no dictionary.

- In Java's standard `java.util` package, `Map` is an interface for a map, and `SortedMap` is an interface for a map with ordered keys (i.e., a dictionary here).

- In the textbook and structure package, `Map` is an interface for a map, and `Table` is an interface for a map with ordered keys (i.e., a dictionary here).

- Wikipedia uses `associative array` for map, and `ordered dictionary` for ordered keys (i.e., a dictionary here).
Implementation
Implementing a Dictionary

We can utilize this additional property of the keys when implementing a dictionary.

In fact, we can significantly improve upon the performance of a generic map.

Since the keys are ordered, we can implement a dictionary with any type of binary search tree (e.g., splay tree, red-black tree, etc).

- The nodes are (key, value) pairs.
- The nodes are ordered by keys.

This approach allows us to replace the linear run-times with logarithmic run-times.

<table>
<thead>
<tr>
<th></th>
<th>get</th>
<th>put</th>
<th>remove</th>
<th>contains Key</th>
<th>contains Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>$O(\log n)$-</td>
<td>$O(\log n)$-</td>
<td>$O(\log n)$-</td>
<td>$O(\log n)$-</td>
<td>$O(n)$-</td>
</tr>
</tbody>
</table>

Worst-case run-times of various dictionary operations.

Note that these run-times assume the use of a self-balancing binary search tree with worst-case logarithmic run-times (e.g. red-black and not splay).
structure Package
Implementation of Table

In the structure package, the term table is used instead of dictionary.

Besides using a binary search tree instead of a linked list, the implementation of the Table class differs from the implementation of MapList in several ways.

- The interface OrderedMap is used instead of Map.
- Each (key, value) pair is a ComparableAssociation rather than an Association.
- Table extends Comparable and AbstractMap whereas MapList does not.
The structure package's implementation of `Table` (aka, dictionary).