Maps or Dictionaries
Dictionaries

- A Dictionary models a searchable collection of key-value entries
- The main operations of a Dictionary are for searching, inserting, and deleting items
- Multiple entries with the same key are not allowed
- Applications:
  - address book
  - student-record database
The Dictionary ADT

- **get(k)**: if the Dictionary M has an entry with key k, return its associated value; else, return null
- **put(k, v)**: insert entry (k, v) into the Dictionary M; if key k is not already in M, then return null; else, ERROR
- **remove(k)**: if the Dictionary M has an entry with key k, remove it from M and return its associated value; else, ERROR
- **size()**, **isEmpty()**
- **entrySet()**: return an iterable collection of the entries in M
- **keySet()**: return an iterable collection of the keys in M
- **values()**: return an iterator of the values in M
## Example

<table>
<thead>
<tr>
<th>Operation</th>
<th>Output</th>
<th>Dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>isEmpty()</td>
<td>true</td>
<td>Ø</td>
</tr>
<tr>
<td>put(5, A)</td>
<td>null</td>
<td>(5, A)</td>
</tr>
<tr>
<td>put(7, B)</td>
<td>null</td>
<td>(5, A), (7, B)</td>
</tr>
<tr>
<td>put(2, C)</td>
<td>null</td>
<td>(5, A), (7, B), (2, C)</td>
</tr>
<tr>
<td>put(8, D)</td>
<td>null</td>
<td>(5, A), (7, B), (2, C), (8, D)</td>
</tr>
<tr>
<td>put(2, E)</td>
<td>C</td>
<td>(5, A), (7, B), (2, C), (8, D)</td>
</tr>
<tr>
<td>get(7)</td>
<td>B</td>
<td>(5, A), (7, B), (2, C), (8, D)</td>
</tr>
<tr>
<td>get(4)</td>
<td>null</td>
<td>(5, A), (7, B), (2, C), (8, D)</td>
</tr>
<tr>
<td>get(2)</td>
<td>E</td>
<td>(5, A), (7, B), (2, C), (8, D)</td>
</tr>
<tr>
<td>size()</td>
<td>4</td>
<td>(5, A), (7, B), (2, C), (8, D)</td>
</tr>
<tr>
<td>remove(5)</td>
<td>A</td>
<td>(7, B), (2, E), (8, D)</td>
</tr>
<tr>
<td>remove(2)</td>
<td>E</td>
<td>(7, B), (8, D)</td>
</tr>
<tr>
<td>get(2)</td>
<td>null</td>
<td>(7, B), (8, D)</td>
</tr>
<tr>
<td>isEmpty()</td>
<td>false</td>
<td>(7, B), (8, D)</td>
</tr>
</tbody>
</table>
A Simple List-Based Dictionary

- We can implement a Dictionary using an unsorted list
  - We store the items of the Dictionary in a list $S$ (based on a linked list), in arbitrary order
The get(k) Algorithm

Algorithm get(k) {
    p = header
    while p is not null do
        if p.element().getKey() = k then
            return p.element().getValue()
        else p = p.next();
    return null {there is no entry with key equal to k}
}

© 2014 Goodrich, Tamassia, Goldwasser
The put(k,v) Algorithm

**Algorithm put(k,v)**

p = header

while p is not null do

  if p.element().getKey() = k then ERROR

  else p = p.next()

p = new node storing (k,v)
p.setNext (header)

header = p

n = n + 1  \{increment variable storing number of entries\}
The remove(k) Algorithm

**Algorithm** remove(k)

```java
p = header
prev = null
while p is not null do
    if p.element().getKey() = k then {
        if prev is not null then
            prev.setNext(p.next())
        else
            header = p.next()
        n = n - 1  \{decrement number of entries\}
    }
    else {
        prev = p
        p = p.next()
    }
```

© 2014 Goodrich, Tamassia, Goldwasser
Performance of a List-Based Dictionary

- Performance:
  - put takes $O(n)$ time since we need to check for duplicated keys
  - get and remove take $O(n)$ time since in the worst case (the item is not found) we traverse the entire sequence to look for an item with the given key
- The unsorted list implementation is effective only for Dictionaries of small size.