Dynamic Dictionaries

Jasleen Kaur

Spring 2014

The Dynamic Dictionary ADT

- Operations?
  - insert, remove
  - find, max, min
  - create, isFull, isEmpty, ...

- Implementation options:
  
<table>
<thead>
<tr>
<th></th>
<th>insert</th>
<th>find</th>
<th>remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>O(1)</td>
<td>O(N)</td>
<td>O(N)</td>
</tr>
<tr>
<td>Sorted array</td>
<td>O(N)</td>
<td>O(log N)</td>
<td>O(N)</td>
</tr>
<tr>
<td>Linked list</td>
<td>O(1)</td>
<td>O(N)</td>
<td>O(N)</td>
</tr>
<tr>
<td>Sorted linked list</td>
<td>O(N)</td>
<td>O(N)</td>
<td>O(1)</td>
</tr>
<tr>
<td>Binary search tree (BST)</td>
<td>O(N)</td>
<td>O(N)</td>
<td>O(N)</td>
</tr>
<tr>
<td>average</td>
<td>O(log N)</td>
<td>O(log N)</td>
<td>O(log N)</td>
</tr>
<tr>
<td>Balanced BST</td>
<td>O(log N)</td>
<td>O(log N)</td>
<td>O(log N)</td>
</tr>
<tr>
<td>Hash tables</td>
<td>O(N)</td>
<td>O(N)</td>
<td>O(N)</td>
</tr>
<tr>
<td>average</td>
<td>O(1)</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
</tbody>
</table>
BINARY SEARCH TREES

Terminology:
- Root, leaves, parents, children, edge, grandparent, grandchild, ancestor, descendant, subtree, height, depth
Implementing Trees

- **Arrays:** memory-inefficient
  - Tree with N nodes may need array of size $O(b^N)$
    - $b$: (upper bound on number of children) must be known!

- **Linked representation**
  - Storing link to each (potential) child – still memory inefficient
    - $b$ must still be known
  - Instead, store links to first child and next sibling

```java
class TreeNode {
    object element;
    TreeNode firstChild;
    TreeNode nextSibling;
}
```