Iterators & Array Representation of Trees

CS 62 - Spring 2016
Michael J. Bannister

Assignments

- **JSON File now mandatory!** Starting with Calculator assignment. If you submit by 8pm today, you will get an email notification if your JSON file is not correct.
- Lab: Compressed grid iterator
- Assignment: Darwin monster simulator
  Due: 3/11 Right before spring break.
  No mentor sessions on 3/12 or 3/13!

Tree Traversals

- Traversals:
  - Pre-Order: root, left subtree, right subtree
  - In-Order: left subtree, root, right subtree
  - Post-Order: left subtree, right subtree, root
- Most algorithms have two parts:
  - Build tree
  - Traverse tree, performing operations on nodes

Recursive In-order

```
if (!isEmpty()){
  left.inOrder()
  doSomething to this.value()
  right.inOrder()
}
```

Types of Iterators

- Pre-order: root, left subtree, right subtree
- Post-order: left subtree, right subtree, root
- In-order: left subtree, root, right subtree.

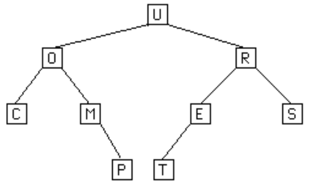
Stack Based Iterators

- Uses a stack to simulate the call stack from recursive implementation
- Each stack “frame” needs to record current line number and current node.
- Example on board.

Array Representations of Trees

Array Representation

- data[0..n-1] can hold values in trees
  - left subtree of node k in 2k+1, right in 2k+2,
  - parent in (k-1)/2



Indices:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
data[]:	U	O	R	C	M	E	S	–	–	–	P	T	–	–	–

Array Representation: Efficiency

- Tree of height h, takes 2^{h+1}-1 slots, even if only has O(h) elements
  - Bad for long, skinny trees
  - Good for full or complete trees.
- *Complete tree* is full except possibly bottom level and has all leaves at that level in leftmost positions.