Lecture 16: Trees in Arrays & Priority Queues

Fall 2016
Kim Bruce & Peter Mawhorter

This Week

- No quiz today.
- Assignment: Calculator
  - Postfix calculator
  - Start with simplified version that requires “enter” before each operation

The Midterm

- Will be graded by next week (perhaps Wednesday)
- There’s a second midterm

How can they Fit?

- How do you fit a 2D data structure into a 1D array?
  - How did \texttt{RowOrder\_Post} do it?
  - Is there an index formula that will work for a tree?
Indices

- data[0...n-1] can hold data for tree of height log n
  - left child of i goes at 2i+1, right child at 2i+2
  - parent is at (i-1)/2

Efficiency

- We need $2^{h-1}$ slots even if we only store $O(h)$ nodes
  - Bad for sticks and skinny trees
  - Good for full and well-balanced trees (topiary)
- A complete tree is full at every layer except the last, where empty spots are all on the right

PriorityQueue

```java
public interface PriorityQueue<E extends Comparable<E>> {
    /**
     * @return true if empty.
     */
    public E remove();
    public E getFirst();
    public void add(E value);
    public boolean isEmpty();
    public int size();
    public void clear();
}
```

First Pass

- Use a regular queue to hold the data
- Search for the min every time we call remove
  - remove is $O(n)$
Second Pass

- Use a regular queue to hold the data
- Every time we insert, put it in the right place
  - insert is now $O(n)$

Hmmmmmm

- What subject have we talked about without a practical use-case?
- Why are we discussing two unrelated topics today?

For Next Time

Heaps