Admin

- Pre-pre enrollment
- OnDiskSort
- Lab tomorrow

Trees

- A set of nodes based on a parent-child relationship
  - Each node has one parent
  - Root has no parent

Binary tree

- Each parent has at most 2 children
Full + Complete?

Full tree: a binary tree where every node has 0 or 2 children

Complete: All levels except the last are completely filled and all
nodes on the last level are on the left

5

Full + Complete?

Complete

Full + Complete

Neither

6

Implementing a binary tree

public class BinaryTree<T> {
    private Node root;
    private class Node {
        private T value;
        private Node left;
        private Node right;
        public Node(Node left, Node right, T value) {
            this.left = left;
            this.right = right;
            this.value = value;
        }
    }
}

7

Recursive data structure!

public class BinaryTree<T> {
    private Node root;
    private class Node {
        private T value;
        private Node left;
        private Node right;
        public Node(Node left, Node right, T value) {
            this.left = left;
            this.right = right;
            this.value = value;
        }
    }
}

public class LinkedList<T> {
    private Node root;
    private class Node {
        private T value;
        public Node next = null;
        public Node(T value) {
            this.value = value;
        }
    }
}

8
Tree traversals

Inorder(node):
  Inorder(left)
  visit node (e.g., print)
  Inorder(right)

What would be printed out?

Tree traversals

Postorder(node):
  Postorder(left)
  Postorder(right)
  visit node (e.g., print)

What would be printed out?
Tree traversals

Preorder(node):
visit node (e.g., print)
Preorder(left)
Preorder(right)

What would be printed out?

F C G E H A B E

Binary heap

A binary tree where the value of a parent is greater than or equal to the value of its children

Additional restriction: the tree must be complete!

Max heap vs. min heap
Binary heap - references

- All nodes in a heap are themselves heaps.
- Parent ≥ child.
- Complete tree.
- Level does not indicate size.

Binary heap - array

```java
int[] A = {16, 14, 10, 8, 7, 9, 3, 2, 4, 1};
```

```java
// LEFT(i)
return 2i;

// RIGHT(i)
return 2i + 1;
```

Note 0 is empty!

Binary heap - array

```java
int[] A = {16, 14, 10, 8, 7, 9, 3, 2, 4, 1};
```

```java
// LEFT(i)
return 2i;

// RIGHT(i)
return 2i + 1;
```

Left child of A[3]?
Binary heap - array

Left child of A[3]?

\[ 2 \times 3 = 6 \]

Parent of A[8]?

\[ \lfloor 8 / 2 \rfloor = 4 \]
Identify the valid heaps

[-, 15, 12, 3, 11, 10, 2, 1, 7, 8]

[-, 20, 18, 10, 17, 16, 15, 9, 14, 13]

What are heaps good for?

What are heaps good for?

What's the largest value in this heap?

Heaps are good at min/max operations (depending on min/max ordering)!
What are heaps good for?

What's the 2nd largest value? The 3rd? The 4th?

ExtractMax
Return and remove the largest element in the set. The rest of the data should stay as a heap

ExtractMax
Return and remove the largest element in the set. The rest of the data should stay as a heap
Return and remove the largest element in the set. The rest of the data should stay as a heap.
Return and remove the largest element in the set. The rest of the data should stay as a heap.

Swap with largest child.

Now what?

Swap with largest child.
Return and remove the largest element in the set. The rest of the data should stay as a heap.

41

When are we done?

42

Return and remove the largest element in the set. The rest of the data should stay as a heap.

43

We're at a leaf

44

Will it always be a leaf?
Return and remove the largest element in the set. The rest of the data should stay as a heap.

What next?

We're done!
Also stop if the node is larger than the two children.
sink/heapify/demote

```java
private void sink(int i) {
    if (left(i) < heap.size()) {
        // find the largest child
        int maxIndex = maxChildIndex(i);
        E current = heap.get(i);
        E maxChild = heap.get(maxIndex);
        if (maxChild.compareTo(current) > 0) {
            swap(i, maxIndex);
            sink(maxIndex);
        }
    }
}
```

sink runtime

```java
private void sink(int i) {
    if (left(i) < heap.size()) {
        // find the largest child
        int maxIndex = maxChildIndex(i);
        E current = heap.get(i);
        E maxChild = heap.get(maxIndex);
        if (maxChild.compareTo(current) > 0) {
            swap(i, maxIndex);
            sink(maxIndex);
        }
    }
}
```

What is the worst case runtime?

sink runtime

What is the worst case runtime?  \(O(\text{height of the tree})\)

Nodes in a binary tree

What is the tallest you can make a complete tree, using the fewest nodes?
What is the tallest you can make a complete tree, using the fewest nodes?

\[1 + 2 + 4 + 8 + \cdots + 2^{h-1} + 1 = 2^h\]

\[n = 2^h \quad \Rightarrow \quad h = \log(n)\]

```java
private void sink(int i) {
    // if we're not a leaf
    if (left(i) < heap.size()) {
        // find the largest child
        int maxIndex = maxChildIndex(i);
        E current = heap.get(i);
        E maxChild = heap.get(maxIndex);
        if (maxChild.compareTo(current) > 0) {
            swap(i, maxIndex);
            sink(maxIndex);
        }
    }
}
```

What is the worst case runtime? \(O(\log n)\)

```java
public E extractMax() {
    E maxVal = data.get(1);
    data.set(1, data.get(data.size()-1));
    data.remove(data.size()-1);
    sink(1);
    return maxVal;
}
```
What is the worst case runtime?

```java
public E extractMax() {
    E maxVal = data.get(1);
    data.set(1, data.get(data.size()-1));
    data.remove(data.size()-1);
    sink(1);
    return maxVal;
}
```

What is the worst case runtime?

```
public E extractMax() {
    E maxVal = data.get(1);
    data.set(1, data.get(data.size()-1));
    data.remove(data.size()-1);
    sink(1);
    return maxVal;
}
```

O(log n)