CS367 Lecture 18

Wednesday 16 July 2014

Announcements/Reminders:

- Readings

Last class:

- Trees (cont'd)
    - Binary Trees
    - Heaps
- Priority Queues
- `Comparable<E>` interface

Today:

- Heaps (finish)
- Binary Search Trees

Inserting into a heap (cont'd)

Heap class data members:

```
private Comparable[] items;
private int nextLoc;
```

Pseudo-code:

```
public void insert(Comparable newItem) {
```

Removing from a heap

Strategy:

Example:

Priority Queue Implementation Options

Data Structure	Insert	removeMax	Notes
Unordered array			
Ordered array			
Unordered chain			
Ordered chain			
Heap			

Using a heap to sort

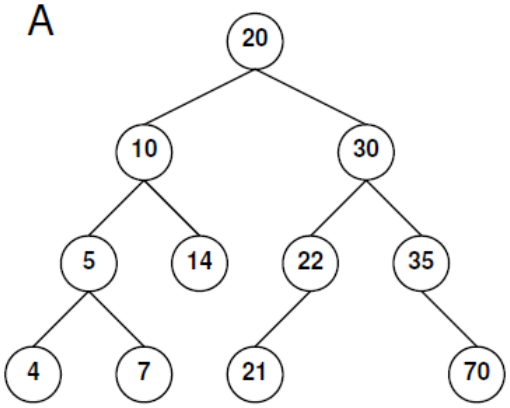
Binary Search Trees (BSTs)

Concept:

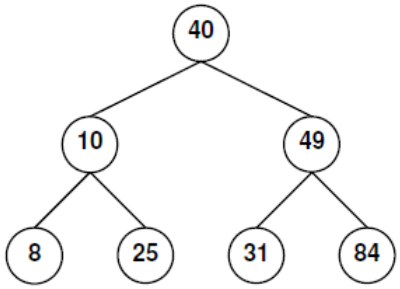
Example:

Practice with BSTs: Which are valid?

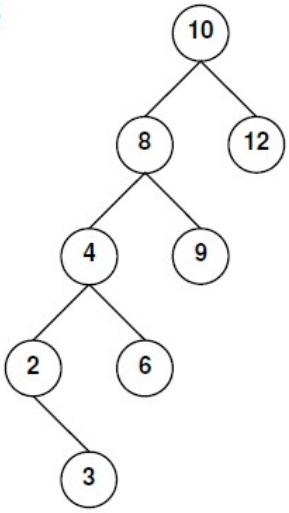
A



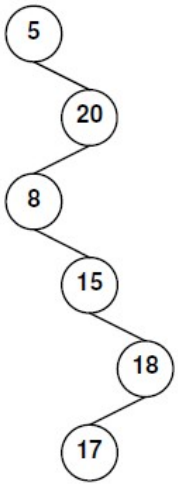
B



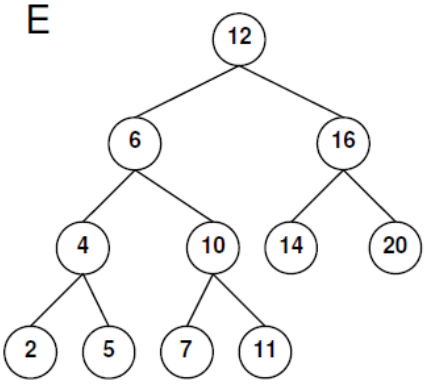
C



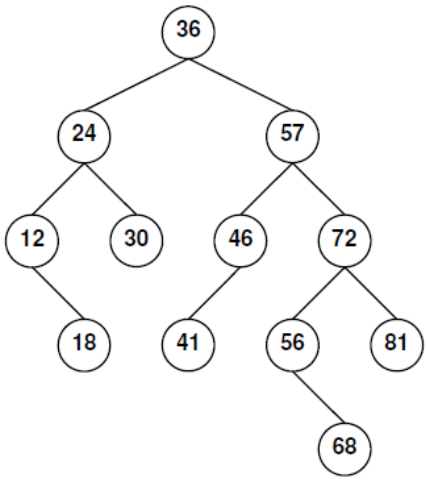
D



E



F



Implementing BSTs

```
class BSTnode<E> {

     private E key;
     private BSTnode<E> left, right, parent;

     public BSTnode(E key, BSTnode<E> left, BSTnode<E> right,
                       BSTNode<E> parent) {
          this.key = key;
          this.parent = parent;
          this.left = left;
          this.right = right;
     }

     public E getKey() { return key; }
     public BSTnode<E> getLeft() { return left; }
     public BSTnode<E> getRight() { return right; }
     public BSTnode<E> getParent() { return parent; }

     public void setKey(E newK) { key = newK; }
     public void setLeft(BSTnode<E> newL) { left = newL; }
     public void setRight(BSTnode<E> newR) { right = newR; }
     public void setParent(BSTnode<E> newP) { parent = newP; }
}
```

Implementing BSTs (cont'd)

```
public class BST<E extends Comparable<E>> {

     private BSTnode<E> root;

     public BST() { root = null; }

     public void insert(E key) throws { ... }

     public void delete(E key) { ... }

     public void boolean lookup(E key) { ... }

     public void print(PrintStream p) { ... }

     ...

}
```

BST `print()` method

Strategy:

Implementation: