Before class

Open TreeNodeExample.java
Change main to:

```java
public static void main(String[] args) {
    for(int j = 4; j < 15; j++) {
        TreeNodeExample tree = new TreeNodeExample();
        double start = System.currentTimeMillis();
        int nodes = (int)Math.pow(2, j);
        for(int i = 0; i < nodes; i++)
            tree.add(i);
        double end = System.currentTimeMillis();
        double time = (end-start)/1000.0;
        System.out.printf("Time: %f Height: %d Nodes: %d (2^%d)\n", time, tree.computeHeight(), nodes, j);
    }
}
```

Trees II

BSTs, Heaps, and PQs
Announcements

• APT Set 4 - Due Feb 28
• DNA - Due March 5

• Office hours tomorrow
  • 1:30 - 2:30

Today

• Trees
  • The importance of balanced trees
  • Traversals
  • Heaps
    • Priority Queues
The following nodes are added to a binary search tree (BST) in order. Draw the resulting BST.

6, 8, 2, 4, 1, 7, 5, 3, 9

```
public void add(int newValue) {
    if (root == null)
        root = new TreeNode(newValue);
    else
        add(newValue, root);
}

public void add(int newValue, TreeNode current) {
    if (newValue < current.myValue) {
        if (current.myLeft == null)
            current.myLeft = new TreeNode(newValue);
        else
            add(newValue, current.myLeft);
    } else
    if (current.myRight == null)
        current.myRight = new TreeNode(newValue);
    else
        add(newValue, current.myRight);
}
```

• My answer
  • 6, 8, 2, 4, 1, 7, 5, 3, 9
The following nodes are added to a binary search tree (BST) in order. Draw the resulting BST.

- 1, 2, 3, 4, 5, 6, 7, 8, 9
Code

Open TreeNodeExample.java
Change main to:

```java
public static void main(String[] args) {
    for(int j = 4; j < 15; j++){
        TreeNodeExample tree = new TreeNodeExample();
        double start = System.currentTimeMillis();
        int nodes = (int)Math.pow(2,j);
        for(int i = 0; i < nodes; i++)
            tree.add(i);
        double end = System.currentTimeMillis();
        double time = (end-start)/1000.0;
        System.out.printf("Time: %f Height: %d Nodes: %d (2^%d)\n", time, tree.computeHeight(), nodes, j);
    }
}
```
**BST**

- Order matters!
  - random input $\Rightarrow$ height $O(\log N)$
  - ordered input $\Rightarrow$ height $O(N)$

**Today**

- Trees
  - The importance of balanced trees
  - Traversals
  - Heaps
    - Priority Queues
Tree traversals

- Given a BST, how would you print the nodes **In Order**?

```
6
  2   8
  1  4   7  9
  3  5
```

- Given a BST, how would you print the nodes **In Order**?

  - go left
  - current
  - go right

```
6
  2   8
  1  4   7  9
  3  5
```
Tree traversals

• Given a BST, how would you print the nodes in **Pre Order**?
  • go left
  • current
  • go right

• Duplicate a tree

- 6
- 2
- 8
- 1
- 4
- 7
- 9
- 3
- 5
Tree traversals

• Given a BST, how would you print the nodes in **Post Order**?
  
  • go left
  • go right
  • current

• Delete a tree

Trees

• Applications
  • Computer graphics
  • Database
  • File storage on your computer
  • Internet protocols
Today

- Trees
  - The importance of balanced trees
- Traversals
- Heaps
  - Priority Queues

Queues and Stacks

```java
public static void main(String[] args) {
    Queue aQueue = new LinkedList();
    Stack aStack = new Stack();
    String[] wordsToAdd = {"compsci", "201", "is", "great"};

    for(String s: wordsToAdd){
        aQueue.add(s); //enqueue
        aStack.push(s);
    }

    while(!aQueue.isEmpty())
        System.out.print(aQueue.remove() + " "); //dequeue
    System.out.println();

    while(!aStack.isEmpty())
        System.out.print(aStack.pop() + " ");
}
```

compsci 201 is great
compsci 201
compsci is great
compsci

2/26/13
Priority Queue

- Airport queue
- First class?

```java
public static void main(String[] args) {
    //Queue aQueue = new LinkedList();
    PriorityQueue<String> aQueue = new PriorityQueue<String>();
    String[] wordsToAdd = {"compsci", "201", "is", "great"};
    for(String s: wordsToAdd) {
        aQueue.add(s);
        aStack.push(s);
    }
    while(!aQueue.isEmpty())
        System.out.print(aQueue.remove() + " ");
}
```

1. compsci 201 is great 2. great is compsci 201
3. is great compsci 201 4. 201 compsci great is
Priority Queue

- What is the output?

```java
PriorityQueue<Integer> ex = new PriorityQueue<Integer>();
ex.add(2);
ex.add(13);
ex.add(9);
ex.add(75);
ex.add(4);
while(!ex.isEmpty()) {
    System.out.println(ex.remove());
}
```

- Add in any order
- Remove smallest first

Heaps

- Common implementation of priority queues
- A tree-like structure
- Almost completely filled
  - All nodes filled expect last level
- Max-Heap - Descendants have values <= to its parent
- Min-Heap - Descendants have values >= to its parent
Heaps

- Why is a heap implemented with a priority queue?
  - Where is the min value?

```
20

20 75 43
75 84 90 57 71
84 96 91 90 93
96 91 93
```

Heaps

- Add 55 to heap
  - add node to first open slot

```
20

20 75 43
75 84 90 57 71
84 96 91 93
96 91 93
```
Heaps

• Add 55 to heap
  • If parent is larger, swap
Heaps

- Add 55 to heap
  - If parent is larger, swap

Heaps as Arrays
Heaps as Arrays

```java
public void add(double d) {
    mySize++;
    myMinHeap[mySize] = d;
    int index = mySize;
    int parentIndex = index/2;
    while((myMinHeap[parentIndex] > myMinHeap[index]) & parentIndex != 0){
        swap(index, parentIndex);
        index = parentIndex;
        parentIndex = index/2;
    }
}

private void swap(int i, int j){
    double temp = myMinHeap[i];
    myMinHeap[i] = myMinHeap[j];
    myMinHeap[j] = temp;
}
```

Heaps as Arrays

```java
public void add(double d) {
    mySize++;
    myMinHeap[mySize] = d;
    int index = mySize;
    int parentIndex = index/2;
    while((myMinHeap[parentIndex] > myMinHeap[index]) & parentIndex != 0){
        swap(index, parentIndex);
        index = parentIndex;
        parentIndex = index/2;
    }
}

private void swap(int i, int j){
    double temp = myMinHeap[i];
    myMinHeap[i] = myMinHeap[j];
    myMinHeap[j] = temp;
}
```
public void add(double d){
    mySize++;
    myMinHeap[mySize] = d;
    int index = mySize;
    int parentIndex = index/2;
    while((myMinHeap[parentIndex] > myMinHeap[index]) & parentIndex != 0){
        swap(index, parentIndex);
        index = parentIndex;
        parentIndex = index/2;
    }
}

private void swap(int i, int j){
    double temp = myMinHeap[i];
    myMinHeap[i] = myMinHeap[j];
    myMinHeap[j] = temp;
}
Heaps as Arrays

```java
public void add(double d)
{
    mySize++;  
    myMinHeap[mySize] = d;  
    int index = mySize;  
    int parentIndex = index/2;  
    while((myMinHeap[parentIndex] > myMinHeap[index]) & parentIndex != 0)
    {
        swap(index, parentIndex);  
        index = parentIndex;  
        parentIndex = index/2;  
    }
}
```

```java
private void swap(int i, int j)
{
    double temp = myMinHeap[i];  
    myMinHeap[i] = myMinHeap[j];  
    myMinHeap[j] = temp;
}
```

Remove

- Remove the root
- Move last value into root
- If a child is smaller than root
- promote the smallest child

- What would the array look like if I called remove()?
Today

- Trees
  - The importance of balanced trees
  - Traversals
- Heaps
  - Priority Queues

Announcements

- APT Set 4 - Due Feb 28
- DNA - Due March 5

- Office hours tomorrow
  - 1:30 - 2:30