Announcements

- DNA – Due October 23
- APT Set 5 – Due October 28
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

```java
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
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);
    }
}
```
• 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
```
Tree traversals

- Given a BST, how would you print the nodes in In Order?
  - go left
  - current
  - go right

Tree traversals

- Given a BST, how would you print the nodes in Pre Order?
  - go left
  - current
  - go right
Tree traversals

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

• Duplicate a tree

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() + " ");
}
```

Priority Queue

- Airport queue
  - First class?

10/22/13
**Priority Queue**

```java
public static void main(String[] args) {
    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

**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 except last level
- Max-Heap - Descendants have values <= to parent
- Min-Heap - Descendants have values >= to parent

Heaps

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

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

Heaps

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

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

```
  20
 /   \
75    43
 /     /
84    55
 /     /
96    91
```

Heaps

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

```
  20
 /   \
55    43
 /     /
84    75
 /     /
96    91
```

```
Heaps as Arrays

1. public void add(double d) {
2.     mySize++;
3.     myMinHeap[mySize] = d;
4.     int index = mySize;
5.     int parentIndex = index/2;
6.     while((myMinHeap[parentIndex] > myMinHeap[index]) &
7.         parentIndex != 0){
8.         swap(index, parentIndex);
9.         index = parentIndex;
10.        parentIndex = index/2;
11.    }
12. }
13. private void swap(int i, int j){
14.     double temp = myMinHeap[i];
15.     myMinHeap[i] = myMinHeap[j];
16.     myMinHeap[j] = temp;
17. }

Heaps as Arrays

20  75  43  84  90  57  71  96  91  93

Layer 1    Layer 2    Layer 3    Layer 4
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)
    {
        index = parentIndex;
    }
}

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

20 75 43 84 90 57 71 96 91 93 55

10/22/13
Heaps as Arrays

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;
}

20 75 43 84 55 57 71 96 91 93 90

10/22/13
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()?

20  55  43  84  75  57  71  96  91  93  90

Practice

- Draw the array for the following heap
  - Add the value 27

![Diagram of a heap](image from wikipedia)
Today

It's a Christmas tree with a heap of presents underneath!

... we're not inviting you home next year.