CSE 100: MORE BSTS, AND C++, PLUS GIT!
Announcements

• Discussion for section B today 4:00pm to 4:50pm
  • Topic: C++, git
• Tutor/TA lab hours posted on website
  • Lab hours will be held in B220.
• Start PA1 early!
CLICKERS OUT
Relating H (height) and N (#nodes)

How many nodes are in a completely filled BST?

A. \( N = \sum_{L=0}^{H-1} 2^L \)
B. \( N = 2^L \)
C. \( N = \sum_{H=0}^{N} 2^H \)
D. \( N = 2^{H+L} - 1 \)
Relating $H$ (height) and $N$ (#nodes)

OK, so...where do we go from here?

\[ N = \sum_{L=0}^{H-1} 2^L \]

A. We are done. We now have a formula that relates $H$ and $N$.
B. We need to represent the sum in closed form.
C. We need to solve for $H$. 
Relating $H$ (height) and $N$ (#nodes)

Representing the sum in closed form:

$$N = \sum_{L=0}^{H-1} 2^L = 2^H - 1$$
Relating $H$ (height) and $N$ (#nodes)

$$N = \sum_{L=0}^{H-1} 2^L = 2^H - 1$$

Finally, what is the height of the tree in terms of $N$?

A. $H = (N + 1) / 2$    B. $H = \log_2(N)$    C. $H = \log_2(N + 1)$

And since we knew finding a node was $O(H)$, we now know it is:
Summary: Running Time of Search in a BST

Worst case running time of search in any generic BST is $O(H)$

Worst case running time of search in a balanced BST is $O(\log_2 N)$
Which of the following statements is a genuine advantage of Binary Search Trees over a linked list data structure?

A. Binary search trees can be faster to retrieve information from
B. Binary search trees use less memory
C. Binary search trees are easier to implement
D. Binary search trees are typically built in to most languages, while linked lists are not
And now... C++

C++’s main priority is getting correct programs to run as fast as it can; incorrect programs are on their own.

Java’s main priority is not allowing incorrect programs to run; hopefully correct programs run reasonably fast, and the language makes it easier to generate correct programs by restricting some bad programming constructs.

-- Mark Allen Weiss, *C++ for Java Programmers*
In Java:

```java
public class BSTNode {
    public BSTNode left;
    public BSTNode right;
    public BSTNode parent;
    public int data;

    public BSTNode(int d) {
        data = d;
    }
}
```

C++, attempt 1:

```cpp
class BSTNode {
    public BSTNode left;
    public BSTNode right;
    public BSTNode parent;
    public int data;

    public BSTNode(int d) {
        data = d;
    }
};
```

Which of the following is a problem with the C++ implementation above?

A. You should not declare the types of your variables in C++
B. The class BSTNode should be declared public
C. The semicolon at the end of the class will cause a compile error
D. In C++ you specify public and private in regions, not on each variable or function
C++, attempt 2:

```cpp
class BSTNode {
public:
    BSTNode left;
    BSTNode right;
    BSTNode parent;
    int const data;

    BSTNode( const int & d ) {
        data = d;
    }
};
```

The code above in red specifies that d is passed by constant reference.

Imagine this code works (it doesn’t yet). If it did, consider creating a new BSTNode as follows:

```cpp
int myInt = 42;
BSTNode* myNode = new BSTNode( myInt );
```

Which of the following diagrams best represents what that means?

A. myInt: 42  d: 42
   d is not allowed to change what’s in its box

B. myInt: 42  d:
   The address in d’s box can’t be changed

C. myInt: 42  d: 42
   d can’t change myInt because there’s no connection
In Java:

```java
public class BSTNode {
    public BSTNode left;
    public BSTNode right;
    public BSTNode parent;
    public int data;

    public BSTNode( int d ) {
        data = d;
    }
}
```

C++, attempt 2:

```cpp
class BSTNode {  
public:
    BSTNode left;
    BSTNode right;
    BSTNode parent;
    int const data;

    BSTNode( const int & d ) {  
        data = d;
    }
};
```

Which of the following is a problem with the C++ implementation above?
A. Because data is a constant variable, the constructor will cause an error.
B. You cannot pass an integer by reference into a function. Integers must be passed by value.
C. Since d is passed by reference, you cannot assign its value to data, which is an int. You need to dereference it first.
D. The constructor needs a semi-colon at the end of its definition.
What is the problem with how we have declared left, right and parent above?
A. They should be `BSTNode*` (pointers to `BSTNodes`) and not `BSTNode` type.
B. They should be declared to be `const`.
C. They should be declared as `BSTNode&` (reference variables).
In Java:

```java
public class BSTNode {
    public BSTNode left;
    public BSTNode right;
    public BSTNode parent;
    public int data;

    public BSTNode( int d ) {
        data = d;
    }
}
```

C++, attempt 4:

```cpp
class BSTNode {
    public:
        BSTNode* left;
        BSTNode* right;
        BSTNode* parent;
        int const data;

        BSTNode( const int & d ) :
            data(d) {
            }
    }

};
```

And now, a little practice with pointers…
Which of the following statements is true about this code?

```c
int a = 5;
int b = a;
int* pt1 = a;
```

A. Both pt1 and b can be used to change the value of a.
B. Only pt1 can be used to change the value of a.
C. This code causes a compile error.
int a = 5;
int b = a;
int* pt1 = &a;

<table>
<thead>
<tr>
<th>address</th>
<th>memory cell</th>
<th>identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>512000</td>
<td>5</td>
<td>a</td>
</tr>
<tr>
<td>512004</td>
<td>5</td>
<td>b</td>
</tr>
<tr>
<td>512008</td>
<td>512000</td>
<td>pt1</td>
</tr>
</tbody>
</table>
int a = 5;
int b = a;
int* pt1 = &a;
How does the diagram change if we change the code as follows. Which is the correct picture now?

```c
int a = 5;
int & b = a;
int* pt1 = &a;
```

A.  
```
 a:  5
 b:   
 pt1: 
```

B.  
```
 a:  5
 b:  5
 pt1: 
```

C.  
```
 a:  5
 b:  5
 pt1: 
```
In Java:

```java
public class BSTNode {
    public BSTNode left;
    public BSTNode right;
    public BSTNode parent;
    public int data;

    public BSTNode( int d ) {
        data = d;
    }
}
```

C++, attempt 4:

```cpp
class BSTNode {
public:
    BSTNode* left;
    BSTNode* right;
    BSTNode* parent;
    int const data;

    BSTNode( const int & d ) :
        data(d) {} }
};
```

Are there any remaining problems with this C++ implementation?
A. Yes
B. No
In Java:

```java
public class BSTNode {
    public BSTNode left;
    public BSTNode right;
    public BSTNode parent;
    public int data;

    public BSTNode( int d ) {
        data = d;
    }
}
```

C++, attempt 5:

```cpp
class BSTNode {
public:
    BSTNode* left;
    BSTNode* right;
    BSTNode* parent;
    int const data;

    BSTNode( const int & d ) :
        data(d) {
            left = right = parent = 0;
        }
};
```

ALWAYS initialize in C++. C++ won’t do it for you. Why not?