CS24 - Problem Solving with Computers II

Binary Search Trees

First… Announcements!

Quiz 1, question 8 had incorrect answer. If you missed it and should have gotten it correct, please submit a regrade request through gradescope

As of now, median grade on quiz was 19 (95%), average was 17.16 (85.8%)

Not many people attending office hours If Wednesdays do not work, let me know.

Lab 3 and PA 01 due dates are switched

The Zybooks for data structures has been made available to the class. Instructions are on the Piazza poll thread.

Course Feedback

About even between people who say the course is going well vs. people saying the class is going too fast

If you're feeling lost, make sure to check out the live lecture and the code from it. Some of the more simple material (like cstrings) were covered in the live lecture but not in the recorded lecture. The live lectures are also recorded and available on Gauchospace

The code is posted on Gauchospace with nearly every line being commented. Play with the code and use Piazza to your advantage. You get points for posting questions so post away. You can post anonymously to your classmates and it will still show the instructors who posted. I'm trying to give at least one week between seeing new material and having to implement it in a lab/programming assignment

Ask questions during live lecture and come to office hours!!

The first couple weeks will have the most material… Most of it should have been stuff you've at least seen before (outside of pointers and memory allocation). Moving forward, we'll essentially cover one data structure per week

Course Feedback

Discussion question:

Won't be able to get to the more advanced topics requested (but feel free to stop by office hours or post on Piazza!)

Pointers, why bother? This course is mostly data structures (like a linked list). Every data structure we learn about will use pointers to organize the data. It will become obvious why they're useful in C++ (which does not automatically manage memory)

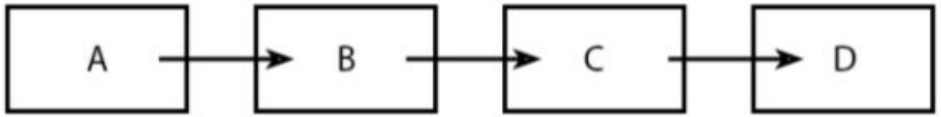
Constructors and destructors, still confused!

Pointers vs references… What's the difference?

Linked Lists (Revisited)

Made of “Node”s where each Node stores a data value and a pointer to the next Node

Singly Linked Lists are uni-directional

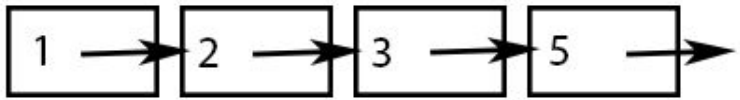


Linked Lists (Revisited)

Sorting/searching a Linked List can be very inefficient, not much benefit to sorting

Searching requires starting at the first node and traversing through every other node until the value is located or you reach the last node

Adding nodes to sorted list is difficult

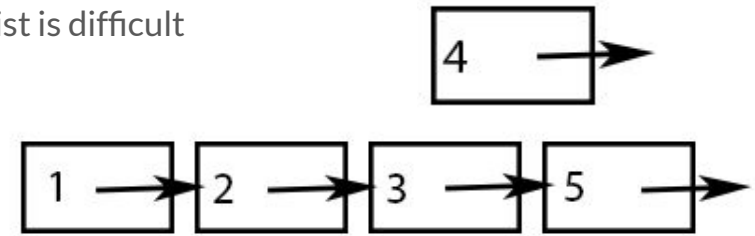


Linked Lists (Revisited)

Sorting/searching a Linked List can be very inefficient, not much benefit to sorting

Searching requires starting at the first node and traversing through every other node until the value is located or you reach the last node

Adding nodes to sorted list is difficult

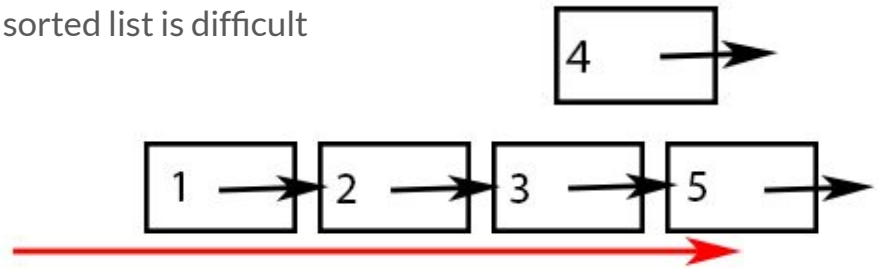


Linked Lists (Revisited)

Sorting/searching a Linked List can be very inefficient, not much benefit to sorting

Searching requires starting at the first node and traversing through every other node until the value is located or you reach the last node

Adding nodes to sorted list is difficult

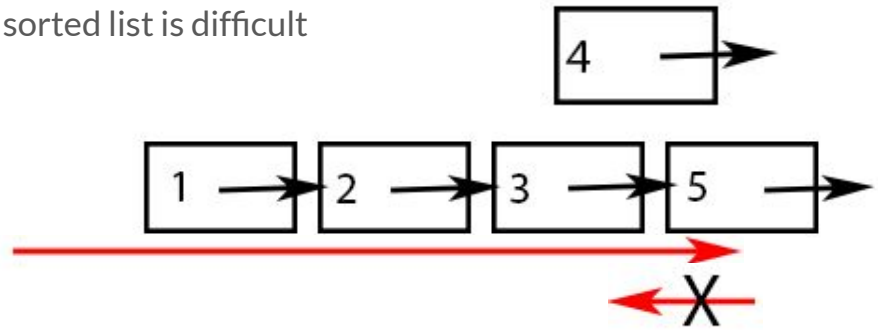


Linked Lists (Revisited)

Sorting/searching a Linked List can be very inefficient, not much benefit to sorting

Searching requires starting at the first node and traversing through every other node until the value is located or you reach the last node

Adding nodes to sorted list is difficult

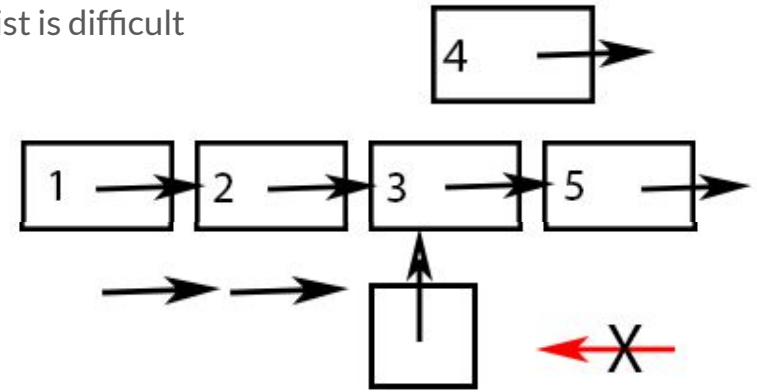


Linked Lists (Revisited)

Sorting/searching a Linked List can be very inefficient, not much benefit to sorting

Searching requires starting at the first node and traversing through every other node until the value is located or you reach the last node

Adding nodes to sorted list is difficult

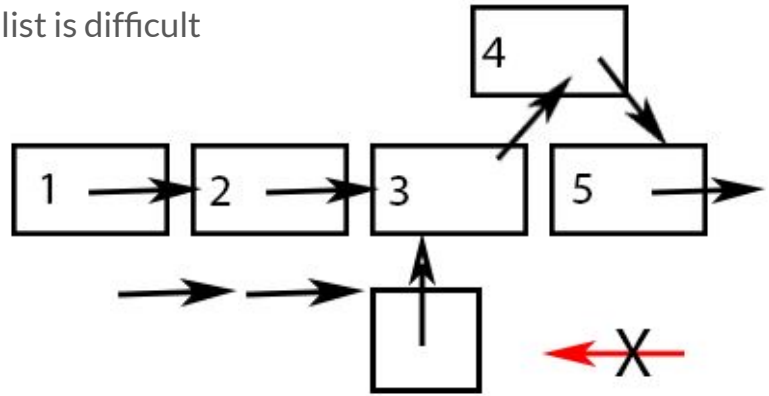


Linked Lists (Revisited)

Sorting/searching a Linked List can be very inefficient, not much benefit to sorting

Searching requires starting at the first node and traversing through every other node until the value is located or you reach the last node

Adding nodes to sorted list is difficult



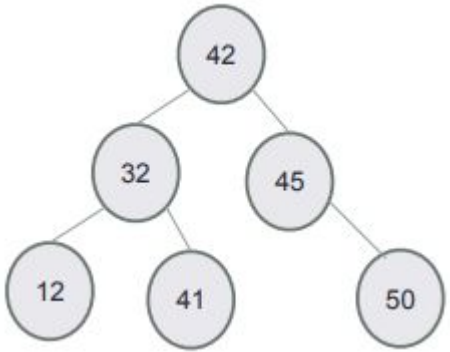
Binary Search Tree

Top Node is the root node

Every other Node has exactly one parent node (parent -> child)

A Node with no children is a leaf node

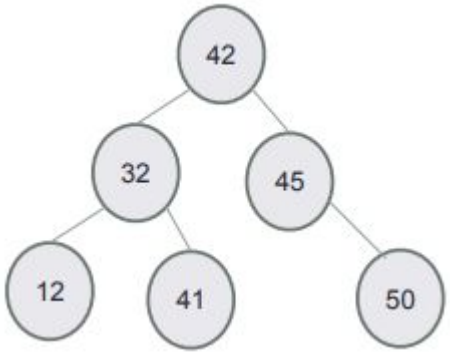
Allow for efficient sorting/search (at the cost of a longer implementation time for Inserting new Nodes)



Binary Search Tree

Typical functions:

- Min/Max
- GetParent/GetLeft/GetRight (Successor and predecessor)
- Search
- Insert
- Print (typically prints in order)



Binary Search Tree

```
class BSTNode {
public:
    BSTNode* left;
    BSTNode* right;
    BSTNode* parent; //optional
    int const data; // can be any type that can be compared with binary operations
    explicit BSTNode( const int & d ) : data(d) { // single parameter constructor
        left = right = parent = nullptr;
    }
}
```

Binary Search Tree

When filling the tree, always go as far to the left as possible before going right

6, 9, 12, 5, 4, 1, 2, 3, 7

Binary Search Tree

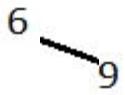
When filling the tree, always go as far to the left as possible before going right

6, 9, 12, 5, 4, 1, 2, 3, 7

Binary Search Tree

When filling the tree, always go as far to the left as possible before going right

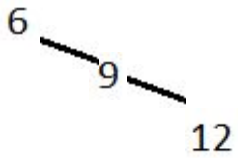
6, 9, 12, 5, 4, 1, 2, 3, 7



Binary Search Tree

When filling the tree, always go as far to the left as possible before going right

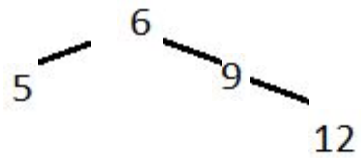
6, 9, 12, 5, 4, 1, 2, 3, 7



Binary Search Tree

When filling the tree, always go as far to the left as possible before going right

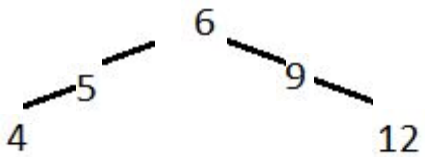
6, 9, 12, 5, 4, 1, 2, 3, 7



Binary Search Tree

When filling the tree, always go as far to the left as possible before going right

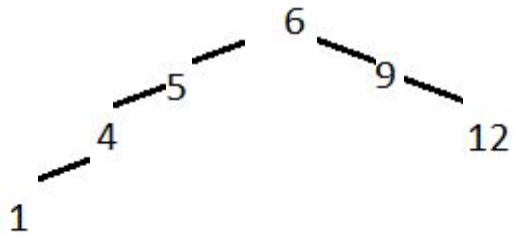
6, 9, 12, 5, 4, 1, 2, 3, 7



Binary Search Tree

When filling the tree, always go as far to the left as possible before going right

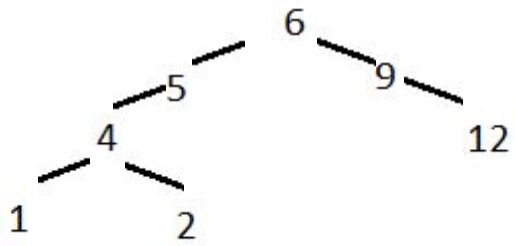
6, 9, 12, 5, 4, 1, 2, 3, 7



Binary Search Tree

When filling the tree, always go as far to the left as possible before going right

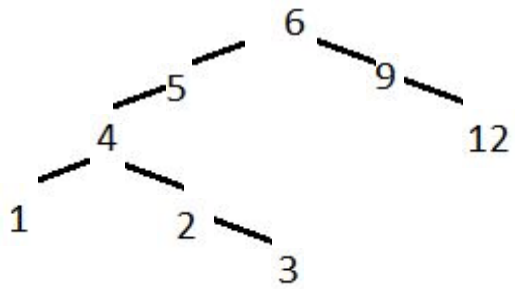
6, 9, 12, 5, 4, 1, 2, 3, 7



Binary Search Tree

When filling the tree, always go as far to the left as possible before going right

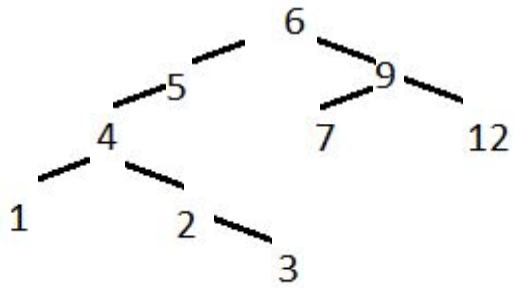
6, 9, 12, 5, 4, 1, 2, 3, 7



Binary Search Tree

When filling the tree, always go as far to the left as possible before going right

6, 9, 12, 5, 4, 1, 2, 3, 7



Binary Search Tree

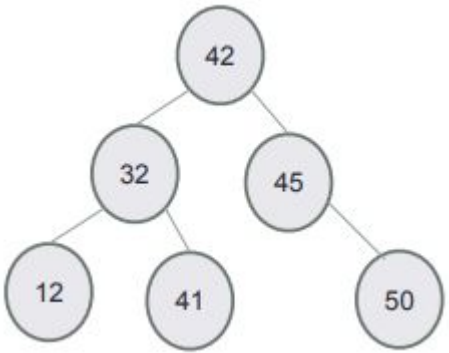
Exercises:

Try to create the binary tree for the following:
[42, 18, 16, 43, 45, 46, 47, 13, 1, 17, 13, 14, 15, 41]

And [1, 2, 3, 4, 5, 6, 7, 8, 9 10, 11…]

What do you do if there's more than one Node with the same data value?

Try to reverse engineer the tree on the right (may have more than one answer, why?):



Next Week...

Run time analysis