Data Structures

GBW ACSL - Contest #4

Clayton O'Neill — clayton@oneill.net

2015-2016

What Are Data Structures?

- Data structures are specific ways of organizing data and connecting data in a program.
- Each data structure is designed to be used with certain algorithms and access patterns.
- Many data structures are organized as a type of graph
- We'll be covering stacks, queues, and binary search trees. [1]

What Is a Stack?

A stack is a data structure that allows you to quickly add items to it, and quickly retrieve the most recently added item. This functions much like a stack of books or a PEZ dispenser. This is called "Last-In, First-Out" or LIFO order.

There are two basic operations that can be used on any stack:

PUSH This takes a single argument which is the value to put on the top of the stack

POP This removes an item and stores it in the variable given.

If you start with an empty stack, what will the following operations produce?

PUSH(A), PUSH(B), POP(X), PUSH(C) [2]

What Is a Queue?

A queue is a data structure that allows you to quickly add items to it, and quickly retrieve the oldest added item. This functions much like a lunch line, where the person first in line gets their lunch first, and the last person gets theirs last.. This is called "First-In, First-Out" or FIFO order.

There are two operations that can be used on a stack:

PUSH This takes a single argument which is the value to put on the end (or tail) of the queue

POP This removes the oldest item from the beginning (or head) of the queue.

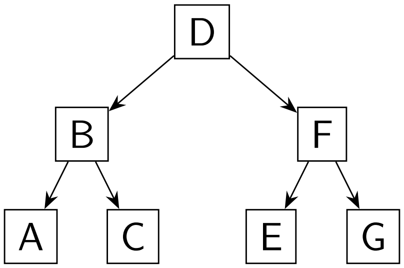
If you start with an empty queue, what will the following operations produce?

PUSH(A), PUSH(B), POP(X), PUSH(C) [3]

What Is a Binary Search Tree?

- A binary search tree is a tree where each node has a value and may have a "left" and a "right" child node below it.
- The value of the left node must be less than or equal to the value of its parent.
- The value of the right node must be greater than the value of its parent.

Below is a binary search tree of the first 7 characters of the alphabet.

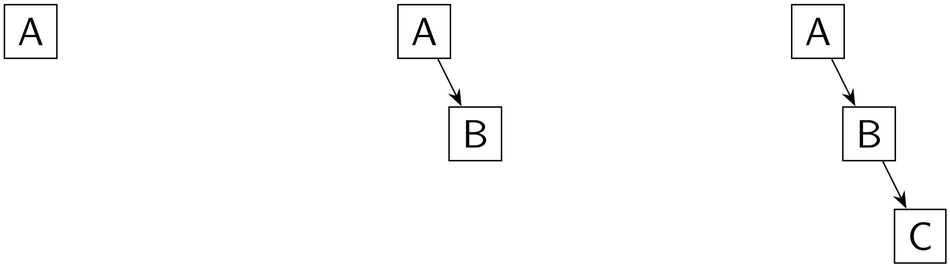


Adding Nodes To a Binary Search Tree

When adding nodes to a binary search tree, follow these steps starting at the root of the tree:

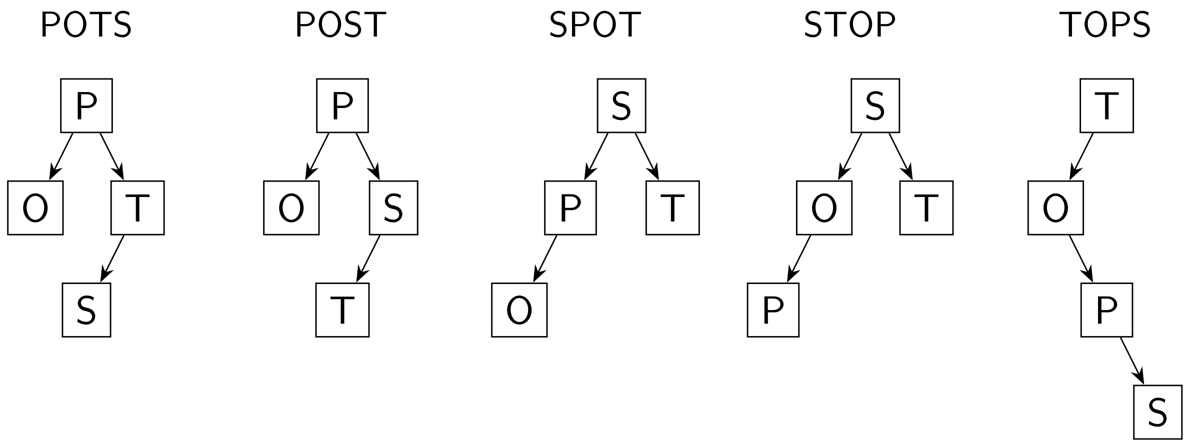
- If the item to be added is less than or equal to the current node look for a left child, otherwise look for a right child
- If there is no child, then add the node there
- Otherwise, move to the child and repeat

Below you'll see adding A, B, and C to a tree.



Order of Insertion Is Important

When adding nodes to a Binary Search Tree, the order the nodes are added impacts the shape of the tree. Below is examples of the trees formed by adding the letters as POTS, POST, SPOT, STOP, and TOPS. Note: All of these words have the same letters but form different trees.



Binary Search Tree Vocabulary

- The *root* of the tree is the single node at the top.
- The *depth* of the tree is the number of levels in the tree below the root. A tree with a single node has a depth of zero.
- An *external node* is is a blank area where a new node could be inserted.

Binary Search Tree Traversal

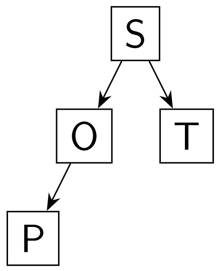
Traversal of a tree is when each node is processed in a specific order from the root of the tree. This is usually used when you want to apply a specific operation once to each node. To traverse all nodes, you start at the root, process the left, right and current nodes as described below.

There are three types of traversal:

Inorder	Processes the left nodes, the current node, and right nodes.
Preorder	Processes the current node, the left nodes, and the right nodes.
Postorder	Processes the left nodes, the right nodes, and the current node.

Traversal example

- Inorder would print POST
- Preorder would print SOPT
- Postorder would print POTS



Miscellaneous

- You should know how to predict the contents of a stack or queue given a list of PUSH and POP operations
- You should know how to draw a binary search tree given a string
- You should know how to calculate the depth of a binary search tree
- You should know how to traverse a binary search tree with inorder, preorder or postorder.
- Interactive Binary Search Tree visualization: https://www.cs.usfca.edu/ galles/visualization/BST.html

Footnotes

1. Priority queues are also covered in the ACSL handout, but don't appear to be on the ACSL classroom division contests
2. A,C
3. B,C