Project 7

1.0

Generated by Doxygen 1.8.6

Wed Apr 29 2015 22:05:13

Contents

Chapter 1

Main Page

This program contains a total of three classes – An interval class with low and high data members – An interval Node class with left, right, and parent node pointers. Aswell as max and color data members. – Interval Tree class for the tree implementation, containing a root and NIL node pointers.

Everything is tested and appears to be working properly, main program runs through the various operations that can be preformed on the tree.

Comments / Questions : `liamgomez@nevada.unr.edu`

Date

4/29/2015

Chapter 2

intervalTree

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Class Documentation

5.1 iNode< T > Class Template Reference

```
#include <intervalTree.h>
```

Public Member Functions

- iNode ()

  Default constructor for interval node class sets the max value of the node to -1.
- iNode (interval copy)

  copy constructor for interval node class

Public Attributes

- iNode< T > * left
- iNode< T > * right
- iNode< T > * parent
- int color
- int max
- interval i

5.1.1 Detailed Description

template<class T>class iNode< T >

Note

uses a template becuase i initially started with my BST imp

5.1.2 Constructor & Destructor Documentation

5.1.2.1 template<class T > iNode< T >::iNode ( )

Default constructor for interval node class sets the max value of the node to -1.

Returns

none

5.1.2.2 template<class T > iNode< T >::iNode (interval *copy*)

copy constructor for interval node class

accepts one interval and copies it to the nodes interval, used throughout program.

Parameters

copy	The interval to copy has low and high values/

Returns

none

5.1.3 Member Data Documentation

5.1.3.1 template<class T> int iNode< T >::color

5.1.3.2 template<class T> interval iNode< T >::i

5.1.3.3 template<class T> iNode<T>∗ iNode< T >::left

Note

I left these as public data members because of all the pointer logic/operations that where needed.

5.1.3.4 template<class T> int iNode< T >::max

5.1.3.5 template<class T> iNode<T>∗ iNode< T >::parent

5.1.3.6 template<class T> iNode<T>∗ iNode< T >::right

The documentation for this class was generated from the following files:

- intervalTree.h
- intervalTree.cpp

5.2 interval Class Reference

```
#include <intervalTree.h>
```

Public Member Functions

- interval ()

  default constructor for interval class
- void printInterval ()

  simple print function to assist in the printing of intervals
- void printOverlap ()

  A simple print function to assist with printing the overlap message and corresponding interval.
- interval & operator= (const interval &copy)

  overloaded assignment operator for interval class

Public Attributes

- int low
- int high

5.2.1 Constructor & Destructor Documentation

5.2.1.1 interval::interval ( )

default constructor for interval class

sets the low and high values to a negative flag value of -1

5.2.2 Member Function Documentation

5.2.2.1 interval & interval::operator= (const interval & *copy*)

overloaded assignment operator for interval class

Simply copies the low and high values of the left side to the current interval. Not really used but thought it might come in handy.

Parameters

copy	The interval to copy

Returns

∗this interval to finalize assignment

5.2.2.2 void interval::printInterval ( )

simple print function to assist in the printing of intervals

5.2.2.3 void interval::printOverlap ( )

A simple print function to assist with printing the overlap message and corresponding interval.

5.2.3 Member Data Documentation

5.2.3.1 int interval::high

5.2.3.2 int interval::low

The documentation for this class was generated from the following files:

- intervalTree.h
- intervalTree.cpp

5.3 intervalTree< T > Class Template Reference

```
#include <intervalTree.h>
```

Public Member Functions

- intervalTree ()

  default constructor for interval tree class
- ∼intervalTree ()

  default destructor for interval tree class
- void destroyTree (iNode< T > ∗sub)

  destroys the tree by preforming a post order deletion
- void insert (const interval input)

  Inserts the input interval into new node and puts it into the tree.
- iNode< T > ∗ getLeftMostNode (iNode< T > ∗sub)

  Obtains the left most node for a given subtree.
- iNode< T > ∗ search (iNode< T > ∗sub, const interval search)

  Searches the tree for a given interval.
- iNode< T > ∗ searchHelper (const interval search)

  The recursive helper for the search function.
- void deleteInterval (iNode< T > ∗target)
- void fixDelete (iNode< T > ∗x)
- void swap (iNode< T > ∗a, iNode< T > ∗b)

  Simple function thats swaps the two nodes a and b.
- void rotateLeft (iNode< T > ∗x)

  Rotates the tree left around the specified x node.
- void rotateRight (iNode< T > ∗x)

  Rotates the tree right around the specified x node.
- void fixInsertion (iNode< T > ∗fix)

  Maintains the Red/Black properties of the tree after an insertion.
- void fixMaxValues (iNode< T > ∗x)

  Finds the largest max values in a portion of the tree.
- bool overlaps (interval a, interval b)

  Determines if two intervals (a) and (b) overlap in some form.
- void findOverlap (iNode< T > ∗sub, const interval &target)

  Finds all the overlaps in a given subtree (root)
- void overlapHelper (const interval &target)

  Recursive helper for the findOverlap function, just uses the root node as the subtree to find all overlap conflicts.
- void preorder (iNode< T > ∗sub)

  Preorder print function for the interval tree.
- void inorderPrint (iNode< T > ∗sub)

  inorder print function for interval tree
- void showTree ()

  Prints the tree inorder then in preorder.

5.3.1 Detailed Description

template<class T>class intervalTree< T >

Note

uses a template becuase i initially started with my BST imp

5.3.2 Constructor & Destructor Documentation

5.3.2.1 template<class T > intervalTree< T >::intervalTree ( )

default constructor for interval tree class

This constructor allocates two nodes, one for the NIL pointer and the other for the root pointer. It sets all the interval and max variables to a very low number to avoid any conflicts. Sets root to NIL to make the tree empty. To maintain RB tree properties the NIL and root nodes both start with the color black.

Returns

none

5.3.2.2 template<class T > intervalTree< T >::∼intervalTree ( )

default destructor for interval tree class

uses the method destroy tree, on the root node to preform a post order deletion of the nodes in the tree.

Precondition

the program must end

Postcondition

the tree will be deleted and the memory returned to the operating system.

Returns

none

5.3.3 Member Function Documentation

5.3.3.1 template<class T > void intervalTree< T >::deleteInterval (iNode< T > ∗ *target*)

5.3.3.2 template<class T > void intervalTree< T >::destroyTree (iNode< T > ∗ *sub*)

destroys the tree by preforming a post order deletion

This function is used in the default destructor for the interval tree class.

Parameters

sub	The root pointer of the tree

Returns

none

5.3.3.3 template<class T > void intervalTree< T >::findOverlap (iNode< T > ∗ *sub,* const interval & *target*)

Finds all the overlaps in a given subtree (root)

This function recursively goes through the tree finding overlapping intervals, once one is found it is printed using the interval class methods.

Parameters

sub	The subtree to search for overlaps in (initially should be root)
target	The target interval to search for overlap conflicts with

Returns

none

5.3.3.4 template<class T > void intervalTree< T >::fixDelete (iNode< T > ∗ *x*)

5.3.3.5 template<class T > void intervalTree< T >::fixInsertion (iNode< T > ∗ *fix*)

Maintains the Red/Black properties of the tree after an insertion.

This function uses five different cases to maintain the Red/black tree properties; Property 3 - All leaves must be black, Property 4

- Both children of a red node are always black, and Property 5 - All paths from a node to its leaf contain the same number of black nodes

Precondition

Should only be called after the insertion operation is preformed on the tree.

Postcondition

The tree will be fixed, and its RB properties maintained.

Parameters

fix	The pointer to the subtree / tree to fix

Returns

none

5.3.3.6 template<class T > void intervalTree< T >::fixMaxValues (iNode< T > ∗ *x*)

Finds the largest max values in a portion of the tree.

This function finds the highest max value from a given subtree, it is used to maintain the max data values throughout insertion / deletion operations. It works by simply looping through the tree comparing max values at each stop.

Parameters

x	The interval node to find the max value in.

Returns

none

5.3.3.7 template<class T > iNode< T > ∗ intervalTree< T >::getLeftMostNode (iNode< T > ∗ *sub*)

Obtains the left most node for a given subtree.

Gets the farthest node to the left and returns it, the subtree is usually the root node, but it doesn't matter either way.

Parameters

sub	The subtree in which to obtain the left most node

Returns

The interval node that is the farthest left for the given subtree.

5.3.3.8 template<class T > void intervalTree< T >::inorderPrint (iNode< T > ∗ *sub*)

inorder print function for interval tree

Parameters

sub	The subtree to print

Returns

none

5.3.3.9 template<class T > void intervalTree< T >::insert (const interval *input*)

Inserts the input interval into new node and puts it into the tree.

This insertion function first finds the appropriate place for the new node, using the interval low value (i.low) as the key. It then checks the various cases, and prepares the node for insertion fix.

Precondition

The interval should not already exist in the tree, and should probably not be negative although i haven't tested it.

Postcondition

The new interval will be placed into the tree with no children, and be the color RED to prepare it for insertion fix.

Parameters

input	The interval to be inputed into the interval tree

Returns

none

5.3.3.10 template<class T > void intervalTree< T >::overlapHelper (const interval & *target*)

Recursive helper for the findOverlap function, just uses the root node as the subtree to find all overlap conflicts.

Parameters

target	The interval to search for overlaps in the tree with

Returns

none

5.3.3.11 template< class T > bool intervalTree< T >::overlaps (interval *a*, interval *b*)

Determines if two intervals (a) and (b) overlap in some form.

Does a simple comparison to determine if two intervals overlap used to find overlapping intervals in the overlap-Helper

Parameters

a	First interval for overlap testing
b	Second interval for overlap testing

Returns

Boolean result of the test, states if they are overlapping or not.

5.3.3.12 template< class T > void intervalTree< T >::preorder (iNode< T > ∗ *sub*)

Preorder print function for the interval tree.

Parameters

sub	The subtree to print

Returns

none

5.3.3.13 template< class T > void intervalTree< T >::rotateLeft (iNode< T > ∗ *x*)

Rotates the tree left around the specified x node.

This function will rotate the tree around the interval node x, does not change the order of the elements.

Precondition

should only be used by fix functions

Parameters

x	The pivot node for which to rotate on

Returns

none

5.3.3.14 template< class T > void intervalTree< T >::rotateRight (iNode< T > ∗ *x*)

Rotates the tree right around the specified x node.

This function will rotate the tree around the interval node x, does not change the order of the elements.

Precondition

should only be used by fix functions

Parameters

x	The pivot node for which to rotate on

Returns

none

5.3.3.15 template<class T > iNode< T > ∗ intervalTree< T >::search (iNode< T > ∗ *sub,* const interval *find*)

Searches the tree for a given interval.

This function recursively searches the tree, going either left or right based on the key value.

Precondition

the subtree to search for the interval should be the root initially, Should only be called using the searchHelper function or bad things will happen.

Postcondition

The given interval will either be found, or a null pointer will be returned.

Parameters

sub	The subtree to find the interval in (Should be root!)
find	The interval to find in the tree

Returns

The interval node that contains the target interval

5.3.3.16 template<class T > iNode< T > ∗ intervalTree< T >::searchHelper (const interval *find*)

The recursive helper for the search function.

Simply calls the search function on the root node and returns the found node to caller.

Precondition

the interval should exist in the tree, if not the caller should force conditions for no run on nullptr return

Parameters

find	The interval to find

Returns

The interval node that contains the specified interval, nullptr if the interval could not be found.

5.3.3.17 template<class T > void intervalTree< T >::showTree ( )

Prints the tree inorder then in preorder.

calls both functions on the root node to display entire tree, also prints the current interval at the root node and color of each node.

Precondition

the tree should not be empty

Postcondition

the interval tree contents will be displayed to the user in both orders.

Returns

none

5.3.3.18 template<class T > void intervalTree< T >::swap (iNode< T > * *a*, iNode< T > * *b*)

Simple function thats swaps the two nodes a and b.

Parameters

a	First interval node to swap
b	Second interval node to swap

Returns

none

The documentation for this class was generated from the following files:

- intervalTree.h
- intervalTree.cpp

Chapter 6

File Documentation

6.1 intervalTree.cpp File Reference

Implementation file for interval tree and all other necassary classes.

```
#include "intervalTree.h"
```

Functions

- int findMax (int a, int b)

  Finds the max number between two numbers and returns it.

6.1.1 Detailed Description

Implementation file for interval tree and all other necassary classes.

Author

Liam Gomez

Note

All extra credit is implemented, tested and working properly.

Date

4/29/2015

Author

Liam Gomez

6.1.2 Function Documentation

6.1.2.1 int findMax (int *a,* int *b*)

Finds the max number between two numbers and returns it.

simple comparison to find the max of two inputs, used throughout the interval tree.

Parameters

a	The first number to compare
b	The second number to compare

Returns

An integer, the maximum of the two. If equal b will be returned instead.

6.2 intervalTree.h File Reference

Class Spec for interval tree and all other necassary classes.

```
#include <iostream>
#include <string>
#include "intervalTree.cpp"
```

Classes

- class interval
- class iNode< T >
- class intervalTree< T >

Functions

- int findMax (int a, int b)

  Finds the max number between two numbers and returns it.

Variables

- const int BLACK = 0
- const int RED = 1

6.2.1 Detailed Description

Class Spec for interval tree and all other necassary classes.

Author

Liam Gomez

Date

4/29/2015

6.2.2 Function Documentation

6.2.2.1 int findMax (int *a*, int *b*)

Finds the max number between two numbers and returns it.

simple comparison to find the max of two inputs, used throughout the interval tree.

Parameters

a	The first number to compare
b	The second number to compare

Returns

An integer, the maximum of the two. If equal b will be returned instead.

6.2.3 Variable Documentation

6.2.3.1 const int BLACK = 0

6.2.3.2 const int RED = 1

6.3 main.cpp File Reference

```
#include "intervalTree.h"
#include <iostream>
#include <fstream>
#include <string>
```

Functions

- int main ()

6.3.1 Function Documentation

6.3.1.1 int main ( )

6.4 README.md File Reference

Index