SSL Inlab 6

Generated by Doxygen 1.8.17

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 BinarySearchTree Class Reference

Class for a Binary Search Tree.

Collaboration diagram for BinarySearchTree:

Public Types

- enum **order** { **PRE**, **IN**, **POST** }

Public Member Functions

- BinarySearchTree ()

  This is a constructor method to create a Binary search tree. Sets root to NULL.
- void insert (ll val)

  This is a member function to insert a new element. Inserts a new node with data as the element in it.
- void traverse (BSTNode ∗T, order tt)

  This is a printer function to print the tree in the traversal order given.
- ll height (BSTNode ∗T)

  This is a member function to get the height of the node.

Public Attributes

- BSTNode ∗ root

  Pointer to the root of the tree.

3.1.1 Detailed Description

Class for a Binary Search Tree.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 BinarySearchTree()

```
BinarySearchTree::BinarySearchTree ( )  [inline]
```

This is a constructor method to create a Binary search tree. Sets root to NULL.

Parameters

in	//	val

3.1.3 Member Function Documentation

3.1.3.1 height()

```
ll BinarySearchTree::height (
             BSTNode * T )  [inline]
```

This is a member function to get the height of the node.

Parameters

in	*BSTNode*∗	T
out	//	height

Returns

height of the node

3.1.3.2 insert()

```
void BinarySearchTree::insert (
             ll val )  [inline]
```

This is a member function to insert a new element. Inserts a new node with data as the element in it.

Parameters

in	//	data

Returns

NULL

3.1.3.3 traverse()

```
void BinarySearchTree::traverse (
             BSTNode * T,
             order tt )  [inline]
```

This is a printer function to print the tree in the traversal order given.

Parameters

in	*BSTNode∗*	T
in	*order*	TT
out	*prints*	the binary search tree in the order given

Returns

NULL

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

- DSA.cpp

3.2 BSTNode Class Reference

Node in a Binary Search tree.

Collaboration diagram for BSTNode:

Public Member Functions

- BSTNode (ll val)

  This is a constructor method to create a DoublyLinkedListNode. Sets info to val. Sets level to 0. Sets left to NULL. Sets right to NULL.

Public Attributes

- ll info

  Contains the element.
- ll level

  Contains level of the node in the tree.
- BSTNode ∗ left

  Pointer to the left node.
- BSTNode ∗ right

  Pointer to the right node.

3.2.1 Detailed Description

Node in a Binary Search tree.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 BSTNode()

```
BSTNode::BSTNode (
              ll val )  [inline]
```

This is a constructor method to create a DoublyLinkedListNode. Sets info to val. Sets level to 0. Sets left to NULL. Sets right to NULL.

Parameters

`in`	*ll*	val

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

- DSA.cpp

3.3 DoublyLinkedList Class Reference

Class for a Doubly Linked List.

Collaboration diagram for DoublyLinkedList:

Public Member Functions

- DoublyLinkedList ()

  This is a constructor method to create a SinglyLinkedList. Sets head to NULL. Sets tail to NULL.
- void insert (ll data)

  This is a member function to insert a new element. Inserts a new node with data as the element in it.
- void printer (string sep=", ")

  This is a printer function to print the values in the list.
- void reverse ()

  This is a member function to reverse the order of the list.

Public Attributes

- DoublyLinkedListNode ∗ head

  Pointer to head of the list.
- DoublyLinkedListNode ∗ tail

  Pointer to tail of the list.

3.3.1 Detailed Description

Class for a Doubly Linked List.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 DoublyLinkedList()

```
DoublyLinkedList::DoublyLinkedList ( )  [inline]
```

This is a constructor method to create a SinglyLinkedList. Sets head to NULL. Sets tail to NULL.

Parameters

in	*NULL*	

3.3.3 Member Function Documentation

3.3.3.1 insert()

```
void DoublyLinkedList::insert (
            ll data )  [inline]
```

This is a member function to insert a new element. Inserts a new node with data as the element in it.

Parameters

in	*ll*	data

Returns

NULL

3.3.3.2 printer()

```
void DoublyLinkedList::printer (
            string sep = ", " )  [inline]
```

This is a printer function to print the values in the list.

Parameters

in	*NULL*	
out	*Prints*	the list

Returns

NULL

3.3.3.3 reverse()

```
void DoublyLinkedList::reverse ( )  [inline]
```

This is a member function to reverse the order of the list.

Parameters

in	*NULL*	

Returns

NULL

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

- DSA.cpp

3.4 DoublyLinkedListNode Class Reference

Node in a Doubly Linked List.

Collaboration diagram for DoublyLinkedListNode:

Public Member Functions

- DoublyLinkedListNode ()

  This is a constructor method to create a DoublyLinkedListNode. Sets data to -1. Sets next to NULL. Sets prev to NULL.
- DoublyLinkedListNode (ll val)

  This is a constructor method to create a DoublyLinkedListNode. Sets data to val. Sets next to NULL. Sets prev to NULL.

Public Attributes

- ll data

  Data in the node.
- DoublyLinkedListNode ∗ next

  Pointer to next node.
- DoublyLinkedListNode ∗ prev

  Pointer to the previous node.

3.4.1 Detailed Description

Node in a Doubly Linked List.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 DoublyLinkedListNode() `[1/2]`

```
DoublyLinkedListNode::DoublyLinkedListNode ( )  [inline]
```

This is a constructor method to create a DoublyLinkedListNode. Sets data to -1. Sets next to NULL. Sets prev to NULL.

Parameters

in	*NULL*	

3.4.2.2 DoublyLinkedListNode() `[2/2]`

```
DoublyLinkedListNode::DoublyLinkedListNode (
            ll val )  [inline]
```

This is a constructor method to create a DoublyLinkedListNode. Sets data to val. Sets next to NULL. Sets prev to NULL.

Parameters

in	*ll*	val

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

- DSA.cpp

3.5 Heap Class Reference

Class for a binary heap.

Public Member Functions

- Heap (int cap)

  This is a constructor method to create a Heap. Initializes a new array with cap as the number of elements Sets n to 0.
- int parent (int i)

  This is a member function to find the parent of a node.
- int left (ll i)

  This is a member function to find the left child of a node.
- int right (ll i)

  This is a member function to find the right child of a node.
- void insert (int val)

  This is a member function to insert a new element. Inserts a new node with data as the element in it.
- int min ()

  This is a member function to find the minimum element in a heap.
- void Heapify (int root)

  This is to make it into a heap when both the left and right subheaps satisfy the heap property but not the whole heap.
- void deleteMin ()

  This is to delete the minimum element in a heap.

Public Attributes

- int cap

  Maximum number of elements in the heap.

3.5.1 Detailed Description

Class for a binary heap.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Heap()

```
Heap::Heap (
             int cap )  [inline]
```

This is a constructor method to create a Heap. Initializes a new array with cap as the number of elements Sets n to 0.

Parameters

in	*int*	cap

3.5.3 Member Function Documentation

3.5.3.1 deleteMin()

```
void Heap::deleteMin ( )  [inline]
```

This is to delete the minimum element in a heap.

Returns

NULL

3.5.3.2 Heapify()

```
void Heap::Heapify (
             int root )  [inline]
```

This is to make it into a heap when both the left and right subheaps satisfy the heap property but not the whole heap.

Parameters

`in`	*int*	root

Returns

NULL

3.5.3.3 insert()

```
void Heap::insert (
            int val )  [inline]
```

This is a member function to insert a new element. Inserts a new node with data as the element in it.

Parameters

`in`	*int*	val

Returns

NULL

3.5.3.4 left()

```
int Heap::left (
            ll i )  [inline]
```

This is a member function to find the left child of a node.

Parameters

`in`	*int*	i
`out`	*int*	2∗i+1

Returns

The index of the left child of element

3.5.3.5 min()

```
int Heap::min ( )  [inline]
```

This is a member function to find the minimum element in a heap.

Returns

The element with the minimum value

3.5.3.6 parent()

```
int Heap::parent (
             int i )  [inline]
```

This is a member function to find the parent of a node.

Parameters

in	*int*	i
out	*int*	(i-1)/2

Returns

The index of the parent of element

3.5.3.7 right()

```
int Heap::right (
             ll i )  [inline]
```

This is a member function to find the right child of a node.

Parameters

in	*int*	i
out	*int*	2∗(i+1)

Returns

The index of the left child of element

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

- DSA.cpp

3.6 SinglyLinkedList Class Reference

Class for a Singly Linked List.

Collaboration diagram for SinglyLinkedList:

Public Member Functions

- SinglyLinkedList ()

  This is a constructor method to create a SinglyLinkedList. Sets head to NULL. Sets tail to NULL.
- void insert (ll data)

  This is a member function to insert a new element. Inserts a new node with data as the element in it.
- SinglyLinkedListNode ∗ find (ll data)

  This is a member function to find an element.
- bool deleteVal (ll data)

  This is a member function to delete an element.
- void printer (string sep=", ")

  This is a printer function to print the values in the list.
- void reverse ()

  This is a member function to reverse the order of the list.

Public Attributes

- SinglyLinkedListNode ∗ head

  Pointer to the head of the list.
- SinglyLinkedListNode ∗ tail

  Pointer to the tail of the list.

3.6.1 Detailed Description

Class for a Singly Linked List.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 SinglyLinkedList()

```
SinglyLinkedList::SinglyLinkedList ( )  [inline]
```

This is a constructor method to create a SinglyLinkedList. Sets head to NULL. Sets tail to NULL.

Parameters

in	*NULL*	

3.6.3 Member Function Documentation

3.6.3.1 deleteVal()

```
bool SinglyLinkedList::deleteVal (
             ll data )  [inline]
```

This is a member function to delete an element.

Parameters

in	*ll*	data
out	*bool*	

Returns

true if successfully deleted else false

3.6.3.2 find()

```
SinglyLinkedListNode* SinglyLinkedList::find (
             ll data )  [inline]
```

This is a member function to find an element.

Parameters

in	*ll*	data
out	*ll*	prev

Returns

NULL if not found else returns pointer to the node containing the element

3.6.3.3 insert()

```
void SinglyLinkedList::insert (
             ll data )  [inline]
```

This is a member function to insert a new element. Inserts a new node with data as the element in it.

Parameters

in	*ll*	data

3.6.3.4 printer()

```
void SinglyLinkedList::printer (
             string sep = ", " )  [inline]
```

This is a printer function to print the values in the list.

Parameters

in	*NULL*	
out	*Prints*	the list

Returns

NULL

3.6.3.5 reverse()

```
void SinglyLinkedList::reverse ( )  [inline]
```

This is a member function to reverse the order of the list.

Parameters

in	*NULL*	

Returns

NULL

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

- DSA.cpp

3.7 SinglyLinkedListNode Class Reference

Node in a Singly Linked List.

Collaboration diagram for SinglyLinkedListNode:

Public Member Functions

- SinglyLinkedListNode ()

  This is a constructor method to create a SinglyLinkedListNode. Sets data to -1. Sets next to NULL.
- SinglyLinkedListNode (ll val)

  This is a constructor method to create a SinglyLinkedListNode. Sets data to val. Sets next to NULL.

Public Attributes

- ll data

    Data stored in the node.
- SinglyLinkedListNode ∗ next

    Pointer to the next node.

3.7.1 Detailed Description

Node in a Singly Linked List.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 SinglyLinkedListNode() [1/2]

```
SinglyLinkedListNode::SinglyLinkedListNode ( )  [inline]
```

This is a constructor method to create a SinglyLinkedListNode. Sets data to -1. Sets next to NULL.

Parameters

in	*NULL*	

3.7.2.2 SinglyLinkedListNode() [2/2]

```
SinglyLinkedListNode::SinglyLinkedListNode (
            ll val )  [inline]
```

This is a constructor method to create a SinglyLinkedListNode. Sets data to val. Sets next to NULL.

Parameters

in	*ll*	val

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

- DSA.cpp

3.8 Trie Class Reference

Class for a suffix trie.

Public Member Functions

- Trie ()
- bool find (Trie ∗T, char c)

  This is a member function to find an element.
- void insert (string s)

  This is a member function to insert a new element. Inserts a new node with data as the element in it.
- bool checkPrefix (string s)

  This is a member function to check if a prefix is present in the trie.
- ll countPrefix (string s)

  This is a member function to get the number of count of matches of a prefix in the trie.

Public Attributes

- ll count

  Keeps count of nodes below it.
- map< char, Trie ∗ > nodes

  Node in a suffix trie.

3.8.1 Detailed Description

Class for a suffix trie.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 Trie()

```
Trie::Trie ( )  [inline]
```

This is a constructor method to create a Suffix Trie. Sets count to 0. Sets nodes to empty map.

Parameters

in	*NULL*	

3.8.3 Member Function Documentation

3.8.3.1 checkPrefix()

```
bool Trie::checkPrefix (
             string s )  [inline]
```

This is a member function to check if a prefix is present in the trie.

Parameters

in	*string*	s

Returns

true if found else false

3.8.3.2 countPrefix()

```
ll Trie::countPrefix (
            string s )  [inline]
```

This is a member function to get the number of count of matches of a prefix in the trie.

Parameters

in	*string*	s
out	*ll*	countprefix

Returns

number of matches

3.8.3.3 find()

```
bool Trie::find (
            Trie * T,
            char c )  [inline]
```

This is a member function to find an element.

Parameters

in	*Trie∗*	T
in	*ll*	char c
out	*bool*	

Returns

true if found else false

3.8.3.4 insert()

```
void Trie::insert (
             string s )  [inline]
```

This is a member function to insert a new element. Inserts a new node with data as the element in it.

Parameters

in	*string*	s

Returns

NULL

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

- DSA.cpp

Chapter 4

File Documentation

4.1 DSA.cpp File Reference

This file contains 4 different types of data structures.

```
#include <bits/stdc++.h>
```

Include dependency graph for DSA.cpp:

Index