My Project

Generated by Doxygen 1.9.5

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

BinarySearchTree It has constructor with no parameters BinarySearchTree ()
It has following member functions (i)void insert(ll val) (ii)void traverse (BSTNode∗ T, order tt) (iii)ll height(BSTNode ∗T)

.

```
#include <DSA.h>
```

Public Types

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

  order

Public Member Functions

- BinarySearchTree ()
- void insert (ll val)

  It is a member function and having 1 parameter1
  First it traverse through BST to find correct position to insert this new node and then change parent of this node to which we should make node a child and also make child of prev node to new node
  .
- void traverse (BSTNode ∗T, order tt)

  It is a member function and having 2 parameters
  It traverses through Binary search tree according to whether it is pre/IN/POST and then prints all nodes according to it.
- ll height (BSTNode ∗T)

  It is a member function and having 1 parameter
  It traverses through Binary search tree and find height of tree by using recursion
  .

Public Attributes

- BSTNode ∗ **root**

  root Datatype BSTNode∗

3.1.1 Detailed Description

BinarySearchTree It has constructor with no parameters BinarySearchTree ()
It has following member functions (i)void insert(ll val) (ii)void traverse (BSTNode∗ T, order tt) (iii)ll height(BSTNode ∗T)

.

Parameters

root	Datatype BSTNode∗

3.1.2 Constructor & Destructor Documentation

3.1.2.1 BinarySearchTree()

```
BinarySearchTree::BinarySearchTree ( )
```

constructor taking no parameters
Whenever this constructor is called it initialises variable root to NULL

3.1.3 Member Function Documentation

3.1.3.1 height()

```
ll BinarySearchTree::height (
             BSTNode ∗ T )
```

It is a member function and having 1 parameter
It traverses through Binary search tree and find height of tree by using recursion
.

Parameters

in	*T*	of datatype BSTNode∗

Returns

1+max of height of left tree and right tree

3.1.3.2 insert()

```
BinarySearchTree::insert (
            ll val )
```

It is a member function and having 1 parameter1
First it traverse through BST to find correct position to insert this new node and then change parent of this node to which we should make node a child and also make child of prev node to new node
.

Parameters

in	*val*	of datatype ll

3.1.3.3 traverse()

```
BinarySearchTree::traverse (
            BSTNode * T,
            order tt )
```

It is a member function and having 2 parameters
It traverses through Binary search tree according to whether it is pre/IN/POST and then prints all nodes according to it.

Parameters

in	*T*	of datatype BSTNode∗
in	*tt*	of datatype order

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

- DSA.h
- DSA.cpp

3.2 BSTNode Class Reference

BSTNode It contains a constructor BSTNode(ll val)
It has no member functions
.

```
#include <DSA.h>
```

Public Member Functions

- BSTNode (ll val)

  constructor taking 1 parameter
  Whenever this constructor is called it initialises variable info to val and variable level to o and variable left to NULL and variable right to NULL

Public Attributes

- ll **info**

  info Datatype ll
- ll **level**

  level Datatype ll
- BSTNode ∗ **left**

  left Datatype BSTNode∗
- BSTNode ∗ **right**

  right Datatype BSTNode∗

3.2.1 Detailed Description

BSTNode It contains a constructor BSTNode(ll val)
It has no member functions
.

Parameters

info	Datatype ll
level	Datatype ll
left	Datatype BSTNode∗
right	Datatype BSTNode∗

3.2.2 Constructor & Destructor Documentation

3.2.2.1 BSTNode()

```
BSTNode::BSTNode (
            ll val )
```

constructor taking 1 parameter
Whenever this constructor is called it initialises variable info to val and variable level to o and variable left to NULL and variable right to NULL

Parameters

in	*val*	of datatype ll

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

- DSA.h
- DSA.cpp

3.3 DoublyLinkedList Class Reference

DoublyLinkedList
It contains a constructor with no parameters DoublyLinkedListNode()
It has following member functions (i)void insert (ll data) (ii)void printer (string sep = ", ") (iii)void reverse ()
.

```
#include <DSA.h>
```

Public Member Functions

- DoublyLinkedList ()
- void insert (ll data)

  It is a member function and has 1 parameter
  First it instantiates class SinglyListNode(data) and equates to node if head is NULL then it points head to node else points next of tail to node and points prev of node to tail then points tail to node
  .
- void printer (string sep=", ")

  It is a member function and has 1 parameter
  It traverse through list and prints all nodes until tail starting from head
  .
- void **reverse** ()

  It is a member function and has no parameters
  It traverse through list and reverse list by just replacing left and right nodes and moving from ends to center.

Public Attributes

- DoublyLinkedListNode ∗ **head**

  head Datatype DoublyLinkedListNode∗
- DoublyLinkedListNode ∗ **tail**

  tail Datatype DoublyLinkedListNode∗

3.3.1 Detailed Description

DoublyLinkedList
It contains a constructor with no parameters DoublyLinkedListNode()
It has following member functions (i)void insert (ll data) (ii)void printer (string sep = ", ") (iii)void reverse ()
.

Parameters

head	Datatype DoubleLinkedListNode∗
tail	Datatype DoubleLinkedListNode∗

3.3.2 Constructor & Destructor Documentation

3.3.2.1 DoublyLinkedList()

```
DoublyLinkedList::DoublyLinkedList ( )
```

constructor taking no parameters
Whenever this constructor is called it initialises variable head to NULL and variable tail to NULL

3.3.3 Member Function Documentation

3.3.3.1 insert()

```
void DoublyLinkedList::insert (
             ll data )
```

It is a member function and has 1 parameter
First it instantiates class SinglyListNode(data) and equates to node if head is NULL then it points head to node else points next of tail to node and points prev of node to tail then points tail to node
.

Parameters

in	*data*	of datatype ll

3.3.3.2 printer()

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

It is a member function and has 1 parameter
It traverse through list and prints all nodes until tail starting from head
.

Parameters

in	*sep*	of datatype sep

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

- DSA.h
- DSA.cpp

3.4 DoublyLinkedListNode Class Reference

DoublyLinkedListNode
It contains 2 types of constructor (i)DoublyLinkedListNode () (ii)DoublyLinkedListNode (ll val) which creates node with given value by default value is -1 and point next and prev to NULL
It has no member functions
.

```
#include <DSA.h>
```

Public Member Functions

- **DoublyLinkedListNode** ()

  constructor taking no parameters
  Whenever this constructor is called it initialises variable data to -1 and variable next to NULL and variable prev to NULL
- DoublyLinkedListNode (ll val)

  constructor taking 1 parameter
  Whenever this constructor is called it initialises variable data to val and variable next to NULL and variable prev to NULL

Public Attributes

- ll **data**

  data Datatype ll
- DoublyLinkedListNode ∗ **next**

  next Datatype DoublyLinkedListNode∗
- DoublyLinkedListNode ∗ **prev**

  prev Datatype DoublyLinkedListNode∗

3.4.1 Detailed Description

DoublyLinkedListNode
It contains 2 types of constructor (i)DoublyLinkedListNode () (ii)DoublyLinkedListNode (ll val) which creates node with given value by default value is -1 and point next and prev to NULL
It has no member functions
.

Parameters

data	Datatype ll
next	Datatype DoubleLinkedListNode∗
prev	Datatype DoubleLinkedListNode∗

3.4.2 Constructor & Destructor Documentation

3.4.2.1 DoublyLinkedListNode()

```
DoublyLinkedListNode::DoublyLinkedListNode (
            ll val )
```

constructor taking 1 parameter
Whenever this constructor is called it initialises variable data to val and variable next to NULL and variable prev to NULL

Parameters

in	*val*	of datatype ll

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

- DSA.h
- DSA.cpp

3.5 SinglyLinkedList Class Reference

SinglyLinkedList It contains constructor with no parameters SinglyLinkedList ()which when instantiated points head and tail to NULL
It contains member functions (i)void insert(ll data) (ii)SinglyLinkedListNode∗ find (ll data) (iii)bool deleteVal (ll data) (iv)void printer (string sep = ", ")
(v)void reverse ()
.

```
#include <DSA.h>
```

Public Member Functions

- SinglyLinkedList ()
- void insert (ll data)

  It is a member function and has 1 parameter
  First it instantiates class SinglyListNode(data) and equates to node if head is NULL then it points head to node else points next of tail to node then points tail to node
  .
- SinglyLinkedListNode ∗ find (ll data)

  It is a member function and has 1 parameter
  First it creates two variables ptr and prev of datatypes SinglyLinkedListNode∗ and initialises to head and prev then by using while loop it traverse through list to find node if founf then returns
  .
- bool deleteVal (ll data)

  It is a member function and has 1 parameter
  It goes to node which is to be deleted and delete taht node and retirn true if found else return false
  .
- void printer (string sep=", ")

  It is a member function and has 1 parameter
  It traverse through list and prints all nodes until tail starting from head
  .
- void **reverse** ()

  It is a member function and has no parameters
  It traverse through list and reverse list by just replacing left and right nodes and moving from ends to center.

Public Attributes

- SinglyLinkedListNode ∗ **head**

  head Datatype SingleListNode∗
- SinglyLinkedListNode ∗ **tail**

  tail Datatype SingleListNode∗

3.5.1 Detailed Description

SinglyLinkedList It contains constructor with no parameters SinglyLinkedList ()which when instantiated points head and tail to NULL
It contains member functions (i)void insert(ll data) (ii)SinglyLinkedListNode∗ find (ll data) (iii)bool deleteVal (ll data) (iv)void printer (string sep = ", ")
(v)void reverse ()
.

Parameters

head	Datatype SingleListNode∗
tail	Datatype SingleListNode∗

3.5.2 Constructor & Destructor Documentation

3.5.2.1 SinglyLinkedList()

```
SinglyLinkedList::SinglyLinkedList ( )
```

constructor taking no parameters
Whenever this constructor is called it initialises variable head to NULL and variable tail to NULL

3.5.3 Member Function Documentation

3.5.3.1 deleteVal()

```
bool SinglyLinkedList::deleteVal (
             ll data )
```

It is a member function and has 1 parameter
It goes to node which is to be deleted and delete taht node and retirn true if found else return false
.

Parameters

in	*data*	of datatype ll

Returns

which returns true/false according to function

3.5.3.2 find()

```
SinglyLinkedListNode * SinglyLinkedList::find (
            ll data )
```

It is a member function and has 1 parameter
First it creates two variables ptr and prev of datatypes SinglyLinkedListNode∗ and initialises to head and prev then by using while loop it traverse through list to find node if founf then returns
.

Parameters

in	*data*	of datatype ll

Returns

which returns prev to function

3.5.3.3 insert()

```
void SinglyLinkedList::insert (
            ll data )
```

It is a member function and has 1 parameter
First it instantiates class SinglyListNode(data) and equates to node if head is NULL then it points head to node else points next of tail to node then points tail to node
.

Parameters

in	*data*	of datatype ll

3.5.3.4 printer()

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

It is a member function and has 1 parameter
It traverse through list and prints all nodes until tail starting from head
.

Parameters

in	*sep*	of datatype string

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

- DSA.h
- DSA.cpp

3.6 SinglyLinkedListNode Class Reference

SinglyLinkedListNode It contains 2 types of constructor (i)SinglyLinkedListNode () (ii)SinglyLinkedListNode (ll val) which creates node with given value by default value is -1 and point next to NULL
It has no member functions
.

```
#include <DSA.h>
```

Public Member Functions

- **SinglyLinkedListNode** ()

  constructor taking no parameters
  Whenever this constructor is called it initialises variable data to -1 and variable next to NULL
- SinglyLinkedListNode (ll val)

  constructor taking 1 parameter
  Whenever this constructor is called it initialises variable data to val and variable next to NULL

Public Attributes

- ll **data**

  data Datatype ll
- SinglyLinkedListNode ∗ **next**

  next Datatype SinglyLinkedListNode∗

3.6.1 Detailed Description

SinglyLinkedListNode It contains 2 types of constructor (i)SinglyLinkedListNode () (ii)SinglyLinkedListNode (ll val) which creates node with given value by default value is -1 and point next to NULL
It has no member functions
.

Parameters

data	Datatype ll
next	Datatype SinglyLinkedListNode∗

3.6.2 Constructor & Destructor Documentation

3.6.2.1 SinglyLinkedListNode()

```
SinglyLinkedListNode::SinglyLinkedListNode (
             ll val )
```

constructor taking 1 parameter
Whenever this constructor is called it initialises variable data to val and variable next to NULL

Parameters

in	*val*	of datatype ll

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

- DSA.h
- DSA.cpp

3.7 Trie Class Reference

Trie
It has constructor with no parameters Trie ()
It has following member functions (i)bool find(Trie∗ T, char c) (ii)void insert(string s) (iii)bool checkPrefix(string s) (iv)ll countPrefix(string s)
.

```
#include <DSA.h>
```

Public Member Functions

- **Trie** ()

  constructor taking no parameters
  Whenever this constructor is called it initialises variable count to 0 and variable nodes to map<char,Trie∗>();
- bool find (Trie ∗T, char c)

  It is a member function and having 2 parameter
  It returns true if c is present in Trie else return false
  .

- void insert (string s)

  It is a member function and having 1 parameter
  If c is not present in Trie then it inserts a new c into Trie
  .
- bool checkPrefix (string s)

  It is a member function and having 1 parameter
  It checks whether string s is prefix for any word or not
  .
- ll countPrefix (string s)

  It is a member function and having 1 parameter
  It counts for how many words string s is prefix
  .

Public Attributes

- ll **count**

  count Datatype ll
- map< char, Trie ∗ > **nodes**

  nodes of Datatype map<char,Trie∗>

3.7.1 Detailed Description

Trie
It has constructor with no parameters Trie ()
It has following member functions (i)bool find(Trie∗ T, char c) (ii)void insert(string s) (iii)bool checkPrefix(string s) (iv)ll countPrefix(string s)
.

Parameters

count	Datatype ll

3.7.2 Member Function Documentation

3.7.2.1 checkPrefix()

```
bool Trie::checkPrefix (
            string s )
```

It is a member function and having 1 parameter
It checks whether string s is prefix for any word or not
.

Parameters

in	*s*	of datatype string

3.7.2.2 countPrefix()

```
ll Trie::countPrefix (
             string s )
```

It is a member function and having 1 parameter
It counts for how many words string s is prefix
.

Parameters

in	*s*	of datatype string

Returns

which returns count of prefix

3.7.2.3 find()

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

It is a member function and having 2 parameter
It returns true if c is present in Trie else return false
.

Parameters

in	*T*	of datatype Trie∗
in	*c*	of datatype char

Returns

which returns true/false

3.7.2.4 insert()

```
void Trie::insert (
             string s )
```

It is a member function and having 1 parameter
If c is not present in Trie then it inserts a new c into Trie
.

Parameters

`in`	*s*	of datatype string

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

- DSA.h
- DSA.cpp

Chapter 4

File Documentation

4.1 DSA.cpp File Reference

This file contains classes SingleLinkedListNode,SingleLinkedList,DoublyClassLinkedListNode,DoublyLinkedList,BSTNode,BinarySea
which contains all required definitions and basic utilities functions of datastructures like Singly Linked List,Doubly Linked List,Binary Search Tree,Suffix Trie.

```
#include "DSA.h"
```

Functions

- ostream & operator<< (ostream &out, const SinglyLinkedListNode &node)
- SinglyLinkedList merge (SinglyLinkedList list1, SinglyLinkedList list2)

  It is a function and has 2 parameters
  It merges two Singlylinkedlists and returns a combined lists
  .
- ostream & operator<< (ostream &out, const DoublyLinkedListNode &node)
- ostream & operator<< (ostream &out, const BSTNode &node)

4.1.1 Detailed Description

This file contains classes SingleLinkedListNode,SingleLinkedList,DoublyClassLinkedListNode,DoublyLinkedList,BSTNode,BinarySea
which contains all required definitions and basic utilities functions of datastructures like Singly Linked List,Doubly Linked List,Binary Search Tree,Suffix Trie.

Author

Narkedamilli Harika

Date

21/09/2022

4.1.2 Function Documentation

4.1.2.1 merge()

```
SinglyLinkedList merge (
            SinglyLinkedList list1,
            SinglyLinkedList list2 )
```

It is a function and has 2 parameters
It merges two Singlylinkedlists and returns a combined lists
.

Parameters

in	*list1*	of datatype SinglyLinkedList
in	*list2*	of datatype SinglyLinkedList

Returns

which returns a merged SinglyLinkedList

4.1.2.2 operator<<() [1/3]

```
ostream & operator<< (
            ostream & out,
            const BSTNode & node )
```

defines the operator <<,the function takes two parameters

Parameters

in	*out*	
in	*node*	

Returns

ostream&

4.1.2.3 operator<<() [2/3]

```
ostream & operator<< (
            ostream & out,
            const DoublyLinkedListNode & node )
```

defines the operator <<,the function takes two parameters

Parameters

in	*out*	
in	*node*	

Returns

ostream&

4.1.2.4 operator<<() [3/3]

```
ostream & operator<< (
              ostream & out,
              const SinglyLinkedListNode & node )
```

defines the operator <<,the function takes two parameters

Parameters

in	*out*	
in	*node*	

Returns

ostream&

4.2 DSA.h

```

#include <bits/stdc++.h>
#define ll long long int
#define vi vector<int>
#define vll vector<ll>
using namespace std;

/* ------------------------------- Data Structures ---------------------------------- */

// ------------------------------- Singly Linked List ----------------------------


class SinglyLinkedListNode
{
    public:
        ll data;
        SinglyLinkedListNode* next;
        SinglyLinkedListNode () ;
        SinglyLinkedListNode (ll val) ;
};
ostream& operator«(ostream &out, const SinglyLinkedListNode &node);
class SinglyLinkedList {

    public:
        SinglyLinkedListNode *head;
        SinglyLinkedListNode *tail;

        SinglyLinkedList () ;

        void insert (ll data) ;

         SinglyLinkedListNode* find (ll data) ;

        bool deleteVal (ll data) ;

        void printer (string sep = ", ");

        void reverse () ;

};
SinglyLinkedList merge (SinglyLinkedList list1, SinglyLinkedList list2) ;

// ------------------------------- Doubly Linked List -----------------------------
class DoublyLinkedListNode {
    public:
        ll data;
        DoublyLinkedListNode *next;
        DoublyLinkedListNode *prev;

        DoublyLinkedListNode () ;

        DoublyLinkedListNode (ll val) ;

};

ostream& operator«(ostream &out, const DoublyLinkedListNode &node);

class DoublyLinkedList {
    public:
        DoublyLinkedListNode *head;
        DoublyLinkedListNode *tail;
        DoublyLinkedList () ;

        void insert (ll data) ;

        void printer (string sep = ", ") ;

        void reverse () ;

};

// ------------------------------- Binary Search Tree -----------------------------
class BSTNode {
    public:
        ll info;
        ll level;
        BSTNode *left;
        BSTNode *right;

        BSTNode (ll val);
};
ostream& operator«(ostream &out, const BSTNode &node) ;
class BinarySearchTree {
    public:
        BSTNode *root;
        enum order {PRE, IN, POST};
        BinarySearchTree () ;

        void insert(ll val) ;

        void traverse (BSTNode* T, order tt) ;

        ll height(BSTNode *T) ;
};

// ------------------------------- Suffix Trie -----------------------------
class Trie {
    public:
        ll count;
        map<char,Trie*> nodes;

        Trie () ;

        bool find(Trie* T, char c) ;

        void insert(string s) ;

        bool checkPrefix(string s) ;

        ll countPrefix(string s) ;

};
```

Index