

My Project

Generated by Doxygen 1.8.17

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 BinarySearchTree Class Reference	5
3.1.1 Detailed Description	6
3.1.2 Member Function Documentation	6
3.1.2.1 height()	6
3.1.2.2 insert()	6
3.1.2.3 traverse()	6
3.2 BSTNode Class Reference	7
3.2.1 Detailed Description	7
3.2.2 Constructor & Destructor Documentation	8
3.2.2.1 BSTNode()	8
3.3 DoublyLinkedList Class Reference	8
3.3.1 Detailed Description	9
3.3.2 Member Function Documentation	9
3.3.2.1 insert()	9
3.3.2.2 printer()	9
3.4 DoublyLinkedListNode Class Reference	9
3.4.1 Detailed Description	10
3.4.2 Constructor & Destructor Documentation	10
3.4.2.1 DoublyLinkedListNode()	10
3.5 Heap Class Reference	11
3.5.1 Constructor & Destructor Documentation	11
3.5.1.1 Heap()	11
3.5.2 Member Function Documentation	11
3.5.2.1 Heapify()	12
3.5.2.2 insert()	12
3.5.2.3 left()	12
3.5.2.4 min()	12
3.5.2.5 parent()	13
3.5.2.6 right()	13
3.6 SinglyLinkedList Class Reference	13
3.6.1 Detailed Description	14
3.6.2 Member Function Documentation	14
3.6.2.1 deleteVal()	15
3.6.2.2 find()	16
3.6.2.3 insert()	16
3.6.2.4 printer()	16

3.7 SinglyLinkedListNode Class Reference	17
3.7.1 Detailed Description	17
3.7.2 Constructor & Destructor Documentation	17
3.7.2.1 SinglyLinkedListNode()	17
3.8 Trie Class Reference	18
3.8.1 Detailed Description	18
3.8.2 Member Function Documentation	18
3.8.2.1 checkPrefix()	18
3.8.2.2 countPrefix()	19
3.8.2.3 find()	19
3.8.2.4 insert()	20
4 File Documentation	21
4.1 DSA.cpp File Reference	21
4.1.1 Detailed Description	22
4.1.2 Function Documentation	22
4.1.2.1 merge()	22
4.1.2.2 operator<<() [1/3]	23
4.1.2.3 operator<<() [2/3]	23
4.1.2.4 operator<<() [3/3]	24
4.1.2.5 swap()	24
Index	25

Chapter 1

Class Index

1.1 Class List

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

BinarySearchTree	
BinarySearchTree class. Available member functions include default constructor, insert, traverse and height	5
BSTNode	
BST Node class. Available member functions include initialising constructor	7
DoublyLinkedList	
Doubly Linked List class. Available member functions include default constructor, insert, printer and reverse	8
DoublyLinkedListNode	
Doubly Linked List Node class. Available member functions include default constructor and initialising constructor	9
Heap	11
SinglyLinkedList	
Singly Linked List class. Available member functions include default constructor, insert, find, deleteVal, printer and reverse	13
SinglyLinkedListNode	
Singly Linked List Node class. Available member functions include default constructor and initialising constructor	17
Trie	
Trie class. Available member functions include default constructor, find, insert, checkPrefix and countPrefix	18

Chapter 2

File Index

2.1 File List

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

DSA.cpp	C++ implementations of data structures and their utility functions	21
-------------------------	--	----

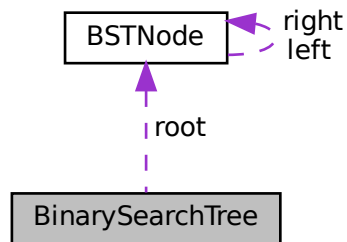
Chapter 3

Class Documentation

3.1 BinarySearchTree Class Reference

[BinarySearchTree](#) class. Available member functions include default constructor, insert, traverse and height.

Collaboration diagram for BinarySearchTree:



Public Types

- enum [order](#) { **PRE**, **IN**, **POST** }
types of traversals

Public Member Functions

- [BinarySearchTree](#) ()
Construct a new Binary Search Tree object.
- void [insert](#) (ll val)
Insert an element into the tree.
- void [traverse](#) (BSTNode *T, [order](#) tt)
Traverse the tree.
- ll [height](#) (BSTNode *T)
Calculate height of the tree.

Public Attributes

- [BSTNode * root](#)
pointer to the root

3.1.1 Detailed Description

[BinarySearchTree](#) class. Available member functions include default constructor, insert, traverse and height.

3.1.2 Member Function Documentation

3.1.2.1 height()

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

Calculate height of the tree.

Parameters

in	<i>T</i>	tree root pointer
----	----------	-------------------

Returns

||

3.1.2.2 insert()

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

Insert an element into the tree.

Parameters

in	<i>val</i>	value to be inserted
----	------------	----------------------

3.1.2.3 traverse()

```
void BinarySearchTree::traverse (
```

```

BSTNode * T,
order tt ) [inline]

```

Traverse the tree.

Parameters

in	T	tree root pointer
	tt	type of traversal

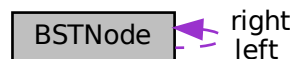
The documentation for this class was generated from the following file:

- [DSA.cpp](#)

3.2 BSTNode Class Reference

BST Node class. Available member functions include initialising constructor.

Collaboration diagram for BSTNode:



Public Member Functions

- [BSTNode \(ll val\)](#)
Construct a new [BSTNode](#) object.

Public Attributes

- [ll info](#)
info stored in the node
- [ll level](#)
level of the node
- [BSTNode * left](#)
pointer to the left child
- [BSTNode * right](#)
pointer to the right child

3.2.1 Detailed Description

BST Node class. Available member functions include initialising constructor.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 BSTNode()

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

Construct a new [BSTNode](#) object.

Parameters

in	val	input value of the node
----	-----	-------------------------

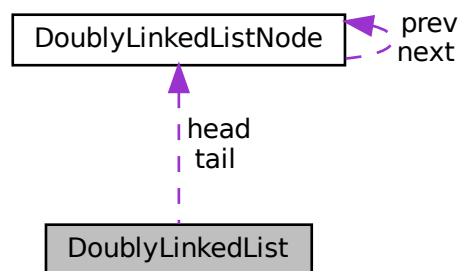
The documentation for this class was generated from the following file:

- [DSA.cpp](#)

3.3 DoublyLinkedList Class Reference

Doubly Linked List class. Available member functions include default constructor, insert, printer and reverse.

Collaboration diagram for DoublyLinkedList:



Public Member Functions

- [DoublyLinkedList](#) ()
Construct a new Doubly Linked List object.
- void [insert](#) (ll data)
Insert an element into the list.
- void [printer](#) (string sep=" ", "
Print the list.
- void [reverse](#) ()
Reverse the list.

Public Attributes

- [DoublyLinkedListNode * head](#)
pointer to the head
- [DoublyLinkedListNode * tail](#)
pointer to the tail

3.3.1 Detailed Description

Doubly Linked List class. Available member functions include default constructor, insert, printer and reverse.

3.3.2 Member Function Documentation

3.3.2.1 insert()

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

Insert an element into the list.

Parameters

in	<i>data</i>	value to be inserted
----	-------------	----------------------

3.3.2.2 printer()

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

Print the list.

Parameters

in	<i>sep</i>	separator string
----	------------	------------------

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

- [DSA.cpp](#)

3.4 DoublyLinkedListNode Class Reference

Doubly Linked List Node class. Available member functions include default constructor and initialising constructor.

Collaboration diagram for DoublyLinkedListNode:



Public Member Functions

- [DoublyLinkedListNode \(\)](#)
Construct a new Doubly Linked List Node object.
- [DoublyLinkedListNode \(ll val\)](#)
Construct a new Doubly Linked List Node object.

Public Attributes

- [ll data](#)
data stored in the node
- [DoublyLinkedListNode * next](#)
pointer to the next node
- [DoublyLinkedListNode * prev](#)
pointer to the previous node

3.4.1 Detailed Description

Doubly Linked List Node class. Available member functions include default constructor and initialising constructor.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 DoublyLinkedListNode()

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

Construct a new Doubly Linked List Node object.

Parameters

in	val	input value for the node
----	-----	--------------------------

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

- [DSA.cpp](#)

3.5 Heap Class Reference

Public Member Functions

- [Heap](#) (int capacity)
Construct a new [Heap](#) object.
- int [parent](#) (int i)
Parent function.
- int [left](#) (int i)
Left child function.
- int [right](#) (int i)
Right child function.
- void [insert](#) (int e)
Insert function.
- int [min](#) ()
Minimum of heap.
- void [Heapify](#) (int root)
Heapify function.
- void [deleteMin](#) ()
Delete minimum.

3.5.1 Constructor & Destructor Documentation

3.5.1.1 Heap()

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

Construct a new [Heap](#) object.

Parameters

in	<i>capacity</i>	max capacity of heap
----	-----------------	----------------------

3.5.2 Member Function Documentation

3.5.2.1 Heapify()

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

Heapify function.

Parameters

in	<i>root</i>	root node
----	-------------	-----------

3.5.2.2 insert()

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

Insert function.

Parameters

in	<i>e</i>	value to insert
----	----------	-----------------

3.5.2.3 left()

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

Left child function.

Parameters

in	<i>i</i>	index of node
----	----------	---------------

Returns

int

3.5.2.4 min()

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

Minimum of heap.

Returns

int

3.5.2.5 parent()

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

Parent function.

Parameters

in	<i>i</i>	index of node
----	----------	---------------

Returns

int

3.5.2.6 right()

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

Right child function.

Parameters

in	<i>i</i>	index of node
----	----------	---------------

Returns

int

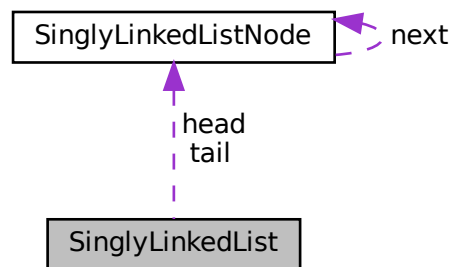
The documentation for this class was generated from the following file:

- [DSA.cpp](#)

3.6 SinglyLinkedList Class Reference

Singly Linked List class. Available member functions include default constructor, insert, find, deleteVal, printer and reverse.

Collaboration diagram for SinglyLinkedList:



Public Member Functions

- [SinglyLinkedList](#) ()
Construct a new Singly Linked List object.
- void [insert](#) (ll data)
Insert an element into the list.
- [SinglyLinkedListNode](#) * [find](#) (ll data)
Find an element in the list.
- bool [deleteVal](#) (ll data)
Delete an element from list.
- void [printer](#) (string sep=", ")
Print the list.
- void [reverse](#) ()
Reverse the list.

Public Attributes

- [SinglyLinkedListNode](#) * [head](#)
pointer to the head
- [SinglyLinkedListNode](#) * [tail](#)
pointer to the tail

3.6.1 Detailed Description

Singly Linked List class. Available member functions include default constructor, insert, find, deleteVal, printer and reverse.

3.6.2 Member Function Documentation

3.6.2.1 deleteVal()

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

Delete an element from list.

Parameters

in	<i>data</i>	value to be deleted
----	-------------	---------------------

Returns

true/false

3.6.2.2 find()

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

Find an element in the list.

Parameters

in	<i>data</i>	value to be found
----	-------------	-------------------

Returns

SinglyLinkedListNode*

3.6.2.3 insert()

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

Insert an element into the list.

Parameters

in	<i>data</i>	value to be inserted
----	-------------	----------------------

3.6.2.4 printer()

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

Print the list.

Parameters

<code>in</code>	<code>sep</code>	separator string
-----------------	------------------	------------------

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

- [DSA.cpp](#)

3.7 SinglyLinkedListNode Class Reference

Singly Linked List Node class. Available member functions include default constructor and initialising constructor.

Collaboration diagram for SinglyLinkedListNode:



Public Member Functions

- [SinglyLinkedListNode](#) ()
Construct a new Singly Linked List Node object.
- [SinglyLinkedListNode](#) (ll val)
Construct a new Singly Linked List Node object.

Public Attributes

- [ll data](#)
data stored in the node
- [SinglyLinkedListNode](#) * [next](#)
pointer to the next node

3.7.1 Detailed Description

Singly Linked List Node class. Available member functions include default constructor and initialising constructor.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 SinglyLinkedListNode()

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

Construct a new Singly Linked List Node object.

Parameters

in	val	input value for the node
----	-----	--------------------------

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

- [DSA.cpp](#)

3.8 Trie Class Reference

[Trie](#) class. Available member functions include default constructor, find, insert, checkPrefix and countPrefix.

Public Member Functions

- [Trie](#) ()
Construct a new [Trie](#) object.
- bool [find](#) ([Trie](#) *T, char c)
Find a char in a trie.
- void [insert](#) (string s)
Insert string into a trie.
- bool [checkPrefix](#) (string s)
Check if a string is a prefix.
- [ll](#) [countPrefix](#) (string s)
Count how many strings have the given prefix.

Public Attributes

- [ll](#) [count](#)
count of the string from root to leaf
- map< char, [Trie](#) * > [nodes](#)
map/dictionary of nodes

3.8.1 Detailed Description

[Trie](#) class. Available member functions include default constructor, find, insert, checkPrefix and countPrefix.

3.8.2 Member Function Documentation

3.8.2.1 checkPrefix()

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

Check if a string is a prefix.

Parameters

in	<i>s</i>	string to check
----	----------	-----------------

Returns

true/false

3.8.2.2 countPrefix()

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

Count how many strings have the given prefix.

Parameters

in	<i>s</i>	prefix to check
----	----------	-----------------

Returns

||

3.8.2.3 find()

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

Find a char in a trie.

Parameters

in	<i>T</i>	trie root pointer
in	<i>c</i>	char to find

Returns

true/false

3.8.2.4 insert()

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

Insert string into a trie.

Parameters

in	s	string to insert
----	---	------------------

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

- [DSA.cpp](#)

Chapter 4

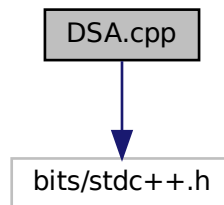
File Documentation

4.1 DSA.cpp File Reference

C++ implementations of data structures and their utility functions.

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

Include dependency graph for DSA.cpp:



Classes

- class [SinglyLinkedListNode](#)
Singly Linked List Node class. Available member functions include default constructor and initialising constructor.
- class [SinglyLinkedList](#)
Singly Linked List class. Available member functions include default constructor, insert, find, deleteVal, printer and reverse.
- class [DoublyLinkedListNode](#)
Doubly Linked List Node class. Available member functions include default constructor and initialising constructor.
- class [DoublyLinkedList](#)
Doubly Linked List class. Available member functions include default constructor, insert, printer and reverse.
- class [BSTNode](#)
BST Node class. Available member functions include initialising constructor.
- class [BinarySearchTree](#)
BinarySearchTree class. Available member functions include default constructor, insert, traverse and height.
- class [Trie](#)
Trie class. Available member functions include default constructor, find, insert, checkPrefix and countPrefix.
- class [Heap](#)

Macros

- `#define ll long long int`
long long int macro
- `#define vi vector<int>`
int vector macro
- `#define vll vector<ll>`
long long vector macro

Functions

- `ostream & operator<<` (`ostream &out`, `const SinglyLinkedListNode &node`)
Function to print a Singly Linked List Node.
- `SinglyLinkedList merge` (`SinglyLinkedList list1`, `SinglyLinkedList list2`)
Merge two sorted linked lists.
- `ostream & operator<<` (`ostream &out`, `const DoublyLinkedListNode &node`)
Function to print a Doubly Linked List Node.
- `ostream & operator<<` (`ostream &out`, `const BSTNode &node`)
Function to print a BST Node.
- `void swap` (`int &x`, `int &y`)
Swap function.

4.1.1 Detailed Description

C++ implementations of data structures and their utility functions.

Author

Sarthak Mittal (200050129@iitb.ac.in)

Version

1

Date

2022-09-17

4.1.2 Function Documentation

4.1.2.1 merge()

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

Merge two sorted linked lists.

Parameters

in	<i>list1</i>	first list
in	<i>list2</i>	second list

Returns

[SinglyLinkedList](#)

4.1.2.2 operator<<() [1/3]

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

Function to print a BST Node.

Parameters

in	<i>out</i>	standard output stream
in	<i>node</i>	the node to be printed

Returns

ostream&

4.1.2.3 operator<<() [2/3]

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

Function to print a Doubly Linked List Node.

Parameters

in	<i>out</i>	standard output stream
in	<i>node</i>	the node to be printed

Returns

ostream&

4.1.2.4 operator<<() [3/3]

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

Function to print a Singly Linked List Node.

Parameters

in	<i>out</i>	standard output stream
in	<i>node</i>	the node to be printed

Returns

ostream&

4.1.2.5 swap()

```
void swap (
    int & x,
    int & y )
```

Swap function.

Parameters

in	<i>x</i>	first value
in	<i>y</i>	second value

Index

- BinarySearchTree, [5](#)
 - height, [6](#)
 - insert, [6](#)
 - traverse, [6](#)
- BSTNode, [7](#)
 - BSTNode, [8](#)
- checkPrefix
 - Trie, [18](#)
- countPrefix
 - Trie, [19](#)
- deleteVal
 - SinglyLinkedList, [14](#)
- DoublyLinkedList, [8](#)
 - insert, [9](#)
 - printer, [9](#)
- DoublyLinkedListNode, [9](#)
 - DoublyLinkedListNode, [10](#)
- DSA.cpp, [21](#)
 - merge, [22](#)
 - operator<<, [23](#), [24](#)
 - swap, [24](#)
- find
 - SinglyLinkedList, [16](#)
 - Trie, [19](#)
- Heap, [11](#)
 - Heap, [11](#)
 - Heapify, [11](#)
 - insert, [12](#)
 - left, [12](#)
 - min, [12](#)
 - parent, [13](#)
 - right, [13](#)
- Heapify
 - Heap, [11](#)
- height
 - BinarySearchTree, [6](#)
- insert
 - BinarySearchTree, [6](#)
 - DoublyLinkedList, [9](#)
 - Heap, [12](#)
 - SinglyLinkedList, [16](#)
 - Trie, [19](#)
- left
 - Heap, [12](#)
- merge
 - DSA.cpp, [22](#)
- min
 - Heap, [12](#)
- operator<<
 - DSA.cpp, [23](#), [24](#)
- parent
 - Heap, [13](#)
- printer
 - DoublyLinkedList, [9](#)
 - SinglyLinkedList, [16](#)
- right
 - Heap, [13](#)
- SinglyLinkedList, [13](#)
 - deleteVal, [14](#)
 - find, [16](#)
 - insert, [16](#)
 - printer, [16](#)
- SinglyLinkedListNode, [17](#)
 - SinglyLinkedListNode, [17](#)
- swap
 - DSA.cpp, [24](#)
- traverse
 - BinarySearchTree, [6](#)
- Trie, [18](#)
 - checkPrefix, [18](#)
 - countPrefix, [19](#)
 - find, [19](#)
 - insert, [19](#)