Message Passing

Generated by Doxygen 1.9.8

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 files with brief descriptions:

Chapter 3

Class Documentation

3.1 BinaryBalanced Class Reference

`#include <BinaryBalanced.h>`

Public Member Functions

- BinaryBalanced ()
- ~BinaryBalanced ()
- void DeleteGraph (Node *node)
- void LoadGraph (const std::string &filename)
- void Insert (short inp)
- void GetPathFrom (short id_destination)
- bool CheckNearbyStatuses (Node *node)
- short SendMessage (short start_id, bool print)
- void ResetStatuses ()
- void ShortestPath ()

3.1.1 Constructor & Destructor Documentation

3.1.1.1 BinaryBalanced()

```
BinaryBalanced::BinaryBalanced ( )
```

3.1.1.2 ~BinaryBalanced()

```
BinaryBalanced::~BinaryBalanced ( )
```

3.1.2 Member Function Documentation

3.1.2.1 CheckNearbyStatuses()

```
bool BinaryBalanced::CheckNearbyStatuses (
            Node * node )
```

Checks if the given node has some neighbors that has not been checked yet

Parameters

node	Current node that should be checked

Return values

TRUE	unchecked neighbors do exist
FALSE	unchecked neighbors do not exist

3.1.2.2 DeleteGraph()

```
void BinaryBalanced::DeleteGraph (
            Node * node )
```

3.1.2.3 GetPathFrom()

```
void BinaryBalanced::GetPathFrom (
            short id_destination )
```

Fills the path vector with the path from the root to the node

Parameters

id_destination	ID of the destination node

3.1.2.4 Insert()

```
void BinaryBalanced::Insert (
            short inp )
```

Inserts a node into the Graph (in a BALANCED way)

Parameters

inp	input number of the new node

3.1.2.5 LoadGraph()

```
void BinaryBalanced::LoadGraph (
            const std::string & filename )
```

Loads graph form input file

Parameters

filename	name of the input file

3.1.2.6 ResetStatuses()

```
void BinaryBalanced::ResetStatuses ( )
```

Resets statuses of each node in the graph

3.1.2.7 SendMessage()

```
short BinaryBalanced::SendMessage (
            short start_id,
            bool print )
```

Sends a message though the graph starting from the given node

Parameters

print	Prints each iteration (step) of the shortest path
start_number	ID from which the path is starting

Returns

Time period for the message to travel trough the graph

3.1.2.8 ShortestPath()

```
void BinaryBalanced::ShortestPath ( )
```

Calculates the shortest time period for the message to travel over the graph

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

- src/BinaryBalanced.h
- src/BinaryBalanced.cpp

3.2 BinaryUnbalanced Class Reference

```
#include <BinaryUnbalanced.h>
```

Public Member Functions

- BinaryUnbalanced ()
- ~BinaryUnbalanced ()
- void DeleteGraph (Node *node)
- void LoadGraph (const std::string &filename)
- void Insert (short inp)
- bool CheckNearbyStatuses (Node *node)
- Node * SearchNode (short inp)
- short SendMessage (short inp, bool print)
- void ResetStatuses ()
- void ShortestPath ()

3.2.1 Constructor & Destructor Documentation

3.2.1.1 BinaryUnbalanced()

```
BinaryUnbalanced::BinaryUnbalanced ( )
```

3.2.1.2 ~BinaryUnbalanced()

```
BinaryUnbalanced::~BinaryUnbalanced ( )
```

3.2.2 Member Function Documentation

3.2.2.1 CheckNearbyStatuses()

```
bool BinaryUnbalanced::CheckNearbyStatuses (
            Node * node )
```

Checks if the given node has some neighbors that has not been checked yet

Parameters

node	Current node that should be checked

Return values

TRUE	unchecked neighbors do exist
FALSE	unchecked neighbors do not exist

3.2.2.2 DeleteGraph()

```
void BinaryUnbalanced::DeleteGraph (
            Node * node )
```

3.2.2.3 Insert()

```
void BinaryUnbalanced::Insert (
            short inp )
```

Inserts a node into the Graph (UNBALANCED WAY = Binary Search Tree)

Parameters

inp	input number of the new node

3.2.2.4 LoadGraph()

```
void BinaryUnbalanced::LoadGraph (
            const std::string & filename )
```

Loads graph form input file

Parameters

filename	name of the input file

3.2.2.5 ResetStatuses()

```
void BinaryUnbalanced::ResetStatuses ( )
```

Resets statuses of each node in the graph

3.2.2.6 SearchNode()

```
Node * BinaryUnbalanced::SearchNode (
            short inp )
```

Searches for a node with the given value

Parameters

inp	Seeked value

Returns

Pointer to searched node

3.2.2.7 SendMessage()

```
short BinaryUnbalanced::SendMessage (
            short inp,
            bool print )
```

Sends a message though the graph starting from the given node

Parameters

print	Prints each iteration (step) of the shortest path
start_number	ID from which the path is starting

Returns

Time period for the message to travel trough the graph

3.2.2.8 ShortestPath()

```
void BinaryUnbalanced::ShortestPath ( )
```

Calculates the shortest time period for the message to travel over the graph

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

- src/BinaryUnbalanced.h
- src/BinaryUnbalanced.cpp

3.3 Node Struct Reference

```
#include <Node.h>
```

Public Member Functions

- Node (short value, short id)
- Node (short value, short id, Node *parent)

Public Attributes

- short value
- short ID
- bool status
- Node * parent
- Node * left
- Node * right

3.3.1 Constructor & Destructor Documentation

3.3.1.1 Node() [1/2]

```
Node::Node (
            short value,
            short id )  [inline]
```

3.3.1.2 Node() [2/2]

```
Node::Node (
            short value,
            short id,
            Node * parent )  [inline]
```

3.3.2 Member Data Documentation

3.3.2.1 ID

```
short Node::ID
```

3.3.2.2 left

```
Node* Node::left
```

3.3.2.3 parent

```
Node* Node::parent
```

3.3.2.4 right

```
Node* Node::right
```

3.3.2.5 status

```
bool Node::status
```

3.3.2.6 value

```
short Node::value
```

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

- src/Node.h

Chapter 4

File Documentation

4.1 src/BinaryBalanced.cpp File Reference

```
#include "BinaryBalanced.h"
```

4.2 src/BinaryBalanced.h File Reference

```
#include "Node.h"
#include <stack>
#include <queue>
#include <vector>
#include <fstream>
```

Classes

- class BinaryBalanced

4.3 BinaryBalanced.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "Node.h"
00003 #include <stack>
00004 #include <queue>
00005 #include <vector>
00006 #include <fstream>
00007
00008 class BinaryBalanced{
00009     private:
00010         short size;
00011         Node* Graph;
00012         std::stack<bool> path;
00013
00014     public:
00015         BinaryBalanced();
00016         ~BinaryBalanced();
00017         void DeleteGraph(Node* node);
00018
```

```
00023         void LoadGraph(const std::string& filename);
00024
00029         void Insert(short inp);
00030
00035         void GetPathFrom(short id_destination);
00036
00044         bool CheckNearbyStatuses(Node* node);
00045
00052         short SendMessage(short start_id, bool print);
00053
00057         void ResetStatuses();
00058
00062         void ShortestPath();
00063 };
```

4.4 src/BinaryUnbalanced.cpp File Reference

```
#include "BinaryUnbalanced.h"
```

4.5 src/BinaryUnbalanced.h File Reference

```
#include "Node.h"
#include <stack>
#include <queue>
#include <vector>
#include <fstream>
```

Classes

- class BinaryUnbalanced

4.6 BinaryUnbalanced.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "Node.h"
00003 #include <stack>
00004 #include <queue>
00005 #include <vector>
00006 #include <fstream>
00007
00008 class BinaryUnbalanced{
00009 private:
00010     short size;
00011     Node* Graph;
00012     std::stack<bool> path;
00013
00014 public:
00015     BinaryUnbalanced();
00016     ~BinaryUnbalanced();
00017     void DeleteGraph(Node* node);
00018
00023     void LoadGraph(const std::string& filename);
00024
00029     void Insert(short inp);
00030
00038     bool CheckNearbyStatuses(Node* node);
00039
00045     Node* SearchNode(short inp);
00046
00053     short SendMessage(short inp, bool print);
00054
00058     void ResetStatuses();
00059
00063     void ShortestPath();
00064 };
```

4.7 src/Node.h File Reference

`#include <iostream>`

Classes

- struct Node

Enumerations

- enum {
  UNCHECKED = false , CHECKED = true , LEFT = true , RIGHT = false ,
  UNASSIGNED = -1 }

4.7.1 Enumeration Type Documentation

4.7.1.1 anonymous enum

`anonymous enum`

Enumerator

UNCHECKED	
CHECKED	
LEFT	
RIGHT	
UNASSIGNED	

4.8 Node.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include <iostream>
00003
00004 enum {
00005     UNCHECKED = false,
00006     CHECKED = true,
00007
00008     LEFT = true,
00009     RIGHT = false,
00010
00011     UNASSIGNED = -1
00012 };
00013
00014 struct Node {
00015     short value;
00016     short ID;
00017     bool status;
00018     Node *parent;
00019     Node *left;
00020     Node *right;
00021
00022     Node(short value, short id) {
00023         this->value = value;
```

```
00024         this->ID = id;
00025         this->status = UNCHECKED;
00026         this->parent = nullptr;
00027         this->left = nullptr;
00028         this->right = nullptr;
00029     }
00030     Node(short value, short id, Node *parent) {
00031         this->value = value;
00032         this->ID = id;
00033         this->status = UNCHECKED;
00034         this->parent = parent;
00035         this->left = nullptr;
00036         this->right = nullptr;
00037     }
00038 };
```

Index