My Project

Generated by Doxygen 1.9.8

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

Chapter 2

File Documentation

2.1 project/project.cpp File Reference

```
#include "source.h"
```

Functions

- int main (int argc, const char ∗argv[ ])

2.1.1 Function Documentation

2.1.1.1 main()

```
int main (
            int argc,
            const char * argv[] )
```

Main function, starting point for program execution

Parameters

argc	number of parameters when starting the program
argv	an array of pointers to char arrays used when running the program

Returns

if the program executed successfully, 0 is returned

2.2 project/source.cpp File Reference

```
#include "source.h"
```

Functions

- bool read_params (int count, const char *params[], std::string &input_file, std::string &output_file)
- NodeMap get_data_and_create_map (const std::string &filename)
- void DFS_on_every_node (NodeMap &MY_MAP, DiscoveredCycles &DISCOVERED_CYCLES)
- void DFS (const int start_node, const int cur_node, std::vector< int > visited, NodeMap &MY_MAP, DiscoveredCycles &DISCOVERED_CYCLES)
- void save_data (const std::string outputFileName, const DiscoveredCycles &DISCOVERED_CYCLES)
- void print_cycles (const DiscoveredCycles &DISCOVERED_CYCLES)

2.2.1 Function Documentation

2.2.1.1 DFS()

```
void DFS (
            const int start_node,
            const int cur_node,
            std::vector< int > visited,
            NodeMap & MY_MAP,
            DiscoveredCycles & DISCOVERED_CYCLES )
```

This function performs a depth-first search (DFS) to find cycles in a graph starting from a specified node. It populates the discovered cycles in the DISCOVERED_CYCLES vector.

Parameters

start_node	The node from which the search for cycles begins.
cur_node	The current node being explored during the search.
visited	A vector representing the nodes visited during the current exploration.
MY_MAP	The graph represented as a NodeMap (adjacency list).
DISCOVERED_CYCLES	Reference to a vector of vectors where discovered cycles are stored.

2.2.1.2 DFS_on_every_node()

```
void DFS_on_every_node (
            NodeMap & MY_MAP,
            DiscoveredCycles & DISCOVERED_CYCLES )
```

This function performs a depth-first search (DFS) on every node in the graph and stores the discovered cycles in the DISCOVERED_CYCLES vector.

Parameters

MY_MAP	The graph represented as a NodeMap (adjacency list).
DISCOVERED_CYCLES	Reference to a vector of vectors where discovered cycles are stored.

2.2.1.3 get_data_and_create_map()

```
NodeMap get_data_and_create_map (
```

```
            const std::string & filenamet )
```

This function reads data from the specified file and creates a NodeMap, where each node is associated with a list of connected nodes. The file is expected to contain lines representing edges in the format "from_node - to_node".

Parameters

filename	The name of the file containing edge information.

Returns

Returns a NodeMap representing the connectivity information read from the file. The NodeMap is a std↩ ::unordered_map<int, std::vector<int>>, where each key-value pair represents a node and its connected nodes.

2.2.1.4 print_cycles()

```
void print_cycles (
            const DiscoveredCycles & DISCOVERED_CYCLES )
```

This function prints the discovered cycles in the graph.

Parameters

DISCOVERED_CYCLES	Reference to a vector of vectors containing the discovered cycles.

2.2.1.5 read_params()

```
bool read_params (
            int count,
            const char * params[],
            std::string & input_file,
            std::string & output_file )
```

This function parses command-line parameters to extract the input and output file paths. It expects the parameters to be in the form of "-g input_file -c output_file".

Parameters

count	The number of command-line parameters.
params	An array of C-style strings representing command-line parameters.
input_file	Reference to a std::string where the input file path will be stored.
output_file	Reference to a std::string where the output file path will be stored.

Returns

Returns true if the input and output file paths were successfully extracted, false otherwise. In case of failure, an error message is printed to std::cout.

2.2.1.6 save_data()

```
void save_data (
            std::string outputFileName,
            const DiscoveredCycles & DISCOVERED_CYCLES )
```

This function takes a vector of discovered cycles and writes them to an output file. Each cycle is represented as a space-separated sequence of node IDs.

Parameters

outputFileName	The name of the output file where cycles will be saved.
DISCOVERED_CYCLES	Reference to a vector of vectors containing discovered cycles.

2.3 project/source.h File Reference

```
#include <unordered_map>
#include <fstream>
#include <sstream>
#include <vector>
#include <iostream>
```

Typedefs

- using NodeMap = std::unordered_map< int, std::vector< int > >
- using DiscoveredCycles = std::vector< std::vector< int > >

Functions

- bool read_params (int count, const char *params[], std::string &input_file, std::string &output_file)
- NodeMap get_data_and_create_map (const std::string &filenamet)
- void DFS_on_every_node (NodeMap &MY_MAP, DiscoveredCycles &DISCOVERED_CYCLES)
- void DFS (const int start_node, const int cur_node, std::vector< int > visited, NodeMap &MY_MAP, DiscoveredCycles &DISCOVERED_CYCLES)
- void save_data (std::string outputFileName, const DiscoveredCycles &DISCOVERED_CYCLES)
- void print_cycles (const DiscoveredCycles &DISCOVERED_CYCLES)

2.3.1 Typedef Documentation

2.3.1.1 DiscoveredCycles

```
using DiscoveredCycles = std::vector<std::vector<int> >
```

2.3.1.2 NodeMap

```
using NodeMap = std::unordered_map<int, std::vector<int> >
```

2.3.2 Function Documentation

2.3.2.1 DFS()

```
void DFS (
            const int start_node,
            const int cur_node,
            std::vector< int > visited,
            NodeMap & MY_MAP,
            DiscoveredCycles & DISCOVERED_CYCLES )
```

This function performs a depth-first search (DFS) to find cycles in a graph starting from a specified node. It populates the discovered cycles in the DISCOVERED_CYCLES vector.

Parameters

start_node	The node from which the search for cycles begins.
cur_node	The current node being explored during the search.
visited	A vector representing the nodes visited during the current exploration.
MY_MAP	The graph represented as a NodeMap (adjacency list).
DISCOVERED_CYCLES	Reference to a vector of vectors where discovered cycles are stored.

2.3.2.2 DFS_on_every_node()

```
void DFS_on_every_node (
            NodeMap & MY_MAP,
            DiscoveredCycles & DISCOVERED_CYCLES )
```

This function performs a depth-first search (DFS) on every node in the graph and stores the discovered cycles in the DISCOVERED_CYCLES vector.

Parameters

MY_MAP	The graph represented as a NodeMap (adjacency list).
DISCOVERED_CYCLES	Reference to a vector of vectors where discovered cycles are stored.

2.3.2.3 get_data_and_create_map()

```
NodeMap get_data_and_create_map (
            const std::string & filenamet )
```

This function reads data from the specified file and creates a NodeMap, where each node is associated with a list of connected nodes. The file is expected to contain lines representing edges in the format "from_node - to_node".

Parameters

filename	The name of the file containing edge information.

Returns

Returns a NodeMap representing the connectivity information read from the file. The NodeMap is a std↩ ::unordered_map<int, std::vector<int>>, where each key-value pair represents a node and its connected nodes.

2.3.2.4 print_cycles()

```
void print_cycles (
            const DiscoveredCycles & DISCOVERED_CYCLES )
```

This function prints the discovered cycles in the graph.

Parameters

DISCOVERED_CYCLES	Reference to a vector of vectors containing the discovered cycles.

2.3.2.5 read_params()

```
bool read_params (
            int count,
            const char * params[],
            std::string & input_file,
            std::string & output_file )
```

This function parses command-line parameters to extract the input and output file paths. It expects the parameters to be in the form of "-g input_file -c output_file".

Parameters

count	The number of command-line parameters.
params	An array of C-style strings representing command-line parameters.
input_file	Reference to a std::string where the input file path will be stored.
output_file	Reference to a std::string where the output file path will be stored.

Returns

Returns true if the input and output file paths were successfully extracted, false otherwise. In case of failure, an error message is printed to std::cout.

2.3.2.6 save_data()

```
void save_data (
            std::string outputFileName,
            const DiscoveredCycles & DISCOVERED_CYCLES )
```

This function takes a vector of discovered cycles and writes them to an output file. Each cycle is represented as a space-separated sequence of node IDs.

Parameters

outputFileName	The name of the output file where cycles will be saved.
DISCOVERED_CYCLES	Reference to a vector of vectors containing discovered cycles.

2.4 source.h

Go to the documentation of this file.

```
00001 #ifndef SOURCE_H
00002 #define SOURCE_H
00003
00004 #include <unordered_map>
00005 #include <fstream>
00006 #include <sstream>
00007 #include <vector>
00008 #include <iostream>
00009
00010 // #include <algorithm>
00011 // #include <functional>
00012
00013 using NodeMap = std::unordered_map<int, std::vector<int»;
00014 using DiscoveredCycles = std::vector<std::vector<int»;
00015
00028 bool read_params(int count, const char* params[], std::string& input_file, std::string& output_file);
00029
00030
00042 NodeMap get_data_and_create_map(const std::string& filenamet);
00043
00051 void DFS_on_every_node(NodeMap& MY_MAP, DiscoveredCycles& DISCOVERED_CYCLES);
00052
00063 void DFS(const int start_node, const int cur_node, std::vector<int> visited, NodeMap& MY_MAP,
      DiscoveredCycles& DISCOVERED_CYCLES);
00064
00072 void save_data(std::string outputFileName, const DiscoveredCycles& DISCOVERED_CYCLES);
00073
00079 void print_cycles(const DiscoveredCycles& DISCOVERED_CYCLES);
00080
00081 #endif
```