Chromatic number

Generated by Doxygen 1.8.11

Contents

Chapter 1

Chromatic number - Graph coloring problem

1.1 Introduction

This code was developed in education purpose as a project for the university.

Using this code we tried to resolve the problem of coloring undirected graph nodes.

More about the graph coloring problem you can find on Wikipedia: `Graph coloring`

1.2 Compilation

To compile this project you will need to use the **make** command. Follow the steps to compile the project:

- Steps to compile the project
    1. Open the terminal and navigate to the project directory
    2. In the **root** of the project directory using the terminal run the command:

       ```
       make
       ```

       Files that need to be compiled are added in to the **Makefile** file.

       Warning

       > You don't need to modify this file. Please pay attention if you do it.

       Compilation process will generate the executable file: **Application**

1.3 Run the program

1.3.1 Run using the terminal

To start this program using the **terminal** run the following command in to the root of the project directory:

Linux:

```
./Application
```

Windows:

```
Application
```

1.3.2 Run without the terminal

To start the program in user-frendly manner, open the project directory and click on **Application**

1.4 Open source

This project is *open source*. You can check the full project code on this link: `Github`

Chapter 2

Bug List

File main.c

No common bugs in this project. Contact us if you find any of them

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Class Documentation

5.1 node_struct Struct Reference

This is the structure of nodes that we are creating for Welsh-Powell algorithms. We are using this structure to compare the properties between nodes such as id, sum_connection, colors. When we are resolving the graph coloring problem with Welsh-Powell algorithm, we are coloring nodes starting from the nodes with a bigger number of connections.

```
#include <node_structure.h>
```

Public Attributes

- int id
- int sum_connection
- int color

5.1.1 Detailed Description

This is the structure of nodes that we are creating for Welsh-Powell algorithms. We are using this structure to compare the properties between nodes such as id, sum_connection, colors. When we are resolving the graph coloring problem with Welsh-Powell algorithm, we are coloring nodes starting from the nodes with a bigger number of connections.

5.1.2 Member Data Documentation

5.1.2.1 int node_struct::color

Shows us the node color represented as a number.

Warning

Default is **-1** before coloring.

5.1.2.2 int node_struct::id

ID - unique for every node. We are using this value to select the row in the **graph matrix**

Warning

ID represend one row in the graph matrix

5.1.2.3 int node_struct::sum_connection

Sum of connections shows us how many nodes we are connected with.

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

- nodes/node_structure.h

Chapter 6

File Documentation

6.1 algorithms/backtracking/backtracking.c File Reference

```
#include <stdio.h>
#include "./backtracking.h"
```
Include dependency graph for backtracking.c:

6.2 algorithms/backtracking/backtracking.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- int graph_min_colors_backtracing (int ∗graph_matrix, int num_of_nodes)

  *This functions resolve the **Chromatic number** - graph coloring problem using the **Backtracking** algorithm.*

6.2.1 Function Documentation

6.2.1.1 int graph_min_colors_backtracing (int ∗ *graph_matrix,* int *num_of_nodes*)

This functions resolve the ***Chromatic number*** - graph coloring problem using the ***Backtracking*** algorithm.

Author

Denis Lapadatovic

Parameters

∗*graph_matrix*	Using this parameter we have the access to the graph matrix.
num_of_nodes	This parameter will give as the number of nodes

Returns

Minimum number of colors for coloring graph nodes determined by the algorithm

Function returns the minimum number of colors used to color the graph

Set min_colors to be equal **1**

NOTE: Because of the first node we know that we will use minimum one color

Increase pointer start + number of nodes (columns)

NOTE: We will skip the first row, so we need to increase the starting pointer

Start with searching and creating colors

Set the first color to the node

To through the graph matrix columns

We don't need to compare the same row and column

Get the column value from the graph matrix

Check if value is not **0**

Warning

If value is 0 means that nodes are not conneted so we don't need to compare the colors

If colors are the same, increase the start color

Set the color to the node

If start color is bigger then min colors means that we need more colors, so we need to increase the variable **min←_colors**

Increase the position of the pointer

Return minimum number of colors when algorithm is finished

6.3 algorithms/welsh_powell/welsh_powell.c File Reference

```
#include <stdio.h>
#include "./../../nodes/node_structure.h"
#include "./welsh_powell.h"
```
Include dependency graph for welsh_powell.c:

Functions

- int graph_min_colors_welsh_powell (int ∗graph_matrix, int num_of_nodes, struct node_struct ∗nodes)
  *This functions resolve the **Chromatic number** - graph coloring problem using the **Welsh Powell** algorithm.*
- int get_node_color (int node_id, struct node_struct ∗nodes, int num_of_nodes)
  This function return the value of the color for the node.

6.3.1 Function Documentation

6.3.1.1 int get_node_color (int *node_id*, struct node_struct ∗ *nodes*, int *num_of_nodes*)

This function return the value of the color for the node.

Author

Denis Lapadatovic

Parameters

node_id	Input variable used to identify the node using the unique ID
∗*nodes*	This variable is used for the iteration through the nodes with properties
num_of_nodes	This parameter will be used to get the number of nodes.

Returns

The color of the node with ID passed in the function

Default value for color

Return the color of the node

6.3.1.2 int graph_min_colors_welsh_powell (int ∗ *graph_matrix,* int *num_of_nodes,* struct node_struct ∗ *nodes*)

This functions resolve the ***Chromatic number*** - graph coloring problem using the ***Welsh Powell*** algorithm.

Author

Denis Lapadatovic

Parameters

∗*graph_matrix*	Using this parameter we have the access to the graph matrix.
num_of_nodes	This parameter will be used to get the number of nodes.
∗*nodes*	This variable is used for the iteration through the nodes with properties

Returns

Minimum number of colors for coloring graph nodes determined by the algorithm

Select the node from 0 - num_of nodes

If node doesn't have color set

Get the node ID (row number in the graph table)

Go through the the row ID $>$ columns

Get the values from the column (row cell)

Check if nodes are conneted (row ID and column ID) : 0 - not connected $|$ $>$ 0 - connected

Nodes are connected get the color of the node that we compare

Chech if color of the neighbor node is the same as current 'min_colors'

If node can be colored with min color set the color

Return minimum number of colors when algorithm is finished

6.4 algorithms/welsh_powell/welsh_powell.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- int graph_min_colors_welsh_powell (int ∗graph_matrix, int num_of_nodes, struct node_struct ∗nodes)
  *This functions resolve the **Chromatic number** - graph coloring problem using the **Welsh Powell** algorithm.*
- int get_node_color (int node_id, struct node_struct ∗nodes, int num_of_nodes)
  This function return the value of the color for the node.

6.4.1 Function Documentation

6.4.1.1 int get_node_color (int *node_id,* struct **node_struct** ∗ *nodes,* int *num_of_nodes*)

This function return the value of the color for the node.

Author

Denis Lapadatovic

Parameters

node_id	Input variable used to identify the node using the unique ID
∗nodes	This variable is used for the iteration through the nodes with properties
num_of_nodes	This parameter will be used to get the number of nodes.

Returns

The color of the node with ID passed in the function

Default value for color

Return the color of the node

6.4.1.2 int graph_min_colors_welsh_powell (int ∗ *graph_matrix,* int *num_of_nodes,* struct **node_struct** ∗ *nodes*)

This functions resolve the ***Chromatic number*** - graph coloring problem using the ***Welsh Powell*** algorithm.

Author

Denis Lapadatovic

Parameters

∗graph_matrix	Using this parameter we have the access to the graph matrix.
num_of_nodes	This parameter will be used to get the number of nodes.
∗nodes	This variable is used for the iteration through the nodes with properties

Returns

Minimum number of colors for coloring graph nodes determined by the algorithm

Select the node from 0 - num_of nodes

If node doesn't have color set

Get the node ID (row number in the graph table)

Go through the the row ID $>$ columns

Get the values from the column (row cell)

Check if nodes are conneted (row ID and column ID) : 0 - not connected $|$ $>$ 0 - connected

Nodes are connected get the color of the node that we compare

Chech if color of the neighbor node is the same as current 'min_colors'

If node can be colored with min color set the color

Return minimum number of colors when algorithm is finished

6.5 graph/create_graph.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "create_graph.h"
```

Include dependency graph for create_graph.c:

Functions

- int ∗ create_graph (int ∗graph_matrix, int ∗num_of_nodes)
  *This functions create the graph matrix. User enter the data for the graph. This funtion is used in **Custom** environment.*

6.5.1 Function Documentation

6.5.1.1 int∗ create_graph (int ∗ *graph_matrix,* int ∗ *num_of_nodes*)

This functions create the graph matrix. User enter the data for the graph. This funtion is used in **Custom** environment.

Author

Denis Lapadatovic

Parameters

∗*graph_matrix*	Using this parameter we have the access to the graph matrix. ∗**graph_matrix** is initialize out of this function.
∗*num_of_nodes*	This parameter will be used to add the number of nodes.

Returns

Pointer to the new created graph matrix

Calculate the graph matrix size

Realloc the size of graph matrix

Create the helper variables

Iterate through the graph matrix row

Iterate through the graph matrix column

Ask user to enter if node is conneted with another node

Save the value in the graph matrix

Increase the position of graph matrix pointer

Return the pointer of the graph matrix.

Warning

Realloc() returns the pointer to the beginning of newly allocated memory.

6.6 graph/create_graph.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- int ∗ create_graph (int ∗graph_matrix, int ∗num_of_nodes)
  This functions create the graph matrix. User enter the data for the graph. This funtion is used in ***Custom*** *environment.*

6.6.1 Function Documentation

6.6.1.1 int∗ create_graph (int ∗ *graph_matrix,* int ∗ *num_of_nodes*)

This functions create the graph matrix. User enter the data for the graph. This funtion is used in **Custom** environment.

Author

Denis Lapadatovic

Parameters

∗*graph_matrix*	Using this parameter we have the access to the graph matrix. ∗**graph_matrix** is initialize out of this function.
∗*num_of_nodes*	This parameter will be used to add the number of nodes.

Returns

Pointer to the new created graph matrix

Calculate the graph matrix size

Realloc the size of graph matrix

Create the helper variables

Iterate through the graph matrix row

Iterate through the graph matrix column

Ask user to enter if node is conneted with another node

Save the value in the graph matrix

Increase the position of graph matrix pointer

Return the pointer of the graph matrix.

Warning

Realloc() returns the pointer to the beginning of newly allocated memory.

6.7 graph/load_graph.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "./load_graph.h"
```
Include dependency graph for load_graph.c:

Functions

- int ∗ load_graph (int ∗graph_matrix, int ∗num_of_nodes)

  *This functions load the test graph matrix from the files. This funtion is used in **Test** environment.*

6.7.1 Function Documentation

6.7.1.1 int∗ load_graph (int ∗ *graph_matrix*, int ∗ *num_of_nodes*)

This functions load the test graph matrix from the files. This funtion is used in **Test** environment.

Author

Denis Lapadatovic

Parameters

∗*graph_matrix*	Using this parameter we have the access to the graph matrix.
num_of_nodes	This parameter will be used to add the number of nodes.

Returns

graph_matrix - Pointer to the new created graph matrix

Solution for fixing the problem with the string with " " character : `Solution`

temp statement to clear buffer

Add path to variable

Add extension to the file_name

Concat path folder with file name

Open file with graph data

First string (number) in the file is represents the number of nodes

Calculate the size of new graph matrix

Realloc the 'graph_matrix' variable and increase the size of it

Initialite the primary graph matrix pointer with new created pointer

Crete the pointer for iteration

Go through the graph matrix from file and create the 'graph_matrix_init'

Get the value from file as a string

Convert string value to integer and add to matrix

Increment the position of pointer

Close the file when reading is finished

Restart the position the of helper variable

6.8 graph/load_graph.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- int ∗ load_graph (int ∗graph_matrix, int ∗num_of_nodes)
  This functions load the test graph matrix from the files. This funtion is used in ***Test*** *environment.*

6.8.1 Function Documentation

6.8.1.1 int∗ load_graph (int ∗ *graph_matrix,* int ∗ *num_of_nodes*)

This functions load the test graph matrix from the files. This funtion is used in **Test** environment.

Author

Denis Lapadatovic

Parameters

∗graph_matrix	Using this parameter we have the access to the graph matrix.
num_of_nodes	This parameter will be used to add the number of nodes.

Returns

graph_matrix - Pointer to the new created graph matrix

Solution for fixing the problem with the string with " " character : `Solution`

temp statement to clear buffer

Add path to variable

Add extension to the file_name

Concat path folder with file name

Open file with graph data

First string (number) in the file is represents the number of nodes

Calculate the size of new graph matrix

Realloc the 'graph_matrix' variable and increase the size of it

Initialite the primary graph matrix pointer with new created pointer

Crete the pointer for iteration

Go through the graph matrix from file and create the 'graph_matrix_init'

Get the value from file as a string

Convert string value to integer and add to matrix

Increment the position of pointer

Close the file when reading is finished

Restart the position the of helper variable

6.9 graph/show_graph.c File Reference

```
#include <stdio.h>
#include "show_graph.h"
```
Include dependency graph for show_graph.c:

Functions

- void show_graph_matrix (int ∗graph_matrix, int num_of_nodes)

  This functions shows the graph matrix.

6.9.1 Function Documentation

6.9.1.1 void show_graph_matrix (int ∗ *graph_matrix,* int *num_of_nodes*)

This functions shows the graph matrix.

Author

Denis Lapadatovic

Parameters

∗graph_matrix	Using this parameter we have the access to the graph matrix.
num_of_nodes	This parameter will be used to get the number of nodes.

Returns

Nothing

Iterator through the rows

Iterator through the columns

Copy of **graph_matrix**

Show the value of the graph matrix cell

6.10 graph/show_graph.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- void show_graph_matrix (int ∗graph_matrix, int num_of_nodes)
  This functions shows the graph matrix.

6.10.1 Function Documentation

6.10.1.1 void show_graph_matrix (int ∗ *graph_matrix,* int *num_of_nodes*)

This functions shows the graph matrix.

Author

Denis Lapadatovic

Parameters

∗graph_matrix	Using this parameter we have the access to the graph matrix.
num_of_nodes	This parameter will be used to get the number of nodes.

Returns

Nothing

Iterator through the rows

Iterator through the columns

Copy of **graph_matrix**

Show the value of the graph matrix cell

6.11 main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "nodes/node_structure.h"
#include "nodes/nodes.h"
#include "algorithms/backtracking/backtracking.h"
#include "algorithms/welsh_powell/welsh_powell.h"
#include "graph/create_graph.h"
#include "graph/show_graph.h"
#include "graph/load_graph.h"
```

Include dependency graph for main.c:

Functions

- void algorithm_menu (int ∗graph_matrix, int num_of_nodes)

  This function will be used to give the posibility to the user to select the algorithm that he/she wants to use for resolving the graph coloring problem.
- int main ()

6.11.1 Detailed Description

Author

Denis Lapadatovic

Version

v0.1

Date

2018-05-30

Bug No common bugs in this project. Contact us if you find any of them

6.11.2 Function Documentation

6.11.2.1 void algorithm_menu (int ∗ *graph_matrix,* int *num_of_nodes*)

This function will be used to give the posibility to the user to select the algorithm that he/she wants to use for resolving the graph coloring problem.

Author

Denis Lapadatovic

Parameters

∗*graph_matrix*	This pointer parameter will be provided in to the functions for resolving graph problem. ∗**graph_matrix** is initialize out of this function.
num_of_nodes	This variable will be provided in to the functions for resolving graph problem. **num_of_nodes** is initialize out of this function.

Returns

Nothing

In this variable we are storing the minimum number of colors needed to color the graph

In this variable we are storing the menu decision for the desired algorithm for resolving the problem

Let user to enter the decision about the algorithm and save it in the variable

Check the user algorithm option

If user enter **1** resolve the problem using the **Backtracing** algorithm

Run the function and save the final result in to the variable

If user enter **2** resolve the problem using the **Welsh-Powell** algorithm

Create all nodes data with 3 properties (needed for **Welsh-Powell** algorithm)

Run the function to initialize the nodes with ID, Sum of connections and Color

Show nodes before sort

Sort the nodes in descending order by number of connections per node

Show nodes after sort

Run the function and save the final result in to the variable

Show nodes after sort

If user select the other number for algorithm

Warning

Algorithm options are **1** for **Backtracing** **2** for **Welsh-Powell**

6.11.2.2 int main ( )

Variable used to save the number of nodes of our graph

Pointer to the graph matrix. In this pointer we will initialize the grapg matrix

Variable used for the menu selection. To choose between **Test** and **Custom** environment

Check if program environment is selected. If **TRUE** continue the program

Allocate the memory for graph matrix with N rows and N columns

Let user to decide which environment wants to run Test - Load dummy data from files located in the folder **data** Custom - Create own graph

Environment menu

If user select the first environment **Test** run the code below

Save the return value from the **load_graph** function in **graph_matrix** pointer

Set variable to **1** - means that environment is selected

If user select the second environment **Custom** run the code below

Let user enter the number of nodes for the graph and save it in **num_of_nodes** variable

Call the function to create the graph

Warning

Graph matrix need to be created for this environment using the **create_graph** function

Save the created graph in **graph_matrix** pointer

Set variable to **1** - means that environment is selected

If user select the other number for environment

Warning

Environment options are **1** for **TEST 2** for **CUSTOM**

Show the graph_matrix

Show the menu to select the algorithm

At the end of the program remove the **graph matrix** pointer from the memory

6.12 nodes/node_structure.h File Reference

C library with node structure details.

This graph shows which files directly or indirectly include this file:

Classes

- struct node_struct

  This is the structure of nodes that we are creating for Welsh-Powell algorithms. We are using this structure to compare the properties between nodes such as id, sum_connection, colors. When we are resolving the graph coloring problem with Welsh-Powell algorithm, we are coloring nodes starting from the nodes with a bigger number of connections.

Typedefs

- typedef struct node_struct node_struct

  This is the structure of nodes that we are creating for Welsh-Powell algorithms. We are using this structure to compare the properties between nodes such as id, sum_connection, colors. When we are resolving the graph coloring problem with Welsh-Powell algorithm, we are coloring nodes starting from the nodes with a bigger number of connections.

6.12.1 Detailed Description

C library with node structure details.

6.12.2 Typedef Documentation

6.12.2.1 typedef struct **node_struct node_struct**

This is the structure of nodes that we are creating for Welsh-Powell algorithms. We are using this structure to compare the properties between nodes such as id, sum_connection, colors. When we are resolving the graph coloring problem with Welsh-Powell algorithm, we are coloring nodes starting from the nodes with a bigger number of connections.

6.13 nodes/nodes.c File Reference

```
#include <stdio.h>
#include "./node_structure.h"
#include "nodes.h"
```
Include dependency graph for nodes.c:

Functions

- void show_nodes_data (struct node_struct ∗nodes, int num_of_nodes)

  This function shows the details for the nodes in one list.
- void create_nodes (struct node_struct ∗nodes, int num_of_nodes, int ∗graph_matrix)

  This function initialize/create the nodes with default values.
- void sort_nodes (struct node_struct ∗nodes, int num_of_nodes)

  This function sort the nodes in descending order by sum of connections.

6.13.1 Function Documentation

6.13.1.1 void create_nodes (struct **node_struct** ∗ *nodes*, int *num_of_nodes*, int ∗ *graph_matrix*)

This function initialize/create the nodes with default values.

Author

Denis Lapadatovic

Parameters

∗*nodes*	This variable is used for the iteration and showing the nodes with properties
num_of_nodes	This variable give us the number of nodes.
∗*graph_matrix*	Using this parameter we have the access to the graph matrix.

Returns

Nothing

Iterator_1 - Iterate through the nodes

Iterator_2 - Iterate through the rows and columns of the graph matrix

Copy of the 'graph_matrix'

Set node **ID**

Set node default color **-1** - not colored

Set the sum of connections to **0** - default

Iterate through the graph matrix and check for number of connections of the node

If matrix value is not 0 - means that nodes are connected

Increase the number of connections

6.13.1.2 void show_nodes_data (struct node_struct ∗ *nodes,* int *num_of_nodes*)

This function shows the details for the nodes in one list.

Author

Denis Lapadatovic

Parameters

∗*nodes*	This variable is used for the iteration and showing the nodes with properties
num_of_nodes	This variable give us the number of nodes. **num_of_nodes** is initialize out of this function.

Returns

Nothing

Helper variable used for iteration through the nodes

Iterate and show the nodes

Increase the position of the pointer

6.13.1.3 void sort_nodes (struct node_struct ∗ *nodes,* int *num_of_nodes*)

This function sort the nodes in descending order by sum of connections.

Author

Denis Lapadatovic

Parameters

∗*nodes*	This variable is used for the iteration and showing the nodes with properties
num_of_nodes	This variable give us the number of nodes.

Returns

Nothing

Iterate through the all nodes

Iterate through the all nodes to compare with selected node

Save the sum of connections for the node

Save the color for the node

Save ID for the node

SELECTED NODE: Iterate through the node and compare with other nodes

Iteratete through the rest of nodes and compare with **SELECTED NODE**

If number of connections of the **SELECTED NODE** is smaller then other node change the positions of the nodes

Save the values for the **SELECTED NODE** in aux variables

Replace the values of the first node with the second node

Replace the values of the second node with aux variables

6.14 nodes/nodes.h File Reference

C library for manipulation with graph nodes.

This graph shows which files directly or indirectly include this file:

Functions

- void show_nodes_data (struct node_struct ∗nodes, int num_of_nodes)

  This function shows the details for the nodes in one list.
- void create_nodes (struct node_struct ∗nodes, int num_of_nodes, int ∗graph_matrix)

  This function initialize/create the nodes with default values.
- void sort_nodes (struct node_struct ∗nodes, int num_of_nodes)

  This function sort the nodes in descending order by sum of connections.

6.14.1 Detailed Description

C library for manipulation with graph nodes.

6.14.2 Function Documentation

6.14.2.1 void create_nodes (struct node_struct ∗ *nodes*, int *num_of_nodes*, int ∗ *graph_matrix*)

This function initialize/create the nodes with default values.

Author

Denis Lapadatovic

Parameters

∗*nodes*	This variable is used for the iteration and showing the nodes with properties
num_of_nodes	This variable give us the number of nodes.
∗*graph_matrix*	Using this parameter we have the access to the graph matrix.

Returns

Nothing

Iterator_1 - Iterate through the nodes

Iterator_2 - Iterate through the rows and columns of the graph matrix

Copy of the 'graph_matrix'

Set node **ID**

Set node default color **-1** - not colored

Set the sum of connections to **0** - default

Iterate through the graph matrix and check for number of connections of the node

If matrix value is not 0 - means that nodes are connected

Increase the number of connections

6.14.2.2 void show_nodes_data (struct node_struct ∗ *nodes*, int *num_of_nodes*)

This function shows the details for the nodes in one list.

Author

Denis Lapadatovic

Parameters

∗*nodes*	This variable is used for the iteration and showing the nodes with properties
num_of_nodes	This variable give us the number of nodes. **num_of_nodes** is initialize out of this function.

Returns

Nothing

Helper variable used for iteration through the nodes

Iterate and show the nodes

Increase the position of the pointer

6.14.2.3 void sort_nodes (struct node_struct ∗ *nodes,* int *num_of_nodes*)

This function sort the nodes in descending order by sum of connections.

Author

Denis Lapadatovic

Parameters

∗*nodes*	This variable is used for the iteration and showing the nodes with properties
num_of_nodes	This variable give us the number of nodes.

Returns

Nothing

Iterate through the all nodes

Iterate through the all nodes to compare with selected node

Save the sum of connections for the node

Save the color for the node

Save ID for the node

SELECTED NODE: Iterate through the node and compare with other nodes

Iteratete through the rest of nodes and compare with **SELECTED NODE**

If number of connections of the **SELECTED NODE** is smaller then other node change the positions of the nodes

Save the values for the **SELECTED NODE** in aux variables

Replace the values of the first node with the second node

Replace the values of the second node with aux variables

Index