Graph Coloring

Generated by Doxygen 1.8.13

Contents

Chapter 1

Chromatic-Number

The minimum number of colours needed to colour each node in a undirected graph

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Data Structure Documentation

4.1 a_graph Struct Reference

Data Fields

- int number_of_nodes

  Struct that defines a graph type.
- int is_initialized
- int * **adjacent_matrix**
- int * **node_value**
- int **number_of_vertices**

4.1.1 Detailed Description

Definition at line 8 of file iograph.h.

4.1.2 Field Documentation

4.1.2.1 is_initialized

```
int is_initialized
```

TRUE (1) =is initialized FALSE (0) =is not initialized

Definition at line 18 of file iograph.h.

4.1.2.2 number_of_nodes

`number_of_nodes`

Struct that defines a graph type.

Stores number of nodes in our graph

Definition at line 13 of file iograph.h.

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

- iograph.h

4.2 a_list_node Struct Reference

Data Fields

- int **info**
- struct a_list_node * **next**

4.2.1 Detailed Description

Definition at line 8 of file graphcommands.h.

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

- graphcommands.h

Chapter 5

File Documentation

5.1 coloringAlgorithms.c File Reference

C library for diferent algorithms of Graph Coloring.

```
#include <stdio.h>
#include "iograph.h"
#include "graphcommands.h"
#include "coloringAlgorithms.h"
```

5.1.1 Detailed Description

C library for diferent algorithms of Graph Coloring.

5.2 coloringAlgorithms.h File Reference

C library for implementation of diferent algorithms of Graph Coloring.

Functions

- void find_color_greedy_alg_basic (struct a_graph ∗my_graph)
- void find_color_greedy_alg_with_bfs (struct a_graph ∗my_graph, int ∗bfs_order_vector)
- void print_solution (int ∗colors, struct a_graph ∗my_graph, int ∗bfs_order_vector)
- int **is_safe** (int node, struct a_graph ∗my_graph, int ∗colors, int c, int ∗bfs_order_vector)
- int graph_coloring_tool (struct a_graph ∗my_graph, int m, int ∗colors, int node, int ∗bfs_order_vector)
- int graph_coloring_backtracking (struct a_graph ∗my_graph, int m, int ∗bfs_order_vector)

5.2.1 Detailed Description

C library for implementation of diferent algorithms of Graph Coloring.

5.2.2 Function Documentation

5.2.2.1 find_color_greedy_alg_basic()

```
void find_color_greedy_alg_basic (
            struct a_graph * my_graph )
```

Colors the graph and prints the colors, using the greedy algorithm A greedy algorithm is an algorithmic paradigm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum

Parameters

∗my_graph	Pointer to our graph

Definition at line 10 of file coloringAlgorithms.c.

5.2.2.2 find_color_greedy_alg_with_bfs()

```
void find_color_greedy_alg_with_bfs (
            struct a_graph * my_graph,
            int * bfs_order_vector )
```

Colors the graph and prints the colors, using the greedy algorithm A greedy algorithm is an algorithmic paradigm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum

Parameters

∗my_graph	Pointer to our graph
∗bfs_order_vector	BFS order of our graph

Definition at line 60 of file coloringAlgorithms.c.

5.2.2.3 graph_coloring_backtracking()

```
int graph_coloring_backtracking (
            struct a_graph * my_graph,
            int m,
            int * bfs_order_vector )
```

This functions solves the coloring problem using backtracking, mainly by using the tool created before If the problem can be solved with 'm' or less colors, then it returns true and prints the solution, if not, it returns false

Parameters

∗my_graph	Pointer to our graph
m	The number of colors we try to color our graph with
bfs_order_vector	BFS order of our graph

Definition at line 215 of file coloringAlgorithms.c.

5.2.2.4 graph_coloring_tool()

```
int graph_coloring_tool (
              struct a_graph * my_graph,
              int m,
              int * colors,
              int node,
              int * bfs_order_vector )
```

Solving the

Parameters

∗my_graph	Pointer to our graph
m	The number of colors we try to color our graph with
∗colors	Vector to store the colors
node	Current node
bfs_order_vector	BFS order of our graph

Definition at line 178 of file coloringAlgorithms.c.

5.2.2.5 print_solution()

```
print_solution (
              int * colors,
              struct a_graph * my_graph,
              int * bfs_order_vector )
```

Print solution

Parameters

∗colors	Vector to store the colors
∗my_graph	Pointer to our graph
bfs_order_vector	BFS order of our graph

Definition at line 117 of file coloringAlgorithms.c.

5.3 graphcommands.c File Reference

C library for Graph commands(check for adjacency, bfs sorting)

```
#include <stdio.h>
#include "iograph.h"
#include "graphcommands.h"
```

5.3.1 Detailed Description

C library for Graph commands(check for adjacency, bfs sorting)

5.4 graphcommands.h File Reference

C library for implementation of Graph commands(check for adjacency, bfs sorting)

Data Structures

- struct a_list_node

Functions

- int is_adjacent (struct a_graph *my_graph, int node1, int node2)
- void push_begining_list (struct a_list_node *head, int new_element_value)
- int pop_end_list (struct a_list_node *head)
- void graph_bfs (struct a_graph *my_graph, int start_node, int *bfs_order_vector)

5.4.1 Detailed Description

C library for implementation of Graph commands(check for adjacency, bfs sorting)

5.4.2 Function Documentation

5.4.2.1 graph_bfs()

```
void graph_bfs (
            struct a_graph * my_graph,
            int start_node,
            int * bfs_order_vector )
```

Order the graph nodes in a Breadth-first manner

Parameters

∗my_graph	The graph
start_node	The node that we consider as a "head"
∗bfs_order_vector	A vector that's gonna store the order in which bfs crosses the graph

Definition at line 89 of file graphcommands.c.

5.4.2.2 is_adjacent()

```
int is_adjacent (
              struct a_graph * my_graph,
              int node1,
              int node2 )
```

Check if two nodes are adjacent

Parameters

∗my_graph	Pointer to our graph
node1	First node
node2	Second node

Definition at line 9 of file graphcommands.c.

5.4.2.3 pop_end_list()

```
int pop_end_list (
              struct a_list_node * head )
```

Pop element from the end of the list

Parameters

∗head	The head of the list

Definition at line 54 of file graphcommands.c.

5.4.2.4 push_begining_list()

```
void push_begining_list (
              struct a_list_node * head,
              int new_element_value )
```

Push element at the begining of the list

Parameters

∗head	The head of the list
new_element_value	New element value

Definition at line 29 of file graphcommands.c.

5.5 iograph.c File Reference

C library for implementation of Input/Output of graphs.

```
#include <stdio.h>
#include "iograph.h"
```

5.5.1 Detailed Description

C library for implementation of Input/Output of graphs.

5.6 iograph.h File Reference

C library for Input/Output of graphs.

Data Structures

- struct a_graph

Functions

- void non_trivial_graph_generator (struct a_graph ∗my_graph)
- void print_graph (struct a_graph ∗my_graph)
- void read_graph (struct a_graph ∗my_graph)
- void read_graph_file (struct a_graph ∗my_graph)

5.6.1 Detailed Description

C library for Input/Output of graphs.

5.6.2 Function Documentation

5.6.2.1 non_trivial_graph_generator()

```
void non_trivial_graph_generator (
            struct a_graph * my_graph )
```

Generator for a random adjecency matrix

Definition at line 8 of file iograph.c.

5.6.2.2 print_graph()

```
void print_graph (
            struct a_graph * my_graph )
```

Prints a graph

Parameters

∗my_graph	Pointer to our graph

Definition at line 41 of file iograph.c.

5.6.2.3 read_graph()

```
void read_graph (
            struct a_graph * my_graph )
```

Prints a graph

Parameters

∗my_graph	Pointer to our graph

Definition at line 71 of file iograph.c.

5.6.2.4 read_graph_file()

```
void read_graph_file (
            struct a_graph * my_graph )
```

Prints a graph

Parameters

∗my_graph	Pointer to our graph

Definition at line 116 of file iograph.c.

5.7 main.c File Reference

Chromatic Number Problem.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "iograph.h"
#include "graphcommands.h"
#include "coloringAlgorithms.h"
```

5.7.1 Detailed Description

Chromatic Number Problem.

The minimum number of colors needed to colors each node in an undirected graph

Index