C++ Programming Work

Graph Coloring Algorithm Program Code Description

April 19, 2021

Jeremy J. Williams

1. Pseudocode

- Begin Program
- First it reads the XML graph file to get the Vertices and Edges and sets those vertices and edges ID number serially.
- Detects and sets total color by counting total vertices and then it sets Edges to a Graph list to find the Vertices color.
- Process first vertices and first color ID using its default value.
- Process rest of the vertices to color it by using Greedy Coloring algorithm.
- Finally, shows the Vertex coloring result.
- End Program

2. Source Code

2.1 Global Declarations

The following code is global declarations of headers, variables and functions marker.

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <cstdlib>
#include <stdio.h>

using namespace std;

struct edge {
        string src;
        string dst;
};


// Function prototypes
string get file(string fileName);
int get_vertices_edges(string xml, vector<string> &vertices, vector<edge> &edges);
int get_vertice_id(string vertice, vector <string> vertices);
string get_vertice_Name(size_t id, vector <string> vertices);
void greedyColoring(int maxnum, vector<int> &color, vector<vector<int>> graph);
```

2.2 Main

The following code is the main function for the entry point of the program.

The code starts the program and reads the XML file to process Greedy Color Graph.

```
int main()
{
	int n = 0;
	int e = 0;
	string fileName = "greedy_color_graph.xml";
	vector <string> vertices;
	vector <edge> edges;
	vector <vector<int>> graph;
	vector <int> color;
	string text = {};
	int nReply = 0;

	text = get_file(fileName);
	if (text == "")  goto CANCEL;

	nReply = get_vertices_edges(text, vertices, edges);
	if (nReply == -1) goto CANCEL;

	cout << "Total Number of Vertices: " << vertices.size() << endl;
	cout << "Total Number of Edges: " << edges.size() << endl;

	if (vertices.size() < 2) { cout << "Vertices must be more than one." << endl; goto CANCEL;
}
	if (edges.size() < 1) { cout << "Edges cannot be zero." << endl; goto CANCEL; }

	n = vertices.size();
	e = edges.size();
	graph.resize(n);
	color.resize(n);


	for (int i = 0; i < e; i++)
	{
		int x = get_vertice_id(edges[i].src, vertices);
		if (x == -1) { cout << "id not found of vertice " << edges[i].src << endl; goto
CANCEL; }
		int y = get_vertice_id(edges[i].dst, vertices);
		if (x == -1) { cout << "id not found of vertice " << edges[i].dst << endl; goto
CANCEL; }

		graph[x].push back(y);
		graph[y].push_back(x);
	}

	for (size_t i = 0; i < graph.size(); i++)
	{
		for (size_t j = 0; j < graph[i].size(); j++)
		{
		  cout << "Vertex " << get_vertice_Name(i,vertices) << " or " << i << " is
			  connected with " << "Vertex " << get_vertice_Name(graph[i][j],vertices)
			 << " or " << graph[i][j] <<  endl;
		}
	}

	greedyColoring(n, color, graph);

	for (int i = 0; i<n; i++)
	{

		cout << "Vertex " << get_vertice_Name(i,vertices) << " is coloured " << color[i] +
		1 << "\n";
	}
CANCEL:
	system("PAUSE");
    return 0;
}
```

2.3 Greedy Coloring Process

The following code is for Greedy Coloring process.

It uses the minimum color for coloring the graph vertices.

```
void greedyColoring(int maxnum, vector<int> &color, vector<vector<int>> graph)
{
	int i = 0;
	size_t j = 0;
	int n = maxnum;
	bool* unused;
	unused = new bool[n];

	color[0] = 0;
	for (i = 1; i<n; i++)
		color[i] = -1;


	for (i = 0; i<n; i++)
		unused[i] = 0;

	for (i = 1; i < n; i++)
	{
		for (j = 0; j<graph[i].size(); j++)
			if (color[graph[i][j]] != -1)
				unused[color[graph[i][j]]] = true;
		int cr;
		for (cr = 0; cr<n; cr++)
			if (unused[cr] == false)
				break;

		color[i] = cr;

		for (j = 0; j<graph[i].size(); j++)
			if (color[graph[i][j]] != -1)
				unused[color[graph[i][j]]] = false;
	}

}
```

2.4 Vertex ID

The following code retrieves vertices ID from vertices list.

```
int get_vertice_id(string vertice, vector <string> vertices){

	for (size_t i = 0; i < vertices.size(); i++)
	{
		if (vertices[i] == vertice) return i;
	}
	return -1;
}
```

2.5 Vertex Name

The following code retrieves the vertices Name from vertices list.

```
string get_vertice_Name(size_t id, vector <string> vertices) {
	if (id < vertices.size()) return vertices[id];
	return "";
}
```

2.6 Vertex and Edge List

The following code retrieves vertices and edges list from XML text file.

```
int get_vertices_edges(string xml, vector<string> &vertices, vector<edge> &edges) {
	string tagGraph = "<graph>";
	string tagGraphEnd = "</graph>";
	string tagNodes = "<nodes>";
	string tagNodesEnd = "</nodes>";
	string tagNode = "<node";
	string Node = {};
	string Edge = {};
	string tagEdges = "<edges>";
	string tagEdgesEnd = "</edges>";
	string tagEdge = "<edge";
	string tagEnd = "/>";
	string atbName = "name=";
	string atbSrc = "src=";
	string atbDst = "dst=";
	string Src = {};
	string Dst = {};
	size_t nStart = 0;
	size_t nEnd = 0;
	size_t nOffset = 0;
	bool isValid = false;

	// scan for valid graph xml node
	isValid = false;
	nStart = xml.find(tagGraph);
	if (nStart != string::npos)
	{
		nStart = xml.find(tagGraphEnd, tagGraph.length());
		if (nStart != string::npos) isValid = true;
	}
	if (isValid == false)
	{
		cout << "Invalid graph xml node found." << endl;
		return -1;
	}

	// scan for valid nodes xml node
	isValid = false;
	nStart = xml.find(tagNodes);
	if (nStart != string::npos)
	{
		nStart = xml.find(tagNodesEnd, tagNodes.length());
		if (nStart != string::npos) isValid = true;
	}
	if (isValid == false)
	{
		cout << "Invalid nodes xml node found." << endl;
		return -1;
	}

	// scan for valid edges xml node
	isValid = false;
	nStart = xml.find(tagEdges);
	if (nStart != string::npos)
	{
		nStart = xml.find(tagEdgesEnd, tagEdges.length());
		if (nStart != string::npos) isValid = true;
	}
	if (isValid == false)
	{
		cout << "Invalid edges xml node found." << endl;
		return -1;
	}

	// find Nodes
	nStart = xml.find(tagGraph);
	if (nStart != string::npos)
	{
		nOffset = tagGraph.length();
		nStart = xml.find(tagNodes, nOffset);
		if (nStart != string::npos)
```

```
	{
		nOffset = nStart + tagNodes.length();
		nStart = xml.find(tagNode, nOffset);
		while (nStart != string::npos)
		{
			nOffset = nStart + tagNode.length();
			nStart = xml.find(atbName, nOffset);
			if (nStart != string::npos)
			{
				nOffset = nStart + atbName.length();
				nEnd = xml.find(tagEnd, nOffset);
				if (nEnd != string::npos)
				{
					Node = xml.substr(nOffset, nEnd - nOffset);
					Node = Node.substr(1, Node.length() - 2);
					vertices.push_back(Node);
					cout << "Vertex " << Node << " is " <<
					get_vertice_id(Node, vertices) << endl;
					nOffset = nEnd + tagEnd.length();
					nStart = xml.find(tagNode, nOffset);
				}
				else
				{
					cout << "Invalid end of node." << endl;
					return -1;
				}

			}
			else
			{
				cout << "Attribute 'Name' not found." << endl;
				return -1;
			}
		}
	}
}

// find Edges
nStart = xml.find(tagGraph);
if (nStart != string::npos)
{
	nOffset = tagGraph.length();
	nStart = xml.find(tagEdges, nOffset);
	if (nStart != string::npos)
	{
		nOffset = nStart + tagEdges.length();
		nStart = xml.find(tagEdge, nOffset);
		while (nStart != string::npos)
		{
			nOffset = nStart + tagEdge.length();
			nStart = xml.find(atbSrc, nOffset);
			if (nStart != string::npos)
			{
				nOffset = nStart + atbSrc.length();
				nEnd = xml.find(atbDst, nOffset);
				if (nEnd != string::npos)
				{
					Src = xml.substr(nOffset, nEnd - nOffset);
					Src = Src.substr(1, Src.length() - 3);
					//cout << "edge source " << Src << endl;
					nStart = nEnd;
					nOffset = nEnd + atbDst.length();
					nEnd = xml.find(tagEnd, nOffset);
					if (nEnd != string::npos)
					{
						Dst = xml.substr(nOffset, nEnd - nOffset);
						Dst = Dst.substr(1, Dst.length() - 3);
						//cout << "edge destination " << Dst << endl;
					}
					else
					{
						cout << "Invalid end of edge node." << endl;
						return -1;
					}
					edge e;
					e.src = Src; e.dst = Dst;
					edges.push_back(e);
					cout << "Edge " << Src << "--" << Dst << endl;
```

```
                                        nOffset = nEnd + tagEnd.length();
                                        nStart = xml.find(tagEdge, nOffset);
                                }
                                else
                                {
                                        cout << "Invalid end of attribute 'Dst'." << endl;
                                        return -1;
                                }

                        }
                        else
                        {
                                cout << "Attribute 'Src' not found." << endl;
                                return -1;
                        }
                }
        }

        return 0;
}
```

2.6 XML Data

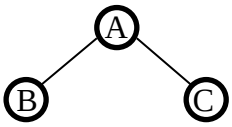
The following code read all XML text from the XML file.

```
string get_file(string fileName)
{
        string buffer;
        char c;

        ifstream in(fileName);   if (!in) { cout << fileName << " not found \n";
}
        while (in.get(c)) buffer += c;
        in.close();

        return buffer;
```

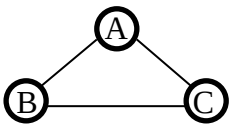
3. Execution of Source Code

See execution screenshots using the following sample graphs:

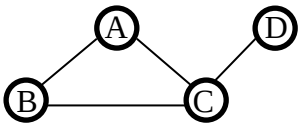
Sample 1)



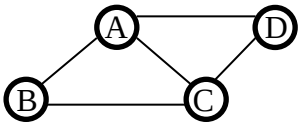
Sample 2)



Sample 3)

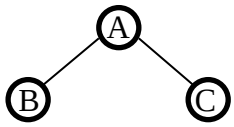


Sample 4)



3.1 Execution #1

Sample 1)



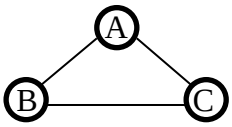
main.cpp greedy_color_grap...

```
<graph>
        <nodes>
                <node name="A"/>
                <node name="B"/>
                <node name="C"/>
        </nodes>
        <edges>
                <edge src="A" dst="B" />
                <edge src="A" dst="C" />
        </edges>
</graph>

```

```
Vertex A is 0
Vertex B is 1
Vertex C is 2
Edge A--B
Edge A--C
Total Number of Vertices: 3
Total Number of Edges: 2
Vertex A or 0 is connected with Vertex B or 1
Vertex A or 0 is connected with Vertex C or 2
Vertex B or 1 is connected with Vertex A or 0
Vertex C or 2 is connected with Vertex A or 0
Vertex A is coloured 1
Vertex B is coloured 2
Vertex C is coloured 2
sh: 1: PAUSE: not found


...Program finished with exit code 0
Press ENTER to exit console.
```

3.2 Execution #2

Sample 2)



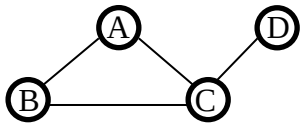
main.cpp greedy_color_grap...

```
<graph>
        <nodes>
                <node name="A"/>
                <node name="B"/>
                <node name="C"/>
        </nodes>
        <edges>
                <edge src="A" dst="B" />
                <edge src="A" dst="C" />
                <edge src="B" dst="C" />
        </edges>
</graph>
```

```
Vertex A is 0
Vertex B is 1
Vertex C is 2
Edge A--B
Edge A--C
Edge B--C
Total Number of Vertices: 3
Total Number of Edges: 3
Vertex A or 0 is connected with Vertex B or 1
Vertex A or 0 is connected with Vertex C or 2
Vertex B or 1 is connected with Vertex A or 0
Vertex B or 1 is connected with Vertex C or 2
Vertex C or 2 is connected with Vertex A or 0
Vertex C or 2 is connected with Vertex B or 1
Vertex A is coloured 1
Vertex B is coloured 2
Vertex C is coloured 3
sh: 1: PAUSE: not found


...Program finished with exit code 0
Press ENTER to exit console.
```

3.3 Execution #3

Sample 3)



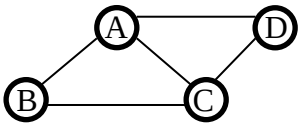
main.cpp greedy_color_grap...

```
<graph>
        <nodes>
                <node name="A"/>
                <node name="B"/>
                <node name="C"/>
                <node name="D"/>
        </nodes>
        <edges>
                <edge src="A" dst="B" />
                <edge src="A" dst="C" />
                <edge src="B" dst="C" />
                <edge src="C" dst="D" />
        </edges>
</graph>
```

```
Vertex A is 0
Vertex B is 1
Vertex C is 2
Vertex D is 3
Edge A--B
Edge A--C
Edge B--C
Edge C--D
Total Number of Vertices: 4
Total Number of Edges: 4
Vertex A or 0 is connected with Vertex B or 1
Vertex A or 0 is connected with Vertex C or 2
Vertex B or 1 is connected with Vertex A or 0
Vertex B or 1 is connected with Vertex C or 2
Vertex C or 2 is connected with Vertex A or 0
Vertex C or 2 is connected with Vertex B or 1
Vertex C or 2 is connected with Vertex D or 3
Vertex D or 3 is connected with Vertex C or 2
Vertex A is coloured 1
Vertex B is coloured 2
Vertex C is coloured 3
Vertex D is coloured 1
sh: 1: PAUSE: not found


...Program finished with exit code 0
Press ENTER to exit console.
```

3.4 Execution #4

Sample 4)



main.cpp greedy_color_gra...

```
<graph>
        <nodes>
                <node name="A"/>
                <node name="B"/>
                <node name="C"/>
                <node name="D"/>
        </nodes>
        <edges>
                <edge src="A" dst="B" />
                <edge src="A" dst="C" />
                <edge src="B" dst="C" />
                <edge src="C" dst="D" />
                <edge src="D" dst="A" />
        </edges>
</graph>
```

```
Vertex A is 0
Vertex B is 1
Vertex C is 2
Vertex D is 3
Edge A--B
Edge A--C
Edge B--C
Edge C--D
Edge D--A
Total Number of Vertices: 4
Total Number of Edges: 5
Vertex A or 0 is connected with Vertex B or 1
Vertex A or 0 is connected with Vertex C or 2
Vertex A or 0 is connected with Vertex D or 3
Vertex B or 1 is connected with Vertex A or 0
Vertex B or 1 is connected with Vertex C or 2
Vertex C or 2 is connected with Vertex A or 0
Vertex C or 2 is connected with Vertex B or 1
Vertex C or 2 is connected with Vertex D or 3
Vertex D or 3 is connected with Vertex C or 2
Vertex D or 3 is connected with Vertex A or 0
Vertex A is coloured 1
Vertex B is coloured 2
Vertex C is coloured 3
Vertex D is coloured 2
sh: 1: PAUSE: not found


...Program finished with exit code 0
Press ENTER to exit console.
```

References

[**1**] **Wikipedia contributors**. "Greedy coloring." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 7 Aug. 2020. Web. 18 Apr. 2021

[**2**] **Sanfoundry.com**, C++ Program to Perform Greedy Coloring, https://www.sanfoundry.com/cpp-program-perform-greedy-coloring/, Web. 18 Apr. 2021

[**3**] **GeeksforGeeks.org**, Graph Coloring - Set 2 (Greedy Algorithm), https://www.geeksforgeeks.org/graph-coloring-set-2-greedy-algorithm/, Web. 18 Apr. 2021

[**4**] **OpenGenus IQ.org**, Graph Coloring Greedy Algorithm [O(V^2 + E) time complexity], https://iq.opengenus.org/graph-colouring-greedy-algorithm/, Web. 18 Apr. 2021

[**4**] **Brilliant.org**, Greedy Algorithms, https://iq.opengenus.org/graph-colouring-greedy-algorithm/, Web. 18 Apr. 2021

[**5**] **Wikipedia contributors**. "Greedy algorithm." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 18 Mar. 2021. Web. 18 Apr. 2021.

[**6**] **Encyclopedia of Mathematics**. Greedy algorithm. https://encyclopediaofmath.org/index.php?title=Greedy_algorithm, Web. 18 Apr. 2021.

[**7**] **OnlineGDB.com**, Online_C++_compiler, https://www.onlinegdb.com/online_c++_compiler, Web. 18 Apr. 2021.

[**8**] **Github.com**, Research-Programming-Work, https://github.com/jeremyjwsc/Research-Programming-Work, Web. 18 Apr. 2021.