**Introduction**

In this report, we will discuss the algorithm we developed for one of the well-known problems in computer engineering which known as “graph coloring”.

This report includes, in order:

* Problem Definition
* Detailed explanation of the algorithm developed for the problem
* References

**Problem Definition**

Coloring all vertices in the graph with the condition that the two adjacent vertices are not the same color. At the same time, while doing this, find the minimum number of colors we can use.

**Algorithm Explanation**

***Inspiration:*** The problem described above is a problem belonging to the “NP-Hard” class which means that it is very hard to find the optimal solution especially for large instances. Therefore, we designed an algorithm based on the **greedy technique** to get the optimal solution in polynomial time.

***Explanation:*** The algorithm consists of two stages. We used the algorithm known as **Welsh-Powell** as a cornerstone in both stage with slight difference.

* **First Stage:** We implement \***Welsh-Powell** algorithm to find acceptable solution for vertex coloring and to find the initial upper bound of the function we will use in the second step, in terms of color number.

\***Welsh-Powell**: it is very effective algorithm that gives an acceptable solution for coloring problem. Since it is based on greedy technique it is not guarantee that it gives the best solution.

*Step-by-Step description of Welsh-Powell:*

1. Find the degree of each vertex in a list.
2. Arrange the vertices in order of descending degree.
3. For each Color X proceed thorough each of vertices in order, if the vertex hasn’t been colored yet and it’s not adjacent to a vertex of color X, assign the same color for the X, else go the next vertex. When the reaching vertex is last vertex, move to the Color X+1 (next color) and start coloring same manner.

*Illustration of algorithm:*

* *Initially all the vertices are not colored. (Programmatically defined as -1)*
* *List all vertices according to their degrees.*
* *Start coloring from highest degree vertex (E or H) according to the algorithm.*

|  |  |
| --- | --- |
| Vertex | Degree |
| E | 4 |
| H | 4 |
| B | 3 |
| D | 3 |
| F | 3 |
| A | 2 |
| C | 2 |
| G | 1 |

* *First iteration*
* *Vertex E, G, B are defined as 0 (which is illustrating red to show clearly)*

|  |  |
| --- | --- |
| Vertex | Degree |
| **E** | **4** |
| H | 4 |
| **B** | **3** |
| D | 3 |
| F | 3 |
| A | 2 |
| C | 2 |
| **G** | **1** |

* *Second iteration*
* *Vertex H and C are defined as 1 (which is illustrating blue to show clearly)*

|  |  |
| --- | --- |
| Vertex | Degree |
| **E** | **4** |
| H | 4 |
| **B** | **3** |
| D | 3 |
| F | 3 |
| A | 2 |
| C | 2 |
| **G** | **1** |

* *Third iteration*
* *Vertex H and C are defined as 1 (which is illustrating blue to show clearly)*

|  |  |
| --- | --- |
| Vertex | Degree |
| **E** | **4** |
| H | 4 |
| **B** | **3** |
| D | 3 |
| F | 3 |
| A | 2 |
| C | 2 |
| **G** | **1** |

* We use 3 different colors in above example. The **chromatic number** is 3 in this specific example.

* **Second Stage:** In the second stage, we use the algorithm used in the first stage by modifying it. Instead of ordering the vertices according to their degrees, we store them randomly and we run algorithm several times to find optimal result among these trials. As the first upper bound we use the optimal value we found in the first stage. In each trial, if the algorithm in the second step gives a better optimal result, the new upper bound is returned to this value. If not, the trial is terminated immediately and continues with the next trial. At the end of the trials, we get optimal solution as possible as we can.

**References**

[*https://www.kleemans.ch/static/fourcolors/welsh-powell.pdf*](https://www.kleemans.ch/static/fourcolors/welsh-powell.pdf) *-* [*https://en.wikipedia.org/wiki/Graph\_coloring#Greedy\_coloring*](https://en.wikipedia.org/wiki/Graph_coloring#Greedy_coloring)