**Abstract**

A greedy coloring algorithm results in a valid solution on a 4x4 Sudoku board where no color is used twice in a sub square, row, or column. However, when the same algorithm is run through a 3x3 Sudoku grid it results in a graph that uses 14 different colors to color all 81 vertices. This results in a non-valid solution of a Sudoku board and reveals that a simple greedy coloring algorithm will not work for a 9x9 Sudoku board.

**Introduction**

The main idea of this paper is to explore multiple graph coloring algorithms and their efficiencies when applied to the puzzle game Sudoku. Graph coloring can be traced back to the problem of coloring the countries of a map in such a way that no two adjacent countries share the same color. A map of countries can be easily translated into a graph by having each country be a vertex known as Vk where k denotes the number of the vertex in the graph. Each border that two countries share can represented as an edge, E between those two countries. This problem of coloring countries led to the famous Four Color Problem which proposed that the number of colors needed to color a planar graph was 4. By recreating a Sudoku board into a graph, the same logic can be applied to color it using nine colors.

**Topic Details**

Graph coloring can be applied to Sudoku easily by creating an undirected graph with 81 total vertices and arranging the vertices from left to right, top to bottom like so:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 |
| 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 |
| 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 |
| 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 |
| 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 |

An edge is placed between two vertices if they are in the same column or row and if they are in the same 3x3 subgraph. For example, V1 will share an edge with V2,V3,V10,V11,V12, V19,V20,V21 because they are in the same 3x3 subgraph and V1 will share an edge with every vertex in the same row and column. Thus the number of adjacent vertices each vertex will have can be represented by the equation, (n-1)+2(n-sqrt(n)) where n is the dimension of the board, 9. For a 9x9 Sudoku board, each vertex will therefore be adjacent with 20 vertices. A graph coloring of a sudoku board will use 9 colors represented by the integers 1-9 and will be colored so that no adjacent vertices will have the same color.

In order to represent this in code, I have created a graph class that relies on the use of the map object from the C++ Standard Library. These map objects will be named vertices, which maps an integer to a vector, and color, which maps an integer to another integer. The vertices map is used to keep an adjacency list of all the vertices adjacent to the Vi and the color map is used to keep a record of each vertices’ color. The constructor for this class works by taking in an integer which is the dimension of the Sudoku grid, 4x4 or 9x9 and initializes the map vertices to find and insert all the adjacent edges to each vertex in the graph.

The simple greedy coloring algorithm works by iterating through each vertex setting int c as 1 and a bool sameColor to true. Then a while loop runs while sameColor is true. The first line of the while loop sets sameColor to false and then a for loop that iterates through a vertex’s adjacency list is used to see if any of vertex in the adjacency list matches that of int c. If any do match, then that color is not used to color the current vertex and c is incremented. As soon as the inside for loop has a pass where c is not found in the adjacency list, the while loop ends, and that vertex is colored c.

**Conclusion**

A greedy coloring algorithm results in a valid solution on a 4x4 Sudoku board where no color is used twice in a sub square, row, or column. However, when the same algorithm is run through a 3x3 Sudoku grid it results in a graph that uses 14 different colors to color all 81 vertices. This results in a non-valid solution of a Sudoku board and reveals that a simple greedy coloring algorithm will not work for a 9x9 Sudoku board.

Future developments planned include the use of a backtracking algorithm that will allow previous vertices that were colored to be altered and changed if the current vertex requires it to do so. This will be used to make my graph coloring algorithm work on a 9x9 sudoku grid. Another planned feature will be to populate an empty Sudoku board with colors scattered around the board to emulate the hints given to a user when trying to solve a board. The algorithm will take this board populated with hints and be able to solve it. Another future development will be to include the option for the user to choose if they want a 4x4 or a 9x9 board. Lastly, timed data will be taken of the algorithms when fully completed.

**Bibliography**

Rioux-Maldague, L. (2014). Graph Coloring Algorithms.

This textbook provides me with some example graph coloring algorithms which I used as a basis for my simple greedy graph coloring algorithm.

Voloshin, Vitaly. (2009). Graph Coloring: History, results and open problems. Alabama Journal of Mathematics.

This source gives me insight into the history of graph coloring problems. It states the beginning of coloring theory as coloring countries on a map which evolved into the Four Color Problem.

**Appendix**