Map Coloring Problem:

Graph-ic Design
MATH 381 HW 3

Tim Ferrin, 1764309

October 23, 2020

I Background Information

What is the minimum number of colors needed to color any map such that no two bordering divisions (countries, states, counties, etc.) are colored the same? According to the four color map theorem (you might have figured out the answer by its name) which was proved in 1976 by Kenneth Apple and Wolfgang Haken, any map only needs at most four different colors to distinguish its neighboring divisions. Some maps might need fewer than four colors, but the most any will need is four. See Figure 1 below for an example of such a map coloring.

This paper focuses on coloring a map of the territories and lands of the vassal houses of the North in Westeros, featured in George R. R. Martin's novels of the *A Song of Ice and Fire* series. An uncolored version of the map used can be seen in Figure 2 below.

II Mathematical Model

Any map can be represented as a graph of vertices/nodes, where each subdivision is a vertex and each pair of regions that share a border is connected by an edge. The map of the North

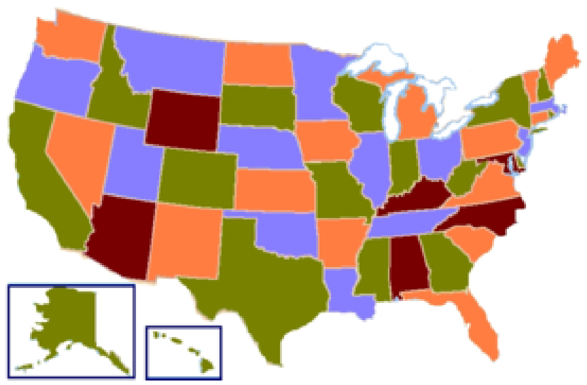
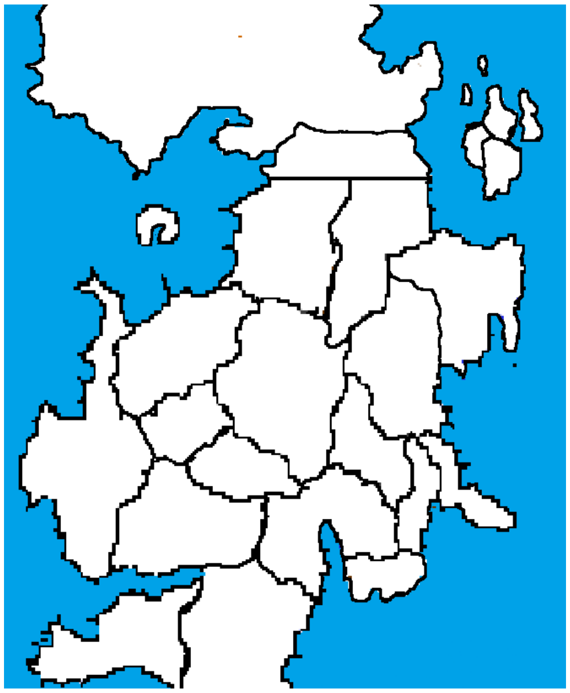


Figure 1: A four-coloring of a map of the states of the United States.



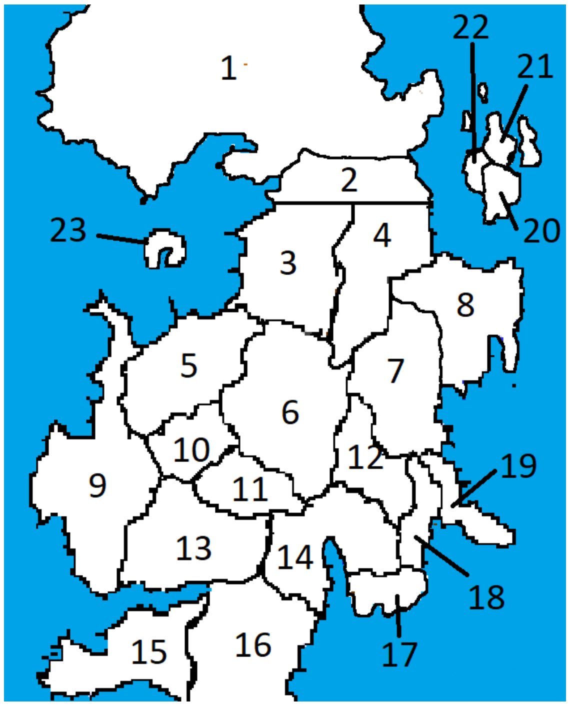


Figure 2: (left) Uncolored vassal lands of the North of Westeros.

Figure 3: (right) The divided regions are now labeled with numbers for graph representation.

can thus be numbered and represented as a graph shown in Figures 3 and 4 respectively.

For those interested, the numbered lands correspond to the following noble houses/groups:

1. The Wildlings and The White Walkers
2. The Night's Watch
3. Assorted (Burley, Harclay, Norrey, Knott, Wull)
4. Umber
5. Glover
6. Stark
7. Bolton
8. Karstark
9. Ryswell
10. Tallhart
11. Cerwyn

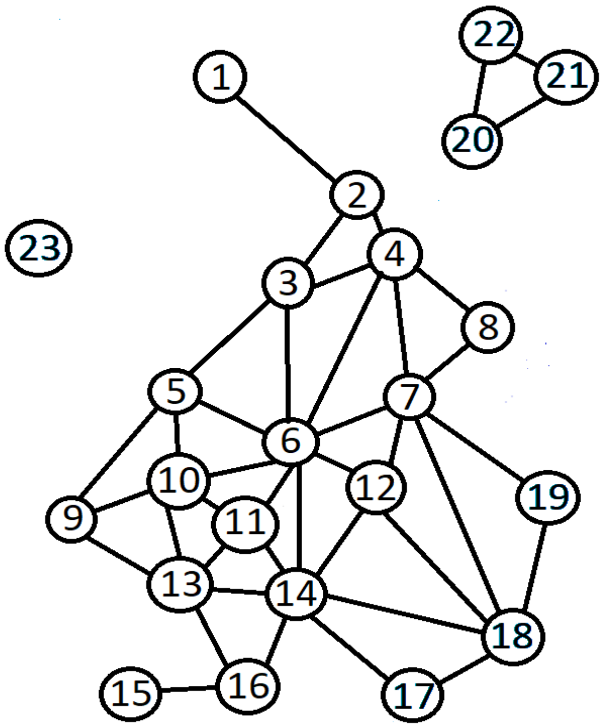


Figure 4: The numbers of the assigned regions are transferred to a graph representation, where bordering lands are connected by an edge.

12. Hornwood
13. Dustin
14. Manderly
15. Flint of Flint's Finger
16. Reed
17. Locke
18. Woolfield
19. Flint of Widow's Watch
20. Magnar
21. Crowl
22. Stane
23. Mormont

This simplifies our map-coloring program into a graph-coloring problem of 23 vertices and 40 edges, with the node/vertex set being

$$V = \{1, 2, 3, 4, 5, ..., 23\}$$

and the edge set being

$$\begin{aligned} E = \{&(1,2),(2,3),(2,4),(3,4),(3,5),(3,6),(4,6),(4,7),(4,8),(5,6),(5,9),(5,10),\\ &(6,7),(6,10),(6,11),(6,12),(6,14),(7,8),(7,12),(7,18),(7,19),(9,10),(9,13),\\ &(10,11),(10,13),(11,13),(11,14),(12,14),(12,18),(13,14),(13,16),(14,16),\\ &(14,17),(14,18),(15,16),(17,18),(18,19),(20,21),(20,22),(21,22)\},\end{aligned}$$

so the overall graph is

$$G = (V, E).$$

It should be noted that the edge $(1, 2)$ is the same as $(2, 1)$ as this is an undirected graph.

III Linear Program

To formulate this graph-coloring problem into a linear program, we will introduce some new variables. If there are n nodes in a graph, then we will need at most n colors to color it, using one color per node. Note that all these colors will not necessarily be used (assuming we have no knowledge of the four color map theorem.)

Let us define a list of variables $c_i \in \{0,1\}, i = 1,2,...,23,$ with $c_i = 1$ if color number i is used in coloring the graph and $c_i = 0$ if it is not.

Let us also define $x_{ik} \in \{0,1\}, i = 1,2,...,23, k = 1,2,...,23,$ with $x_{ik} = 1$ if color k is used to color vertex i and $x_{ik} = 0$ if it is not.

Since we are trying to minimize the number of colors we need to use, our objective function is just the sum of all c variables,

$$\sum_{i=1}^{23} c_i \tag{1}$$

Our first constraint will come from the fact that each vertex must be exactly one color. Formulated mathematically this is

$$\sum_{k=1}^{23} x_{ik} = 1, \quad i = 1,2,...,23. \tag{2}$$

Our second constraint can be thought of as the correlation between the x and c variables. If $x_{ik} = 0$, then c_k can either be 0 or 1, since we don't know if color k is used for other vertices. If $x_{ik} = 1$, then $c_k = 1$ because a vertex cannot be colored by an unused color. Formulated mathematically this is

$$x_{ik} \leq c_k, \quad i,k = 1,2,...,23. \tag{3}$$

Our third constraint is where we specify that no two lands in the map that share a border can be the same color, so in our graph, no two nodes that form an edge can have the same color. Formulated mathematically this is

$$x_{ik} + x_{jk} \leq 1 \quad \text{for all } (v_i, v_j) \in E \text{ and } k = 1,2,...,23. \tag{4}$$

Our fourth constraint is that we won't use color two if we don't use color one, we won't use color three if we don't use color two, etc. Note that this not necessary for the LP, but cuts down on the number of solution states and thus can speed up calculation time. Formulated mathematically this is

$$c_k \leq c_{k-1}, \quad k = 2,3,...,23. \tag{5}$$

Our final constraint is that all of the x and c variables must be binary. Formulated mathematically this is

$$x_{ik}, c_k \in \{0,1\}, \quad i = 1,2,...,23, k = 1,2,...,23. \tag{6}$$

Thus our summarizing LP is the following:

Minimize

$$\sum_{i=1}^{23} y_i$$

subject to

$$\begin{aligned} \sum_{k=1}^{23} x_{ik}, &\quad i = 1, 2, ..., 23 \\ x_{ik} \leq c_k, &\quad i, k = 1, 2, ..., 23 \\ x_{ik} + x_{jk} \leq 1 &\quad \text{for all } (v_i, v_j) \in E \text{ and } k = 1, 2, ..., 23 \\ c_k \leq c_{k-1}, &\quad k = 2, 3, ..., 23 \\ x_{ik}, c_k \in \{0, 1\}, &\quad i = 1, 2, ..., 23, k = 1, 2, ..., 23 \end{aligned}$$

IV Code

The code to generate the linear program, which helpfully specifies which nodes are colored with what number, was written in MATLAB. This linear program was written to an .lp file which actually solves the LPs. The commented MATLAB code is written below.

```
% Below is an adjacency matrix where a 1 in row i column j signifies that
% there is an edge between nodes i and j
Adj = [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
       1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
       0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
       0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
       0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0;
       0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0;
       0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0;
       0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
       0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0;
       0 0 0 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0;
       0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 1 0 0 0 0 0 0 0;
       0 0 0 0 0 1 0 0 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0;
       0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0;
       0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0;
       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0;
       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0;
       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
n = 23;
```

```
nodes = 1:23;
fileID = fopen(['Graph_Coloring_North.lp'], 'w');

% Prints the objective function
fprintf(fileID, 'min: ');
for k=1:(n-1) % For each color
    fprintf(fileID, ['c_',num2str(k),' + ']);
end
fprintf(fileID, ['c_',num2str(n),';\n']);

% Ensures all vertices are colored with exactly one color
for i=1:n % For each node
    for k=1:(n-1) % For each color
        fprintf(fileID, ['x_',num2str(i),'_',num2str(k),' + ']);
    end
    fprintf(fileID, ['x_',num2str(i),'_',num2str(n),' = 1;\n']);
end

% Ensures a vertex cannot be colored with an unused color
for i=1:n % For each node
    for k=1:n % For each color
        fprintf(fileID, ['x_',num2str(i),'_',num2str(k),' <= c_',num2str(k),';\n']);
    end
end

% Ensures that adjacent vertices have different colors
for i=1:n % For each node
    % Next line removes the redundant edge constraints (not necessary)
    triAdj = triu(Adj);
    cons = nodes(triAdj(i,:)==1);
    for j=1:length(cons) % For each connected node
        for k=1:n % For each color
            fprintf(fileID, ['x_',num2str(i),'_',num2str(k),' + x_',num2str(cons(j)),'_'
        end
    end
end

% Ensures that we don't use color 2 if we don't use color 1, we don't use
% color 3 if we don't use color 2, etc. This constraint is not necessary
% but often speeds up computation time
for k=2:n % For each color
    fprintf(fileID, ['c_',num2str(k),' <= c_',num2str(k-1),';\n']);
end

% Ensures all variables are binary
```

```
fprintf(fileID, 'bin ');
for i=1:n % For each node
    for k=1:n % For each color
        fprintf(fileID, ['x_',num2str(i),'_',num2str(k),', ']);
    end
end
for k=1:(n-1) % For each color
    fprintf(fileID, ['c_',num2str(k),', ']);
end
fprintf(fileID, ['c_',num2str(n),';']);
```

V LPSolve Input File

The following is the LPSolve input file for this graph coloring problem:

```
min: c_1 + c_2 + ... + c_23;
(23 lines of the following type:
ensure that exactly 1 color is used for each vertex)
x_1_1 + x_1_2 + ... + x_1_23 = 1;
.
.
x_23_1 + x_23_2 + ... + x_23_23 = 1;
(23^2 = 529 lines of the following type:
ensure that each vertex is colored by a color that is being used)
x_1_1 <= c_1;
.
.
x_23_23 <= c_23;
(23*40 = 920 lines of the following type:
ensure that each pair of connected vertices are different colors)
x_1_1 + x_2_1 <= 1;
.
.
x_21_23 + x_22_23 <= 1;
(22 lines of the following type:
ensure that we if we don't use color i, we also don't use color i+1)
c_2 <= c_1;
.
.
c_23 <= c_22;
(ensure all variables are binary)
bin x_1_1, x_1_2,...,c_23;
```

VI LPSolve Output and Results

The following is the output for the LP:

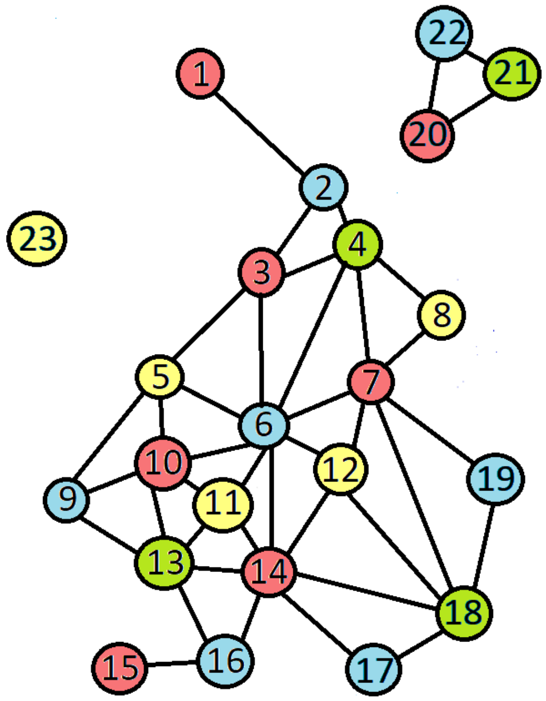
```
Value of objective function: 4.00000000

Actual values of the variables:
c_1                             1
c_2                             1
c_3                             1
c_4                             1
x_1_1                           1
x_2_2                           1
x_3_1                           1
x_4_3                           1
x_5_4                           1
x_6_2                           1
x_7_1                           1
x_8_4                           1
x_9_2                           1
x_10_1                          1
x_11_4                          1
x_12_4                          1
x_13_3                          1
x_14_1                          1
x_15_1                          1
x_16_2                          1
x_17_2                          1
x_18_3                          1
x_19_2                          1
x_20_1                          1
x_21_3                          1
x_22_2                          1
x_23_4                          1
```

All other variables are 0.

The solution shows that four colors are needed and corresponds to a colored graph and map as seen in Figures 5 and 6.

Considering only the subgraph pictured in Figure 7, it's clearly evident that four colors are needed to color such an arrangement. As this is a portion of the overall graph, at least four



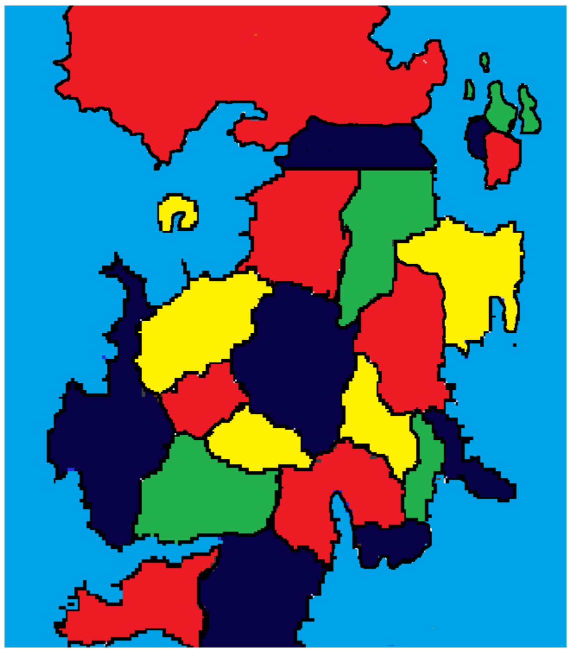


Figure 5: (left) Colored in graph.

Figure 6: (right) Colored in map (light blue is the ocean).

colors are also needed for the map coloring.

A previous iteration of the MATLAB code additionally created lines in the LPSolve input file for the symmetric, repeated edges (e.g. $(1, 2)$ and $(2, 1)$). This resulted in 920 less constraint rows and a different output coloring, where the colors for nodes 11 and 12 switched with the colors for 13 and 18. This exemplifies the fact that there is often more than one way to color a graph using it's minimum number of colors.

VII References

As stated, all maps and names come from cannon/partially cannon sources based on George R. R. Martin's series *A Song of Ice and Fire*

Figure 1 is from the Wikipedia page on the four color map theorem, which can be found here

The detailed map of the North and the associated noble houses before *A Game of Thrones* was taken from this Quora page (unable to track down its original source.)

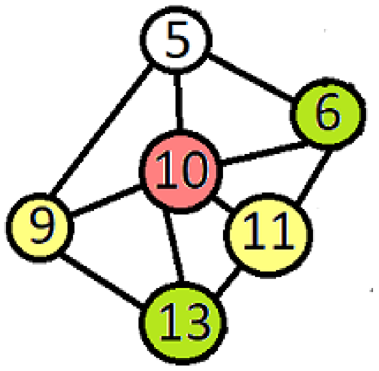


Figure 7: An (unsuccesful) attempt to color a subgraph of the graphs from Figures 4 and 5 with only three colors. Since all vertices in each triangle must all be different colors, all the triangles share two vertices with each of their neighboring triangles, and there are 5 triangles in a ring, vertex 5 must be colored with a fourth color.