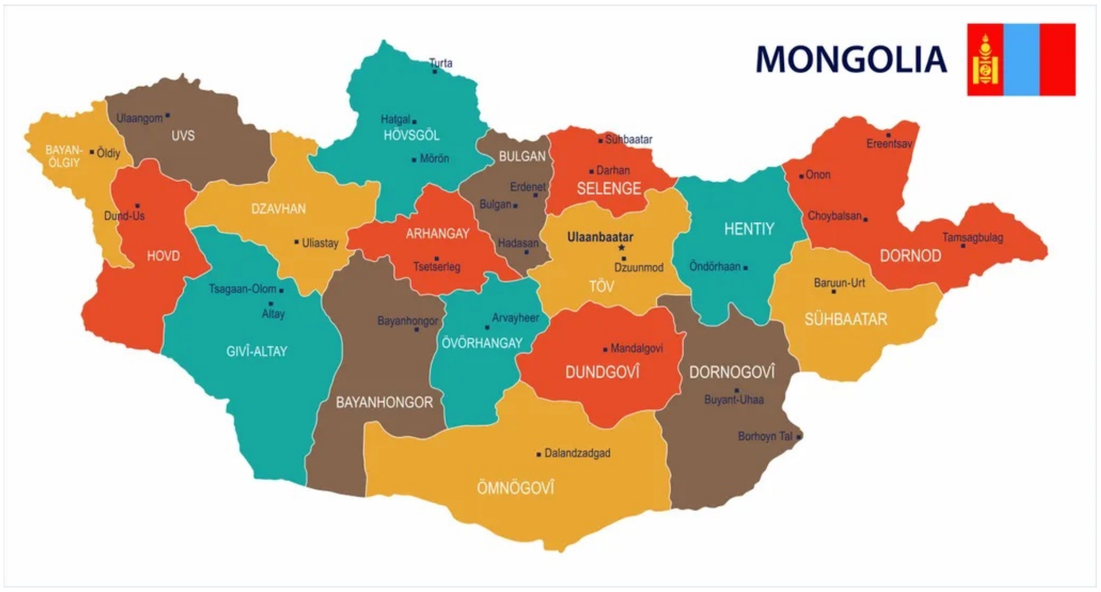
Math381 Assignment 3

Yuchen AN

June 28, 2023

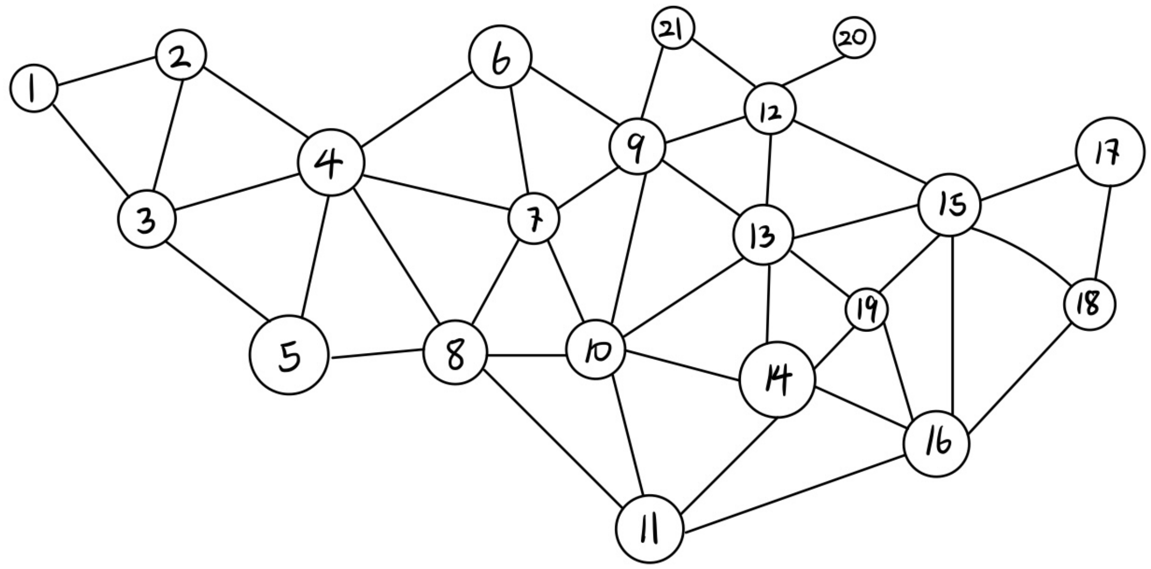
For assignment 3, we try to solve a coloring problem of a real world map. The coloring problem refers to graph coloring, which is the procedure of assignment of colors to each vertex of a graph G such that no adjacent vertices get same color. By solving such problem, we can find out what is the minimum number of colors needed to color the real world map and the way to color the graph with that number of colors.

The map I will be coloring is the map of Mongolia showing subdivisions.



Resources: https://www.mappr.co/political-maps/mongolia/

I draw a graph G that resembles the layout of the map. The vertices are corresponding to each region in the map. The edges between each pair of vertices show if the corresponding regions share a border or not.



I made a table including the vertex label and corresponding region to help reader get better understanding of this graph coloring.

1	Bayan-Ölgii	2	UVS	3	HOVD	4	DZAVHAN
5	GIVI-ALTAY	6	HOVSGOL	7	ARHANGAY	8	BAYANHONGOR
9	BULGAN	10	OVORHANGAY	11	OMNOGOVI	12	SELENGE
13	TOV	14	DUNDGOVI	15	HENTIY	16	DORNOGOVI
17	DORNOD	18	SUHBAATAR	19	CHOIR	20	OARKHAN
21	ERDENET	-	-	-	-	-	-

The graph G = (V,E) with 21 vertices. V = $\{v_1, ..., v_{21}\}$. We know that the smallest number of needed colors, the chromatic number of the graph, is less than or equal to the number of vertices of G. However, We might or might not use all of the colors. Based on the four color theorem, the map of Mongolia can be colored using four or fewer colors. Let y_k be a specific color, 1 $\leq$ k $\leq$4. y_k is a binary variable which indicate whether or not the color k is being used. Let x_{ik} be a binary variable which indicate whether or not the vertex i have color k,1 $\leq$ i $\leq$21, 1 $\leq$ k $\leq$4..

$$y_k = 1 \text{ if the color i is in use.}$$
$$y_k = 0 \text{ if the color i is not in use.}$$
$$x_{ik} = 1 \text{ if the vertex i have color k.}$$
$$x_{ik} = 0 \text{ if the vertex i doesn't have color k.}$$

Here is the mathematical formulation of LP I am going to solve:
We want to find the minimum number of colors to color the graph G and meet all constraints . By four color theorem, we can color the graph G with four or fewer colors.
The objective function is :

$$\text{Minimize:} \sum_{k=1}^{4} y_k$$

Subject to constraints:
1.All vertices need to be colored with exactly one color. For example, if vertex 1 is colored with color

1, other colors cannot be used to color vertex 1. Therefore, $x_{11} = 1$. $x_{1k} = 0$, where k $\neq$ 1 and 2 $\leq$ k $\leq$4. Thus, $\sum_{k=1}^{4} x_{1k} = 1$.
We have the constraint:

$$\sum_{k=1}^{4} x_{ik} = 1, i = 1, ..., 21$$

2. A vertex cannot be colored with an unused colored. To be specific, if we color vertex i with color k, then we are using color k. For example, if $x_{11} = 1$, $y_1 = 1$. If $x_{11} = 0$, y_1 can be 0 or 1. Color 1 may be not be used or it is used to color vertex other than 1. Therefore, $x_{11} \leq y_1$.
We have the constraint:

$$x_{ik} \leq y_k, i = 1, ..., 21, k = 1, ..., 4$$

3. Adjacent vertices in the graph must have different colors. Vertices are adjacent if two vertices in a graph are connected by an edge. If there is an edge between v_i and v_j, then x_{ik} and x_{jk} cannot both be equal to one for any color k.
We have the constraint:

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

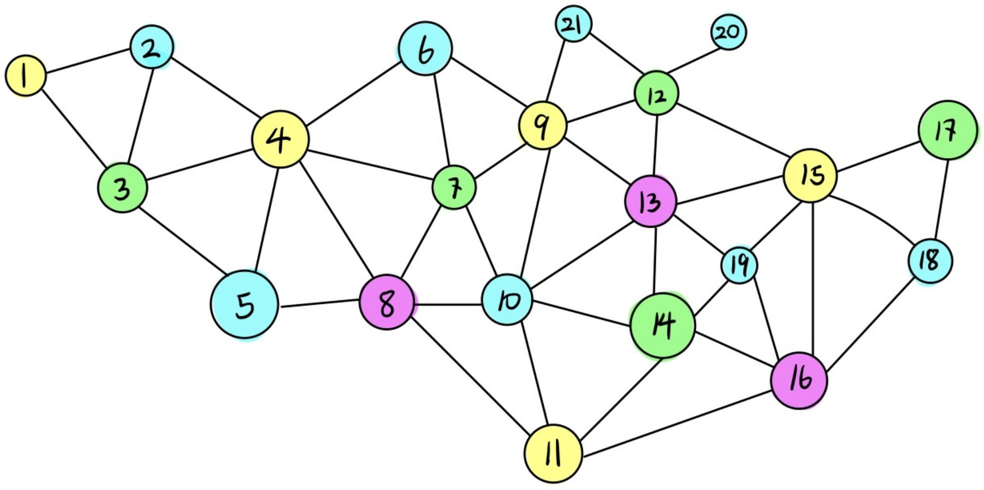
4. To improve the speed of computation, we add another constraint which is not necessary. This constraint says that we will not use color 2 if we do not use color 1. We will not use color 3 if we do not use color 2. This constraint reduces the computation time by reducing the combination of colors need to be checked.
We have the constraint:

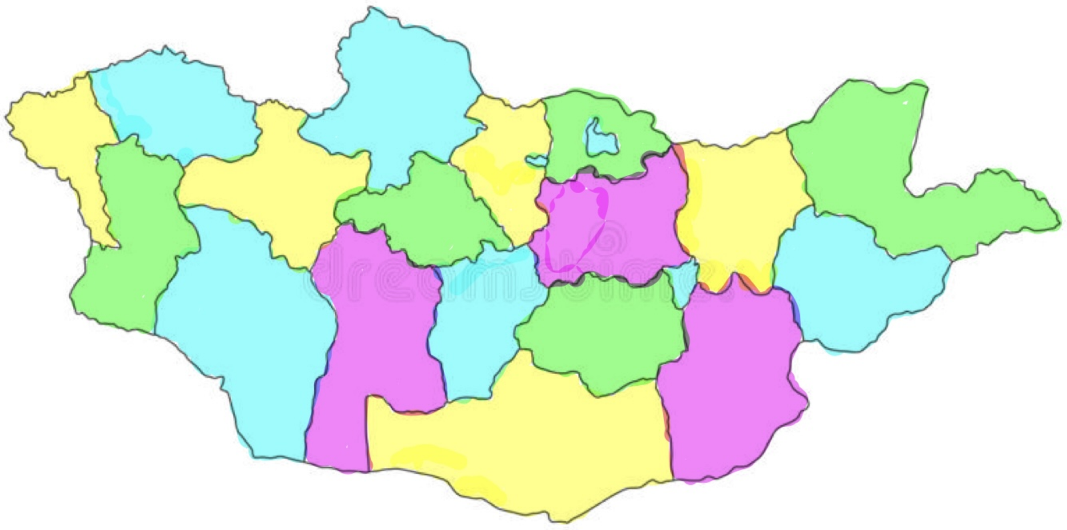
$$y_k \leq y_{k-1}, \text{ k=2,...,4}$$

The complete mathematical formulation of LP looks like this:

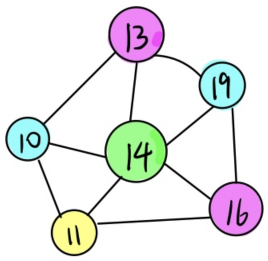
$$\begin{gathered}
\text{Minimize:} \sum_{k=1}^{4} y_k \\
\text{Subject to:} \sum_{k=1}^{4} x_{ik} = 1, i = 1, ..., 21 \\
x_{ik} \leq y_k, i = 1, ..., 21, k = 1, ..., 4 \\
x_{ik} + x_{jk} \leq 1 \text{ for all}(v_i,\ v_j) \in \text{E and k=1,...,4} \\
y_k \leq y_{k-1}, \text{ k=2,...,4} \\
x_{ik}, y_k \in \{0, 1\}, 1 \leq i \leq 21, 1 \leq k \leq 4
\end{gathered}$$

The code for generating the lpsolve input file, lpsolve input file and output can be found in the appendix section. The optimal value for the objective function is 4. We need 4 different colors such that each vertex of graph G has no adjacent vertices with same color. Using this result, we color the map and graph as follows:





Check:
The solution for this coloring problem is 4. We find a sub-graph with center vertex 14. Vertex 14 connecting 5 adjacent vertex forms a wheel graph. We call this wheel graph W_5. This wheel graph need to be colored with 4 colors to meet all constraints. That is $x(W_5) = 4$. We can not color W_5 with 3 colors. To be specific, vertex 14, 19, 16 must have different colors because they form a triangle. Then, vertex 13 can have the same color as vertex 16. Vertex 10 can have the same color as vertex 19. However, vertex 11 must have different color with vertex 10, 14, 16. Thus, four colors is the minimum. Our solutions are valid.
Colored W_5 looks like this:



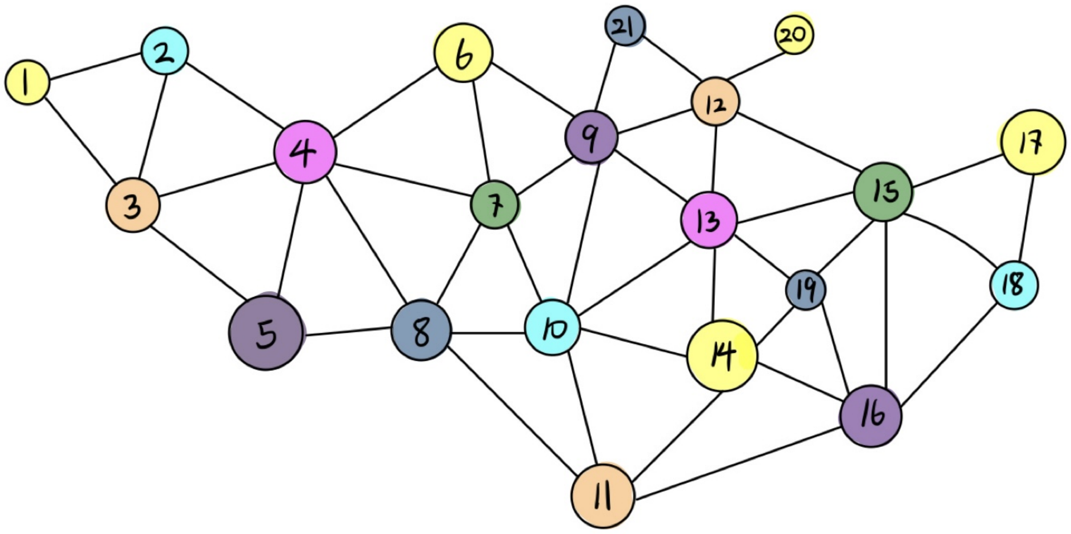
Additional constraint:
Two regions that border the same region cannot be colored the same. To be specific, if two regions both have a border in common with a third region, they cannot be given the same color. We add a constraint that said if two regions both share border with same third region, they cannot be assigned to the same color. Basically, we just add edge between two vertices that are connected to the same vertex.
We have the constraint:

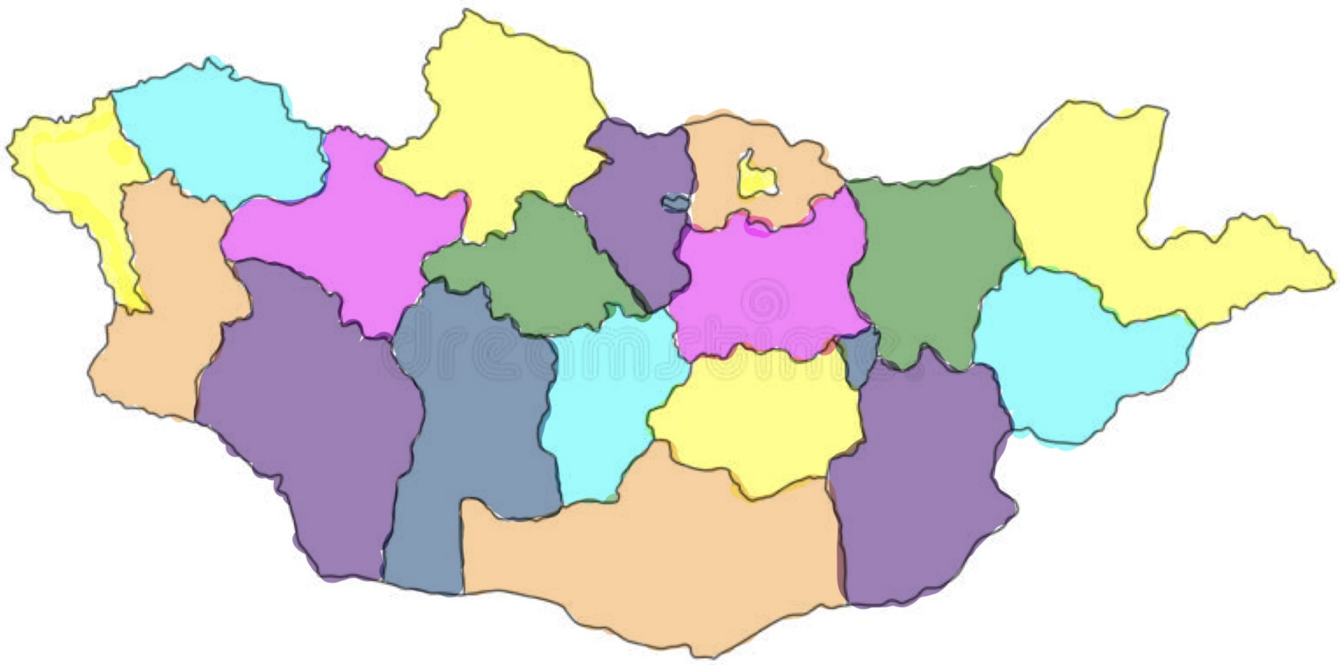
$$x_{ik} + x_{jk} \leq 1 \text{ for all}(v_i,\, v_m),\, (v_m,\, v_j) \in \text{E and k=1,...,n}$$

We apply the same LP in the previous part and add this new constraint. In coding part, we just add some new edges. We find that vertex 13 have the most edges 6. We know the chromatic number of the graph, x(G), is equal to the largest degree of the vertices in G plus 1. That is x(G) $= \triangle(G)+1$, where $\triangle(G)$ represents the largest degree of the vertices in G. Therefore, we now need at least 7 colors to meet the additional constraint. We modified code for generating the lpsolve input file. Also, to speed up the computation, we label three assignment lines. Since vertex 1, 2, 3 must be different color, we label those three variables.
For example, part of the LP could be written:

```
...
x_1_1 = 1
x_2_2 = 1
X_3_3 = 1
...
```

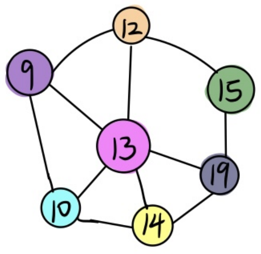
Using the results from Lpsolve, we color the map and graph as follows:





Check:
The solution for this coloring problem is 7.The sub-graph with center vertex 13 connects 6 adjacent

vertices forms a wheel graph. We call this graph W_6. 6 adjacent vertices of vertex 13 must have different colors with each other. Vertex 13 has different color with others. So we cannot color W_6 with 6 colors. It is obvious that the wheel graph colored with 7 colors can meet all constraints. Thus, seven colors are the minimum. Our solutions are valid.
Colored W_6 looks like this:



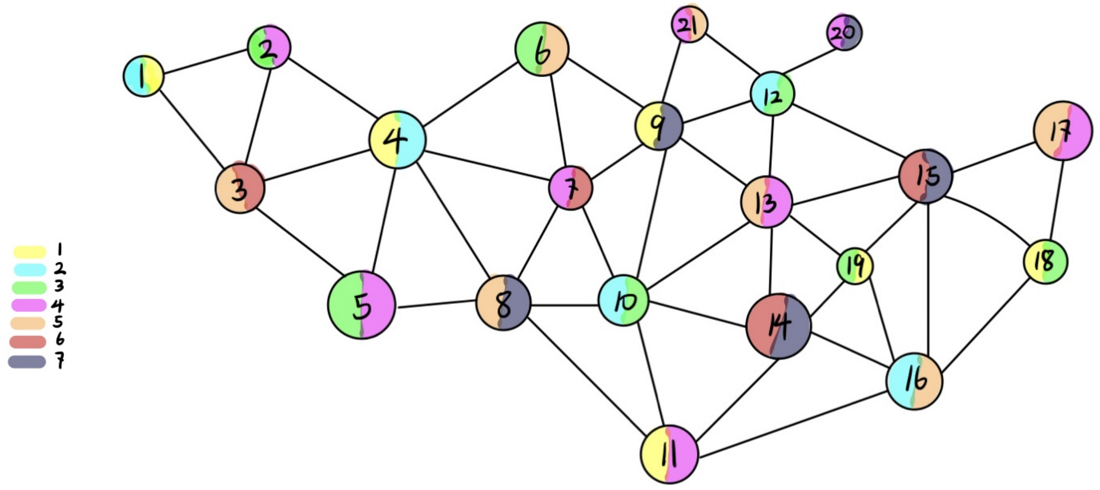
Assign to colors to each region:
Now, we want to assign two colors to each region such that adjacent regions share no colors without the additional constraint mentioned above. To do this, we modify the first constraint on original LP. The new constraint looks like this:

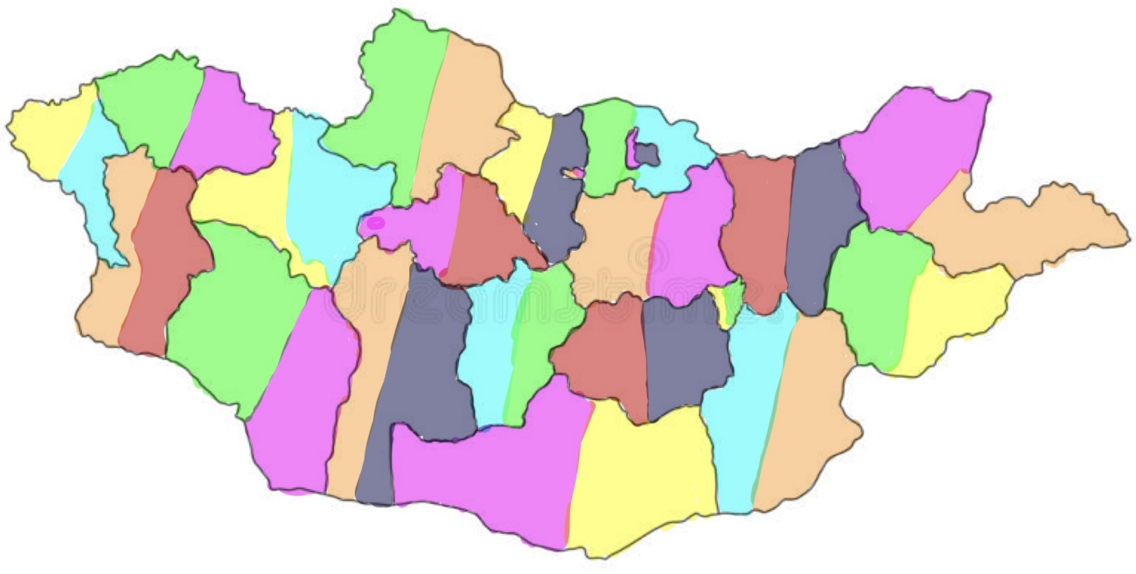
$$\sum_{k=1}^{n} x_{ik} = 2, \text{ i = 1,..., 21}$$

By four color theorem, the solution for originial LP is 4. We now assign two colors to each region. The chromatic number of the current graph can be at most as the chromatic number of the original graph. That is $x_2(G) \leq 2x(G) = 8$.
The complete new mathematical formulation of LP looks like this:

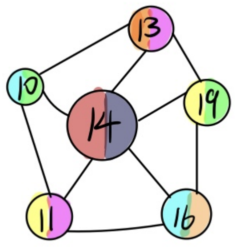
$$\begin{gathered}
\text{Minimize:} \sum_{k=1}^{n} y_k \\
\text{Subject to:} \sum_{k=1}^{n} x_{ik} = 2, i = 1, ..., 21 \\
x_{ik} \leq y_k, i = 1, ..., 21, k = 1, ..., n \\
x_{ik} + x_{jk} \leq 1 \text{ for all} (v_i,\ v_j) \in \text{E and k=1,...,n} \\
y_k \leq y_{k-1}, \text{ k=2,...,n} \\
x_{ik}, y_k \in \{0, 1\}, 1 \leq i \leq 21, 1 \leq k \leq n
\end{gathered}$$

The optimal solution that Lpsolver give is 7, which means that we only need 7 colors to color the graph G with new constraints. Using the result from Lpsolve, we color the graph and map as follows:





Check:
Vertex 14 connecting 5 adjacent vertex forms a wheel graph. We call this wheel graph W_5. This wheel graph need to be colored with 7 colors to meet all constraints. That is $x_2(W_5) = 7$. We can not color W_5 with 6 colors. Vertex 19, 13, 10 are colored respectively by $(1,3),(4,5),(2,3)$. Vertex 14 connects with vertex 10, 13, 19. So vertex 11 cannot have color 1,2,3,4,5. We need two new colors to color vertex 14. Thus, seven colors are the minimum. Our solutions are valid.
Colored W_6 looks like this:



Appendix:

```
Python code for part 1:
# The following code generates a lpsolve input file.
# This file determines the what is the minimum number of colors needed to color the real world map.

import math

# define edges in the graph which show if the corresponding regions share a border or not.
A = [[1,2],[1,3],[2,3],[2,4],[3,4],[3,5],[4,5],[4,6],[4,7],[4,8],[5,8],[6,7],[6,9],[7,9],
     [7,8],[7,10],[8,10],[8,11],[9,12],[9,10],[9,13],[9,21],[10,13],[10,14],[10,11],
     [11,14],[11,16],[12,13],[12,15],[12,21],[12,20],[13,15],[13,14],[13,19],
     [14,16],[14,19],[15,16],[15,17],[15,18],[15,19],[16,18],[16,19],[17,18]]

#### generate lpsolve input file
# cn means color number.
# rn means number of vertices.
cn = 4 + 1
rn = 21+1
# objective function
fullString = ""
for a in range(1,cn):
    fullString += "+y"+"_"+str(a)
print("min: "+fullString+";")

## constraint : all vertices need to be colored  with exactly one color.
for i in range(1,rn):
    one_color = ""
    for j in range(1,cn):
        one_color += "+x"+"_"+str(i)+"_"+str(j)
    print(one_color+"="+"1"+";")

## constraint: a vertex cannot be colored with an unused color.
for i in range(1, rn):
    for j in range(1,cn):
        used_color = ""
        used_color = "+x"+"_"+str(i)+"_"+str(j) + "<=" + "+y"+"_"+str(j)
        print(used_color+";")

## constraint: adjacent vertices in the graph must have different colors.
for i in range(1,cn):
    for j in A:
        adjacent_vertex = ""
        adjacent_vertex = "+x"+"_"+str(j[0])+"_"+str(i)+"+x"+"_"+str(j[1])+"_"+str(i)
        print(adjacent_vertex+"<="+"1"+";")


## constraint: To improve the speed of computation.
for i in range(2,cn):
    improve_com = ""
    improve_com = "+y"+"_"+str(i)+"<="+"+y"+"_"+str(i-1)
    print(improve_com+";")

# declare all varaibles as binary
binString = "bin "
for j in range(1,cn):
    for i in range(1, rn):
        binString += "x"+"_"+str(i)+"_"+str(j)+","
for i in range(1,cn):
    binString += "y" + "_"+str(i)+","
print(binString+";")

The input file for part1 looks like this:
min: +y_1+y_2+y_3+y_4;
(21 lines of the following type:
ensure all vertices need to be colored  with exactly one color.)
+x_1_1+x_1_2+x_1_3+x_1_4=1;
.
.
+x_21_1+x_21_2+x_21_3+x_21_4=1;
(84 lines of the following type:
ensure a vertex cannot be colored with an unused color.)
+x_1_1<=+y_1;
+x_1_2<=+y_2;
.
.
+x_21_3<=+y_3;
+x_21_4<=+y_4;
(172 lines of the following type:
ensure adjacent vertices in the graph must have different colors.)
+x_1_1+x_2_1<=1;
+x_1_1+x_3_1<=1;
.
.
+x_16_4+x_19_4<=1;
+x_17_4+x_18_4<=1;
(To speed up computation)
+y_2<=+y_1;
+y_3<=+y_2;
+y_4<=+y_3;
(declare all varaibles as binary)
bin x_1_1,x_2_1,...,x_20_4,x_21_4,y_1,y_2,y_3,y_4;

The output file for part1 looks like this:
Value of objective function: 4.00000000

Actual values of the variables:
y_1                             1
y_2                             1
```

```
y_3                            1
y_4                            1
x_1_1                          1
x_2_2                          1
x_3_3                          1
x_4_1                          1
x_5_2                          1
x_6_2                          1
x_7_3                          1
x_8_4                          1
x_9_1                          1
x_10_2                         1
x_11_1                         1
x_12_3                         1
x_13_4                         1
x_14_3                         1
x_15_1                         1
x_16_4                         1
x_17_3                         1
x_18_2                         1
x_19_2                         1
x_20_2                         1
x_21_2                         1

All other variables are zero.
```

Python code for part 2:

```
# The following code generates a lpsolve input file.
# This file determines the what is the minimum number of colors needed to color the real world map.

import math

# define edges in the graph which show if the corresponding regions
share a border or not. I add more edges.
A = [[1,2],[1,3],[2,3],[2,4],[3,4],[3,5],[4,5],[4,6],[4,7],[4,8],[5,8],
[6,7],[6,9],[7,9],[7,8],[7,10],
[8,10],[8,11],[9,12],[9,10],[9,13],[9,21],[10,13],[10,14],
[10,11],[11,14],[11,16],[12,13],[12,15],[12,21],[12,20],[13,15],[13,14],
[13,19], [14,16],[14,19],[15,16],[15,17],[15,18],[15,19],
[16,18],[16,19],[17,18],[12,21],[9,21],[12,20],[16,21],
[9,19],[12,19],[12,20],[1,4],[1,5],[2,5],[2,8],[2,6],
[2,7],[3,8],[3,6],[3,7],[4,9],[4,10],[4,11],[5,10],[5,7],
[5,11],[5,6],[6,21],[6,12],[6,13],[6,10],[6,8],[7,21],[7,12],
[7,13],[7,11],[7,14],[8,13], [8,9],[8,14],[8,16],[9,20],
[9,19],[9,11],[9,14],[9,15],[10,21],[10,12],[10,15],[10,19],
[10,16], [11,13],[11,19],[11,18],[11,15],[12,18],[12,17],
[12,16],[12,14],[12,19],[13,17],[13,18],[13,16],[13,20],
[13,21],[14,15],[14,18],[15,20],[15,21],[16,17],[17,19],[18,19],[20,21]]

#### generate lpsolve input file
cn = 7 + 1
rn = 21+1
# objective function
fullString = ""
for a in range(1,cn):
    fullString += "+y"+"_"+str(a)
```

```
print("min: "+fullString+";")

## constraint: all vertices need to be colored  with exactly one color.
for i in range(1,rn):
    one_color = ""
    for j in range(1,cn):
        one_color += "+x"+"_"+str(i)+"_"+str(j)
    print(one_color+"="+"1"+";")

## constraint: a vertex cannot be colored with an unused color.
for i in range(1, rn):
    for j in range(1,cn):
        used_color = ""
        used_color = "+x"+"_"+str(i)+"_"+str(j) + "<=" + "+y"+"_"+str(j)
        print(used_color+";")

## constraint: adjacent vertices in the graph must have different colors.
for i in range(1,cn):
    for j in A:
        adjacent_vertex = ""
        adjacent_vertex = "+x"+"_"+str(j[0])+"_"+str(i)+"+x"+"_"+str(j[1])+"_"+str(i)
        print(adjacent_vertex+"<="+"1"+";")


## constraint: To improve the speed of computation.
for i in range(2,cn):
    improve_com = ""
    improve_com = "+y"+"_"+str(i)+"<="+"+y"+"_"+str(i-1)
    print(improve_com+";")

# declare all varaibles as binary
binString = "bin "
for j in range(1,cn):
    for i in range(1, rn):
        binString += "x"+"_"+str(i)+"_"+str(j)+","
for i in range(1,cn):
    binString += "y" + "_"+str(i)+","
print(binString+";")

The input file for part2 looks like this:
min: +y_1+y_2+y_3+y_4+y_5+y_6+y_7;
x_1_1 = 1;
x_2_2 = 1;
X_3_3 = 1;
(21 lines of the following type:
ensure all vertices need to be colored  with exactly one color.)
+x_1_1+x_1_2+x_1_3+x_1_4+x_1_5+x_1_6+x_1_7=1;
.
.
+x_21_1+x_21_2+x_21_3+x_21_4+x_21_5+x_21_6+x_21_7=1;
(146 lines of the following type:
ensure a vertex cannot be colored with an unused color.)
+x_1_1<=+y_1;
+x_1_2<=+y_2;
.
.
```

```
+x_21_6<=+y_6;
+x_21_7<=+y_7;
(784 lines of the following type:
ensure adjacent vertices in the graph must have different colors.)
+x_1_1+x_2_1<=1;
+x_1_1+x_3_1<=1;
.
.
+x_18_7+x_19_7<=1;
+x_20_7+x_21_7<=1;
(6 lines of the following type:
 to speed up computation)
+y_2<=+y_1;
.
.
+y_7<=+y_6;
(declare all varaibles as binary)
bin x_1_1,x_2_1,...,x_19_7,x_20_7,x_21_7,y_1,...,y_7;

The output file for part2 looks like this:
Value of objective function: 7.00000000

Actual values of the variables:
y_1                             1
y_2                             1
y_3                             1
y_4                             1
y_5                             1
y_6                             1
y_7                             1
x_1_1                           1
x_2_2                           1
x_3_3                           1
x_4_4                           1
x_5_5                           1
x_6_1                           1
x_7_6                           1
x_8_7                           1
x_9_5                           1
x_10_2                          1
x_11_3                          1
x_12_3                          1
x_13_4                          1
x_14_1                          1
x_15_6                          1
x_16_5                          1
x_17_1                          1
x_18_2                          1
x_19_7                          1
x_20_2                          1
x_21_7                          1

All other variables are zero.

Python code for part 3:
# The following code generates a lpsolve input file.
# This file determines the what is the minimum number of colors needed to color the real world map.
import math

# define edges in the graph which show if the corresponding regions share a border or not.
A = [[1,2],[1,3],[2,3],[2,4],[3,4],[3,5],[4,5],[4,6],[4,7],[4,8],
[5,8],[6,7],[6,9],[7,9],[7,8],[7,10],
[8,10],[8,11],[9,12],[9,10],[9,13],[9,21],[10,13],[10,14],
[10,11],[11,14],[11,16],[12,13],[12,15],[12,21],[12,20],[13,15],[13,14],
[13,19], [14,16],[14,19],[15,16],[15,17],[15,18],[15,19],[16,18],[16,19],[17,18]]

#### generate lpsolve input file
cn = 8 + 1
rn = 21+1
# objective function
fullString = ""
for a in range(1,cn):
    fullString += "+y"+"_"+str(a)
print("min: "+fullString+";")

## constraint: all vertices need to be colored  with two colors.
for i in range(1,rn):
    one_color = ""
    for j in range(1,cn):
        one_color += "+x"+"_"+str(i)+"_"+str(j)
    print(one_color+"="+"2"+";")

## constraint: a vertex cannot be colored with an unused color.
for i in range(1, rn):
    for j in range(1,cn):
        used_color = ""
        used_color = "+x"+"_"+str(i)+"_"+str(j) + "<=" + "+y"+"_"+str(j)
        print(used_color+";")

## constraint: adjacent vertices in the graph must have different colors.
for i in range(1,cn):
    for j in A:
        adjacent_vertex = ""
        adjacent_vertex = "+x"+"_"+str(j[0])+"_"+str(i)+"+x"+"_"+str(j[1])+"_"+str(i)
        print(adjacent_vertex+"<="+"1"+";")


## constraint: To improve the speed of computation.
for i in range(2,cn):
    improve_com = ""
    improve_com = "+y"+"_"+str(i)+"<="+"+y"+"_"+str(i-1)
    print(improve_com+";")

# declare all varaibles as binary
binString = "bin "
for j in range(1,cn):
    for i in range(1, rn):
        binString += "x"+"_"+str(i)+"_"+str(j)+","
for i in range(1,cn):
    binString += "y" + "_"+str(i)+","
print(binString+";")
```

```
The input file for part3 looks like this:
min: +y_1+y_2+y_3+y_4+y_5+y_6+y_7+y_8;
(ensure all vertices need to be colored  with exactly one color.)
+x_1_1+x_1_2+x_1_3+x_1_4+x_1_5+x_1_6+x_1_7+x_1_8=2;
.
.
+x_21_1+x_21_2+x_21_3+x_21_4+x_21_5+x_21_6+x_21_7+x_21_8=2;
(ensure a vertex cannot be colored with an unused color.)
+x_1_1<=+y_1;
.
.
+x_21_8<=+y_8;
(ensure adjacent vertices in the graph must have different colors.)
+x_1_1+x_2_1<=1;
+x_1_1+x_3_1<=1;
.
.
+x_16_8+x_19_8<=1;
+x_17_8+x_18_8<=1;
(To speed up computation)
+y_2<=+y_1;
+y_3<=+y_2;
+y_4<=+y_3;
+y_5<=+y_4;
+y_6<=+y_5;
+y_7<=+y_6;
+y_8<=+y_7;
(declare all varaibles as binary)
bin x_1_1,x_2_1,...,x_20_8,x_21_8,y_1,...,y_8,;

The output file for part3 looks like this:
Value of objective function: 7.00000000

Actual values of the variables:
y_1                             1
y_2                             1
y_3                             1
y_4                             1
y_5                             1
y_6                             1
y_7                             1
x_1_1                           1
x_1_2                           1
x_2_3                           1
x_2_4                           1
x_3_5                           1
x_3_6                           1
x_4_1                           1
x_4_2                           1
x_5_3                           1
x_5_4                           1
x_6_3                           1
x_6_5                           1
x_7_4                           1
x_7_6                           1
```

```
x_8_5                          1
x_8_7                          1
x_9_1                          1
x_9_7                          1
x_10_2                         1
x_10_3                         1
x_11_1                         1
x_11_4                         1
x_12_2                         1
x_12_3                         1
x_13_4                         1
x_13_5                         1
x_14_6                         1
x_14_7                         1
x_15_6                         1
x_15_7                         1
x_16_2                         1
x_16_5                         1
x_17_4                         1
x_17_5                         1
x_18_1                         1
x_18_3                         1
x_19_1                         1
x_19_3                         1
x_20_4                         1
x_20_7                         1
x_21_4                         1
x_21_5                         1

All other variables are zero.
```