MATH 381 Assignment 3

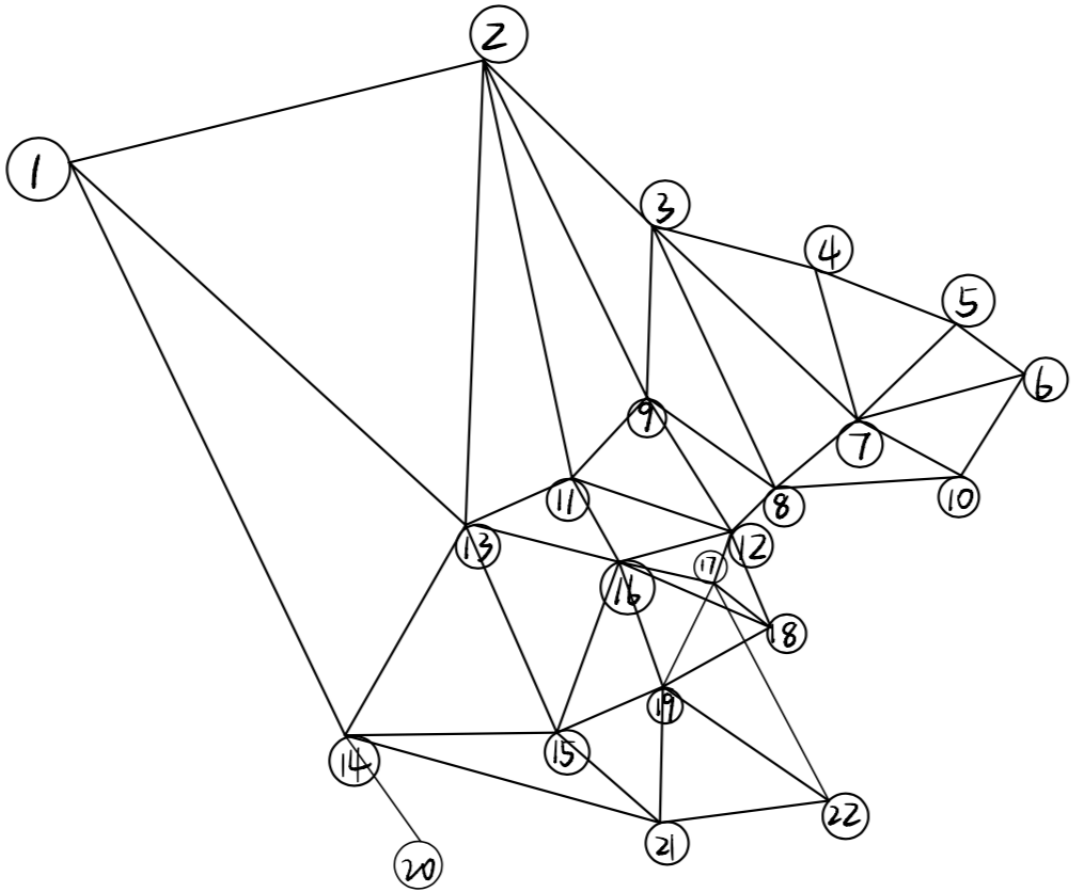
Xinghan Guo

January 28, 2023

1 Original Map and Graph

We will color the map of Sichuan Province, China using minimum number of colors. Here is the source of the map and it's original figure:
https://paintmaps.com/map-charts/389c/Sichuan-map-chart

This map contains 22 regions in Sichuan Province. Let each region corresponds to a vertex, then we have 22 vertices. A edge connects two vertices if these two regions share a border. Therefore, the orginal graph can be created as:



We are going to color this graph using the minimum number of colors in three different situation.

2 Design LPs

2.1 Variables

First, Let's define the variables:
Since we have 22 vertices in total and each vertex can be colored using only one single color, the maximum number of colors we have is 22.
Define binary variable $y_k \in \{0, 1\}$, $k = 1, ..., 22$, with $y_k = 1$ iff we use color k.

Define binary variable $x_{i,k} \in \{0, 1\}$, $i = 1, ..., 22$, $k = 1, ..., 22$, with $x_{i,k} = 1$ iff vertex i will have color k.

2.2 Objective Function

Our objective is to minimize the number of colors. Therefore, our objective function is just the sum over all y values: $\sum_{k=1}^{22} y_k$

2.3 Constraints

2.3.1

First, we require all vertices to be colored with exact one color. Therefore, for each vertex i, there must be and only be one color which can be used to color vertex i. Therefore, the sum over $x_{i,k}$, $k = 1, ..., 22$ must be equal to 1. We can express this constraint as:
$\sum_{k=1}^{22} x_{i,k} = 1$, $i = 1, ..., 22$

2.3.2

Next, we want to make sure that if we color vertex i with color k, then we are using color k. In other words, if we use color k in any of the vertices, then y_k must be equal to one. If color k is not used in any vertex, then y_k is equal to zero. We can express this constraints as:
$x_{i,k} \leqslant y_k$, i, $k = 1, ..., 22$

2.3.3

A main principle we use to color our graph is that adjacent vertices (connecting with one edge) must have different colors. i.e. If there is an edge between Vertex i and Vertex j, Vertex i and Vertex j can not be colored using same color. Then, the sum of $x_{i,k}$ and $x_{j,k}$ must less or equal to one. We can express this constraints as:
$x_{i,k} + x_{j,k} \leqslant 1$ for all Vertex i, Vertex j that are connected by an edge, and $k = 1, ..., 22$

2.3.4

Our LPs require complex calculations. To reduce the running time, we can speed up the calculation by reducing the space of set of colors. For example, if we use four colors in total, then we assume to use Color 1, 2, 3 and 4 but not other colors in our set of colors. We can

also consider that if we do not use Color 5, then we will also not use Color 6, 7,..., 22. We can set this constraint by letting $y_k \leqslant y_{k-1}$, for all $k = 2, ..., 22$.
In this case, if we don't use y_{k-1} (i.e. $y_{k-1} = 0$), then we will not use y_k, y_{k+1}, y_{k+2}, ... y_{22}, which will all be equal to zero.

2.3.5

The last constraint we need to set is to make sure that x and y are binary variables. i.e. $x_{i,k}$, $y_k \in \{0, 1\}$, $i = 1, ..., 22$, $k = 1, ..., 22$

2.4 Other scenarios

2.4.1 Second Coloring

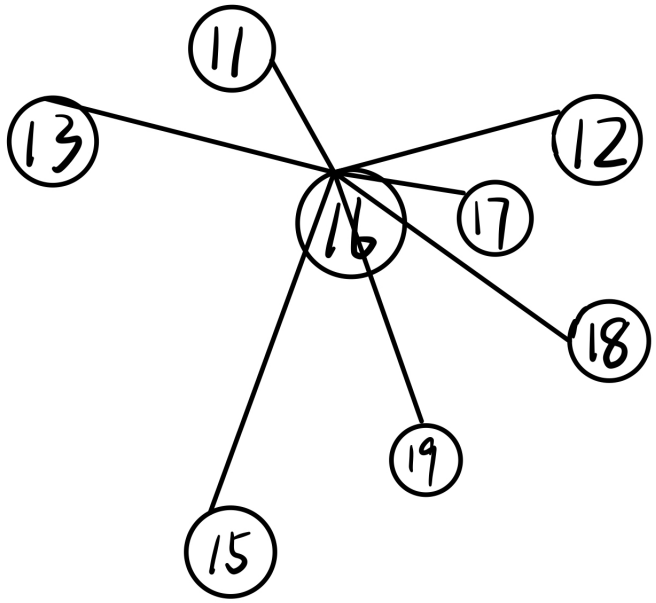
What if we want to make two regions that border the same region cannot be colored the same? Specifically, two regions that share a border, or that both have a border in common with a third region, cannot both be given the same color.

The number of regions (vertices) doesn't change and is still 22. We can solve this problem by adding edges based on first coloring.

Consider Vertex i, j, h. Suppose Vertex i and Vertex j is connected by an edge, Vertex$j$ and Vertex h is connected by an edge. Vertex i and Vertex h are currently not adjacent. According to the new requirement, Vertex i and Vertex h can not be colored same. We can achieve this by adding an edge between Vertex i and Vertex h and setting new constraint: $x_{i,k} + x_{h,k} \leqslant 1$ for all Vertex i and Vertex h if they have an edge with a common vertex j, where $i, j, h, k = 1, ..., 22$

In addition, to speed up the calculation, we can pre-assign color to the complete subgraph.

The subgraph below shows the maximum degree in our map. Vertex 16 has edges with seven different vertices, so the degree is 7.



According to the new requirement in our second coloring, the seven vertices having edge with single Vertex 16 should be different colors. Also, these seven vertices is adjacent with Vertex 16 so Vertex 16 should be colored in different color, too. Therefore, we can make sure that our map needs to use at least eight different colors.

Then, we can add labels to pre-assign color numbers for these eight vertices. The space of sets of colors can be reduced and the calculations will be speeded up.

2.4.2 Third Coloring

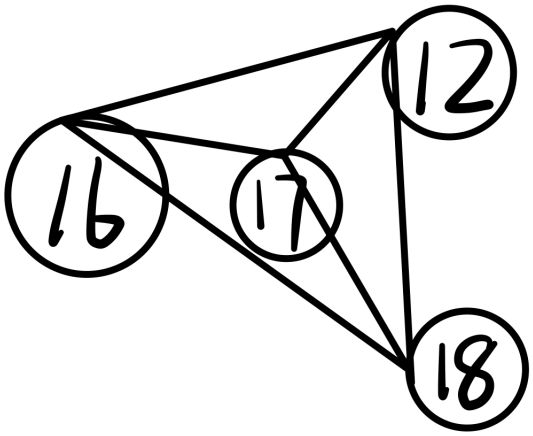
Now, based on our first coloring, suppose we want to assign two colors to each region such that adjacent regions share no colors.

The number of regions (vertices) doesn't change and is 22, but the maximum number of colors in our color set doubles. To solve this problem, we only need to make a little change of the constraint in 2.3.1.

We require all vertices to be colored with exact two colors. Therefore, for each vertex i, the sum over $x_{i,k}$, $k = 1, ..., 22$ must be equal to 2. We can express this constraint as: $\sum_{k=1}^{22} x_{i,k} = 2$, $i = 1, ..., 22$

In addition, to speed up the calculation, we can pre-assign color the complete subgraph.

In our map, we can find a complete graph K_4. This subgraph connects Vertex 16, 17,



12, and 18 as shown below.

We can see that the four vertices are connected to all other vertices, so Vertex 16, 17, 12, and 18 has to be different colors. Each vertex must be colored in two colors. Therefore, we can make sure that our third map needs to use at least eight different colors.

Then, we can add labels to pre-assign color numbers for these four vertices. The space of sets of colors can be reduced and the calculations will be speeded up.

2.5 LPs in Compact Form

Therefore, these are our LPs:

2.5.1 First coloring

min: $\sum_{i=1}^{22} y_i$
$\sum_{k=1}^{22} x_{i,k} = 1$, $i = 1, ..., 22$
$x_{i,k} \leqslant y_k$, $i, k = 1, ..., 22$
$x_{i,k} + x_{j,k} \leqslant 1$ for all Vertex i, Vertex j that are connected by an edge, and $k = 1, ..., 22$
$y_k \leqslant y_{k-1}$, for all $k = 2, ..., 22$
$x_{i,k}, y_k \in \{0, 1\}$, $i = 1, ..., 22$, $k = 1, ..., 22$

2.5.2 Second coloring

min: $\sum_{i=1}^{22} y_i$
$\sum_{k=1}^{22} x_{i,k} = 1$, $i = 1, ..., 22$
$x_{i,k} \leqslant y_k$, $i, k = 1, ..., 22$
$x_{i,k} + x_{j,k} \leqslant 1$ for all Vertex i, Vertex j that are connected by an edge, and $k = 1, ..., 22$
$x_{i,k} + x_{h,k} \leqslant 1$ for all Vertex i and Vertex h if they have an edge with a common vertex j, where $i, j, h, k = 1, ..., 22$ and $i \neq h$.

$y_k \leqslant y_{k-1}$, for all $k = 2, ..., 22$
$x_{16,1} = 1$
$x_{11,2} = 1$
$x_{12,3} = 1$
$x_{17,4} = 1$
$x_{18,5} = 1$
$x_{19,6} = 1$
$x_{15,7} = 1$
$x_{13,8} = 1$
$\sum_{i=1}^{22} y_i \geqslant 8$
$x_{i,k}, y_k \in \{0, 1\}, i = 1, ..., 22, k = 1, ..., 22$

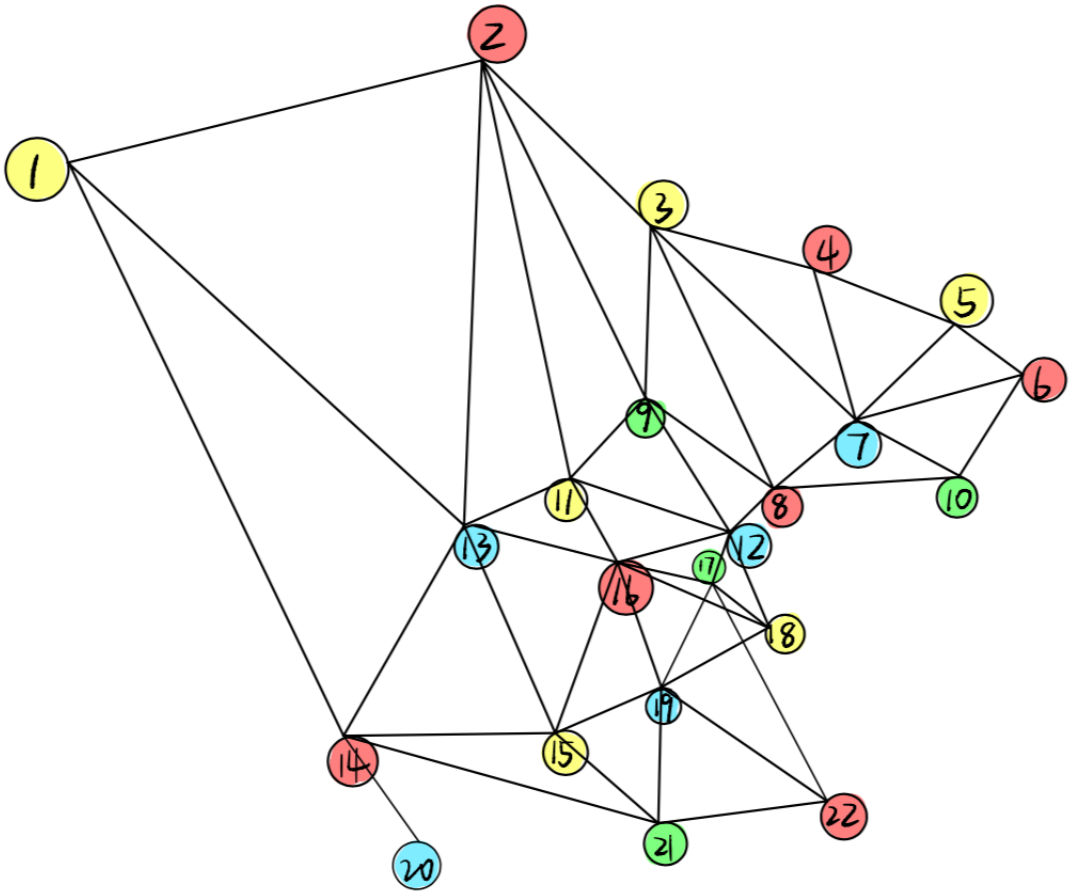
2.5.3 Third coloring

min: $\sum_{i=1}^{44} y_i$
$\sum_{k=1}^{44} x_{i,k} = 2, i = 1, ..., 22$
$x_{i,k} \leqslant y_k, i = 1, ..., 22, k = 1, ..., 44$
$x_{i,k} + x_{j,k} \leqslant 1$ for all Vertex i, Vertex j that are connected by an edge, and $k = 1, ..., 44$
$y_k \leqslant y_{k-1}$, for all $k = 2, ..., 44$
$x_{16,1} = 1$
$x_{17,2} = 1$
$x_{12,3} = 1$
$x_{18,4} = 1$
$x_{16,5} = 1$
$x_{17,6} = 1$
$x_{12,7} = 1$
$x_{18,8} = 1$
$\sum_{i=1}^{44} y_i \geqslant 8$
$x_{i,k}, y_k \in \{0, 1\}, i = 1, ..., 22, k = 1, ..., 44$

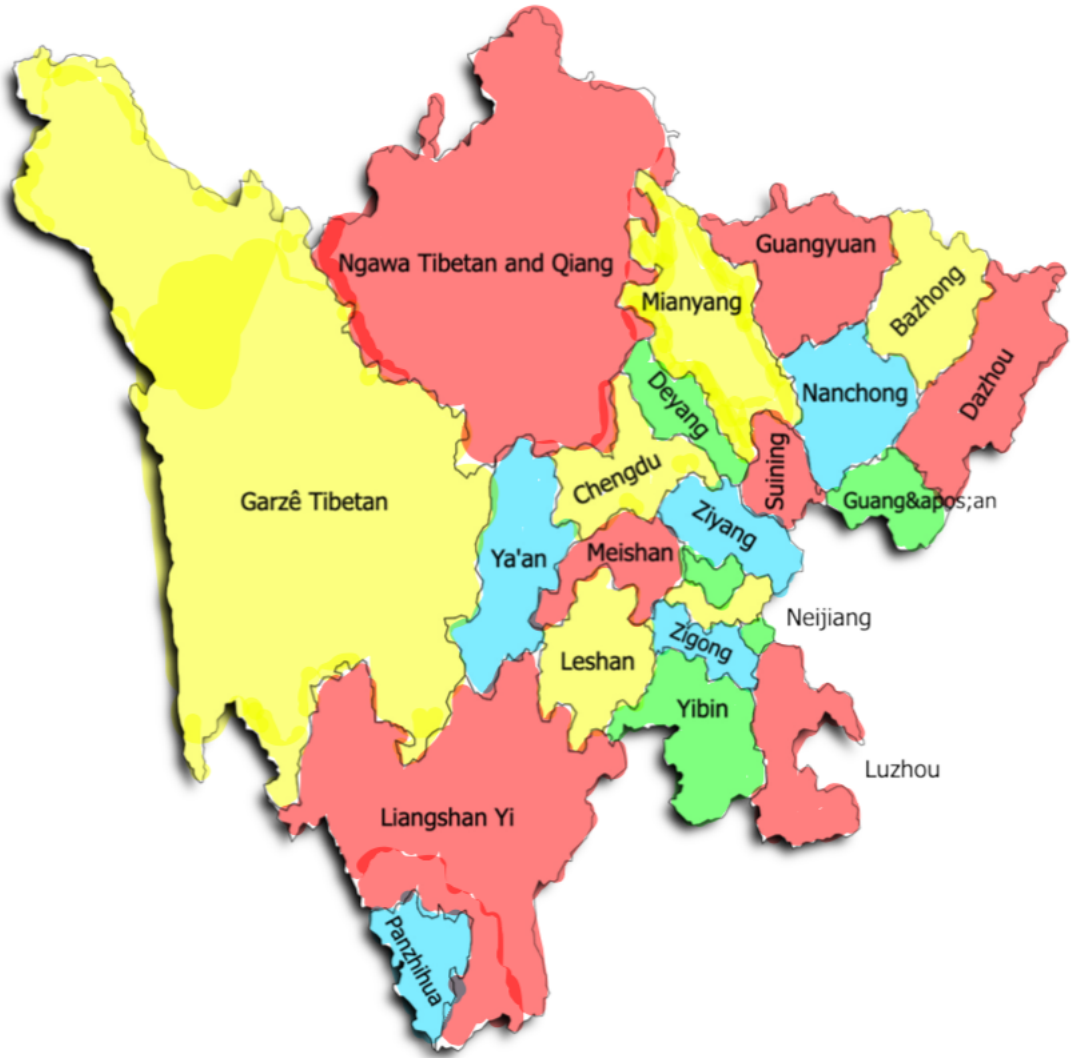
3 Graph and Map Drawings

3.1 First coloring

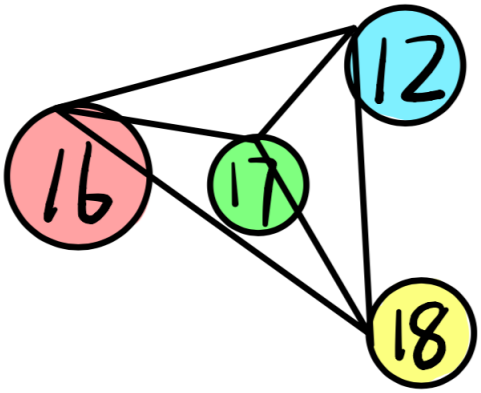
For the first coloring, we require all vertices to be colored with exact one color and adjacent regions should be colored differently. Through our LP, we need at least **FOUR** different colors to color this map. The result conforms to Four Color Theorem.
Here is the drawing of the graph:



Here is the drawing of the map:



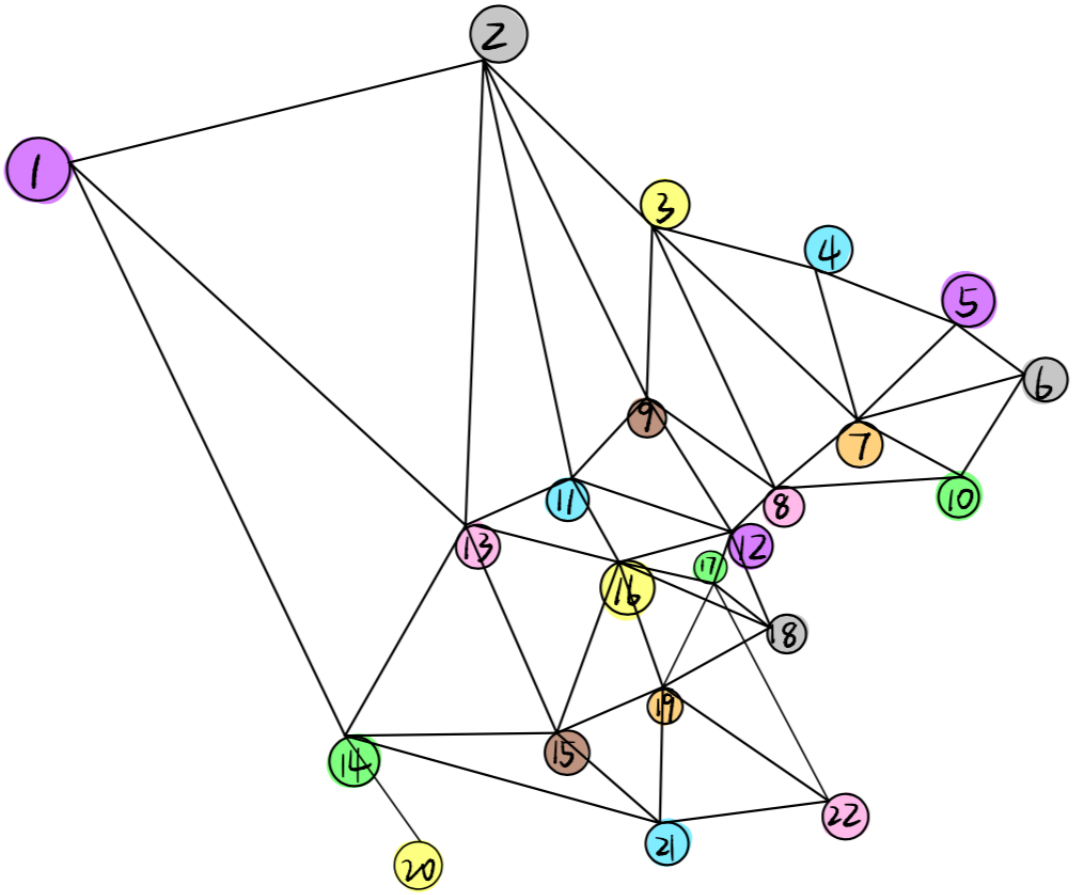
To test our result, we can see a subgraph as shown below.



Since each vertex is connected with all other vertices, this subgraph is a complete graph K_4. We must use four different colors for these vertices. Therefore, our result of first graph is valid. Through the graph/ map, we can also discover that every pair of collected vertices/ adjacent regions are colored differently.

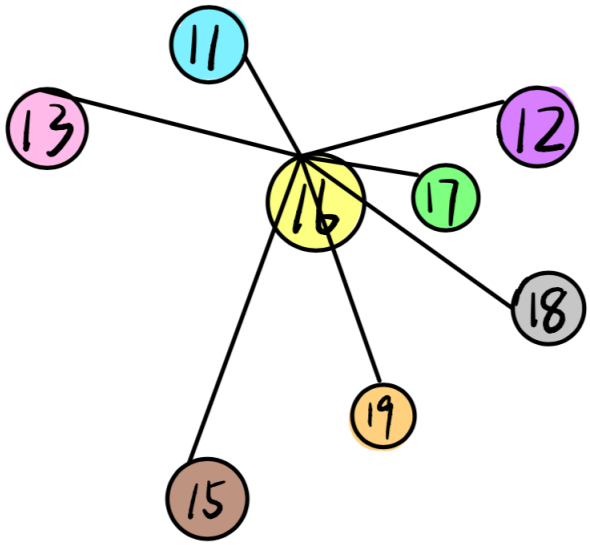
3.2 Second coloring

For the second coloring, we require that two regions that share a border, or that both have a border in common with a third region, cannot both be given the same color. Through our LP, we need at least **EIGHT** different colors to color this map.
Here is the drawing of the graph:



Here is the drawing of the map:

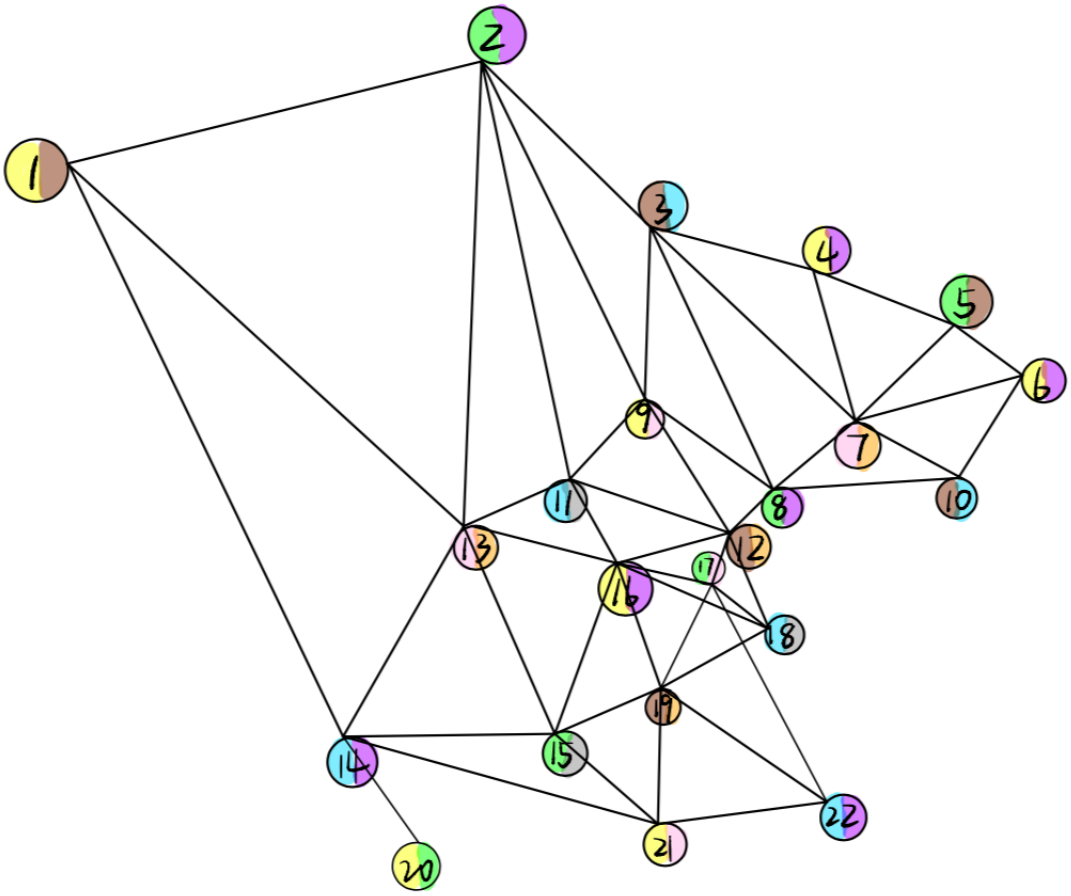
To test our result, we can see a subgraph as shown below.



Vertex 16 has the maximum degree of 7. According to the new requirement in our second coloring, the seven vertices having edge with single Vertex 16 should be different colors. Also, these seven vertices is adjacent with Vertex 16 so Vertex 16 should be colored in different color, too. Therefore, we can make sure that our map needs to use at least eight different colors. Therefore, our result of second graph is valid. Through the graph/ map, we can also discover that two regions that border the same region are colored differently.

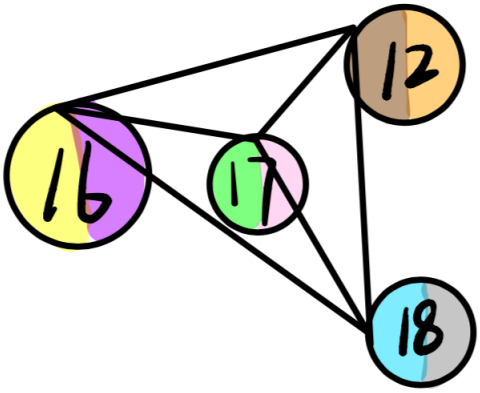
3.3 third coloring

For the third coloring, we require all vertices to be colored with two colors and adjacent regions should be colored differently. Through our LP, we need at least **EIGHT** different colors to color this map.
Here is the drawing of the graph:



Here is the drawing of the map:

To test our result, we can see a subgraph as shown below.



Since each vertex is connected with all other vertices, this subgraph is a complete graph K_4. Therefore, we must use eight different colors for these vertices. Our result of third graph is also proved to be valid! Through the graph/ map, we can also discover that each region contains two colors and every pair of collected vertices/ adjacent regions are colored differently.

4 Appendix

4.1 First coloring

4.1.1 Code

Here is my Python code of generating LP input file for the first coloring:

```
import math
# number of vertices
n=22
# number of maximum colors
k=22

# define edges
edges=[[1,2],[1,14],[1,13],[2,3],[2,13],[2,11],[2,9],[3,9],[3,8],[3,7],
       [3,4],[4,7],[4,5],[5,6],[5,7],[6,7],[6,10],[7,8],[7,10],[8,9],
       [8,12],[8,10],[9,12],[9,11],[11,13],[11,16],[11,12],[12,16],
       [12,17],[12,18],[13,14],[13,15],[13,16],[14,15],[14,20],[14,21],
       [15,16],[15,19],[15,21],[16,19],[16,18],[16,17],[17,18],[17,19],
       [17,22],[18,19],[19,21],[19,22],[21,22]]

##### generate lpsolve input file
# objective function
fullString = ""
```

```
for k in range(1,k+1):
    fullString += "+y_"+str(k)
print("min: "+fullString+";")

# constraint 1: all vertices to be colored with exactly one color.
for i in range(1,n+1):
    constraintString1 = ""
    for k in range(1,k+1):
        constraintString1 += "+x_" + str(i) + "_" + str(k)
    print(constraintString1 + "=1;")

# constraint 2: a vertex cannot be colored with an unused colored
for i in range(1,n+1):
    for k in range(1,k+1):
        print("x_" + str(i) + "_" + str(k) + "<=y_" + str(k) + ";")

# constraint 3: adjacent vertices have different colors.
for k in range(1,k+1):
    for edge in edges:
        print("x_"+str(edge[0])+"_"+str(k) + "+x_" + str(edge[1])+"_"+str(k) + "<=1;")

# constraint 4: speeds up the calculations
for k in range (2,k+1):
    print("y_" + str(k) + "<=y_" + str(k-1) + ";")

# declare all variables as binary
binString = "bin "
for k in range(1,k+1):
    for i in range(1,n+1):
        if (i > 1 or k > 1):
            binString += ","
        binString += "x_"+str(i)+"_"+str(k)
for k in range(1,k+1):
    binString += ",y_"+str(k)
print(binString+";")
```

4.1.2 lpsolve Input

The lpsolve input file for the first coloring can be concluded as:

min: +y_1+y_2+y_3+y_4+...+y_19+y_20+y_21+y_22
(22 lines of the following type:
ensure that every vertex is colored with exactly one color)
+x_1_1+x_1_2+x_1_3+...+x_1_20+x_1_21+x_1_22=1;
.
.
(484 lines of the following type:

ensure that the y variables keep track of colors that are used)
x_1_1<=y_1;
.
.
(1078 lines of the following type:
ensure vertices connected by edges are different colors)
x_1_1+x_2_1<=1;
.
.
(22 lines of the following type:
ensure that lower numbered colors are used before higher numbered colors)
y_2<=y_1;
.
.
y_22<=y_21;
(ensure all variables are binary)
bin x_1_1,...,x_22_22,y_1,...,y_22;

4.1.3 lpsolve Output

The lpsolve output file for the first coloring is:

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_1 1
x_4_2 1
x_5_1 1
x_6_2 1
x_7_3 1
x_8_2 1
x_9_4 1
x_10_4 1
x_11_1 1
x_12_3 1
x_13_3 1

x_14_2 1
x_15_1 1
x_16_2 1
x_17_4 1
x_18_1 1
x_19_3 1
x_20_3 1
x_21_4 1
x_22_2 1
All other variables are zero.

4.2 Second coloring

4.2.1 Code

Here is my Python code of generating LP input file for the second coloring:

```
import math
# number of vertices
n=22
# number of maximum colors
k=22

# define edges
edges=[[1,2],[1,14],[1,13],[2,3],[2,13],[2,11],[2,9],[3,9],[3,8],[3,7],
       [3,4],[4,7],[4,5],[5,6],[5,7],[6,7],[6,10], [7,8],[7,10],[8,9],
       [8,12],[8,10],[9,12],[9,11],[11,13],[11,16],[11,12],[12,16],
       [12,17],[12,18], [13,14],[13,15], [13,16],[14,15],[14,20],
       [14,21],[15,16],[15,19],[15,21],[16,19],[16,18],[16,17],
       [17,18],[17,19],[17,22],[18,19],[19,21],[19,22],[21,22]]

##### generate lpsolve input file
# objective function
fullString = ""
for k in range(1,k+1):
    fullString += "+y_"+str(k)
print("min: "+fullString+";")

#  constraint 1: all vertices to be colored with exactly one color.
for i in range(1,n+1):
    constraintString1 = ""
    for k in range(1,k+1):
        constraintString1 += "+x_" + str(i) + "_" + str(k)
    print(constraintString1 + "=1;")

# constraint 2: a vertex cannot be colored with an unused colored
for i in range(1,n+1):
    for k in range(1,k+1):
```

```
        print("x_" + str(i) + "_" + str(k) + "<=y_" + str(k) + ";")

# constraint 3: adjacent vertices have different colors.
for k in range(1,k+1):
    for edge in edges:
        print("+x_"+str(edge[0])+"_"+str(k) + "+x_" + str(edge[1])+"_"+str(k) + "<=1;")

# constraint 4: speeds up the calculations
for k in range (2,k+1):
    print("y_" + str(k) + "<=y_" + str(k-1) + ";")

# constraint 5: two regions that border the same region cannot be colored the same
for a in edges:
    for b in edges:
        if (a[1] == b[0]) & (a[0] != b[1]):
            for k in range(1,k+1):
                print("x_"+str(a[0])+"_"+str(k) + "+" "x_"+str(b[1])+"_"+str(k)+"<=1;")
        elif (a[1] == b[1]) & (a[0] != b[0]):
            for k in range(1,k+1):
                print("x_" + str(a[0]) + "_" + str(k) + "+" "x_" + str(b[0]) + "_"
                      + str(k) + "<=1;")

# constraint 6: speed up the calculations
print("label1: x_16_1 = 1;")
print("label2: x_11_2 = 1;")
print("label3: x_12_3 = 1;")
print("label4: x_17_4 = 1;")
print("label5: x_18_5 = 1;")
print("label6: x_19_6 = 1;")
print("label7: x_15_7 = 1;")
print("label8: x_13_8 = 1;")
print(fullString+">=8;")

# declare all variables as binary
binString = "bin "
for k in range(1,k+1):
    for i in range(1,n+1):
        if (i > 1 or k > 1):
            binString += ","
        binString += "x_"+str(i)+"_"+str(k)
for k in range(1,k+1):
    binString += ",y_"+str(k)
print(binString+";")
```

4.2.2 lpsolve Input

The lpsolve input file for the second coloring can be concluded as:

min: +y_1+y_2+y_3+y_4+...+y_19+y_20+y_21+y_22

(22 lines of the following type:
ensure that every vertex is colored with exactly one color)
+x_1_1+x_1_2+x_1_3+...+x_1_20+x_1_21+x_1_22=1;
.
.
(484 lines of the following type:
ensure that the y variables keep track of colors that are used)
x_1_1<=y_1;
.
.
(1078 lines of the following type:
ensure vertices connected by edges are different colors)
x_1_1+x_2_1<=1;
.
.
(22 lines of the following type:
ensure that lower numbered colors are used before higher numbered colors)
y_2<=y_1;
.
.
y_22<=y_21;
(4180 lines of the following type:
ensure two regions that border the same region cannot be colored the same)
x_1_1+x_3_1<=1;
.
.
(8 lines of the following type:
speeding up the calculations)
label1: x_16_1 = 1;
.
.
(at least 8 different colors should be used)
+y_1+y_2+y_3+y_4+...+y_19+y_20+y_21+y_22>=8;
(ensure all variables are binary)
bin x_1_1,...,x_22_22,y_1,...,y_22;

4.2.3 lpsolve Output

The lpsolve output file for the second coloring is:

Value of objective function: 8.00000000

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
y_8 1
x_1_3 1
x_2_5 1
x_3_1 1
x_4_2 1
x_5_3 1
x_6_5 1
x_7_6 1
x_8_8 1
x_9_7 1
x_10_4 1
x_11_2 1
x_12_3 1
x_13_8 1
x_14_4 1
x_15_7 1
x_16_1 1
x_17_4 1
x_18_5 1
x_19_6 1
x_20_1 1
x_21_2 1
x_22_8 1
All other variables are zero.

4.3 Third coloring

4.3.1 lpsolve Output

The lpsolve output file for the third coloring is:

Value of objective function: 8.00000000
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
y_8 1
x_1_1 1
x_1_3 1
x_2_2 1
x_2_5 1
x_3_3 1
x_3_4 1
x_4_1 1
x_4_5 1
x_5_2 1
x_5_3 1
x_6_1 1
x_6_5 1
x_7_6 1
x_7_7 1
x_8_2 1
x_8_5 1
x_9_1 1
x_9_6 1
x_10_3 1
x_10_4 1
x_11_4 1
x_11_8 1
x_12_3 1
x_12_7 1
x_13_6 1
x_13_7 1
x_14_4 1
x_14_5 1
x_15_2 1
x_15_8 1
x_16_1 1
x_16_5 1
x_17_2 1

x_17_6 1
x_18_4 1
x_18_8 1
x_19_3 1
x_19_7 1
x_20_1 1
x_20_2 1
x_21_1 1
x_21_6 1
x_22_4 1
x_22_5 1
All other variables are zero.