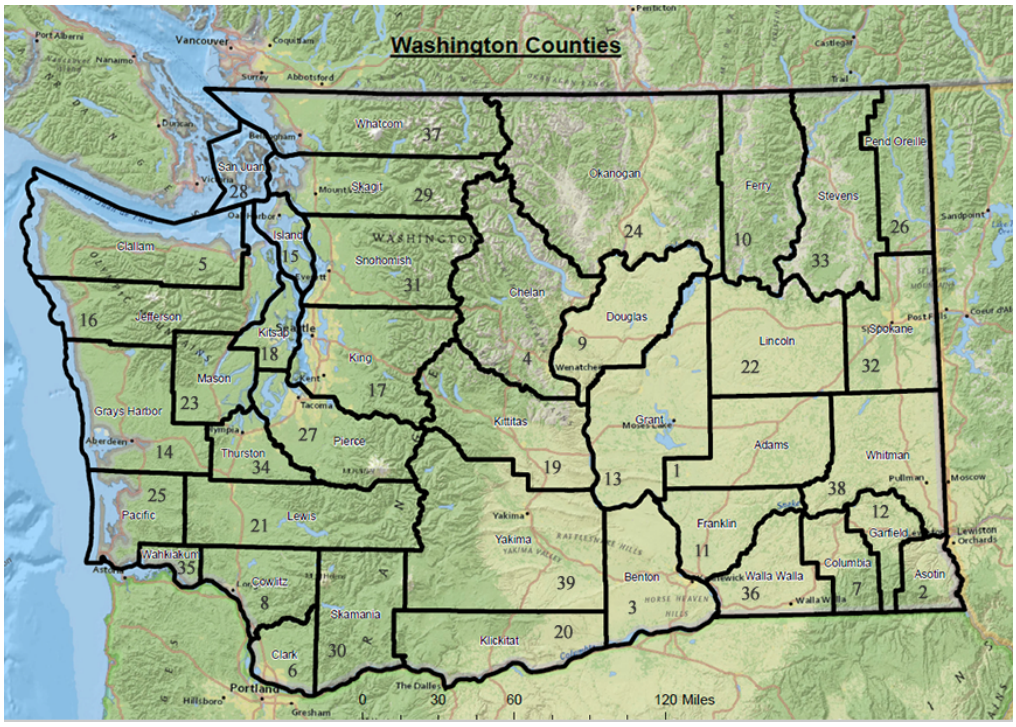
Map Coloring

Jiasui(Maggie) Qin

October 23, 2020

Given a Washington state county map as shown below. How can we use the minimum number of colors to color all counties while adjacent counties are colored with different colors? We define two counties to be adjacent if they share a common boundary of non-zero length. Counties which meet at a single point are not considered to be "adjacent". This map is extracted from http://www.crab.wa.gov/counties/countyMap.cfm. All 39 counties are listed alphabetically and numbered 1 to 39. I will solve this problem using linear programming.

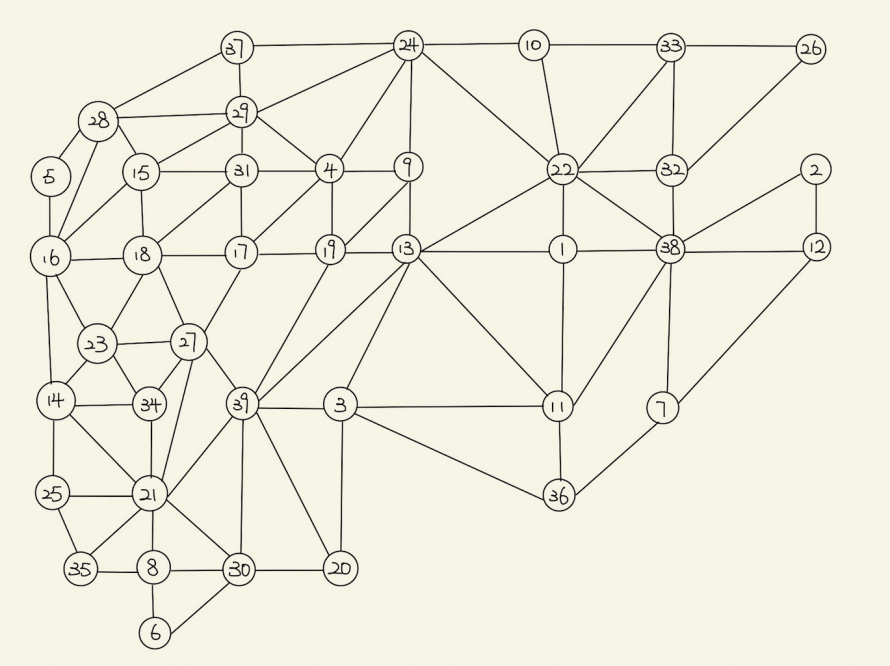


Mathematical formulation:

We define a graph $G = (V, E)$ with 39 vertices. Lable each county with $v_i, i = 1, ..., 39$ in alphabetic order as shown in the map, and let $V = \{v_1, v_2 ... v_{39}\}, E = \{(v_i, v_j) \text{for all adjacent} v_i, v_j, v = 1, ..., 39, i = 1, ..., 39\}$.

Counties will be labled as below:
v_1: Adams County, v_2: Asotin County, v_3: Benton County,
v_4: Chelan County, v_5: Clallam County, v_6: Clark County,
v_7: Columbia County, v_8: Cowlitz County, v_9: Douglas County,
v_{10}: Ferry County, v_{11}: Franklin County, v_{12}: Garfield County,
v_{13}: Grant County, v_{14}: Grays Harbor County, v_{15}: Island County,
v_{16}: Jefferson County, v_{17}: King County, v_{18}: Kitsap County,
v_{19}: Kittitas County, v_{20}: Klickitat County, v_{21}: Lewis County,
v_{22}: Lincoln County, v_{23}: Mason County, v_{24}: Okanogan County,
v_{25}: Pacific County, v_{26}: Pend Orielle County, v_{27}: Pierce County,
v_{28}: San Juan County, v_{29}: Skagit County, v_{30}: Skamania County,
v_{31}: Snohomish County, v_{32}: Spokane County, v_{33}: Stevens County,
v_{34}: Thurston County, v_{35}: Wahkiakum County, v_{36}: Walla Walla County,
v_{37}: Whatcom County, v_{38}: Whitman County, v_{39}: Yakima County

We can then represent the map by creating the graph G with vertices and edges as below:



Suppose we have a set of 39 colors to use to color the map(we won't use all of them since there are non-adjacent counties can be colored with the same color).

Define $c_i \in \{0, 1\}, i = 1, ..., 39$, with $c_i = 1$ if and only if we use color i.

Define $x_{vi} \in \{0, 1\}, v = 1, ..., 39, i = 1, ..., 39$, with $x_{vi} = 1$ if and only if vertex v will have color i.

Our goal is to minimize the number of colors we use. Therefore, the objective function would be

$$\sum_{i=1}^{39} c_i$$

To make sure that every county is colored by exactly one color, we have the following constraint:

$$\sum_{i=1}^{39} x_{vi} = 1, v = 1, ..., 39$$

Since we can only color a county with an existing color, if a vertex has color i, c_i must be one.

$$x_{vi} \leq c_i, v, i = 1, ..., 39$$

Moreover, adjacent counties should be colored with different colors. Therefore, for any adjacent counties v_a and v_b, x_{ai} and x_{bi} can not be both equal to one.

$$x_{ai} + x_{bi} \leq 1, i = 1, ..., 39$$

To speed up the calculation, we make sure that we prefer lower-numbered colors first. That is, if we use m colors, then it only chooses colors from 1 to m but not other sets of m colors. We will not use color 2 if we don't use color 1, and we will not use color 3 if we don't use color 2.

$$c_i \leq c_{i-1}, i = 2, ..., 39$$

Then we define all our variables to be binary.

$$x_{vi}, c_i \in \{0, 1\}, v, i = 1, ..., 39$$

Generate the input file:

To use Lpsolve to solve this LP problem, I wrote the following python code to generate the Lpsolve input file:

```
# Jiasui Qin
#
# This code outputs an lpsolve-formatted LP which
# decides how to use the minimum number of colors
# to color the Washington state county map while
# adjacent counties have different colors

# define the adjacency list by manually adding all
# pairs of edges into a 2d-array
adjacency =
    ↪ [[1,13],[1,22],[1,11],[1,38],[2,38],[2,12],[3,11],[3,13],[3,39],
    ↪
[3,20],[3,36],[4,9],[4,19],[4,17],[4,31],[4,29],[4,24],[5,28],[5,16],[6,8],
    ↪
[6,30],[7,38],[7,12],[7,36],[8,35],[8,21],[8,30],[9,24],[9,13],[9,19],[10,24],
    ↪
[10,33],[10,22],[11,13],[11,36],[11,38],[12,38],[13,19],[13,22],[14,16],
    ↪
[14,23],[14,34],[14,21],[14,25],[15,28],[15,29],[15,31],[15,18],[15,16],
    ↪
[16,18],[16,23],[17,18],[17,31],[17,19],[17,27],[18,31],[18,23],[18,27],
    ↪
[19,39],[20,39],[20,30],[21,34],[21,27],[21,39],[21,30],[21,35],[21,25],
    ↪
[22,24],[22,33],[22,32],[22,38],[23,27],[23,34],[24,29],[24,37],[25,35],
    ↪
[26,32],[26,33],[27,39],[27,34],[28,37],[28,29],[28,16],[29,37],[29,31],
    ↪
[30,39],[32,33],[32,38],[39,13]]

# print the objective function
outstring = "min: "
for i in range(1, 40):
    outstring = outstring + "+c_" + str(i)
outstring = outstring+";"
print(outstring)

# generate a constraint that every county
# is colored by exactly one color
outstring = ""
for v in range(1, 40):
    for i in range(1, 40):
        outstring = outstring + "+x_" + str(v) + "_" + str(i)
    outstring = outstring+"=1;\n"
print(outstring)

# generate a constraint that we color a
# county with an existing color
outstring = ""
for v in range(1, 40):
    for i in range(1, 40):
        outstring = outstring + "+x_" + str(v) + "_" + str(i) + "<=
```

```
            ↪ c_" + str(i) + ";\n"
print(outstring)

# generate a constraint that adjacent
# counties have different colors
outstring = ""
for a,b in adjacency:
    for i in range(1, 40):
        outstring = outstring + "+x_" + str(a) + "_" + str(i) + "+
            ↪ x_" + str(b) + "_" + str(i) + "<=1;\n"
print(outstring)

# generate a constraint that we choose
# lower-numbered colors first
outstring = ""
for i in range(2, 40):
    outstring = outstring + "c_" + str(i) + "<=c_" + str(i-1) + ";\
        ↪ n"
print(outstring)

# define variables to be binary
outstring = "bin "
for i in range(1, 40):
    if i != 1:
        outstring = outstring + ","
    outstring = outstring + "c_" + str(i)
for v in range(1, 40):
    for i in range(1, 40):
        outstring = outstring + ",x_" + str(v) + "_" + str(i)
outstring = outstring+";"
print(outstring)
```

After I ran the python code, I got the following Lpsolve input file:

```
min: +c_1+c_2+c_3+c_4+c_5+c_6+c_7+c_8+c_9+c_10+c_11+c_12+c_13+c_14+
    ↪ c_15+c_16+c_17+c_18+c_19+c_20+c_21+c_22+c_23+c_24+c_25+c_26+
    ↪ c_27+c_28+c_29+c_30+c_31+c_32+c_33+c_34+c_35+c_36+c_37+c_38+
    ↪ c_39;
(39 lines of the following type:
ensure that every county is colored by exactly one color)
+x_1_1+x_1_2+x_1_3+...+x_1_39=1;
.
.
(1521 lines of the following type:
ensure that we color a county with an existing color)
+x_1_1<=c_1;
.
.
(3471 lines of the following type:
ensure that adjacent counties have different colors)
+x_1_1+x_13_1<=1;
.
.
(39 lines of the following type:
ensure that we choose lower-numbered colors first)
c_2<=c_1;
.
.
(ensure that all variables are binary)
bin c_1,c_2,...c_39,x_1_1,x_1_2,...x_39_39;;
```

After running the Lpsolve, it gave me the following result:

```
CPU Time for solving: 89.7529s (89.7826s total since program start)

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_1                              1
x_3_1                              1
x_4_1                              1
x_5_1                              1
x_6_1                              1
x_7_1                              1
x_8_2                              1
x_9_2                              1
x_10_1                             1
x_11_2                             1
x_12_2                             1
x_13_3                             1
x_14_1                             1
x_15_1                             1
x_16_2                             1
x_17_2                             1
x_18_4                             1
x_19_4                             1
x_20_3                             1
x_21_3                             1
x_22_2                             1
x_23_3                             1
x_24_3                             1
x_25_2                             1
x_26_2                             1
x_27_1                             1
x_28_3                             1
x_29_2                             1
x_30_4                             1
x_31_3                             1
x_32_1                             1
x_33_3                             1
x_34_4                             1
x_35_1                             1
x_36_3                             1
x_37_1                             1
x_38_3                             1
x_39_2                             1
```

Conclusion

It turns out that the minimum number of colors to cover the entire map is 4.

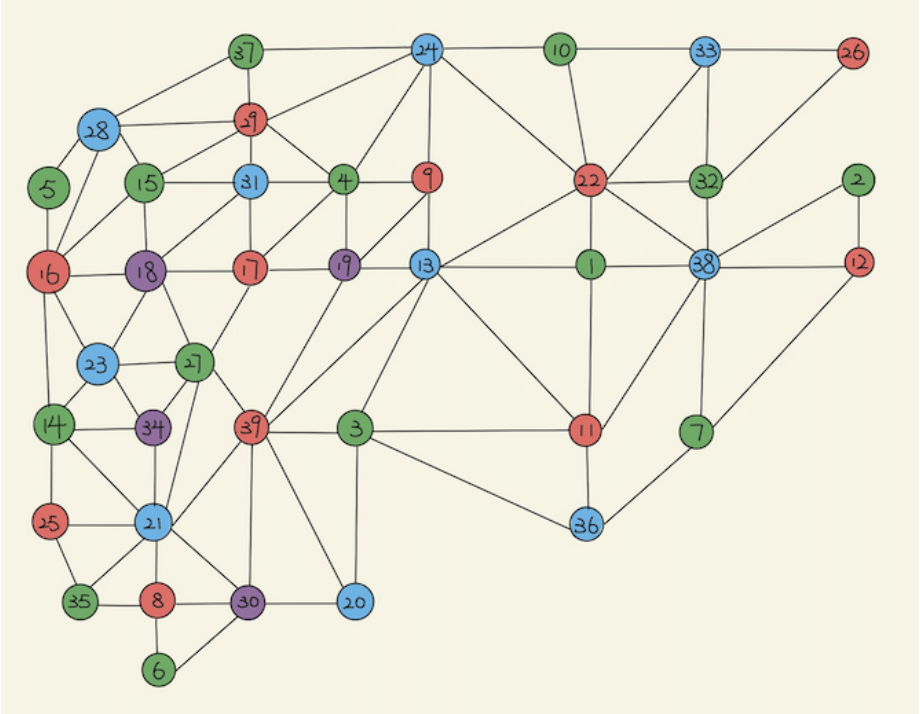
Counties that are colored by color1:
$v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_{10}, v_{14}, v_{15}, v_{27}, v_{32}, v_{35}, v_{37}$

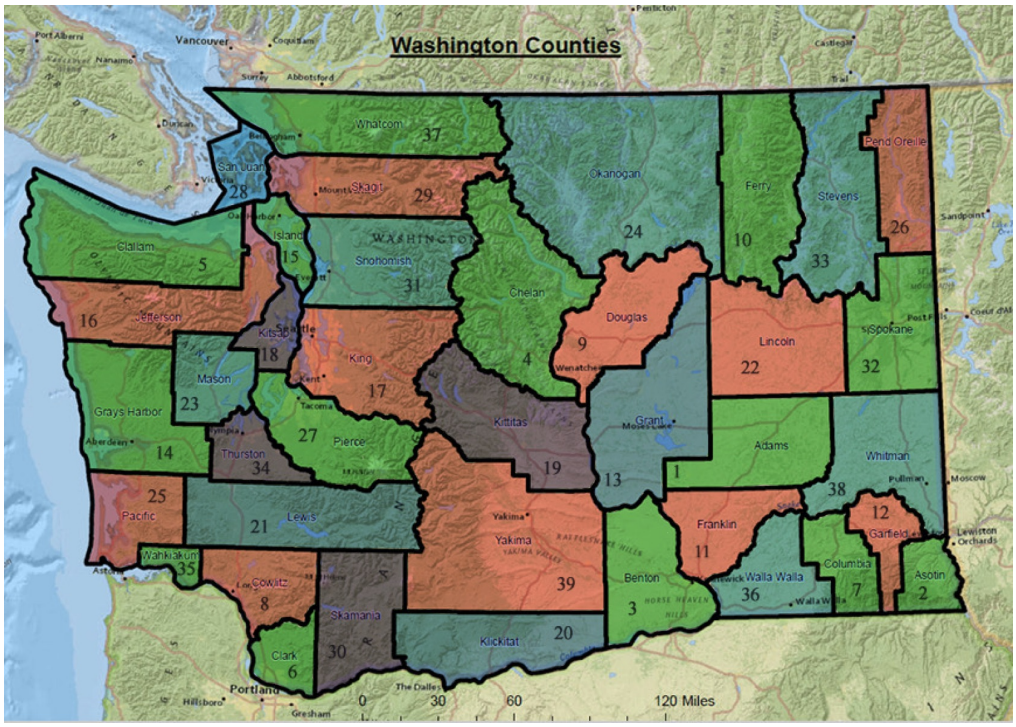
Counties that are colored by color2:
$v_8, v_9, v_{11}, v_{12}, v_{16}, v_{17}, v_{22}, v_{25}, v_{26}, v_{29}, v_{39}$

Counties that are colored by color3:
$v_{13}, v_{20}, v_{21}, v_{23}, v_{24}, v_{28}, v_{31}, v_{33}, v_{36}, v_{38}$

Counties that are colored by color4:
$v_{18}, v_{19}, v_{30}, v_{34}$

Suppose that color1 is green, color2 is red, color3 is blue, color4 is purple, we can then color the graph and the map as below:





The solution is valid since no adjacent counties are colored by the same color. And we can prove that at least 4 colors are needed by looking at the following subgraph involving $v_{16}, v_{18}, v_{23}, v_{27}, v_{14}, v_{34}$. Since v_{23} is adjacent to all other vertices, it needs to have a different color. While all other five vertices are connected, there are at least 3 colors needed to color these five vertices without having adjacent vertices getting the same color. Therefore, this subgraph requires at least 4 colors and proves our solution.