

Problem statement: Map Coloring Problem

A k -coloring of a map is an assignment of k colors, one to each country, in such a way that no two countries sharing a border have the same color. This problem can be translated to a constraint graph. A coloring of a graph G assigns a color to each vertex of G , with the restriction that two adjacent vertices never have the same color.

The chromatic number of G , written $\chi(G)$, is the smallest number of colors needed to color G . In this project, we will experiment with map coloring techniques and compare the observed results in the context of USA and Australia map.

Map coloring problem formulation.

The Map Coloring Problem is an application of the graph theory. It is an assignment of labels traditionally called "colors" to elements of a graph subject to certain constraints. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices are of the same color; this is called a **vertex coloring**. Similarly, an edge coloring assigns a color to each edge so that no two adjacent edges are of the same color, and a **face coloring** of a planar graph assigns a color to each face or region so that no two faces that share a boundary have the same color.

Vertex coloring is the starting point of graph coloring. Other coloring problems can be transformed into a vertex version. For example, an edge coloring of a graph is just a vertex coloring of its line graph, and a face coloring of a plane graph is just a vertex coloring of its dual. However, non-vertex coloring problems are often stated and studied as *is*. That is partly for perspective, and partly because some problems are best studied in non-vertex form, as for instance is edge coloring.

The convention of using colors originates from coloring the countries of a map, where each face is literally colored. This was generalized to coloring the faces of a

graph embedded in the plane. By planar duality it became coloring the vertices, and in this form, it generalizes to all graphs.

In mathematical and computer representations, it is typical to use the first few positive or nonnegative integers as the "colors". In general, one can use any finite set as the "color set". The nature of the coloring problem depends on the number of colors but not on what they are.

Graph coloring enjoys many practical applications as well as theoretical challenges. Beside the classical types of problems, different limitations can also be set on the graph, or on the way a color is assigned, or even on the color itself.

We are trying to solve the K Coloring for the United States Map. Here, we need to identify the least number of colors that is required to color the states present in the United States country. The K here is the chromatic number which is obtained from the Chromatic Polynomial.

Chromatic polynomial

The **chromatic polynomial** counts the number of ways a graph can be colored using no more than a given number of colors. For example, using three colors, the graph in the adjacent image can be colored in 12 ways. With only two colors, it cannot be colored at all. With four colors, it can be colored in $24 + 4 \cdot 12 = 72$ ways: using all four colors, there are $4! = 24$ valid colorings (every assignment of four colors to *any* 4-vertex graph is a proper coloring); and for every choice of three of the four colors, there are 12 valid 3-colorings. So, for the graph in the example, a table of the number of valid colorings would start like this:

Available colors	1	2	3	4	...
Number of colorings	0	0	12	72	...

The chromatic polynomial is a function $P(G, t)$ that counts the number of t -colorings of G . As the name indicates, for a given G the function is indeed a polynomial in t . For the example graph, $P(G, t) = t(t - 1)^2(t - 2)$, and indeed $P(G, 4) = 72$.

The chromatic polynomial includes at least as much information about the colorability of G as does the chromatic number. Indeed, χ is the smallest positive integer that is not a root of the chromatic polynomial

We have used various attempts through programming to calculate the Chromatic number for to color the United States Map. **We identified the optimal chromatic number is 4.**

We can solve this Vortex Coloring Problem using various methods like the following:

1. SEQ Algorithm
2. DSATUR Algorithm
3. DSATUR branch and bound Algorithm:
 - Let UB be an upper bound on χ .
 - Each sub-problem corresponds to a partial coloring of the graph. When this partial coloring uses k colors, and $k \geq \text{UB}$, the sub-problem can be fathomed;
 - When all the vertices are colored and $k < \text{UB}$, then $\text{UB} = k$;
 - From each sub-problem using k colors, up to $k + 1$ new sub-problems are generated, by assigning, when feasible, one of the k colors to the next vertex to be colored, and by coloring it with color $k + 1$ if $k + 1 < \text{UB}$. The vertex with the maximum chromatic degree is chosen as next vertex to be colored.
 - A lower bound (maximal clique) is obtained at the beginning.

Compact MIP formulation

Variables ($i = 1, \dots, n$ $h = 1, \dots, n$):

$$x_i^h = \begin{cases} 1 & \text{vertex } i \text{ takes color } h \\ 0 & \text{otherwise} \end{cases} \quad y^h = \begin{cases} 1 & \text{if color } h \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

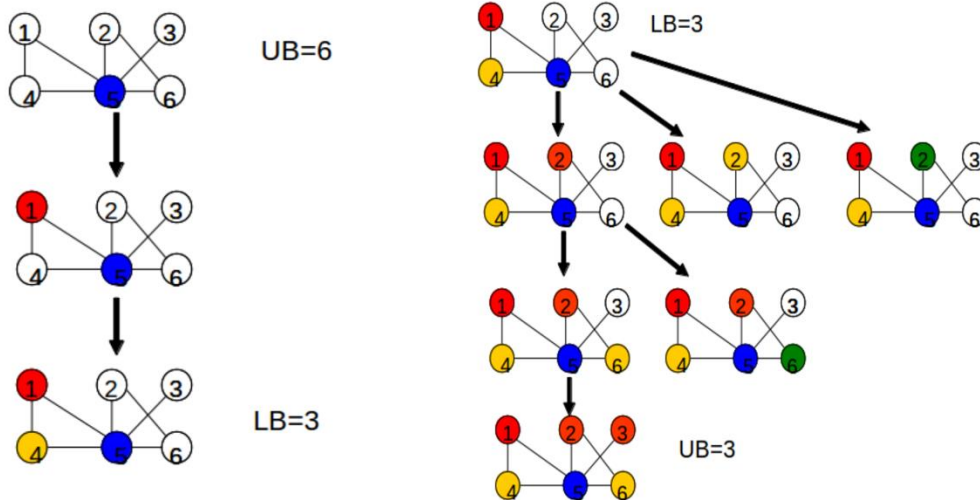
$$\min \sum_{h=1}^n y^h \quad (1)$$

$$\sum_{h=1}^n x_i^h = 1, \quad i = 1, \dots, n \quad (2)$$

$$x_i^h + x_j^h \leq y^h \quad (i, j) \in E, h = 1, \dots, n \quad (3)$$

$$x_i^h \in \{0, 1\} \quad i = 1, \dots, n, h = 1, \dots, n \quad (4)$$

$$y^h \in \{0, 1\} \quad h = 1, \dots, n \quad (5)$$



The above is an example on how a branch and cut algorithm works to solve this problem.

We have used 6 methods described below to solve this problem using the techniques taught in the class.

Global variables

- 1) students: ["Dhruv Dhamani","Saurabh Burange","Akhil Battu", "Kartik Ravi", "Samruddhi Godbole"]
-It is set of students to display details.
- 2) type: "dfs"
-to define the method/ algorithm to execute map coloring problem. Default value set to dfs.
- 3) withHeuristic: false
-to check if map coloring to be executed using heuristic or not. Default value set to false.
- 4) triedColors: 0
-counter to keep track of colors.
- 5) backtracked: 0
-keep the count of backtrack/ number of backtracks required.
- 6) timeTaken: 0
-counter to keep track of execution time.
- 7) startState: ""
-Name of the state that user selects to start the coloring/ initial state name.
- 8) countryName: "usa"
-Name of the country for which map coloring is done.
- 9) usaColors: ['FF3860', '209CEE', '22D160', 'FFDD57']
-set of colors to be considered for map coloring. [hard coded to 4 colors as blue, green, red and yellow]
- 10)usa:
It is the object of the country that holds all its state related data e.g. name, neighboring states, legal colors etc. [for all of the states in USA country].

Function/procedure

We considered total 6 approaches and corresponding functions to solve map coloring problem:

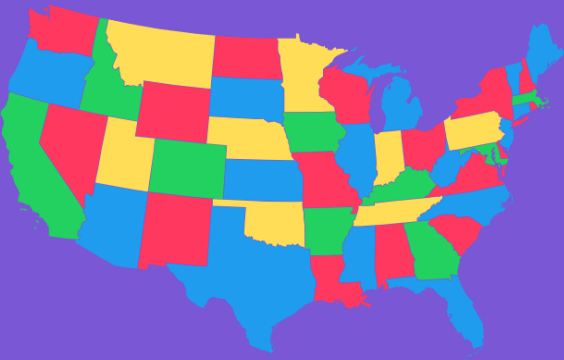
DFS : dfs

Depth first search approach starts with a state [selected input state] and from the state's set of legal colors it chooses a color to be assigned to it. It then moves to color its neighbor such that no 2 neighbors can have same color. It assigns color to the state till either the entire map is colored or the coloring constraint is violated, in this case it backtracks to the previous states. This approach is complete and finds the solution (if exists).

Keeps track of country name [e.g. USA] along with stack[] that contains all the states of the map. Colored[] array to store states that have been colored so far and notFound[] to store the state(s) that could not be assigned a color to. If there exists any states in notFound[] then backtrack or check if any states still remaining to assign the color to, else exit successfully.

Project 3 - Map Coloring

Dhruv Dhamani
Saurabh Burange
Akshil Battu
Kartik Ravi
Samruddhi Godbole



Options

☐ With Heuristic (min remaining values first)

Type: DFS Only

Country: United States

Reset

Click on a state to start DFS!

Started from: Washington

Tried: 727 colors.

Backtracked: 124 times.

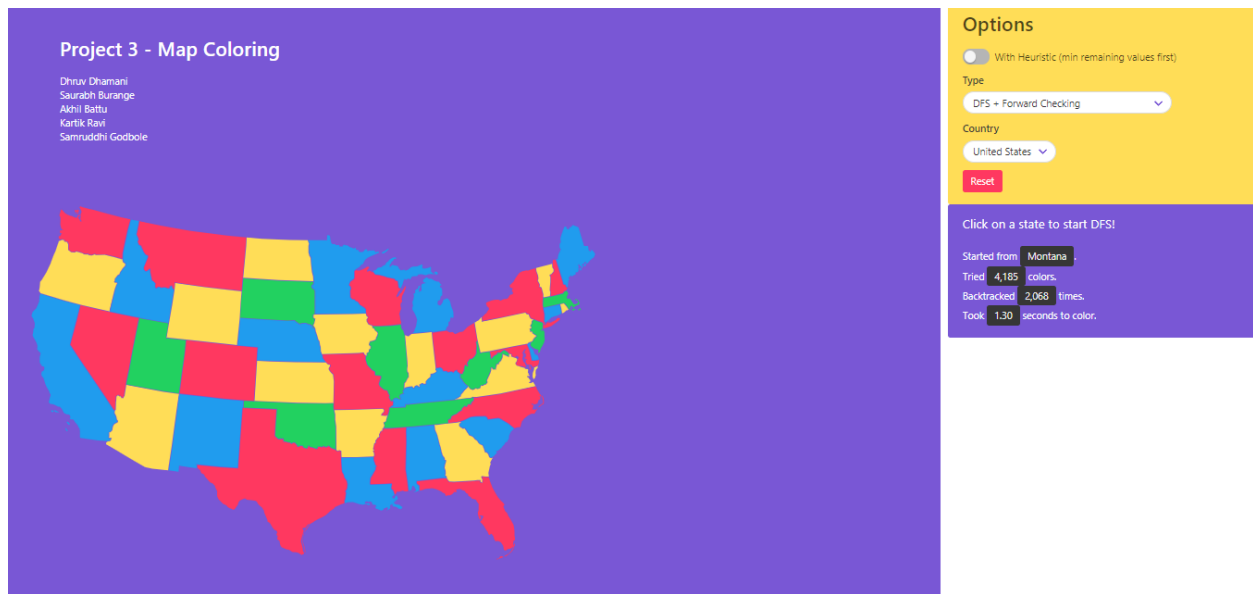
Took: 0.41 seconds to color.

DFS + Forward checking: dfsf

This approach starts by assigning color to initial state and limits the legal colors to its neighboring states.

It imposes the constraints to the neighboring states by making the current state's color 'false' from the state of legal colors. It continues the coloring till either all states are colored or there are no more colors to be assigned to state, in that case it backtracks to previous states and resets the previously assigned constraints. This approach is complete and finds the solution (if exists).

Similar to dfs, it keeps the track of the country name, colored[] , stack[] and notFound[] arrays.

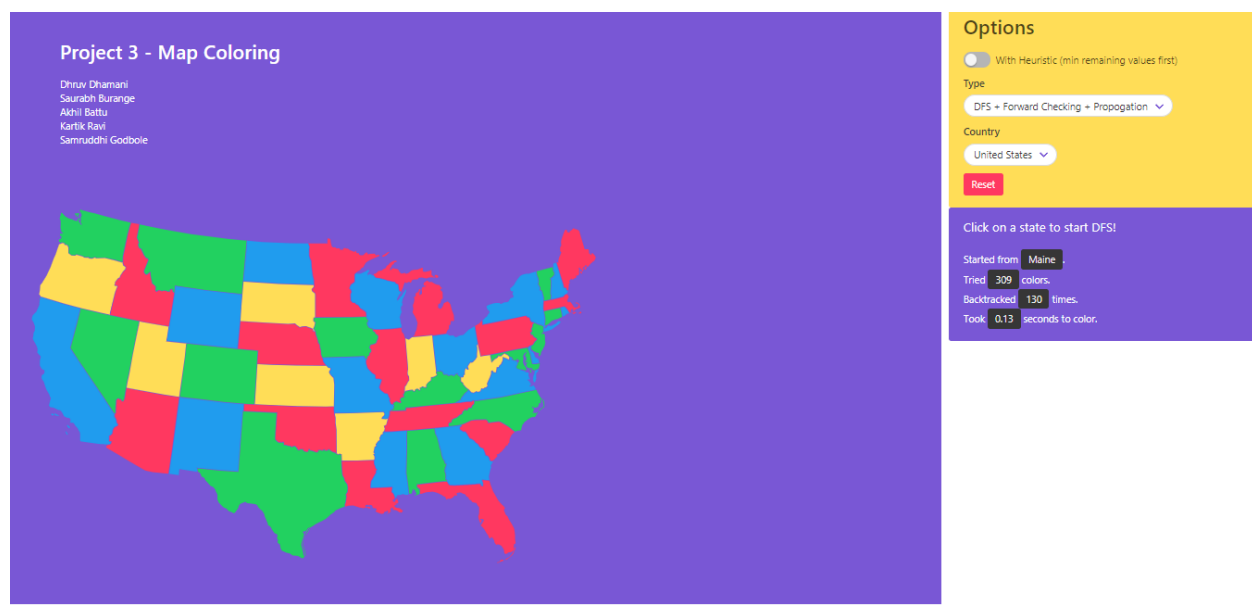


DFS + Forward checking + propagation dfsfp

This approach along with DFS and Forward checking, checks after assigning constraint to the neighboring states. If any of the states has only 1 color left in the legal colors set,

If yes, then It assigns that color to the state. This approach is complete and finds the solution (if exists).

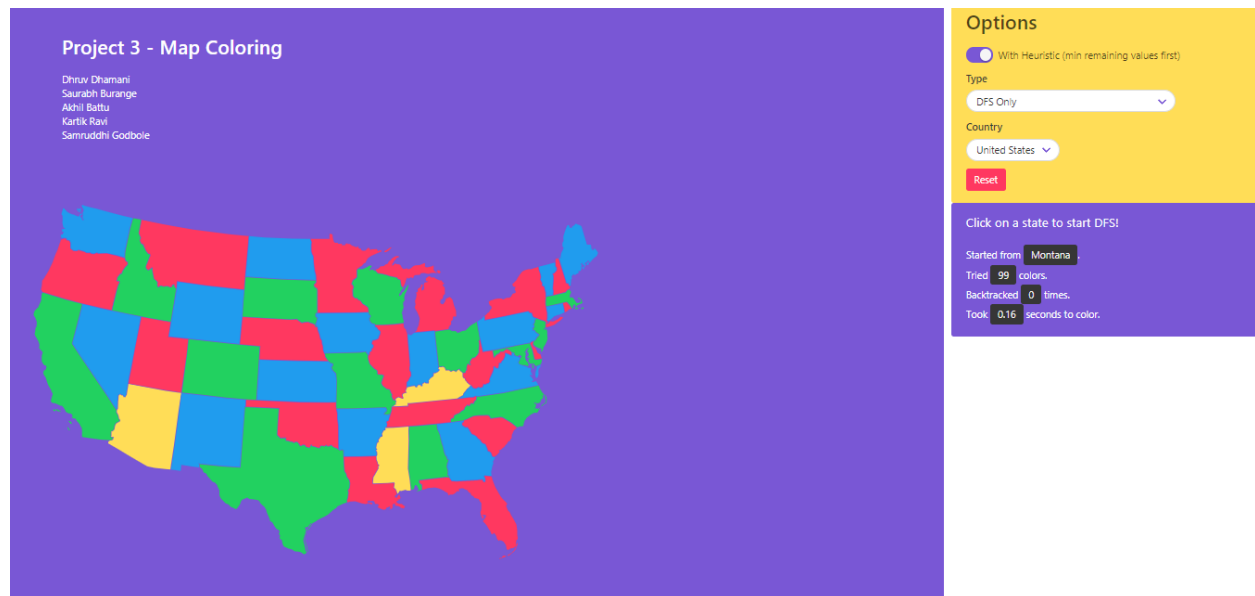
Keeps track of country name [e.g. USA] along with stack[] that contains all the states of the map. Colored[] array to store states that have been colored so far and notFound[] to store the state(s) that could not be assigned a color to. If there exists any states in notFound[] then backtrack or check if any states still remaining to assign the color to, else exit successfully.



DFS with heuristic: dfsH

It is similar to finding solution using dfs, but in addition it uses heuristic value and It rearranges the states using priority queue ordered based on min remaining value. This approach is complete and finds the solution (if exists).

Keeps track of country name [e.g. USA] along with stack[] that contains all the states of the map. Colored[] array to store states that have been colored so far and notFound[] to store the state(s) that could not be assigned a color to. If there exists any states in notFound[] then backtrack or check if any states still remaining to assign the color to, else exit successfully.



DFS + Forward checking with heuristic: dfsfH

It is similar to finding solution using dfs+forward checking, but in addition it uses heuristic value and It rearranges the states using priority queue ordered based on min remaining value. This approach is complete and finds the solution (if exists).

Keeps track of country name [e.g. USA] along with stack[] that contains all the states of the map. Colored[] array to store states that have been colored so far and notFound[] to store the state(s) that could not be assigned a color to. If there exists any states in notFound[] then backtrack or check if any states still remaining to assign the color to, else exit successfully.



DFS + Forward checking + propagation with heuristic: dfsfpH

It is similar to finding solution using dfs+forward+propagation, but in addition it uses heuristic value and It rearranges the states using priority queue ordered based on min remaining value. This approach is complete and finds the solution (if exists).

Keeps track of country name [e.g. USA] along with stack[] that contains all the states of the map. Colored[] array to store states that have been colored so far and notFound[] to store the state(s) that could not be assigned a color to. If there exists any states in notFound[] then backtrack or check if any states still remaining to assign the color to, else exit successfully.

Project 3 - Map Coloring

Dhruv Dhamani
Saurabh Burange
Aksh Gattu
Karik Ravi
Samrudhhi Godbole

Finished coloring the map!



Options

☒ With Heuristic (min remaining values first)

Type

DFS + Forward Checking + Propagation

Country

United States

Reset

Click on a state to start DFS!

Started from: Pennsylvania

Tried: 104 colors.

Backtracked: 0 times.

Took: 0.10 seconds to color.

Other Methods/ functions:

isValid

This function checks for the current country and its state if the coloring constraint is violated i.e. if any of its neighbors have same color. If the constraint is violated, function returns false else it returns true.

allColored

Checks if all the states in the current country are colored. And if any of the states are still not colored /white returns false. Else, returns true

hasSingletonDomain

This function is used for propagation. It returns true if there's only a single legal color available for a particular state.

setColor

takes color as the input and sets 'white' if no color or the assigns the color to the state.

Starting

Prints the starting position or state of the execution.

Finished

After successful execution it prints the message 'Finished coloring the map!',

Limitations:

- 1> This project is implemented for only USA country. However, it can be extended to implement any other country(ies). Map of USA has more states and is complicated and hence this implementation demonstrates efficiently execution of all the mentioned algorithm/ approaches to solve map coloring problem.
- 2> We considered 4 colors to assign to a map namely, Blue Red Green Yellow.
- 3> Without Heuristic the values of the numbers of colors tried, time taken, and backtracked are higher when compared to the values with heuristic.

Below table shows the analysis of the map coloring for 3 different states with and without Heuristic

	S.No	State	Number of colors tried	Number of times Backtracked	Time taken
Without Heuristic	DFS	North Carolina	1347	248	0.21s
		Texas	1179	215	0.22s
		Virginia	782	135	0.13s
	DFS+FC	North Carolina	1077	514	0.16s
		Texas	351	151	0.05s
		Virginia	499	225	0.07s
	DFS+FC+SP	North Carolina	285	118	0.04s
		Texas	227	89	0.03s
		Virginia	291	121	0.04s
With Heuristic	DFS	North Carolina	153	10	0.04s
		Texas	108	0	0.03s
		Virginia	104	0	0.03s
	DFS + FC	North Carolina	107	0	0.02s
		Texas	49	0	0.02s
		Virginia	49	0	0.02s
	DFS + FC + SP	North Carolina	153	10	0.04s
		Texas	108	0	0.03s
		Virginia	104	0	0.03s

Source code:

```
<!DOCTYPE html>

<html>

<head>

  <meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="stylesheet" href="https://unpkg.com/buefy/dist/buefy.min.css">

  <title>Map Coloring - Project 3</title>

</head>


<body>

<div id="app" class="columns">

<div class="column is-three-quarters">

  <section class="hero is-primary">

    <div class="hero-body">

      <div class="container">

        <h1 class="title">

          Project 3 - Map Coloring

        </h1>

        <h3>

          <ul><li v-for="name in students">{{ name }}</li>

          </ul>

        </h3>

      </div>

    </div>

  </div>

</div>
```

```

<div class="hero-foot">
  <div class="section">
    <svg xmlns="http://www.w3.org/2000/svg" width="100%"
height="593" v-if="countryName == 'usa'">
      <g id="outlines" class="state">
        <path id="AL" d="M628.5 466.4l.6.2 1.3-2.7 1.5-4.4 2.3.6
3.1 6v1l-2.7 1.9 2.7.3 5.2-2.5-.3-7.6-2.5-1.8-2-2 .4-4 10.5-1.5 25.7-2.9 6.7-
.6 5.6.1-.5-2.2-1.5-.8-.9-1.1 1-2.6-.4-5.2-1.6-4.5.8-5.1 1.7-4.8-.2-1.7-1.8-
.7-.5-3.6-2.7-3.4-2-6.5-1.4-6.7-1.8-5-3.8-16-3.5-7.9-.8-5.6.1-2.2-9 .8-23.4
2.2-12.2.8-.2 6.4.2 16.7-.7 31-.3 14.1 2.8 18.8 1.6 14.7z">
          <title>Alabama</title>
        </path>
        <path id="AR" d="M587.3 346.1l-6.4-.7.9-3.1 3.1-2.6.6-
2.3-1.8-2.9-31.9 1.2-23.3.7-23.6.3 1.5 6.9.1 8.5 1.4 10.9.3 38.2 2.1 1.6 3-
1.2 2.9 1.2.4 10.1 25.2-.2 26.8-.8.9-1.9-.3-3.8-1.7-3.1 1.5-1.4-1.4-2.2.7-2.4
1.1-5.9 2.7-2.3-.8-2.2 4-5.6 2.5-1.1-.1-1.7-.5-1.7 2.9-5.8 2.5-1.1.2-3.3 2.1-
1.4.9-4.1-1.4-4 4.2-2.4.3-2.1 1.2-4.2.9-3.1z">
          <title>Arkansas</title>
        </path>
        <path id="AZ" d="M135.1 389.7l-.3 1.5.5 1 18.9 10.7 12.1
7.6 14.7 8.6 16.8 10 12.3 2.4 25.4 2.7 6-39.6 7-53.1 4.4-31-24.6-3.6-60.7-11-
.2 1.1-2.6 16.5-2.1 3.8-2.8-.2-1.2-2.6-2.6-.4-1.2-1.1-1.1.1-2.1 1.7-.3 6.8-.3
1.5-.5 12.5-1.5 2.4-.4 3.3 2.8 5 1.1 5.5.7 1.1 1.1.9-.4 2.4-1.7 1.2-3.4 1.6-
1.6 1.8-1.6 3.6-.5 4.9-3 2.9-1.9.9-.1 5.8-.6 1.6.5.8 3.9.4-.9 3-1.7 2.4-
3.7.4z">
          <title>Arizona</title>
        </path>
        <path id="CA" d="M122.7 385.9l-19.7-2.7-10-1.5-.5-1.8v-
9.4l-.3-3.2-2.6-4.2-.8-2.3-3.9-4.2-2.9-4.7-2.7-.2-3.2-.8-.3-1 1.5-.6-.6-3.2-
1.5-2.1-4.8-.8-3.9-2.1-1.1-2.3-2.6-4.8-2.9-3.1H57l-3.9-2.1-4.5-1.8-4.2-.5-
2.4-2.7.5-1.9 1.8-7.1.8-1.9v-2.4l-1.6-1-.5-2.9-1.5-2.6-3.4-5.8-1.3-3.1-1.5-
4.7-1.6-5.3-3.2-4.4-.5-2.9.8-3.9h1.1l2.1-1.6 1.1-3.6-1-2.7-2.7-.5-1.9-2.6-
2.1-3.7-.2-8.2.6-1.9.6-2.3.5-2.4-5.7-6.3V236l.3-.5.3-3.2-1.3-4-2.3-4.8-2.7-
4.5-1.8-3.9 1-3.7.6-5.8 1.8-3.1.3-6.5-1.1-3.6-1.6-4.2L14 184l.8-3.2 1.5-4.2
1.8-.8.3-1.1 3.1-2.6 5.2-11.8.2-7.4 1.69-4.9 38.69 11.8 25.6 6.6-8 31.3-8.67
33.1L88.84 250 131 312.3l17.1 26.1-.4 3.1 2.8 5.2 1.1 5.4 1 1.5.7.6-.2 1.4-
1.4 1-3.4 1.6-1.9 2.1-1.7 3.9-.5 4.7-2.6 2.5-2.3 1.1-.1 6.2-.6 1.9 1 1.7 3
.3-.4 1.6-1.4 2-3.9.6zM48.8 337l1.3 1.5-.2 1.3-3.2-.1-.6-1.2-.6-1.5zm1.9
0l1.2-.6 3.6 2.1 3.1 1.2-.9.6-4.5-.2-1.6-1.6zm20.7 19.8l1.8 2.3.8 1 1.5.6.6-

```

1.5-1-1.8-2.7-2-1.1.2v1.2zm-1.4 8.7l1.8 3.2 1.2 1.9-1.5.2-1.3-1.2s-.7-1.5-.7-1.9v-2.2z">

<title>California</title>

</path>

<path id="CO" d="M380.2 235.5l-36-3.5-79.1-8.6-2.2 22.1-750.4-1.9 13.7 34 3.9 37.5 4.4 34.7 3 14.3.6z">

<title>Colorado</title>

</path>

<path id="CT" d="M852 190.9l3.6-3.2 1.9-2.1.8.6 2.7-1.5 5.2-1.1 7-3.5-.6-4.2-.8-4.4-1.6-6-4.3 1.1-21.8 4.7.6 3.1 1.5 7.3v8.3l-.9 2.1 1.7 2.2z">

<title>Connecticut</title>

</path>

<path id="DE" d="M834.4 247.2l-1 .5-3.6-2.4-1.8-4.7-1.9-3.6-2.3-1-2.1-3.6.5-2 .5-2.3.1-1.1-.6.1-1.7 1-2 1.7-.2.3 1.4 4.1 2.3 5.6 3.7 16.1 5-.3 6-1.1z">

<title>Delaware</title>

</path>

<path id="FL" d="M750.2 445.2l-5.2-.7-.7.8 1.5 4.4-.4 5.2-4.1-1-.2-2.8H737l-5.3.7-32.4 1.9-8.2-.3-1.7-1.7-2.5-4.2H681l-6.6.5-35.4 4.2-.3 2.8 1.6 1.6 2.9 2 .3 8.4 3.3-.6 6-2.1 6-.5 4.4-.6 7.6 1.8 8.1 3.9 1.6 1.5 2.9 1.1 1.6 1.9.3 2.7 3.2-1.3h3.9l3.6-1.9 3.7-3.6 3.1.2.5-1.1-.8-1 .2-1.9 4-.8h2.6l2.9 1.5 4.2 1.5 2.4 3.7 2.7 1 1.1 3.4 3.4 1.6 1.6 2.6 1.9.6 5.2 1.3 1.3 3.1 3 3.7v9.5l-1.5 4.7.3 2.7 1.3 4.8 1.8 4 .8-.5 1.5-4.5-2.6-1-.3-.6 1.6-.6 4.5 1 .2 1.6-3.2 5.5-2.1 2.4 3.6 3.7 2.6 3.1 2.9 5.3 2.9 3.9 2.1 5 1.8.3 1.6-2.1 1.8 1.1 2.6 4 .6 3.6 3.1 4.4.8-1.3 3.9.3 3.6 2.3 3.4 5.2.8 3.4.3 2.9 1.1 1 1.3.5 2.4-1 1.5-1.6 3.9-.2 3.1-1.5 2.7-3.2-.5-1.9-.3-2.4.6-1.9-.3-1.9 2.4-1.3.3-3.4-.6-1.8-.5-12-1.3-7.6-4.5-8.2-3.6-5.8-2.6-5.3-2.9-2.9-2.9-7.4.7-1.4 1.1-1.3-1.6-2.9-4-3.7-4.8-5.5-3.7-6.3-5.3-9.4-3.7-9.7-2.3-7.3zm17.7 132.7l2.4-.6 1.3-.2 1.5-2.3 2.3-1.6 1.3.5 1.7.3.4 1.1-3.5 1.2-4.2 1.5-2.3 1.2zm13.5-5l1.2 1.1 2.7-2.1 5.3-4.2 3.7-3.9 2.5-6.6 1-1.7.2-3.4-.7.5-1 2.8-1.5 4.6-3.2 5.3-4.4 4.2-3.4 1.9z">

<title>Florida</title>

</path>

<path id="GA" d="M750.2 444.2l-5.6-.7-1.4 1.6 1.6 4.7-.3 3.9-2.2-.6-.2-3h-5.2l-5.3.7-32.3 1.9-7.7-.3-1.4-1.2-2.5-4.3-.8-3.3-1.6-.9-.5-.5.9-2.2-.4-5.5-1.6-4.5.8-4.9 1.7-4.8-.2-2.5-1.9-.7-.4-3.2-2.8-3.5-1.9-6.2-1.5-7-1.7-4.8-3.8-16-3.5-8-.8-5.3.1-2.3 3.3-.3 13.6-1.6 18.6-2 6.3-1.1.5 1.4-

2.2.9-.9 2.2.4 2 1.4 1.6 4.3 2.7 3.2-.1 3.2 4.7.6 1.6 2.3 2.8.5 1.7 4.7 1.8 3
2.2 2.3 3 2.3 1.3 2 1.8 1.4 2.7 2.1 1.9 4.1 1.8 2.7 6 1.7 5.1 2.8.7 2.1 1.9 2
5.7 2.9 1.6 1.7-.8.4 1.2-3.3 6.2.5 2.6-1.5 4.2-2.3 10 .8 6.3z">

<title>Georgia</title>

</path>

<path id="IA" d="M556.8 183.6l2.1 2.1.3.7-2 3 .3 4 2.6
4.1 3.1 1.6 2.4.3.9 1.8.2 2.4 2.5 1 .9 1.1.5 1.6 3.8 3.3.6 1.9-.7 3-1.7 3.7-
.6 2.4-2.1 1.6-1.6.5-5.7 1.5-1.6 4.8.8 1.8 1.7 1.5-.2 3.5-1.9 1.4-.7
1.8v2.4l-1.4.4-1.7 1.4-.5 1.7.4 1.7-1.3 1-2.3-2.7-1.4-2.8-8.3.8-10 .6-49.2
1.2-1.6-4.3-.4-6.7-1.4-4.2-.7-5.2-2.2-3.7-1-4.6-2.7-7.8-1.1-5.6-1.4-1.9-1.3-
2.9 1.7-3.8 1.2-6.1-2.7-2.2-.3-2.4.7-2.4 1.8-.3 61.1-.6 21.2-.7z">

<title>Iowa</title>

</path>

<path id="ID" d="M175.3 27.63l-4.8 17.4l166 65.9l-3.4
16.22-.4 9.67 1.2 4.44 3.5 2.66-.2 3.91-3.9 4.4-4.5 6.6-.9 2.9-1.2 1.1-1.8.8-
4.3 5.3-.4 3.1-.4 1.1.6 1 2.6-.1 1.1 2.3-2.4 5.8-1.2 4.2-8.8 35.3 20.7 4.5
39.5 7.9 34.8 6.1 4.9-29.2 3.8-24.1-2.7-2.4-.4-2.6-.8-1.1-2.1 1-.7 2.6-3.2.5-
3.9-1.6-3.8.1-2.5.7-3.4-1.5-2.4.2-2.4 2-2-1.1-.7-4 .7-2.9-2.5-2.9-3.3-2.6-
2.7-13.1-.1-4.7-.3-.1-.2.4-5.1 3.5-1.7-.2-2.9-3.4-.2-3.1 7-17.13-.4-1.94-3.4-
1.15-.6-1.18-2.6-3.46-4.6-10.23-3.2-1.53-2-4.95 1.3-4.63-3.2-7.58 4.4-
21.52z">

<title>Idaho</title>

</path>

<path id="IL" d="M618.7 214.3l-.8-2.6-1.3-3.7-1.6-1.8-
1.5-2.6-.4-5.5-15.9 1.8-17.4 1h-12.3l.2 2.1 2.2.9 1.1 1.4.4 1.4 3.9 3.4.7
2.4-.7 3.3-1.7 3.7-.8 2.7-2.4 1.9-1.9.6-5.2 1.3-1.3 4.1.6 1.1 1.9 1.8-.2 4.3-
2.1 1.6-.5 1.3v2.8l-1.8.6-1.4 1.2-.4 1.2.4 2-1.6 1.3-.9 2.8.3 3.9 2.3 7 7 7.6
5.7 3.7v4.4l.7 1.2 6.6.6 2.7 1.4-.7 3.5-2.2 6.2-.8 3 2 3.7 6.4 5.3 4.8.8 2.2
5.1 2 3.4-.9 2.8 1.5 3.8 1.7 2.1 1.6-.3 1-2.2 2.4-1.7 2.8-1 6.1 2.5.5-.2v-
1.1l-1.2-2.7.4-2.8 2.4-1.6 3.4-1.2-.5-1.3-.8-2 1.2-1.3 1-2.7v-4l.4-4.9 2.5-3
1.8-3.8 2.5-4-.5-5.3-1.8-3.2-.3-3.3.8-5.3-.7-7.2-1.1-15.8-1.4-15.3-.9-11.7z">

<title>Illinois</title>

</path>

<path id="IN" d="M622.9 216.1l1.5 1 1.1-.3 2.1-1.9 2.5-
1.8 14.3-1.1 18.4-1.8 1.6 15.5 4.9 42.6-.6 2.9 1.3 1.6.2 1.3-2.3 1.6-3.6 1.7-
3.2.4-.5 4.8-4.7 3.6-2.9 4 .2 2.4-.5 1.4h-3.5l-1.4-1.7-5.2 3 .2 3.1-.9.2-.5-
.9-2.4-1.7-3.6 1.5-1.4 2.9-1.2-.6-1.6-1.8-4.4.5-5.7 1-2.5 1.3v-2.6l.4-4.7
2.3-2.9 1.8-3.9 2.7-4.2-.5-5.8-1.8-3.1-.3-3.2.8-5.3-.7-7.1-.9-12.6-2.5-
30.1z">

```

<title>Indiana</title>

</path>

<path id="KS" d="M485.9 259.5l-43.8-.6-40.6-1.2-21.7-.9-
4.3 64.8 24.3 1 44.7 2.1 46.3.6 12.6-.3.7-35-1.2-11.1-2.5-2-2.4-3-2.3-3.6.6-3
1.7-1.4v-2.1l-.8-.7-2.6-.2-3.5-3.4z">

<title>Kansas</title>

</path>

<path id="KY" d="M607.2 331.8l12.6-.7.1-4.1h4.3l30.4-3.2
45.1-4.3 5.6-3.6 3.9-2.1.1-1.9 6-7.8 4.1-3.6 2.1-2.4-3.3-2-2.5-2.7-3-3.8-.5-
2.2-2.6-1.4-.9-1.9-.2-6.1-2.6-2-1.9-1.1-.5-2.3-1.3.2-2 1.2-2.5 2.7-1.9-1.7-
2.5-.5-2.4 1.4h-2.3l-1.8-2-5.6-.1-1.8-4.5-2.9-1.5-2.1.8-4.2.2-.5 2.1 1.2
1.5.3 2.1-2.8 2-3.8 1.8-2.6.4-.5 4.5-4.9 3.6-2.6 3.7.2 2.2-.9 2.3-4.5-.1-1.3-
1.3-3.9 2.2.2 3.3-2.4.6-.8-1.4-1.7-1.2-2.7 1.1-1.8 3.5-2.2-1-1.4-1.6-3.7.4-
5.6 1-2.8 1.3-1.2 3.4-1 1 1.5 3.7-4.2 1.4-1.9 1.4-.4 2.2 1.2 2.4v2.2l-1.6.4-
6.1-2.5-2.3.9-2 1.4-.8 1.8 1.7 2.4-.9 1.8-.1 3.3-2.4 1.3-2.1 1.7z">

<title>Kentucky</title>

</path>

<path id="LA" d="M526.9 485.9l18.1-.3 10.3 3.6 6.5 1.1
3.7-1.5 3.2 1.1 3.2 1 .8-2.1-3.2-1.1-2.6.5-2.7-1.6.8-1.5 3.1-1 1.8 1.5 1.8-1
3.2.6 1.5 2.4.3 2.3 4.5.3 1.8 1.8-.8 1.6-1.3.8 1.6 1.6 8.4 3.6 3.6-1.3 1-2.4
2.6-.6 1.8-1.5 1.3 1 .8 2.9-2.3.8.6.6 3.4-1.3 2.3-3.4.8-.5-2.1-.3.8-1.6-.2-
1.5 2.1-.5 1.1-1.3.6.8.6 3.1 4.2.6 4 1.9 1 1.5h2.9l1.1 1 2.3-3.1v4.93h-1.3l-
3.4-2.7-5.8-.8-3.2-2.3 1.1-2.7 2.3.3.2-.6-1.8-1v-.5h3.2l1.8-3.1-1.3-1.9-.3-
2.7-1.5.2-1.9 2.1-.6 2.6-3.1-.6-1-1.8 1.8-1.9 1.9-1.7-2.2-6.5-3.4-3.4 1-7.3-
.2-.5-1.3.2-33.1 1.4-.8-2.4.8-8.5 8.6-14.8-.9-2.6 1.4-.4.4-2-2.2-2 .1-1.9-2-
4.5-.4-5.1.1-.7-26.4.8-25.2.1.4 9.7.7 9.5.5 3.7 2.6 4.5.9 4.4 4.3 6 .3
3.1.6.8-.7 8.3-2.8 4.6 1.2 2.4-.5 2.6-.8 7.3-1.3 3 .2 3.7z">

<title>Louisiana</title>

</path>

<path id="MA" d="M887.5 172.5l-.5-2.3.8-1.5 2.9-1.5.8
3.1-.5 1.8-2.4 1.5v1l1.9-1.5 3.9-4.5 3.9-1.9 4.2-1.5-.3-2.4-1-2.9-1.9-2.4-
1.8-.8-2.1.2-.5.5 1 1.3 1.5-.8 2.1 1.6.8 2.7-1.8 1.8-2.3 1-3.6-.5-3.9-6-2.3-
2.6h-1.8l-1.1.8-1.9-2.6.3-1.5 2.4-5.2-2.9-4.4-3.7 1.8-1.8 2.9-18.3 4.7-13.8
2.5-.6 10.6.7 4.9 22-4.8 11.2-2.8 2 1.6 3.4 4.3 2.9 4.7zm12.5 1.4l2.2-.7.5-
1.7 1 .1 1 2.3-1.3.5-3.9.1zm-9.4.8l2.3-2.6h1.6l1.8 1.5-2.4 1-2.2 1z">

<title>Massachusetts</title>

</path>

```

```
<path id="MD" d="M834.8 264.111.7-3.8.5-4.8-6.3 1.1-
5.8.3-3.8-16.8-2.3-5.5-1.5-4.6-22.2 4.3-37.6 7.6 2 10.4 4.8-4.9 2.5-.7 1.4-
1.5 1.8-2.7 1.6.7 2.6-.2 2.6-2.1 2-1.5 2.1-.6 1.5 1.1 2.7 1.4 1.9 1.8 1.3 1.4
4.8 1.6-.6 2.9 5.8 2.1 2.1-2.6 3.7 2.5-2.1 3.3-.7 3.3-1.8 2.6v2.11.3.8 2 1.3
3.4 1.1 4.3-.1 3.1 1 2.1.3 1-2.1-1.5-2.1v-1.81-2.4-2.1-2.1-5.5 1.3-5.3-.2-
2.1-1.3-1.3s1.5-1.6 1.5-2.3c0-.6.5-2.1.5-2.111.9-1.3 1.9-1.6.5 1-1.5 1.6-1.3
3.7.3 1.1 1.8.3.5 5.5-2.1 1 .3 3.6.5-.2 1.1-1.9 1.6 1.8-1.6 1.3-.3 3.4 2.6
3.4 3.9.5 1.6-.8 3.2 4.2 1 .4zm-14.5.211.1 2.5.2 1.8 1.1 1.9s.9-.9.9-1.2c0-
.3-.7-3.1-.7-3.11-.7-2.3z">
```

<title>Maryland</title>

</path>

```
<path id="ME" d="M865.8 91.911.5.4v-2.61.8-5.5 2.6-4.7
1.5-4-1.9-2.4v-61.8-1 .8-2.7-.2-1.5-.2-4.8 1.8-4.8 2.9-8.9 2.1-
4.2h1.311.3.2v1.111.3 2.3 2.7.6.8-.8v-114-2.9 1.8-1.8 1.5.2 6 2.4 1.9 1 9.1
29.9h61.8 1.9.2 4.8 2.9 2.3h.81.2-.5-.5-1.1 2.8-.5 1.9 2.1 2.3 3.7V851-2.1
4.7-1.9.6-3.4 3.1-4.8 5.5h-1.3c-.6 0-1-2.1-1-2.11-1.8.2-1 1.5-2.4 1.5-1 1.5
1.6 1.5-.5.6-.5 2.7-1.9-.2v-1.61-.3-1.3-1.5.3-1.8-3.2-2.1 1.3 1.3 1.5.3 1.1-
.8 1.3.3 3.1.2 1.6-1.6 2.6-2.9.5-.3 2.9-5.3 3.1-1.3.5-1.6-1.5-3.1 3.6 1 3.2-
1.5 1.3-.2 4.4-1.1 6.3-2.2-.9-.5-3.1-4-1.1-.2-2.5-11.7-37.43zm36.5 15.611.5-
1.5 1.4 1.1.6 2.4-1.7.9zm6.7-5.911.8 1.9s1.3.1 1.3-.2c0-.3.2-2 .2-21.9-.8-.8-
1.8-2 .7z">
```

<title>Maine</title>

</path>

```
<path id="MI" d="M644.5 211119.1-1.9.2 1.1 9.9-1.5 12-
1.7.1-.6.2-1.5 2.1-3.7 2-1.7-.2-5.1 1.6-1.6 1.1-.3.2-3.6 1.5-3 1.1.6.2.6.8.2
1.9-1-.4-9.1-3.2-8.2-2.3-9.1-2.4-3.2-2.6-1.8-1.6 1.1-3.9 1.8-1.9 5-2.7 3.7-
1.1.6-1.5-.6s-2.6-1.5-2.4-2.1c.2-.6.5-.5 .5-513.4-1.3.8-3.4.6-2.6 2.4-1.6-.3-
10-1.6-2.3-1.3-.8-.8-2.1.8-.8 1.6.3.2-1.6-2.6-2.2-1.3-2.6h-2.61-4.5-1.5-5.5-
3.4h-2.71-.6.6-1-.5-3.1-2.3-2.9 1.8-2.9 2.3.3 3.6 1 .3 2.1.5.5.8-2.6.8-2.6.3-
1.5 1.8-.3 2.1.3 1.6.3 5.5-3.6 2.1-.6-.2v-4.211.3-2.4.6-2.4-.8-.8-1.9.8-1
4.2-2.7 1.1-1.8 1.9-.2 1 .6.8-.6 2.6-2.3.5v1.11.8 2.4-1.1 6.1-1.6 4 .6 4.7.5
1.1-.8 2.4-.3.8-.3 2.7 3.6 6 2.9 6.5 1.5 4.8-.8 4.7-1 6-2.4 5.2-.3 2.7-3.2
3.1zm-33.3-72.41-1.3-1.1-1.8-10.4-3.7-1.3-1.7-2.3-12.6-2.8-2.8-1.1-8.1-2.2-
7.8-1-3.9-5.3.7-.5 2.7-.8 3.6-2.3v-11.6-.6 6-1 2.4-1.9 4.4-2.1.2-1.3 1.9-2.9
1.8-.8 1.3-1.8 2.3-2.3 4.4-2.4 4.7-.5 1.1 1.1-.3 1-3.7 1-1.5 3.1-2.3.8-.5
2.4-2.4 3.2-.3 2.6.8.5 1-1.1 3.6-2.9 1.3 1.3h2.313.2 1 1.5 1.1 1.5 3.1 2.7
2.7 3.9-.2 1.5-1 1.6 1.3 1.6.5 1.3-.8h1.111.6-1 4-3.6 3.4-1.1 6.6-.3 4.5-1.9
2.6-1.3 1.5.2v5.71.5.3 2.9.8 1.9-.5 6.1-1.6 1.1-1.1 1.5.5v713.2 3.1 1.3.6 1.3
1-1.3.3-.8-.3-3.7-.5-2.1.6-2.3-.2-3.2 1.5h-1.81-5.8-1.3-5.2.2-1.9 2.6-7 .6-
2.4.8-1.1 3.1-1.3 1.1-.5-.2-1.5-1.6-4.5 2.4h-.61-1.1-1.6-.8.2-1.9 4.4-1 4-3.2
6.9zm-29.6-56.511.8-2.1 2.2-.8 5.4-3.9 2.3-.6.5.5-5.1 5.1-3.3 1.9-2.1.9zm86.2
32.11.6 2.5 3.2.2 1.3-1.2s-.1-1.5-.4-1.6c-.3-.2-1.6-1.9-1.6-1.91-2.2.2-1.6.2-
.3 1.1z">
```

<title>Michigan</title>

</path>

<path id="MN" d="M464.6 66.791-.6 3.91v10.27l11.6 5.03 1.9 3.32.5 9.93 1.8 13.45 1.8 7.3.4 6.4v5.3l-1.6 1.8-1.8 1.3v1.5l.9 1.7 4.1 3.5.7 3.2v35.9l60.3-.6 21.2-.7-.5-6-1.8-2.1-7.2-4.6-3.6-5.3-3.4-.9-2-2.8h-3.2l-3.5-3.8-.5-7 .1-3.9 1.5-3-.7-2.7-2.8-3.1 2.2-6.1 5.4-4 1.2-1.4-.2-8 .2-3 2.6-3 3.8-2.9 1.3-.2 4.5-5 1.8-.8 2.3-3.9 2.4-3.6 3.1-2.6 4.8-2 9.2-4.1 3.9-1.8.6-2.3-4.4.4-.7 1.1h-.6l-1.8-3.1-8.9.3-1 .8h-1l-.5-1.3-.8-1.8-2.6.5-3.2 3.2-1.6.8h-3.1l-2.6-1v-2.1l-1.3-.2-.5.5-2.6-1.3-.5-2.9-1.5.5-.5 1-2.4-.5-5.3-2.4-3.9-2.6h-2.9l-1.3-1-2.3.6-1.1 1.1-.3 1.3h-4.8v-2.1l-6.3-.3-.3-1.5h-4.8l-1.6-1.6-1.5-6.1-.8-5.5-1.9-.8-2.3-.5-.6.2-.3 8.2-30.1-.03z">

<title>Minnesota</title>

</path>

<path id="MO" d="M593.1 338.7l1.5-5.9 4.2-3.4 1.9-1v-2.9l1.7-1.6-1.1-1.6-2.4.3-2.1-2.5-1.7-4.5.9-2.6-2-3.2-1.8-4.6-4.6-.7-6.8-5.6-2.2-4.2.8-3.3 2.2-6 .6-3-1.9-1-6.9-.6-1.1-1.9v-4.1l-5.3-3.5-7.2-7.8-2.3-7.3-.5-4.2.7-2.4-2.6-3.1-1.2-2.4-7.7.8-10 .6-48.8 1.2 1.3 2.6-.1 2.2 2.3 3.6 3 3.9 3.1 3 2.6.2 1.4 1.1v2.9l-1.8 1.6-.5 2.3 2.1 3.2 2.4 3 2.6 2.1 1.3 11.6-.8 40 .5 5.7 23.7-.2 23.3-.7 32.5-1.3 2.2 3.7-.8 3.1-3.1 2.5-.5 1.8 5.2.5 4.1-1.1z">

<title>Missouri</title>

</path>

<path id="MS" d="M604.3 472.5l2.6-4.2 1.8.8 6.8-1.9 2.1.3 1.5.8h5.2l1.4-1.6-1.7-14.8-2.8-19 1-45.1-.2-16.7.2-6.3-4.8.3-19.6 1.6-13 .4-.2 3.2-2.8 1.3-2.6 5.1.5 1.6.1 2.4-2.9 1.1-3.5 5.1.8 2.3-3 2.5-1 5.7-.6 1.9 1.6 2.5-1.5 1.4 1.5 2.8.3 4.2-1.2 2.5-.2.9.4 5 2 4.5-.1 1.7 2.3 2-.7 3.1-.9.3.6 1.9-8.6 15-.8 8.2.5 1.5 24.2-.7 8.2-.7 1.9-.3.6 1.4-1 7.1 3.3 3.3 2.2 6.4z">

<title>Mississippi</title>

</path>

<path id="MT" d="M361.1 70.77l-5.3 57.13-1.3 15.2-59.1-6.6-49-7.1-1.4 11.2-1.9-1.7-.4-2.5-1.3-1.9-3.3 1.5-.7 2.5-2.3.3-3.8-1.6-4.1.1-2.4.7-3.2-1.5-3 .2-2.1 1.9-.9-.6-.7-3.4.7-3.2-2.7-3.2-3.3-2.5-2.5-12.6-.1-5.3-1.6-.8-.6 1-4.5 3.2-1.2-.1-2.3-2.8-.2-2.8 7-17.15-.6-2.67-3.5-1.12-.4-.91-2.7-3.5-4.6-10.41-3.2-1.58-1.8-4.26 1.3-4.63-3.2-7.57 4.4-21.29l222 37.3l18.4 3.4 32.3 5.3 29.3 4 29.2 3.5 30.8 3.07z">

<title>Montana</title>

</path>

```
<path class="is-warning" id="NC" d="M786.7 357.7L774
3501-3.1-.8-16.6 2.1-1.6-3-2.8-2.2-16.7.5-7.4.9-9.2 4.5-6.8 2.7-6.5 1.2-13.4
1.4.1-4.1 1.7-1.3 2.7-.7.7-3.8 3.9-2.5 3.9-1.5 4.5-3.7 4.4-2.3.7-3.2 4.1-
3.8.7 1 2.5.2 2.4-3.6 1.7-.4 2.6.3 1.8-4 2.5-2.4.5-1.8.1-3.5 4.4.1 38.5-5.6
57.5-12.3 2 4.8 3.6 6.5 2.4 2.4.6 2.3-2.4.2.8.6-.3 4.2-2.6 1.3-.6 2.1-1.3
2.9-3.7 1.6-2.4-.3-1.5-.2-1.6-1.3.3 1.3v1h1.9l.8 1.3-1.9 6.3h4.2l.6 1.6 2.3-
2.3 1.3-.5-1.9 3.6-3.1 4.8H828l-1.1-.5-2.7.6-5.2 2.4-6.5 5.3-3.4 4.7-1.9 6.5-
.5 2.4-4.7.5-5.1 1.5zm49.3-26.2l2.6-2.5 3.2-2.6 1.5-.6.2-2-.6-6.1-1.5-2.3-.6-
1.9.7-.2 2.7 5.5.4 4.4-.2 3.4-3.4 1.5-2.8 2.4-1.1 1.2z">
```

<title>North Carolina</title>

</path>

```
<path id="ND" d="M471 126.4l-.4-6.2-1.8-7.3-1.8-13.6l-.5-
9.7-1.9-3.18-1.6-5.32V70.68l.6-3.85-1.8-5.54-28.6-.59-18.6-.6-26.5-1.3-25.2-
2.16-.9 14.42-4.7 50.94 56.8 3.9 56.9 1.7z">
```

<title>North Dakota</title>

</path>

```
<path id="NE" d="M470.3 204.3l-1-2.3-.5-1.6-2.9-1.6-4.8-
1.5-2.2-1.2-2.6.1-3.7.4-4.2 1.2-6-4.1-2.2-2-10.7.6-41.5-2.4-35.6-2.2-4.3 43.7
33.1 3.3-1.4 21.1 21.7 1 40.6 1.2 43.8.6h4.5l-2.2-3-2.6-3.9.1-2.3-1.4-2.7-
1.9-5.2-.4-6.7-1.4-4.1-.5-5-2.3-3.7-1-4.7-2.8-7.9-1-5.3z">
```

<title>Nebraska</title>

</path>

```
<path id="NH" d="M881.7 141.3l1.1-3.2-2.7-1.2-.5-3.1-4.1-
1.1-.3-3-11.7-37.48-.7.08-.6 1.6-.6-.5-1-1-1.5 1.9-.2 2.29.5 8.41 1.9
2.8v4.3l-3.9 4.8-2.4.9v.7l1.1 1.9v8.6l-.8 9.2-.2 4.7 1 1.4-.2 4.7-.5 1.5 1
1.1 5.1-1.2 13.8-3.5 1.7-2.9 4-1.9z">
```

<title>New Hampshire</title>

</path>

```
<path id="NJ" d="M823.7 228.3l1.1-1.5 2.7-1.3 1.7-2.8 1.7-
2.4 3.3-3.2v-1.2l-6.1-4.1-1-2.7-2.7-.3-.1-.9-.7-2.2 2.2-1.1.2-2.9-1.3-1.3.2-
1.2 1.9-3.1V193l2.5-3.1 5.6 2.5 6.4 1.9 2.5 1.2.1 1.8-.5 2.7.4 4.5-2.1 1.9-
1.1 1 .5.5 2.7-.3 1.1-.8 1.6 3.4.2 9.4.6 1.1-1.1 5.5-3.1 6.5-2.7 4-.8 4.8-2.1
2.4h-.8l-.3-2.7.8-1-.2-1.5-4-.6-4.8-2.3-3.2-2.9-1-2z">
```

<title>New Jersey</title>

</path>

```
<path id="NM" d="M270.2 429.4l-16.7-2.6-1.2 9.6-15.8-2 6-
39.7 7-53.2 4.4-30.9 34 3.9 37.4 4.4 32 2.8-.3 10.8-1.4-.1-7.4 97.7-28.4-1.8-
38.1-3.7.7 6.3z">
```

```

<title>New Mexico</title>

</path>

<path id="NV" d="M123.1 173.6l38.7 8.5 26 5.2-10.6 53.1-
5.4 29.8-3.3 15.5-2.1 11.1-2.6 16.4-1.7 3.1-1.6-.1-1.2-2.6-2.8-.5-1.3-1.1-
1.8.1-.9.8-1.8 1.3-.3 7.3-.3 1.5-.5 12.4-1.1 1.8-16.7-25.5-42.1-62.1-12.43-19
8.55-32.6 8.01-31.3z">

<title>Nevada</title>

</path>

<path id="NY" d="M843.4 200l.5-2.7-.2-2.4-3-1.5-6.5-2-6-
2.6-.6-.4-2.7-.3-2-1.5-2.1-5.9-3.3-.5-2.4-2.4-38.4 8.1-31.6 6-.5-6.5 1.6-1.2
1.3-1.1 1-1.6 1.8-1.1 1.9-1.8.5-1.6 2.1-2.7 1.1-1-.2-1-1.3-3.1-1.8-.2-1.9-6.1
2.9-1.8 4.4-1.5 4-1.3 3.2-.5 6.3-.2 1.9 1.3 1.6.2 2.1-1.3 2.6-1.1 5.2-.5 2.1-
1.8 1.8-3.2 1.6-1.9h2.1l1.9-1.1.2-2.3-1.5-2.1-.3-1.5 1.1-2.1v-1.5h-1.8l-1.8-
.8-.8-1.1-.2-2.6 5.8-5.5.6-.8 1.5-2.9 2.9-4.5 2.7-3.7 2.1-2.4 2.4-1.8 3.1-1.2
5.5-1.3 3.2.2 4.5-1.5 7.4-2.2.7 4.9 2.4 6.5.8 5-1 4.2 2.6 4.5.8 2-.9 3.2 3.7
1.7 2.7 10.2v5.8l-.6 10.9.8 5.4.7 3.6 1.5 7.3v8.1l-1.1 2.3 2.1 2.7.5.9-1.9
1.8.3 1.3 1.3-.3 1.5-1.3 2.3-2.6 1.1-.6 1.6.6 2.3.2 7.9-3.9 2.9-2.7 1.3-1.5
4.2 1.6-3.4 3.6-3.9 2.9-7.1 5.3-2.6 1-5.8 1.9-4 1.1-1-.4z">

<title>New York</title>

</path>

<path id="OH" d="M663.8 211.2l1.7 15.5 4.8 41.1 3.9-.2
2.3-.8 3.6 1.8 1.7 4.2 5.4.1 1.8 2h1.7l2.4-1.4 3.1.5 1.5 1.3 1.8-2 2.3-1.4
2.4-.4.6 2.7 1.6 1 2.6 2 .8.2 2-.1 1.2-.6v-2.1l1.7-1.5.1-4.8 1.1-4.2 1.9-1.3
1 .7 1 1.1.7.2.4-.4-.9-2.7v-2.2l1.1-1.4 2.5-3.6 1.3-1.5 2.2.5 2.1-1.5 3-3.3
2.2-3.7.2-5.4.5-5V230l-1.2-3.2 1.2-1.8 1.3-1.2-.6-2.8-4.3-25.6-6.2 3.7-3.9
2.3-3.4 3.7-4 3.9-3.2.8-2.9.5-5.5 2.6-2.1.2-3.4-3.1-5.2.6-2.6-1.5-2.2-1.3z">

<title>Ohio</title>

</path>

<path id="OK" d="M411.9 334.9l-1.8 24.3-.9 18 .2 1.6 4
3.6 1.7.9h.9l.9-2.1 1.5 1.9 1.6.1.3-.2.2-1.1 2.8 1.4-.4 3.5 3.8.5 2.5 1 4.2.6
2.3 1.6 2.5-1.7 3.5.7 2.2 3.1 1.2.1v2.3l2.1.7 2.5-2.1 1.8.6 2.7.1.7 2.3 4.4
1.8 1.7-.3 1.9-4.2h1.3l1.1 2.1 4.2.8 3.4 1.3 3 .8 1.6-.7.7-2.7h4.5l1.9.9 2.7-
1.9h1.4l.6 1.4h3.6l2-1.8 2.3.6 1.7 2.2 3 1.7 3.4.9 1.9 1.2-.3-37.6-1.4-10.9-
.1-8.6-1.5-6.6-.6-6.8.1-4.3-12.6.3-46.3-.5-44.7-2.1-41.5-1.8-.4 10.7z">

<title>Oklahoma</title>

</path>

<path id="OR" d="M67.44 158.9l28.24 7.2 27.52 6.5 17 3.7
8.8-35.1 1.2-4.4 2.4-5.5-.7-1.3-2.5.1-1.3-1.8.6-1.5.4-3.3 4.7-5.7 1.9-.9.9-

```

.8.7-2.7.8-1.1 3.9-5.7 3.7-4 .2-3.26-3.4-2.49-1.2-4.55-13.1-3.83L132.9 851-14.8.37-1.1-1.31-5.1 1.84-4.5-.48-2.4-1.58-1.3.54-4.68-.29-1.96-1.43-4.84-1.77-1.1-.07-4.45-1.27-1.76 1.52-6.26-.24-5.31-3.85.21-9.28-2.05-3.5-4.1-.6-.7-2.5-2.4-.5-5.8 2.1-2.3 6.5-3.2 10-3.2 6.5-5 14.1-6.5 13.6-8.1 12.6-1.9 2.9-.8 8.6-1.3 6 2.71 3.5z">

<title>Oregon</title>

</path>

<path id="PA" d="M736.6 192.2L1.3-.5 5.7-5.5.7 6.9 33.5-6.5 36.9-7.8 2.3 2.3 3.1.4 2 5.6 2.4 1.9 2.8.4.1.1-2.6 3.2v3.1L-1.9 3.1-.2 1.9 1.3 1.3-.2 1.9-2.4 1.1 1 3.4.2 1.1 2.8.3.9 2.5 5.9 3.9v.4L-3.1 3-1.5 2.2-1.7 2.8-2.7 1.2-1.4.3-2.1 1.3-1.6 1.4-22.4 4.3L757 241L-11.3 1.4-3.9.7-5.1-22.4-4.3-25.9z">

<title>Pennsylvania</title>

</path>

<path id="RI" d="M873.6 175.7L-.8-4.4-1.6-6 5.7-1.5 1.5 1.3 3.4 4.3 2.8 4.4-2.8 1.4-1.3-.2-1.1 1.8-2.4 1.9-2.8 1.1z">

<title>Rhode Island</title>

</path>

<path id="SC" d="M759 413.6L-2.1-1-1.9-5.6-2.5-2.3-2.5-.5-1.5-4.6-3-6.5-4.2-1.8-1.9-1.8-1.2-2.6-2.4-2-2.3-1.3-2.2-2.9-3.2-2.4-4.4-1.7-.4-1.4-2.3-2.8-.5-1.5-3.8-5.4-3.4.1-3.9-2.5-1.2-1.2-.2-1.4.6-1.6 2.7-1.3-.8-2 6.4-2.7 9.2-4.5 7.1-.9 16.4-.5 2.3 1.9 1.8 3.5 4.6-.8 12.6-1.5 2.7.8 12.5 7.4 10.1 8.3-5.3 5.4-2.6 6.1-.5 6.3-1.6.8-1.1 2.7-2.4.6-2.1 3.6-2.7 2.7-2.3 3.4-1.6.8-3.6 3.4-2.9.2 1 3.2-5 5.3-2.3 1.6z">

<title>South Carolina</title>

</path>

<path id="SD" d="M471 181.1L-.9 3.2.4 3 2.6 2-1.2 5.4-1.8 4.1 1.5 3.3.7 1.1-1.3.1-.7-1.6-.6-2-3.3-1.8-4.8-1.5-2.5-1.3-2.9.1-3.9.4-3.8 1.2-5.3-3.8-2.7-2.4-10.9.8-41.5-2.4-35.6-2.2L354 162L2.8-34 .4-5 56.9 3.9 56.9 1.7v2.7L-1.3 1.5-2 1.5-.1 2.2 1.1 2.2 4.1 3.4.5 2.7v35.9z">

<title>South Dakota</title>

</path>

<path id="TN" d="M670.8 359.6L-13.1 1.2-23.3 2.2-37.6 2.7-11.8.4.9-.6.9-4.5-1.2-3.6 3.9-2.3.4-2.5 1.2-4.3 3-9.5.5-5.6.3-.2 12.3-.2 13.6-.8.1-3.9 3.5-.1 30.4-3.3 54-5.2 10.3-1.5 7.6-.2 2.4-1.9 1.3.3-.1 3.3-.4 1.6-2.4 2.2-1.6 3.6-2-.4-2.4.9-2.2 3.3-1.4-.2-.8-1.2-1.1.4-4.3 4-.8 3.1-4.2 2.2-4.3 3.6-3.8 1.5-4.4 2.8-.6 3.6-2.5.5-2 1.7-.2 4.8z">

<title>Tennessee</title>

</path>

<path id="TX" d="M282.8 425.6l37 3.6 29.3 1.9 7.4-97.7
54.4 2.4-1.7 23.3-1 18 .2 2 4.4 4.1 2 1.1h1.8l1.5-1.2.7.9 2.4.2 1.1-.6v-.2l1
.5-.4 3.7 4.5.7 2.4.9 4.2.7 2.6 1.8 2.8-1.9 2.7.6 2.2 3.1.8.1v2.1l3.3 1.1
2.5-2.1 1.5.5 2.1.1.6 2.1 5.2 2 2.3-.5 1.9-4h.1l1.1 1.9 4.6.9 3.4 1.3 3.2 1
2.4-1.2.7-2.3h3.6l2.1 1 3-2h.4l.5 1.4h4.7l1.9-1.8 1.3.4 1.7 2.1 3.3 1.9 3.4 1
2.5 1.4 2.7 2 3.1-1.2 2.1.8.7 2.0 .7 9.5.6 4.1 2.6 4.4.9 4.5 4.2 5.9.3
3.1.6.8-.7 7.7-2.9 4.8 1.3 2.6-.5 2.4-.8 7.2-1.3 3 .3 4.2-5.6 1.6-9.9 4.5-1
1.9-2.6 1.9-2.1 1.5-1.3.8-5.7 5.3-2.7 2.1-5.3 3.2-5.7 2.4-6.3 3.4-1.8 1.5-5.8
3.6-3.4.6-3.9 5.5-4 .3-1 1.9 2.3 1.9-1.5 5.5-1.3 4.5-1.1 3.9-.8 4.5.8 2.4 1.8
7 1 6.1 1.8 2.7-1 1.5-3.1 1.9-5.7-3.9-5.5-1.1-1.3.5-3.2-.6-4.2-3.1-5.2-1.1-
7.6-3.4-2.1-3.9-1.3-6.5-3.2-1.9-.6-2.3.6-.6.3-3.4-1.3-.6-.6-1 1.3-4.4-1.6-
2.3-3.2-1.3-3.4-4.4-3.6-6.6-4.2-2.6.2-1.9-5.3-12.3-.8-4.2-1.8-1.9-.2-1.5-6-
5.3-2.6-3.1v-1.1l-2.6-2.1-6.8-1.1-7.4-.6-3.1-2.3-4.5 1.8-3.6 1.5-2.3 3.2-1
3.7-4.4 6.1-2.4 2.4-2.6-1-1.8-1.1-1.9-.6-3.9-2.3v-.6l-1.8-1.9-5.2-2.1-7.4-
7.8-2.3-4.7v-8.1l-3.2-6.5-.5-2.7-1.6-1-1.1-2.1-5-2.1-1.3-1.6-7.1-7.9-1.3-3.2-
4.7-2.3-1.5-4.4-2.6-2.9-1.7-.5zm174.4 141.7l-.6-7.1-2.7-7.2-.6-7 1.5-8.2 3.3-
6.9 3.5-5.4 3.2-3.6.6.2-4.8 6.6-4.4 6.5-2 6.6-.3 5.2.9 6.1 2.6 7.2.5 5.2.2
1.5z">

<title>Texas</title>

</path>

<path id="UT" d="M228.4 305.9l24.6 3.6 1.9-13.7 7-50.5
2.3-22-32.2-3.5 2.2-13.1 1.8-10.6-34.7-6.1-12.5-2.5-10.6 52.9-5.4 30-3.3
15.4-1.7 9.2z">

<title>Utah</title>

</path>

<path id="VA" d="M834.7 265.2l-.2 2.8-2.9 3.8-.4 4.6.5
3.4-1.8 5-2.2 1.9-1.5-4.6.4-5.4 1.6-4.2.7-3.3-.1-1.7zm-60.3 44.6l-38.6 5.6-
4.8-.1-2.2-.3-2.5 1.9-7.3.1-10.3 1.6-6.7.6 4.1-2.6 4.1-2.3v-2.1l5.7-7.3 4.1-
3.7 2.2-2.5 3.6 4.3 3.8.9 2.7-1 2-1.5 2.4 1.2 4.6-1.3 1.7-4.4 2.4.7 3.2-2.3
1.6.4 2.8-3.2.2-2.7-.8-1.2 4.8-10.5 1.8-5.2.5-4.7.7-.2 1.1 1.7 1.5 1.2 3.9-.2
1.7-8.1 3-.6.8-2.6 2.8-2.2 1.1-2.1 1.8-4.3.1-4.6 3.6 1.4 6.6 3.1.3-5.2 3.4
1.2-.6 2.9 8.6 3.1 1.4 1.8-.8 3.3-1.3 1.3-.5 1.7.5 2.4 2 1.3 3.9 1.4 2.9 1
4.9.9 2.2 2.1 3.2.4.9 1.2-.4 4.7 1.4 1.1-.5 1.9 1.2.8-.2 1.4-2.7-.1.1 1.6 2.3
1.5.1 1.4 1.8 1.8.5 2.5-2.6 1.4 1.6 1.5 5.8-1.7 3.7 6.2z">

<title>Virginia</title>

</path>

<path id="VT" d="M832.7 111.3l2.4 6.5.8 5.3-1 3.9 2.5
4.4.9 2.3-.7 2.6 3.3 1.5 2.9 10.8v5.3l11.5-2.1-1-1.1.6-1.9.2-4.3-1-1.4.2-

4.7.8-9.3v-8.5l-1.1-1.8v-1.6l2.8-1.1 3.5-4.4v-3.6l-1.9-2.7-.3-5.79-26.1
6.79z">

<title>Vermont</title>

</path>

<path id="WA" d="M74.5 67.7l-2.3-4.3-4.1-.7-.4-2.4-2.5-
.6-2.9-.5-1.8 1-2.3-2.9.3-2.9 2.7-.3 1.6-4-2.6-1.1.2-3.7 4.4-.6-2.7-2.7-1.5-
7.1.6-2.9v-7.9l-1.8-3.2 2.3-9.4 2.1.5 2.4 2.9 2.7 2.6 3.2 1.9 4.5 2.1 3.1.6
2.9 1.5 3.4 1 2.3-.2V22l1.3-1.1 2.1-1.3.3 1.1.3 1.8-2.3.5-.3 2.1 1.8 1.5 1.1
2.4.6 1.9 1.5-.2.2-1.3-1-1.3-.5-3.2.8-1.8-.6-1.5V19l1.8-3.6-1.1-2.6L91.9
81.3-.8 1.4-.8L98 7.9l9.7 2.7 8.6 1.9 20 5.7 23 5.7 15 3.49-4.8 17.56-4.5
20.83-3.4 16.25-.4 9.18-12.9-3.72-15.3-3.47-14.5.32-1.1-1.53-5.7 2.09-3.9-
.42-2.6-1.79-1.7.65-4.15-.25-1.72-1.32-5.16-1.82-1.18-.16-4.8-1.39-1.92 1.65-
5.65-.25-4.61-3.35zm9.6-55.4l2-.2.5 1.4 1.5-1.6h2.3l.8 1.5-1.5 1.7.6.8-.7 2-
1.4.4s-.9.1-.9-.2c0-.3 1.5-2.6 1.5-2.6l-1.7-.6-.3 1.5-.7.6-1.5-2.3z">

<title>Washington</title>

</path>

<path id="WI" d="M541.4 109.9l2.9.5 2.9-.6 7.4-3.2 2.9-
1.9 2.1-.8 1.9 1.5-1.1 1.1-1.9 3.1-.6 1.9 1 .6 1.8-1 1.1-.2 2.7.8.6 1.1
1.1.2.6-1.1 4 5.3 8.2 1.2 8.2 2.2 2.6 1.1 12.3 2.6 1.6 2.3 3.6 1.2L609
138l1.6 1.4 1.5.9-1.1 2.3-1.8 1.6-2.1 4.7-1.3 2.4.2 1.8 1.5.3 1.1-1.9 1.5-
.8.8-2.3 1.9-1.8 2.7-4 4.2-6.3.8-.5.3 1-.2 2.3-2.9 6.8-2.7 5.7-.5 3.2-.6
2.6.8 1.3-.2 2.7-1.9 2.4-.5 1.8.6 3.6.6 3.4-1.5 2.6-.8 2.9-1 3.1 1.1 2.4.6
6.1 1.6 4.5-.2 3-15.9 1.8-17.5 1H567l-.7-1.5-2.9-.4-2.6-1.3-2.3-3.7-.3-3.6 2-
2.9-.5-1.4-2.1-2.2-.8-3.3-.6-6.8-2.1-2.5-7-4.5-3.8-5.4-3.4-1-2.2-2.8h-3.2l-
2.9-3.3-.5-6.5.1-3.8 1.5-3.1-.8-3.2-2.5-2.8 1.8-5.4 5.2-3.8 1.6-1.9-.2-8.1.2-
2.8 2.4-2.8z">

<title>Wisconsin</title>

</path>

<path id="WV" d="M758.9 254.3l5.8-6 2.6-.8 1.6-1.5 1.5-
2.2 1.1.3 3.1-.2 4.6-3.6 1.5-.5 1.3 1 2.6 1.2 3 3-.4 4.3-5.4-2.6-4.8-1.8-.1
5.9-2.6 5.7-2.9 2.4-.8 2.3-3 .5-1.7 8.1-2.8.2-1.1-1-1.2-2-2.2.5-.5 5.1-1.8
5.1-5 11 .9 1.4-.1 2-2.2 2.5-1.6-.4-3.1 2.3-2.8-.8-1.8 4.9-3.8 1-2.5-1.3-2.5
1.9-2.3.7-3.2-.8-3.8-4.5-3.5-2.2-2.5-2.5-2.9-3.7-.5-2.3-2.8-1.7-.6-1.3-.2-
5.6.3.1 2.4-.2 1.8-1V275l1.7-1.5.1-5.2.9-3.6 1.1-.7.4.3 1 1.1 1.7.5 1.1-1.3-
1-3.1v-1.6l3.1-4.6 1.2-1.3 2 .5 2.6-1.8 3.1-3.4 2.4-4.1.2-5.6.5-4.8v-4.9l-
1.1-3 .9-1.3.8-.7 4.3 19.3 4.3-.8 11.2-1.3z">

<title>West Virginia</title>

</path>

```

        <path id="WY" d="M353 161.91-1.5 25.4-4.4 44-2.7-.3-83.3-
9.1-27.9-3 2-12 6.9-41 3.8-24.2 1.3-11.2 48.2 7 59.1 6.5z">
            <title>Wyoming</title>
        </path>
        <g id="DC">
        </g>
    </g>
</svg>
</div>
</div>
</section>
</div>
<div class="column is-one-third">
    <div class="tile is-ancestor">
        <article class="tile is-child notification is-warning">
            <p class="title">Options</p>
            <div class="field">
                <b-switch v-model="withHeuristic">
                    With Heuristic (min remaining values first)
                </b-switch>
            </div>
            <div class="field">
                <b-field label="Type">
                    <b-select v-model="type" rounded>
                        <option value="dfs">DFS Only</option>
                        <option value="dfsfp">DFS + Forward
Checking</option>
                        <option value="dfsfp">DFS + Forward Checking
+ Propagation</option>
                    </b-select>
                </b-field>

```

```

        </div>
    <div class="field">
        <b-field label="Country">
            <b-select v-model="countryName" rounded>
                <option value="usa">United States</option>
                <!-- <option value="australia">Australia</option>
-->
            </b-select>
        </b-field>
    </div>
    <div class="field">
        <b-button class="is-danger"
onclick="document.location.reload()">Reset</b-button>
    </div>
</article>
</div>
<div class="tile is-ancestor">
    <article class="tile is-child notification is-primary">
        <p class="subtitle">
            Click on a state to start DFS!
        </p>
        <ul>
            <li v-if="startState != ''"> Started from <span class="tag
is-dark is-medium">{{ this[countryName][startState].name }}</span>.</li>
            <li>Tried <span class="tag is-dark is-medium">{{
triedColors.toLocaleString() }}</span> colors.</li>
            <li>Backtracked <span class="tag is-dark is-medium">{{
backtracked.toLocaleString() }}</span> times.</li>
            <li>Took <span class="tag is-dark is-medium">{{
(timeTaken/1000).toFixed(2) }}</span> seconds to color.</li>
        </ul>
    </article>

```

```

</div>
<!-- <div class="tile is-ancestor">
  <article class="tile is-child notification is-warning">
    <p class="subtitle">
      Logs
    </p>

  </article> -->
</div>
</div>
</div>

<script src="https://unpkg.com/vue"></script>
<script src="https://unpkg.com/buefy/dist/buefy.min.js"></script>
<script>
  new Vue({
    el: '#app',
    mounted: function () {
      this.$nextTick(function () {
        for (let key in this[this.countryName]) {
          this[this.countryName][key].setColor =
function(color) {
          this.color = color
          this.legalColors = this.legalColors.filter(c => c
!= color);
          document.getElementById(key).style.fill =
this.color;
        }

        this[this.countryName][key].constraints = []

```

```

function(allColors) {
    this[this.countryName][key].hasSingletonDomain =

        let illegalColors = new Set()

        this.constraints.forEach(con => {
            illegalColors.add(con.color)
        })

        let legalColors = this.legalColors.filter(c =>
!illegalColors.has(c))

        if (legalColors.length == 1)
            return {
                singleton: true,
                color: legalColors.pop()
            }
        else
            return {
                singleton: false
            }
    }
}

let paths = document.getElementsByTagName('path');
for (let i = 0; i < paths.length; i++) {
    paths.item(i).style.fill = 'white';
    paths.item(i).style.pointerEvents = 'visibleFill';
    paths.item(i).onclick = () => {
        this.starting(paths.item(i).id)
    }
}

```

```

if (this.type == "dfs") {
    let start = window.performance.now()
    this.startState = paths.item(i).id
    if (this.withHeuristic)
        this.dfsH(paths.item(i).id)
    else
        this.dfs(paths.item(i).id)
    let end = window.performance.now()

    this.timeTaken = end - start;
    this.finished()
}
else if (this.type == "dfsH") {
    let start = window.performance.now()
    this.startState = paths.item(i).id
    if (this.withHeuristic)
        this.dfsH(paths.item(i).id)
    else
        this.dfs(paths.item(i).id)
    let end = window.performance.now()

    this.timeTaken = end - start;
    this.finished()
}
else {
    let start = window.performance.now()
    this.startState = paths.item(i).id
    if (this.withHeuristic)
        this.dfsH(paths.item(i).id)
    else

```

```

        this.dfsfp(paths.item(i).id)
        let end = window.performance.now()

        this.timeTaken = end - start;
        this.finished()
    }

    for (let i = 0; i < paths.length; i++) {
        paths.item(i).onclick = () => {
            this.$toast.open({
                message: 'Resest the map to start
coloring again!',

                type: 'is-danger',
                duration: 2500,
                queue: true
            })
        }
    }
}

}))
},
data: {
    students: [
        "Dhruv Dhamani",
        "Saurabh Burange",
        "Akhil Battu",
        "Kartik Ravi",
        "Samruddhi Godbole"
    ],

```

```

type: "dfs",
withHeuristic: false,
triedColors: 0,
backtracked: 0,
timeTaken: 0,
startState: "",
delay: "150",
countryName: "usa",
usaColors: ['FF3860', '209CEE', '22D160', 'FFDD57'],
usa: {
  "AL": {
    name: "Alabama",
    color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
    neighbours: ['MS', 'TN', 'GA', 'FL'],
  },
  "AZ": {
    name: "Arizona",
    color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
    neighbours: ['CA','NV','UT','CO','NM']
  },
  "AR": {
    name: "Arkansas",
    color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
    neighbours: ['MO','TN','MS','LA','TX','OK']
  },

```



```

"CA": {
    name: "California",
    color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
    neighbours: ['OR','NV','AZ']
},
"CO": {
    name: "Colorado",    color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
    neighbours: ['WY','NE','KS','OK','NM','AZ','UT']
},
"CT": {
    name: "Connecticut",    color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
    neighbours: ['NY','MA','RI']
},
"DE": {
    name: "Delaware",    color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
    neighbours: ['MD','PA','NJ']
},
"DC": {
    name: "District Of Columbia",    color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
    neighbours: ['MD','VA']
},

```

```

"FL": {
    name: "Florida",    color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

    neighbours: ['AL','GA']
},
"GA": {
    name: "Georgia",    color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

    neighbours: ['FL','AL','TN','NC','SC']
},
"ID": {
    name: "Idaho",      color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

    neighbours: ['MT','WY','UT','NV','OR','WA']
},
"IL": {
    name: "Illinois",   color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

    neighbours: ['IN','KY','MO','IA','WI']
},
"IN": {name: "Indiana",    color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

    neighbours: ['MI','OH','KY','IL']
},
"IA": {
    name: "Iowa",       color: 'white',

```

```

        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['MN','WI','IL','MO','NE','SD']
    },
    "KS": {
        name: "Kansas",      color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['NE','MO','OK','CO']
    },
    "KY": {
        name: "Kentucky",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['IN','OH','WV','VA','TN','MO','IL']
    },
    "LA": {
        name: "Louisiana",   color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['TX','AR','MS']
    },
    "ME": {
        name: "Maine",       color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['NH']
    },
    "MD": {
        name: "Maryland",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

```

```

        neighbours: ['VA','WV','PA','DC','DE']
    },
    "MA": {
        name: "Massachusetts",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
        neighbours: ['RI','CT','NY','NH','VT']
    },
    "MI": {
        name: "Michigan",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
        neighbours: ['WI','IN','OH']
    },
    "MN": {
        name: "Minnesota",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
        neighbours: ['WI','IA','SD','ND']
    },
    "MS": {
        name: "Mississippi",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
        neighbours: ['LA','AR','TN','AL']
    },
    "MO": {
        name: "Missouri",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
        neighbours: ['IA','IL','KY','TN','AR','OK','KS','NE']

```

```

    },
    "MT": {
        name: "Montana",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
        neighbours: ['ND','SD','WY','ID']
    },
    "NE": {
        name: "Nebraska",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
        neighbours: ['SD','IA','MO','KS','CO','WY']
    },
    "NV": {
        name: "Nevada",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
        neighbours: ['ID','UT','AZ','CA','OR']
    },
    "NH": {
        name: "New Hampshire",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
        neighbours: ['VT','ME','MA']
    },
    "NJ": {
        name: "New Jersey",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
        neighbours: ['DE','PA','NY']
    },

```

```

"NM": {
    name: "New Mexico",    color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

    neighbours: ['AZ','UT','CO','OK','TX']
},
"NY": {
    name: "New York",      color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

    neighbours: ['NJ','PA','VT','MA','CT']
},
"NC": {
    name: "North Carolina", color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

    neighbours: ['VA','TN','GA','SC']
},
"ND": {
    name: "North Dakota",   color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

    neighbours: ['MN','SD','MT']
},
"OH": {
    name: "Ohio",           color: 'white',
    legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

    neighbours: ['PA','WV','KY','IN','MI']
},
"OK": {

```

```

        name: "Oklahoma",      color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['KS','MO','AR','TX','NM','CO']
    },
    "OR": {
        name: "Oregon",      color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['CA','NV','ID','WA']
    },
    "PA": {
        name: "Pennsylvania",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['NY','NJ','DE','MD','WV','OH']
    },
    "RI": {
        name: "Rhode Island",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['CT','MA']
    },
    "SC": {
        name: "South Carolina",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['GA','NC']
    },
    "SD": {
        name: "South Dakota",    color: 'white',

```

```

        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['ND','MN','IA','NE','WY','MT']
    },
    "TN": {
        name: "Tennessee",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['KY','VA','NC','GA','AL','MS','AR','MO']
    },
    "TX": {
        name: "Texas",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['NM','OK','AR','LA']
    },
    "UT": {
        name: "Utah",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['ID','WY','CO','NM','AZ','NV']
    },
    "VT": {
        name: "Vermont",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

        neighbours: ['NY','NH','MA']
    },
    "VA": {
        name: "Virginia",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],

```



```

        neighbours: ['NC','TN','KY','WV','MD','DC']
    },
    "WA": {
        name: "Washington",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
        neighbours: ['ID','OR']
    },
    "WV": {
        name: "West Virginia",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
        neighbours: ['OH','PA','MD','VA','KY']
    },
    "WI": {
        name: "Wisconsin",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
        neighbours: ['MI','MN','IA','IL']
    },
    "WY": {
        name: "Wyoming",    color: 'white',
        legalColors: ['FF3860', '209CEE', '22D160',
'FFDD57'],
        neighbours: ['MT','SD','NE','CO','UT','ID']
    },
    },
    methods: {
        isValid: function() {
            let country = this.countryName;

```

```

        let valid = true
        for (let key in this[country]) {
            if (this[country][key].color == 'white')
                continue

            this[country][key].neighbours.forEach(n => {
                if (this[country][n].color == 'white')
                    return

                if (this[country][n].color ==
this[country][key].color) {
                    valid = false
                }
            })
        }

        return valid
    },
    allColored: function() {
        let country = this.countryName;

        for (let key in this[country]) {
            if (this[country][key].color == 'white')
                return false
        }

        return true;
    },
    starting: function (state) {
        this.$toast.open({

```

```

        message: 'Starting from ' +
this[this.countryName][state].name,
        queue: true,
        duration: 2500
    })
},
finished: function () {
    this.$toast.open({
        message: 'Finished coloring the map!',
        type: 'is-success',
        queue: true,
        duration: 2500
    })
},
dfs: function (state) { // Only, No heuristic
    let country = this.countryName;
    let stack = []
    let colored = []
    let notFound = []

    stack.push(state)
    let legalColors = this[country][state].legalColors;

    while (stack.length != 0) {
        let s = stack.pop()
        let found = false

        for (let c of this[country][s].legalColors) {
            this.triedColors = this.triedColors + 1;
            this[country][s].setColor(c)

```

```

        if (this.isValid() && this.allColored())
            return true
        else if (this.isValid()) {
            while (notFound.length != 0) {
                let _s = notFound.pop()
                if (!stack.includes(_s))
                    stack.push(_s)
            }

            this[country][s].neighbours.forEach(n => {
                if (this[country][n].color == 'white' &&
!stack.includes(n))

                    stack.push(n)
            })
            found = true
            colored.push(s)
            break
        } else {
            this[country][s].legalColors =
this[country][s].legalColors.filter(_c => _c != c)
        }
    }

    if (!found) {
        this.backtracked = this.backtracked + 1;
        this.triedColors = this.triedColors + 1;
        this[country][s].setColor('white')

        this[country][s].legalColors = legalColors
    }

```

```

        stack.push(colored.pop())
        notFound.push(s)
        if (colored.length == 0) {
            return false
        }
    }
}
},
dfs: function (state) { // Forward Checking, No Heuristic
    let country = this.countryName;
    let stack = []
    let colored = []
    let notFound = []

    stack.push(state)
    let legalColors = this[country][state].legalColors;

    while (stack.length != 0) {
        let s = stack.pop()
        let found = false

        for (let c of this[country][s].legalColors) {
            let isLegal = true

            this[country][s].constraints.forEach(con => {
                if (con.color == c)
                    isLegal = false
            })
        }
    }
}

```

```

        if (!isLegal)
            continue

        this.triedColors = this.triedColors + 1;
        this[country][s].setColor(c)

        if (this.isValid() && this.allColored())
            return true
        else if (this.isValid()) {
            while (notFound.length != 0) {
                let _s = notFound.pop()
                if (!stack.includes(_s))
                    stack.push(_s)
            }

            this[country][s].neighbours.forEach(n => {
                if (this[country][n].color == 'white' &&
!stack.includes(n))

                    stack.push(n)

                if (this[country][n].color == 'white'){
                    this[country][n].constraints.push({
                        'origin': s,
                        'color': c
                    })
                }
            })
        })
        found = true
        colored.push(s)
        break

```

```

        } else {
            this[country][s].legalColors =
this[country][s].legalColors.filter(_c => _c != c)
        }
    }

    if (!found) {
        this.backtracked = this.backtracked + 1;
        this.triedColors = this.triedColors + 1;
        this[country][s].setColor('white')
        this[country][s].legalColors = legalColors
        stack.push(colored.pop())
        notFound.push(s)
        if (colored.length == 0) {
            return false
        }

        for (let key in this[country]) {
            this[country][key].constraints =
this[country][key].constraints.filter(con => con.origin != s)
        }
    }
},

dfsfp: function (state) { // Foward checking + propogation,
No heuristic

    let country = this.countryName;
    let stack = []
    let colored = []
    let notFound = []

```

```

stack.push(state)
let legalColors = this[country][state].legalColors;

while (stack.length != 0) {
    let s = stack.pop()
    let found = false

    for (let c of this[country][s].legalColors) {
        let isLegal = true

        this[country][s].constraints.forEach(con => {
            if (con.color == c)
                isLegal = false
        })

        if (!isLegal)
            continue

        this.triedColors = this.triedColors + 1;
        this[country][s].setColor(c)

        if (this.isValid() && this.allColored())
            return true
        else if (this.isValid()) {
            while (notFound.length != 0) {
                let _s = notFound.pop()
                if (!stack.includes(_s))
                    stack.push(_s)
            }
        }
    }
}

```



```

        this[country][s].neighbours.forEach(n => {
            if (this[country][n].color == 'white' &&
!stack.includes(n))

                stack.push(n)

            if (this[country][n].color == 'white'){
                this[country][n].constraints.push({
                    'origin': s,
                    'color': c
                })

                let result =
this[country][n].hasSingletonDomain()

                if (result.singleton) {

stack.push(stack.splice(stack.findIndex(v => v == n), 1)[0])

                    }

                })

                found = true
                colored.push(s)
                break
            } else {

                this[country][s].legalColors =
this[country][s].legalColors.filter(_c => _c != c)

            }

        }

        if (!found) {

            this.backtracked = this.backtracked + 1;

```

```

        this.triedColors = this.triedColors + 1;
        this[country][s].setColor('white')
        this[country][s].legalColors = legalColors
        stack.push(colored.pop())
        notFound.push(s)
        if (colored.length == 0) {
            return false
        }

        for (let key in this[country]) {
            this[country][key].constraints =
this[country][key].constraints.filter(con => con.origin != s)
        }
    }
},
dfsH: function (state) { // Only, with heuristic
    let country = this.countryName;
    let stack = []
    let colored = []
    let notFound = []

    stack.push(state)
    let legalColors = this[country][state].legalColors;

    while (stack.length != 0) {
        let s = stack.pop()
        let found = false

        for (let c of this[country][s].legalColors) {

```

```

        this.triedColors = this.triedColors + 1;
        this[country][s].setColor(c)

        if (this.isValid() && this.allColored())
            return true
        else if (this.isValid()) {
            while (notFound.length != 0) {
                let _s = notFound.pop()
                if (!stack.includes(_s))
                    stack.push(_s)
            }

            this[country][s].neighbours.forEach(n => {
                if (this[country][n].color == 'white' &&
!stack.includes(n))

                    stack.push(n)
            })
            found = true
            colored.push(s)
            break
        } else {
            this[country][s].legalColors =
this[country][s].legalColors.filter(_c => _c != c)
        }
    }

    if (!found) {
        this.backtracked = this.backtracked + 1;
        this.triedColors = this.triedColors + 1;
        this[country][s].setColor('white')
    }
}

```

```

        this[country][s].legalColors = legalColors
        stack.push(colored.pop())
        notFound.push(s)
        if (colored.length == 0) {
            return false
        }
    }
}

let min = ""
let minValues = 999999

for (let x of stack) {
    console.log(x)
    if (this[country][x].legalColors.length <
minValues) {
        min = x
        minValues =
this[country][x].legalColors.length
    }
}

stack.push(stack.splice(stack.findIndex(v => v ==
min), 1)[0])
},
dfsFH: function (state) { // Forward Checking, with Heuristic
    let country = this.countryName;
    let stack = []
    let colored = []
    let notFound = []

```

```

stack.push(state)
let legalColors = this[country][state].legalColors;

while (stack.length != 0) {
    let s = stack.pop()
    let found = false

    for (let c of this[country][s].legalColors) {
        let isLegal = true

        this[country][s].constraints.forEach(con => {
            if (con.color == c)
                isLegal = false
        })

        if (!isLegal)
            continue

        this.triedColors = this.triedColors + 1;
        this[country][s].setColor(c)

        if (this.isValid() && this.allColored())
            return true
        else if (this.isValid()) {
            while (notFound.length != 0) {
                let _s = notFound.pop()
                if (!stack.includes(_s))
                    stack.push(_s)
            }
        }
    }
}

```

```

    }

    this[country][s].neighbours.forEach(n => {
        if (this[country][n].color == 'white' &&
!stack.includes(n))

            stack.push(n)

        if (this[country][n].color == 'white'){
            this[country][n].constraints.push({
                'origin': s,
                'color': c
            })
        }
    })

    found = true
    colored.push(s)
    break
} else {
    this[country][s].legalColors =
this[country][s].legalColors.filter(_c => _c != c)
}
}

if (!found) {
    this.backtracked = this.backtracked + 1;
    this.triedColors = this.triedColors + 1;
    this[country][s].setColor('white')
    this[country][s].legalColors = legalColors
    stack.push(colored.pop())
    notFound.push(s)
}

```

```

        if (colored.length == 0) {
            return false
        }

        for (let key in this[country]) {
            this[country][key].constraints =
this[country][key].constraints.filter(con => con.origin != s)
        }
    }

    let min = ""
    let minValues = 999999

    for (let x of stack) {
        console.log(x)
        if (this[country][x].legalColors.length <
minValues) {
            min = x
            minValues =
this[country][x].legalColors.length
        }
    }

    stack.push(stack.splice(stack.findIndex(v => v ==
min), 1)[0])
    },
    dfsfpH: function (state) { // Foward checking + propogation,
with heuristic
        let country = this.countryName;
        let stack = []

```

```

let colored = []
let notFound = []

stack.push(state)
let legalColors = this[country][state].legalColors;

while (stack.length != 0) {
    let s = stack.pop()
    let found = false

    for (let c of this[country][s].legalColors) {
        let isLegal = true

        this[country][s].constraints.forEach(con => {
            if (con.color == c)
                isLegal = false
        })

        if (!isLegal)
            continue

        this.triedColors = this.triedColors + 1;
        this[country][s].setColor(c)

        if (this.isValid() && this.allColored())
            return true
        else if (this.isValid()) {
            while (notFound.length != 0) {
                let _s = notFound.pop()

```



```

        if (!stack.includes(_s))
            stack.push(_s)
    }

    this[country][s].neighbours.forEach(n => {
        if (this[country][n].color == 'white' &&
!stack.includes(n))

            stack.push(n)

        if (this[country][n].color == 'white'){
            this[country][n].constraints.push({
                'origin': s,
                'color': c
            })

            let result =
this[country][n].hasSingletonDomain()

            if (result.singleton) {

stack.push(stack.splice(stack.findIndex(v => v == n), 1)[0])

            }

        })

        found = true
        colored.push(s)
        break
    } else {

        this[country][s].legalColors =
this[country][s].legalColors.filter(_c => _c != c)

    }

}

```

```

    if (!found) {
        this.backtracked = this.backtracked + 1;
        this.triedColors = this.triedColors + 1;
        this[country][s].setColor('white')
        this[country][s].legalColors = legalColors
        stack.push(colored.pop())
        notFound.push(s)
        if (colored.length == 0) {
            return false
        }

        for (let key in this[country]) {
            this[country][key].constraints =
this[country][key].constraints.filter(con => con.origin != s)
        }
    }

    let min = ""
    let minValues = 999999

    for (let x of stack) {
        console.log(x)
        if (this[country][x].legalColors.length <
minValues) {
            min = x
            minValues =
this[country][x].legalColors.length
        }
    }
}

```

```

min), 1)[0])
        stack.push(stack.splice(stack.findIndex(v => v ==
    }
    }
    })
    })
</script>
</body>
</html>

```

References:

- 1> https://en.wikipedia.org/wiki/Graph_coloring