PROJECT 3

Map Coloring Problem

Rishi Kumar Gnanasundaram - 801101490

Deepak VeeraPandian – 801100869

Harshini Koduru - 801103195

Anusha Vudathu - 801102031

Table of contents

1. **Problem/Aim**
   Compute the chromatic number of USA and Australia map using map coloring techniques and compare the observed results.
2. **Language**

   The code is developed in JAVA and the for the visualization HTML is used.

3. **Abstract**

In this project we have used CSP concept to color the USA and Australia maps, one to each country, in such a way that no two countries sharing a border have the same color. In this project, we have experimented CSP using following methods,

- Depth first search only
- Depth first search + forward checking
- Depth first search + forward checking + propagation through singleton domains
- Depth first search + forward checking + propagation through reduced domain

4. **Introduction**

A CSP is composed of a set of variables X1,X2,…,Xn, with domains (possible values) D1,D2,…,Dn. A set of constraints C1,C2, …,Cm. Each constraint Ci limits the values that a subset of variables can take, e.g., V1 ≠ V.

In this project we are implementing the CSP with or without heuristics using any one of the following methods,

1. Depth first search only

This strategy is the most basic approach; it doesn't prune any branches at all. The only check it makes is that all the assigned values so far are consistent with each other.

To perform depth first search only:

- [DFS] After you assign a value to a variable, examine all of the variables you've assigned values so far, and make sure those values are consistent with the constraints. If they aren't, backtrack.

2. Depth first search + forward checking

This strategy eliminates impossible options from neighboring variables.

To perform dfs+forward checking:

- [DFS] After you assign a value to a variable, examine all of the variables you've assigned values so far, and make sure those values are consistent with the constraints. If they aren't, backtrack.
- [FC] After you assign a value to a variable, consider all of its neighbors. Eliminate any options from its neighbors that are incompatible with the value you just assigned.

3. Depth first search + forward checking + propagation through singleton domains

This strategy eliminates impossible options from neighboring variables. Then, if any of those neighboring variables have only one option left, it looks ahead to see what else it can eliminate.

- [DFS] After you assign a value to a variable, examine all the variables you've assigned values so far, and make sure those values are consistent with the constraints. If they aren't, backtrack.
- [FC] After you assign a value to a variable, consider all its neighbors. Eliminate any options from its neighbors that are incompatible with the value you just assigned.
- [PROP-1] If you eliminate options from a neighbor in the previous step, and that neighbor has only one option left, add that neighbor to the list of variables to propagate. Propagate all the variables in the list.

To propagate a singleton variable, take note of its one remaining option and consider each of its neighbors. You want to cross off values in the neighboring variables that are incompatible with the one remaining option. Then, if you cross off options in a neighboring variable, and that neighbor has only one option left, add that neighbor to the list of variables to propagate.

4. Depth first search + forward checking + propagation through reduced domains

This strategy eliminates impossible options from neighboring variables. If it successfully eliminates any options from those variables, it looks ahead to see what else it can eliminate.

- [DFS] After you assign a value to a variable, examine all the variables you've assigned values so far, and make sure those values are consistent with the constraints. If they aren't, backtrack.
- [FC] After you assign a value to a variable, consider all its neighbors. Eliminate any options from its neighbors that are incompatible with the value you just assigned.
- [PROP-ANY] If you eliminate any options from a neighbor in the previous step, add that neighbor to the list of variables to propagate. Propagate all the variables in the list.

To propagate a reduced variable, take note of its remaining options and consider each of its neighbors. You want to cross off values in the neighboring variables that are incompatible with every one of the remaining options. If you cross off any values in a neighboring variable, add that neighbor to the list of variables to propagate.

- **Heuristics:**

  In this project we used three heuristics,

  A. Minimum remaining values (MRV)
  B. Degree constraint, this is used when there is a tie in the minimum remaining values.
  C. Least constraining value

5. **Experimental Analysis**

- **Input**

The program accepts user input to select the map to execute i.e., Australia/USA and then the user has to select the CSP methods he needs to implement. The code takes the states and their constraints values of both maps and do the necessary map coloring operations.

- **Output**

  The application prints the output in the hash map, where value is the country name and the key is the color assigned to it after solving by using selected algorithm. That output is then converted into CSV file which is used to visualize the output in the respective maps.

6. **Classes defined in the program**

6.1 Main Variables

- **folderPath (String)** : Path to the executable html files, csv and json files.
- **Domain (Concurrenthashmap)** : Domain map which has value as string and key as arraylist datatypes
- **Constraints (Hashmap)** : It stores the constraints for each state.
- **Mapcolor (Concurrenthashmap)** : It stores the colors value.
- **Output (Concurrenthashmap)** : The output which has color assigned to the respective state is stored here.
- **failureList (Concurrenthashmap)** : The states which are wrongly assigned are stored here, so that we don't check the same state again.

6.2 Aussie class

- This class contains the main method to take the user input and all the methods and modules to assign the colors to the states, performing backtracking, forward checking if selected. In short, this class contains all the logic which is required to solve the map coloring problem using CSP.

Methods

main	It has the main display logic to solve the map coloring program. It contains the code to take the input from the user, call the respective methods and to generate and display the output.
Singleton	This method contains the singleton method logic, IT stores the output in the hashmap. This method contains if loop, to execute the code with or without using heuristics
fwdCheckWithBacktrack	This method contains the forward checking with backtracking logic. Here, code checks whether the color assigned is valid or not else it backtracks to the previous state and removes the color assigned to that state previously. For this we have used two maps to store the details. One for domain and another for failure.
ReinitialiseColor	This method is used to reinitialize the color in the backtracking process, if the assigned color to the current state is not valid. Here, we compare constraint list with each key in domain.
DFS	This method contains the logic for depth first search process. It is called when the user selects the DFS at the start.
DFSForwardCheck	This method contains logic for the DFS using forward checking process. Here, we call the method fwdCheckWithBacktrack if the current state color is same as the adjacent state. The final output is then converted to the CSV using the method.
shuffleState	This method shuffle the states, so that it generates the random states order to generate the final output.
copyArray	This method copies the colors to the new array.

Heuristics	This method takes constraints and domain map as the parameters. Here, as part of the heuristics we calculate the least legal values, it choose the domain with the least legal value.
Assigncolor	This method first finds the domain with key=minkey and then calls the remove method to remove the color from assigned state. It stores the all the constraints in the list and compares the constraint list with each key in the domain.
Singledomain	This method takes the domain map as the parameter, and then calls the method assignColor.
Backtracking	This method contains the logic for the backtracking process. It accepts minkey, constraints and domain maps as the parameters. Here, it takes a copy of the domain list in domain copy variable. It stores the new available color in the stack and calls the remove method to remove the wrongly assigned color.
heuwithBacktrack	This method checks whether key=minkey. It calls the remove and fwdCheckWithBacktrack methods.
converttoCsv	This method converts the hashmap output to the csv file.
Browseroutput	This method calls the html file and displays the output in the firefox browser.

6.3 Input class

- This class contains Australia and USA states names and their respective constraints for each state.

7. Chromatic Explanation

- For the United states map, we have to use min of 4 colors, here we have chosen red, yellow, green, blue for the map coloring.
- For the Australia map we have chosen there colors which are red, blue and green

8. Results for Australia Map

Heuristics	**Depth First Search**	**Depth First Search with forward checking**	**Depth First Search with forward checking and singleton propagation**
With heuristics	**output:** {Western Australia=blue, New South Wales=green, Tasmania=red, Victoria=blue, Northern Territory=green, South Australia=red, Queensland=blue} **backTrackCount:** 0 **Execution time in milliseconds:** 23	**FINAl OUTPUT:** {Western Australia=blue, New South Wales=green, Victoria=blue, Tasmania=red, Northern Territory=green, South Australia=red, Queensland=blue} **backTrackCount:** 0 **Execution time in milliseconds:** 26	**FINAl OUTPUT:** {Western Australia=blue, New South Wales=green, Victoria=blue, Tasmania=red, Northern Territory=green, South Australia=red, Queensland=blue} **Execution time in milliseconds:** 26

Without heuristics	**output:** {Western Australia=red, New South Wales=green, Victoria=red, Tasmania=red, Northern Territory=green, South Australia=blue, Queensland=red} **backTrackCount:** 0 **Execution time in milliseconds:** 24	**FINAl OUTPUT:** {Western Australia=red, New South Wales=green, Victoria=red, Tasmania=red, Northern Territory=green, South Australia=blue, Queensland=red} **backTrackCount:** 0 **Execution time in milliseconds:** 22	**FINAl OUTPUT:** {Western Australia=blue, New South Wales=red, Victoria=blue, Tasmania=red, Northern Territory=red, South Australia=green, Queensland=blue} **Execution time in milliseconds:** 37

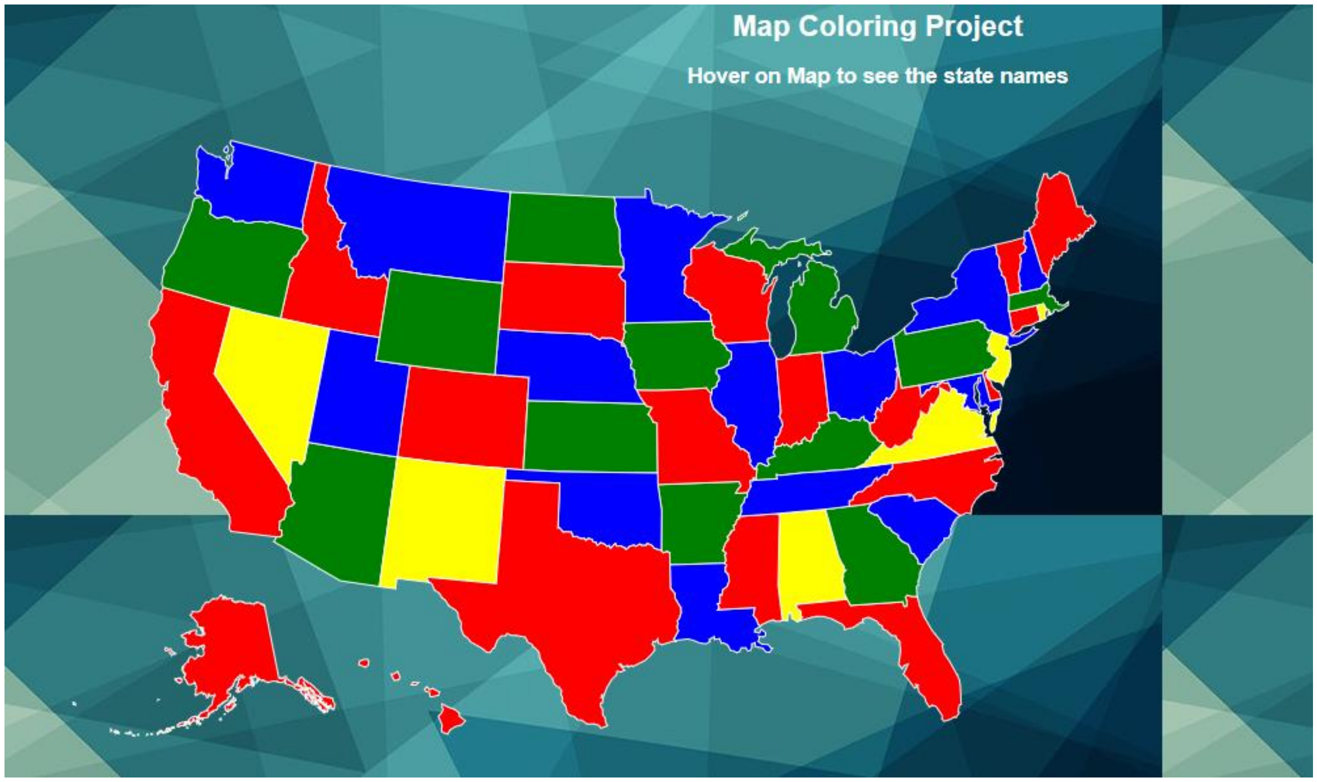
8.1 Results for United States of America Map

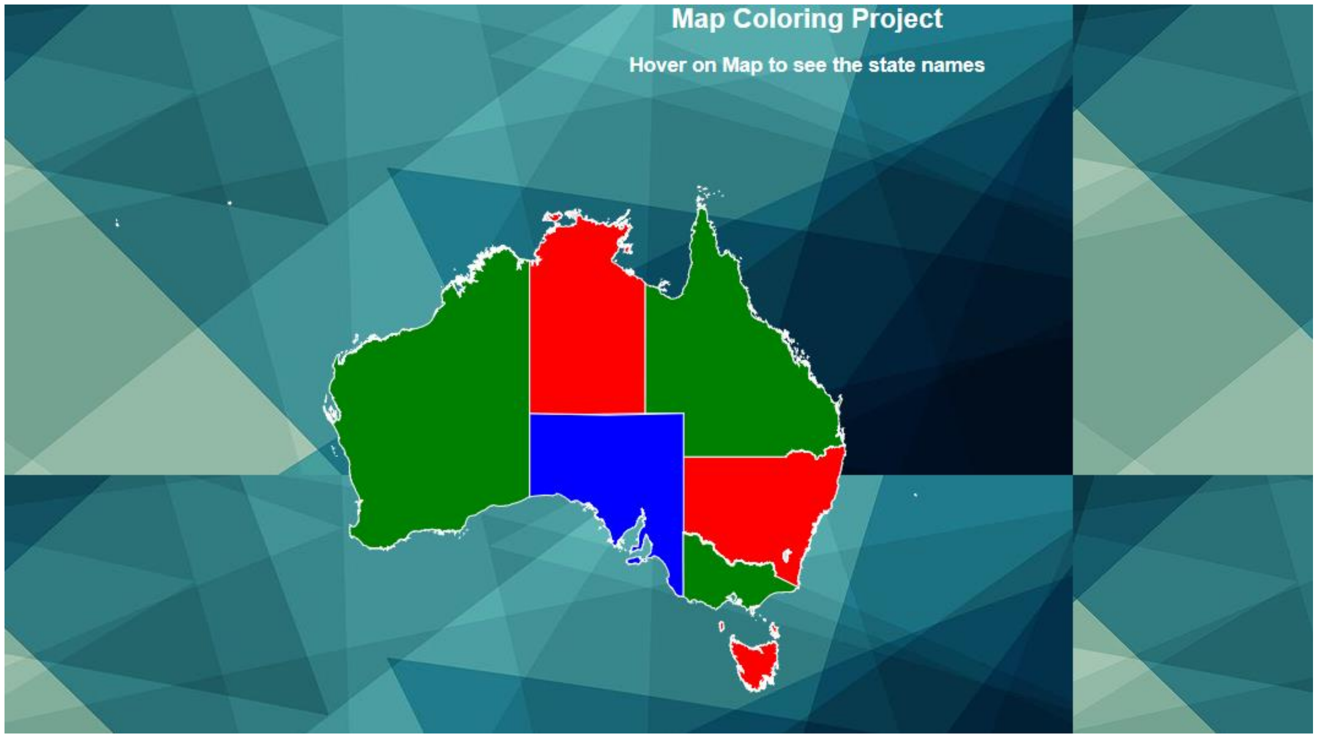
Heuristics	**Depth First Search**	**Depth First Search with forward checking**	**Depth First Search with forward checking and singleton propagation**
With heuristics	**output:** {North Carolina=red, Indiana=red, Wyoming=green, Utah=blue, Arizona=green, Montana=blue, Kentucky=green, California=red, Kansas=green, Delaware=red, Florida=red, Pennsylvania=green, Iowa=green, Mississippi=red, Illinois=blue, Texas=red, Connecticut=red, Georgia=green, Virginia=yellow, Maryland=blue, Idaho=red, Oregon=green, Vermont=red, Tennessee=blue, Oklahoma=blue, Maine=red, Alabama=yellow, Arkansas=green, South Carolina=blue, Washington=blue, Nebraska=blue, West Virginia=red, Colorado=red, Massachusetts=green,	**FINAl OUTPUT:** {North Carolina=red, Indiana=red, Wyoming=green, Utah=blue, Arizona=green, Montana=blue, Kentucky=green, California=red, Kansas=green, Delaware=red, Florida=red, Pennsylvania=green , Iowa=green, Mississippi=red, Illinois=blue, Texas=red, Connecticut=red, Georgia=green, Virginia=yellow, Maryland=blue, Idaho=red, Oregon=green, Vermont=red, Tennessee=blue, Oklahoma=blue, Maine=red, Alabama=yellow, Arkansas=green, South Carolina=blue, Washington=blue, Nebraska=blue, West Virginia=red,	**FINAl OUTPUT:** {North Carolina=green, Indiana=yellow, Wyoming=green, Utah=blue, Arizona=green, Montana=blue, Kentucky=green, Kansas=green, California=red, Delaware=red, Florida=blue, Pennsylvania=green, Iowa=green, Mississippi=red, Illinois=blue, Texas=red, Connecticut=blue, Georgia=red, Virginia=red, Maryland=yellow, Idaho=red, Oregon=green, Vermont=blue, Tennessee=blue, Oklahoma=blue, Maine=green, Alabama=green, Arkansas=green, South Carolina=blue, Washington=blue, Nebraska=blue, West Virginia=blue, Colorado=red, Massachusetts=green, Missouri=red, Alaska=red, North Dakota=green, Wisconsin=red, Nevada=yellow, New York=red, Rhode Island=yellow, South Dakota=red, Hawaii=red, Minnesota=blue, New Jersey=blue, Michigan=green, New Mexico=yellow, New Hampshire=red, Louisiana=blue, Ohio=red} **Execution time in milliseconds:** 64

	Missouri=red, Alaska=red, North Dakota=green, Wisconsin=red, Nevada=yellow, New York=blue, Rhode Island=yellow, South Dakota=red, Hawaii=red, Minnesota=blue, New Jersey=yellow, Michigan=green, New Mexico=yellow, New Hampshire=blue, Louisiana=blue, Ohio=blue} **backTrackCount**: 0 **Execution time in milliseconds:** 31	Colorado=red, Massachusetts=green, Missouri=red, Alaska=red, North Dakota=green, Wisconsin=red, Nevada=yellow, New York=blue, Rhode Island=yellow, South Dakota=red, Hawaii=red, Minnesota=blue, New Jersey=yellow, Michigan=green, New Mexico=yellow, New Hampshire=blue, Louisiana=blue, Ohio=blue} **backTrackCount:** 0 **Execution time in milliseconds:** 30	
Without heuristics	**output:** {North Carolina=blue, Indiana=red, Wyoming=green, Utah=red, Arizona=green, Montana=red, Kentucky=green, California=red, Kansas=green, Florida=red, Delaware=red, Pennsylvania=yellow, Iowa=green, Mississippi=green, Texas=green, Illinois=yellow, Connecticut=red, Georgia=green, Virginia=yellow, Maryland=green, Idaho=yellow, Oregon=green, Vermont=red, Oklahoma=red, Tennessee=red, Maine=red, Alabama=blue, Arkansas=yellow, Washington=red, South Carolina=red, Nebraska=red, West Virginia=red, Colorado=yellow,	**FINAl OUTPUT:** {North Carolina=red, Indiana=yellow, Wyoming=green, Utah=red, Arizona=green, Montana=red, Kentucky=red, California=red, Kansas=green, Florida=green, Delaware=blue, Pennsylvania=green, Iowa=green, Mississippi=green, Texas=green, Illinois=blue, Connecticut=red, Georgia=blue, Virginia=blue, Maryland=yellow, Idaho=yellow, Oregon=green, Vermont=red, Oklahoma=red, Tennessee=green, Maine=red, Alabama=red, Arkansas=yellow, Washington=red, South Carolina=yellow, Nebraska=red, West	**Final Output:**[Kansas, Georgia, Mississippi, New Jersey, Connecticut, North Carolina, Kentucky, Montana, Indiana, Massachusetts, Alabama, New York, Minnesota, New Hampshire, Hawaii, West Virginia, Wisconsin, Florida, Wyoming, Washington, Vermont, Michigan, Texas, South Carolina, Delaware, Virginia, Illinois, Ohio, Arkansas, Pennsylvania, Louisiana, Idaho, Iowa, South Dakota, Arizona, Nebraska, Rhode Island, Missouri, Alaska, Maine, Maryland, North Dakota, Oklahoma, Nevada, Oregon, Utah, Tennessee, Colorado, California, New Mexico] [Florida, Alaska, Connecticut, Texas, California, Georgia, North Dakota, North Carolina, Kansas, New Hampshire, Maine, Iowa, South Dakota, Michigan, Ohio, Montana, Oklahoma, Louisiana, Nebraska, Oregon, Washington, New Mexico, Wyoming, Delaware, Kentucky, Maryland, Wisconsin, Alabama, Nevada, Vermont, Tennessee, Pennsylvania, New York, West Virginia, Idaho, South Carolina, Hawaii, Utah,

	Massachusetts=yellow, Missouri=blue, Alaska=red, North Dakota=green, Wisconsin=blue, Nevada=blue, Rhode Island=green, New York=blue, South Dakota=blue, Hawaii=red, Minnesota=red, New Jersey=green, Michigan=green, New Mexico=blue, New Hampshire=green, Louisiana=red, Ohio=blue} **backTrackCount:** 64 **Execution time in milliseconds:** 40	Virginia=green, Colorado=yellow, Massachusetts=yellow, Missouri=blue, Alaska=red, North Dakota=green, Wisconsin=yellow, Nevada=blue, Rhode Island=green, New York=blue, South Dakota=blue, Hawaii=red, Minnesota=red, New Jersey=yellow, Michigan=green, New Mexico=blue, New Hampshire=green, Louisiana=red, Ohio=blue} **backTrackCount:** 9 **Execution time in milliseconds:** 33	Missouri, Arizona, Virginia, Mississippi, Rhode Island, Indiana, Arkansas, Minnesota, Illinois, Massachusetts, Colorado, New Jersey] FINAl OUTPUT {North Carolina=red, Indiana=yellow, Wyoming=yellow, Utah=red, Arizona=green, Montana=blue, Kentucky=red, California=red, Kansas=red, Florida=red, Delaware=red, Pennsylvania=blue, Iowa=red, Mississippi=red, Texas=red, Illinois=blue, Connecticut=red, Georgia=green, Maryland=green, Virginia=blue, Idaho=green, Oregon=blue, Vermont=green, Maine=green, Oklahoma=green, Tennessee=yellow, Alabama=blue, Arkansas=blue, Washington=red, South Carolina=blue, Nebraska=blue, West Virginia=yellow, Colorado=yellow, Massachusetts=blue, Missouri=yellow, Alaska=red, North Dakota=red, Wisconsin=green, Nevada=yellow, New York=yellow, Rhode Island=green, South Dakota=green, Hawaii=red, Minnesota=blue, New Jersey=green, Michigan=red, New Mexico=blue, New Hampshire=red, Louisiana=green, Ohio=green} **Execution time in milliseconds:** 60

- **SCREENSHOTS:**


Map Coloring Project
Hover on Map to see the state names


Map Coloring Project
Hover on Map to see the state names