Introduction to Artificial Intelligence

18 May, 2023

Proje 2 Report

In this project, I used an algorithm to color the South African countries so that they are not the same color next to each other, and I printed the map with each country in a different color from its neighboring country. In the first part of the code, the countries and the colors we have are given to us as a String type "list". We should use these countries and only the colors here.

First, we write *"pip install plotly"* in the terminal and install the plotly module.

Then we enter the neighbors of each country as an array. This is because countries introduce their neighbors to the program.

```
# Do not modify the line below.
countries = ["Argentina", "Bolivia", "Brazil", "Chile", "Colombia", "Ecuador",
             "Falkland Islands", "Guyana", "Paraguay", "Peru", "Suriname",
              "Uruguay", "Venezuela"]

# Do not modify the line below.
colors = ["blue", "green", "red", "yellow"]
```

Our first method, "**sortCountries"**, takes the graph and unsorted countries as parameters. Our first goal is to rank the countries according to the number of sides, because the number of neighboring countries is different from each other. Some countries may border only one country, while others may border up to five different countries. We should not ignore this situation and include it in our algorithm.

With the following loop, we sort from the country with the highest number of neighboring countries to the country with the least. Each country is added to the "**retCountries**" list and we return this list.

```
i = len(newCountry) - 1
while i > -1:
    retCountries.append(newCountry[i][0])
    i -= 1

return retCountries
```

The “**trueColoring**” method checks whether neighboring countries are the same color as each other. It uses graph and testMap parameters for this. If two neighboring countries have the same color, it returns "false".

The purpose of the “**colorTheCountry**” method, as the name suggests, was written to color the countries.

The variable 'color_index' follows the color index.

The variable 'country_index' follows the country index.

We sort the number of neighbors of the countries using the ‘sortCountries’ method and assign them to the 'sortedCountries' list. This list gives us the sorted state of the countries.

```
    sortedCountries = sortCountries(graph, unSortedCountries)
```

The while loop shown below allows us to iterate over the sorted countries. The solverCounter variable indicates whether the four colors continue before they are exhausted. The 'temp' variable is a temporary index. The colorMap parameter, on the other hand, is constantly updated, and then it checks if the countries sorted with the trueColoring method are correctly colored. If there is a problem with the coloring, the 'country_index' is reduced by one and the 'color_index' is checked.

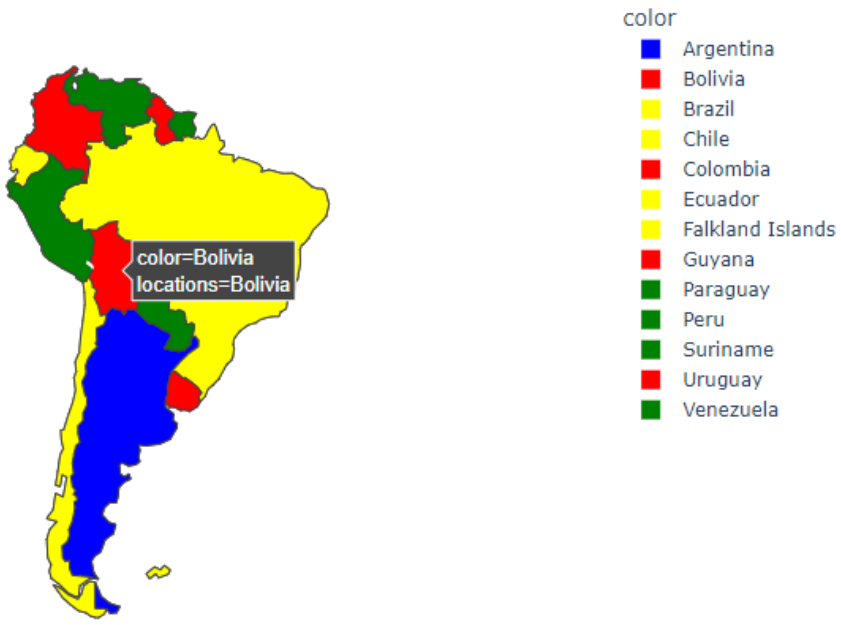
```
 while country_index < len(sortedCountries):
        if solverCounter == len(colors):
            print("Problem not solved.")
            isSolved = False
            break
        temp = country_index
        colorMap[sortedCountries[country_index]] =
colors[color_index]
        if not trueColoring(graph, colorMap):
            country_index -= 1
        color_index = (color_index + 1) % len(colors)
        country_index += 1
        if temp == country_index:
            solverCounter += 1
        else:
            solverCounter = 0
    return colorMap, isSolved
```

With 'colormap_test', it checks if the color map is colored correctly and outputs the result accordingly.

Then an empty map is created with the 'coloringMap' and each of the countries is given one of four colors with the 'colorTheCountry' method. Finally, if the coloring is correct, the 'plot_choropleth' method is called and the map output is as follows.

```
colormap_test True colored.
coloringMap True colored.
PS C:\Users\BÜŞRA>
```

The image below gives the output of the code.



Büşra ZENBİLCİ

20170808054