BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KỲ

Đề tài: LOGIC PROGRAMMING VÀ ỨNG DỤNG TRONG BÀI TOÁN PHÂN TÍCH BẢN ĐỒ THEO LOGIC

Giảng viên hướng dẫn : Trần Tiến Đức
Sinh viên thực hiện : Nguyễn Thị Hồng Thơ
MSSV : 22151305
Lớp : 22133B
Khóa : 2022
Mã lớp : ARIN330585_23_2_05

Thành phố Hồ Chí Minh, tháng 05 năm 2024

DANH SÁCH THAM GIA ĐỀ TÀI

HỌC KÌ I, NĂM HỌC: 2023 – 2024

Lớp: ARIN330585_23_2_05

Tên đề tài: Logic programming và ứng dụng trong bài toán phân tích bản đồ theo logic

HỌ VÀ TÊN SINH VIÊN	MÃ SỐ SINH VIÊN
Nguyễn Thị Hồng Thơ	22151305

Nhận xét của giảng viên

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Ngày 19 tháng 05 năm 2024

Giảng viên chấm điểm

DANH MỤC HÌNH ẢNH

MỤC LỤC

PHẦN MỞ ĐẦU

1. Lý do chọn đề tài

Một yêu cầu quan trọng đối với hệ thống thông minh hay trong trí tuệ nhân tạo là phải có khả năng sử dụng tri thức về thế giới xung quanh và lập luận (reasoning) với tri thức. Rất khó để đạt được những hành vi thông minh mà không có tri thức về thế giới xung quanh và khả năng suy diễn với tri thức đó. Sử dụng tri thức và lập luận sẽ giúp hệ thống dựa trên tri thức có tính mềm dẻo cao. Việc kết hợp tri thức và lập luận (bao gồm suy diễn và suy luận) cho phép tạo ra tri thức khác, giúp hệ thống đạt được những mục tiêu khác nhau, đồng thời có khả năng lập luận về bản thân mục tiêu. Và lập trình logic là một mô hình lập trình dựa trên các quan hệ và suy luận này.

Từ đó, nhận thấy tầm quan trọng của logic trong việc giải quyết bài toán, ứng dụng vào trí tuệ nhân tạo, em đã chọn đề tài: **"Logic programming và ứng dụng trong bài toán phân tích bản đồ theo logic"** để nghiên cứu, tìm hiểu rõ hơn về logic và ứng dụng của nó.

2. Kết cấu đề tài

Ngoài phần Mở đầu, Kết luận, Danh mục tài liệu tham khảo, nội dung của đề tài được kết cấu thành 2 chương:

Chương 1: Tổng quan về logic.

Chương 2: Ứng dụng của logic trong bài toán phân tích bản đồ.

CHƯƠNG 1: TỔNG QUAN VỀ LOGIC

1. Biểu diễn tri thức và lập luận logic

Các hệ thống có sử dụng tri thức được gọi là hệ dựa trên tri thức. Hệ thống loại này gồm thành phần cơ bản là cơ sở tri thức (tiếng Anh là Knowledge Base, viết tắt là KB). Cơ sở tri thức gồm các câu hay các công thức trên một ngôn ngữ nào đó và chứa các tri thức về thế giới của bài toán.

Cùng với cơ sở tri thức, hệ thống còn có khả năng lập luận, gồm cả suy diễn (inference) và suy luận (deduction), cho phép đưa ra các hành động hoặc câu trả lời hợp lý dựa trên tri thức và thông tin quan sát được. Lập luận là hành động sinh ra một phát biểu đúng mới từ các phát biểu đúng có trước. Thực chất, suy diễn hay lập luận là cách tạo ra các câu mới từ những câu đã có. Như vậy, một hệ dựa trên tri thức bao gồm cơ sở tri thức và thủ tục suy diễn.

2. Tổng quan về logic

Lập trình logic là một mô hình lập trình dựa trên các quan hệ và suy luận logic. Logic với vai trò là phương tiện để biểu diễn tri thức và suy diễn. Dạng biểu diễn tri thức cổ điển nhất trong máy tính là logic, với hai dạng phổ biến là logic mệnh đề và logic vị từ. Logic là một ngôn ngữ biểu diễn tri thức trong đó các câu nhận hai giá trị đúng (True) hoặc sai (False).

Như vậy, logic là một ngôn ngữ mà mỗi câu trong ngôn ngữ đó có ngữ nghĩa (giá trị) là đúng hoặc sai, và vì vậy ta có thể lập luận, tức là một câu mới có giá trị đúng không khi các câu trước đó là đúng hay không.

Các câu trước đó được gọi là cơ sở tri thức (Knowledge base – KB), câu cần chứng minh đúng gọi là câu truy vấn (query – q).

Cú pháp của biểu thức logic: bao gồm các ký hiệu và các quy tắc liên kết các ký hiệu để tạo thành câu hay biểu thức logic. Một ví dụ cú pháp là các ký hiệu và quy tắc xây dựng biểu thức toán học trong số học và đại số:

− Ngữ nghĩa của ngôn ngữ cho phép ta xác định ý nghĩa của các câu trong một miền nào đó của thế giới hiện thực, xác định các sự kiện hoặc sự vật phản ánh thế giới thực của câu mệnh đề. Đối với logic, ngữ nghĩa cho phép xác định câu là đúng hay sai trong thế giới của bài toán đang xét. Ví dụ, trong ngôn ngữ toán học, câu a + 1 = 3 là

câu đúng cú pháp. Theo ngữ nghĩa của ngôn ngữ toán học, câu này là đúng trong miền bài toán có a = 2 và sai trong những miền bài toán có a ≠ 2.

− Cơ chế suy diễn là phương pháp cho phép sinh ra các câu mới từ các câu đã có hoặc kiểm tra liệu các câu có phải là hệ quả logic của nhau. Ta có thể sử dụng suy diễn để sinh ra các tri thức mới từ tri thức đã có trong cơ sở tri thức.

2.1 Khái quát về logic mệnh đề

Logic mệnh đề là logic đơn giản nhất. Các phát biểu (câu) trong logic mệnh đề được hình thành từ các ký hiệu mệnh đề và các ký hiệu liên kết ¬ (với ngữ nghĩa là phủ định), ∧ (và), ∨ (hoặc), ⇒ (kéo theo), ⇔ (tương đương).

Ví dụ:

Gọi A là mệnh đề "tôi chăm học", B là mệnh đề "tôi thông minh", C là mệnh đề "tôi thi đạt điểm cao môn Trí tuệ nhân tạo"; Ta có thể biểu diễn các câu sau trong logic mệnh đề:

- "Nếu tôi chăm học thì tôi thi đạt điểm cao môn Trí tuệ nhân tạo": A ⇒ C
- "Tôi vừa chăm học lại vừa thông minh": A ∧ B
- "Nếu tôi chăm học hoặc tôi thông minh thì tôi thi đạt điểm cao môn Trí tuệ nhân tạo": A ∨ B ⇒ C

2.2 Khái quát về logic vị từ cấp một

Nếu logic mệnh đề chỉ có thể biểu diễn được các MỆNH ĐỀ và các liên kết hoặc quan hệ giữa các MỆNH ĐỀ, hạn chế trong việc biểu diễn và suy diễn thì logic vị từ (một mở rộng của logic mệnh đề sẽ cho phép biểu diễn những mệnh đề mang tính phổ quát và những mệnh đề mang tính đặc thù một cách dễ dàng.

Câu đơn của logic vị từ có dạng Vị_từ(chủ_ngữ) hoặc Vị_từ(chủ_ngữ, tân ngữ); chẳng hạn "An là sinh viên" biểu diễn là Sinhvien(An); "An yêu Bình" biểu diễn là Yeu(An,Binh).

Ví dụ:

"An là sinh viên"	Sinhvien(An)
"Nam là cha của Hoàn"	Cha(Nam,Hoàn)
"Mọi sinh viên đu học giỏi"	∀x Sinhvien(x) ⇒ Hocgioi(x)

CHƯƠNG 2: ỨNG DỤNG CỦA LOGIC TRONG BÀI TOÁN PHÂN TÍCH BẢN ĐỒ

1. Áp dụng logic vào bài toán phân tích bản đồ

Hình 1. Bản đồ các bang của nước Mỹ.

Chỉ định thông tin có sẵn về vị trí của các bang của Hoa Kỳ, và sau đó xây dựng chương trình trả lời những câu hỏi.

Cung cấp file: *adjacent_states.txt* và *coastal_states.txt*

Chứa các bang kề nhau, hay các bang kề biển

Ví dụ:

Alabama,Mississippi,Tennessee,Georgia,Florida

Arkansas,Missouri,Tennessee,Mississippi,Louisiana,Texas,Oklahoma

Arizona,California,Nevada,Utah,Colorado,New Mexico

- ⇨ Bang Alabama kề bang Mississippi,Tennessee,Georgia,Florida

Hay những bang kề biển được liệt kê trong file *coastal_states.txt:*

Washington,Oregon,California,Texas,Louisiana,Michigan,Alabama,Georgia,Florida, South Carolina,North Carolina,Virgin Islands,Maryland,Delaware,New Jersey,New York,Connecticut,Rhode Island,Massachusetts,Minnesota,New Hampshire

Bài toán sẽ dựa vào logic, suy luận để tìm ra **những bang nào kề nhau**, **các bang kề một bang nào đó có kề biển hay không, hay một bang kề cả hai bang được chỉ định**.

Cài đặt thư viện logpy dành cho các chương trình logic. Bên cạnh đó, bài toán sử dụng những thư viện:

import tkinter as tk

import numpy as np

import matplotlib.pyplot as plt

from tkinter import ttk

from mpl_toolkits.basemap import Basemap

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

from matplotlib.patches import Polygon

from tkinter import messagebox

from logpy import run, fact, eq, Relation, var

Sử dụng 3 file: st99_d00.dbf, st99_d00.shp, st99_d00.shx để vẽ nên hình dáng cho các bang dựa vào Basemap:

```
m = Basemap(llcrnrlon=-119,llcrnrlat=22,urcrnrlon=-64,urcrnrlat=49,
    projection='lcc',lat_1=33,lat_2=45,lon_0=-95, ax=ax)  # Tạo một bản đồ
m.readshapefile('st99_d00', 'states')  #Đọc file shapefile và vẽ các state lên bản đồ
```

Ví dụ: Chương trình sẽ thực hiện như sau để tìm bang kề với bang Alabama:

Tạo biến logic x: x = var()

>>> fact(adjacent, (“Alabama,Mississippi,Tennessee,Georgia,Florida”)

>>>run(0, x, adjacent('Alabama', x))

(‘Mississippi,Tennessee,Georgia,Florida’)

1.1 Code gốc:

```
from logpy import run, fact, eq, Relation, var

adjacent = Relation()
coastal = Relation()

file_coastal = 'coastal_states.txt'
file_adjacent = 'adjacent_states.txt'
```

```
# Read the file containing the coastal states
with open(file_coastal, 'r') as f:
    line = f.read()
    coastal_states = line.split(',')

# Add the info to the fact base
for state in coastal_states:
    fact(coastal, state)

# Read the file containing the coastal states
with open(file_adjacent, 'r') as f:
    adjlist = [line.strip().split(',') for line in f if line and line[0].isalpha()]

# Add the info to the fact base
for L in adjlist:
    head, tail = L[0], L[1:]
    for state in tail:
        fact(adjacent, head, state)

# Initialize the variables
x = var()
y = var()

# Is Nevada adjacent to Louisiana?
output = run(0, x, adjacent('Nevada', 'Louisiana'))
print('\nIs Nevada adjacent to Louisiana?:')
print('Yes' if len(output) else 'No')

# States adjacent to Oregon
output = run(0, x, adjacent('Oregon', x))
print('\nList of states adjacent to Oregon:')
for item in output:
    print(item)

# States adjacent to Mississippi that are coastal
output = run(0, x, adjacent('Mississippi', x), coastal(x))
print('\nList of coastal states adjacent to Mississippi:')
for item in output:
    print(item)

# List of 'n' states that border a coastal state
n = 7
output = run(n, x, coastal(y), adjacent(x, y))
print('\nList of ' + str(n) + ' states that border a coastal state:')
for item in output:
    print(item)
```

```
# List of states that adjacent to the two given states
output = run(0, x, adjacent('Arkansas', x), adjacent('Kentucky', x))
print('\nList of states that are adjacent to Arkansas and Kentucky:')
for item in output:
    print(item)
```[1]

Kết quả in ra màn hình của code trên:

```
Is Nevada adjacent to Louisiana?:
No

List of states adjacent to Oregon:
California
Nevada
Idaho
Washington

List of coastal states adjacent to Mississippi:
Louisiana
Alabama

List of 7 states that border a coastal state:
Connecticut
Georgia
New Jersey
Florida
Massachusetts
Tennessee
Oklahoma

List of states that are adjacent to Arkansas and Kentucky:
Missouri
Tennessee
```

1.2. Code giao diện GUI:

Ý tưởng: tạo một danh sách từ những *output* như*: output = run(0, x, adjacent('Nevada', 'Louisiana')).* Sao đó thêm các bang tìm được vào mảng *states_to_fill = []* để tô màu.

Code:

[1] Alberto Artasanchez and Prateek Joshi, Artificial Intelligence with Python, https://github.com/PacktPublishing/Artificial-Intelligence-with-Python-Second-Edition/blob/master/Chapter09/states.py

```
import tkinter as tk
import numpy as np
import matplotlib.pyplot as plt
from tkinter import ttk
from mpl_toolkits.basemap import Basemap
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.patches import Polygon
from tkinter import messagebox
from logpy import run, fact, eq, Relation, var
class App(tk.Tk):
    def __init__(self):
        super().__init__()

        self.adjacent = Relation()
        self.coastal = Relation()

        self.file_coastal = 'coastal_states.txt'
        self.file_adjacent = 'adjacent_states.txt'

        # Read the file containing the coastal states
        with open(self.file_coastal, 'r') as f:
            line = f.read()
            coastal_states = line.split(',')

        # Add the info to the fact base
        for state in coastal_states:
            fact(self.coastal, state)

        # Read the file containing the coastal states
        with open(self.file_adjacent, 'r') as f:
            adjlist = [line.strip().split(',') for line in f if line and line[0].isalpha()]
```

```
        # Add the info to the fact base
        for L in adjlist:
            head, tail = L[0], L[1:]
            for state in tail:
                fact(self.adjacent, head, state)

        # Initialize the variables
        self.x = var()

        self.title('Analyzing Geography')
        self.cvs_map = tk.Canvas(self, width=620, height=480, relief = tk.SUNKEN,
border = 2)
        self.ve_ban_do([])
        lbl_frm_menu = tk.LabelFrame(self, text = 'Menu', width = 300, height = 100)
        self.lbl_frm_menu_2 = tk.LabelFrame(self, text = 'The result board', width =
300, height = 100)
        self.lst_city = ["Alabama", "Arizona", "Arkansas",
            "California", "Colorado", "Connecticut", "Delaware",
            "Florida", "Georgia", "Idaho", "Illinois", "Indiana",
            "Iowa", "Kansas", "Kentucky", "Louisiana", "Maine", "Maryland",
"Massachusetts",
            "Michigan", "Minnesota", "Mississippi", "Missouri", "Montana", "Nebraska",
            "Nevada", "New Hampshire", "New Jersey", "New Mexico", "New York",
"North Carolina",
            "North Dakota", "Ohio", "Oklahoma", "Oregon",
            "Pennsylvania", "Rhode Island", "South Carolina", "South Dakota",
            "Tennessee", "Texas", "Utah", "Vermont", "Virginia",
            "Washington", "West Virginia", "Wisconsin", "Wyoming"]

        self.notebook = ttk.Notebook(self.lbl_frm_menu_2)
```

```
self.tab_bang_ke = ttk.Frame(self.notebook)
self.notebook.add(self.tab_bang_ke, text = 'Adjacent States')
self.tab_ke_ven_bien = ttk.Frame(self.notebook)
self.notebook.add(self.tab_ke_ven_bien, text = 'Coastal States \nAdjacent')
self.tab_ke_2_bang = ttk.Frame(self.notebook)
self.notebook.add(self.tab_ke_2_bang, text = 'States that border \nthe two chosen states')
self.notebook.grid(row = 0, column = 0, padx=5, pady=7, sticky = tk.EW)

self.text_widget_1 = tk.Text(self.tab_bang_ke , width = 35, height = 15)
self.text_widget_1.tag_configure('center', justify='center', font=('Georgia', 12))
self.text_widget_1.grid(row=0, column=0, padx=5, pady=5, sticky = tk.EW)
self.text_widget_2 = tk.Text(self.tab_ke_ven_bien , width = 35, height = 15)
self.text_widget_2.tag_configure('center', justify='center', font=('Georgia', 12))
self.text_widget_2.grid(row=0, column=0, padx=5, pady=5, sticky = tk.EW)
self.text_widget_3 = tk.Text(self.tab_ke_2_bang , width = 35, height = 15)
self.text_widget_3.tag_configure('center', justify='center', font=('Georgia', 12))
self.text_widget_3.grid(row=0, column=0, padx=5, pady=5, sticky = tk.EW)

btn_click_bang_ke = ttk.Button(lbl_frm_menu, text = 'Adjacent States', command= self.btn_bang_ke_click)
btn_click_ke_ven_bien = ttk.Button(lbl_frm_menu, text = 'Coastal States Adjacent', command= self.btn_ke_ven_bien_click)
btn_click_ke_2_bang = ttk.Button(lbl_frm_menu, text = 'States that border the two chosen states',command= self.btn_ke_2_bang_click)

lbl_chon_bang = ttk.Label(lbl_frm_menu, text = 'Select State')
self.cbo_chon_bang = ttk.Combobox(lbl_frm_menu, value = self.lst_city)
self.cbo_chon_bang.set('')
self.cbo_chon_bang.bind("<<ComboboxSelected>>", self.cbo_chon_bang_click)
```

```
        lbl_chon_bang.grid(row=0, column=0, padx = 5, pady = 0, sticky = tk.W)
        self.cbo_chon_bang.grid(row=1, column=0, padx = 5, pady = 5, sticky = tk.EW)

        lbl_ke_2_bang = ttk.Label(lbl_frm_menu, text = 'Select The Second State')
        self.cbo_ke_2_bang = ttk.Combobox(lbl_frm_menu, value = self.lst_city)
        self.cbo_ke_2_bang.set('')
        self.cbo_ke_2_bang.bind("<<ComboboxSelected>>",
self.cbo_ke_2_bang_click)
        lbl_ke_2_bang.grid(row=6, column=0, padx = 5, pady = 0, sticky = tk.W)
        self.cbo_ke_2_bang.grid(row=7, column=0, padx = 5, pady = 5, sticky = tk.EW)

        btn_click_bang_ke.grid(row=4, column=0, padx=5, pady=5, sticky = tk.EW)
        btn_click_ke_ven_bien.grid(row=5, column=0, padx=5, pady=5, sticky = tk.EW)
        btn_click_ke_2_bang.grid(row=8, column=0, padx=5, pady=5, sticky = tk.EW)

        self.cvs_map.grid(row=0, column=0, padx =5, pady = 5)
        lbl_frm_menu.grid(row = 0, column=1, padx=5, pady=7, sticky = tk.N)
        self.lbl_frm_menu_2.grid(row = 0, column = 1 , padx=5, pady=7, sticky =
tk.EW)

    def ve_ban_do(self, states_to_fill):
        state_coords = {
            'California': (-119.4179, 36.7783),
            'Texas': (-99.9018, 31.9686),
            'Florida': (-81.5158, 27.6648),
            'New York': (-75.3060, 42.7128),
            'Illinois': (-89.3985, 40.6331),
            'Pennsylvania': (-77.4945, 40.9033),
            'Ohio': (-82.9071, 40.4173),
            'Georgia': (-83.5389, 32.1656),
            'North Carolina': (-79.0193, 35.4596),
```

'Michigan': (-84.5068, 43.1148),

'6': (-74.4057, 40.0583),

'Virginia': (-78.2494, 37.5407),

'Washington': (-120.3321, 46.8062),

'Arizona': (-111.2937, 34.4484),

'3': (-71.4589, 42.3601),

'Tennessee': (-86.7816, 35.8627),

'Indiana': (-86.1349, 39.7684),

'Missouri': (-92.3295, 38.5767),

'8': (-76.7413, 39.0458),

'Wisconsin': (-89.4012, 44.2731),

'Colorado': (-105.9821, 39.2501),

'Minnesota': (-94.4650, 46.0778),

'South Carolina': (-81.1637, 33.8361),

'Alabama': (-86.6023, 33.3182),

'Louisiana': (-91.8749, 30.0843),

'Kentucky': (-84.7700, 37.4393),

'Oregon': (-120.6765, 43.8393),

'Oklahoma': (-97.0929, 35.0078),

'5': (-73.0877, 41.6032),

'Iowa': (-93.6977, 41.8780),

'Mississippi': (-89.4985, 32.3547),

'Arkansas': (-92.3311, 34.7465),

'Utah': (-111.8535, 39.5608),

'Nevada': (-116.4398, 39.1699),

'Kansas': (-98.4842, 38.4119),

'New Mexico': (-105.6056, 34.5199),

'Nebraska': (-99.9018, 41.4925),

'9': (-80.7549, 38.3498),

'Idaho': (-114.7420, 44.0682),

'Maine': (-69.0455, 45.2538),

```
        '1': (-71.5724, 43.1939),
        '4': (-71.4828, 41.5801),
        'Montana': (-109.3626, 46.8797),
        '7': (-75.5277, 38.9108),
        'South Dakota': (-99.9018, 44.3683),
        'North Dakota': (-100.7837, 47.5515),
        '2': (-72.6778, 44.1588),
        'Wyoming': (-107.2903, 43.0760)
    }
    states_digit = {'1': 'New Hamsphire', '2': 'Vermont', '3': 'Massachusetts', '4':
'Rhode Island',
                    '5': 'Connecticut', '6': 'New Jersey', '7': 'Delaware', '8': 'Maryland',
                    '9': 'West Virginia'}

    fig = plt.figure(figsize=(10,8))  # Tạo một figure
    ax = fig.add_subplot(1,1,1)  # Tạo một subplot

    m = Basemap(llcrnrlon=-119,llcrnrlat=22,urcrnrlon=-64,urcrnrlat=49,
        projection='lcc',lat_1=33,lat_2=45,lon_0=-95, ax=ax)  # Tạo một bản đồ

    m.readshapefile('st99_d00', 'states')  #Đọc file shapefile và vẽ các state lên bản
đồ
    for state, coords in state_coords.items():
        lon, lat = coords
        x, y = m(lon, lat)
        plt.text(x, y, state, fontsize=7, fontweight = 'bold', ha='center', va='center',
color='red', fontname = 'Georgia')
    for info, shape in zip(m.states_info, m.states):
        if info['NAME'] in states_to_fill:
            poly = Polygon(shape, facecolor='yellow')
            plt.gca().add_patch(poly)
```

```
        # Tạo chú thích và scatter cho các bang có tên là số
        for i, name in states_digit.items():
            lon, lat = state_coords[i]  # Lấy tọa độ tương ứng
            x, y = m(lon, lat)  # Chuyển đổi tọa độ
            plt.scatter(x, y, label=f'{i}: {name}')

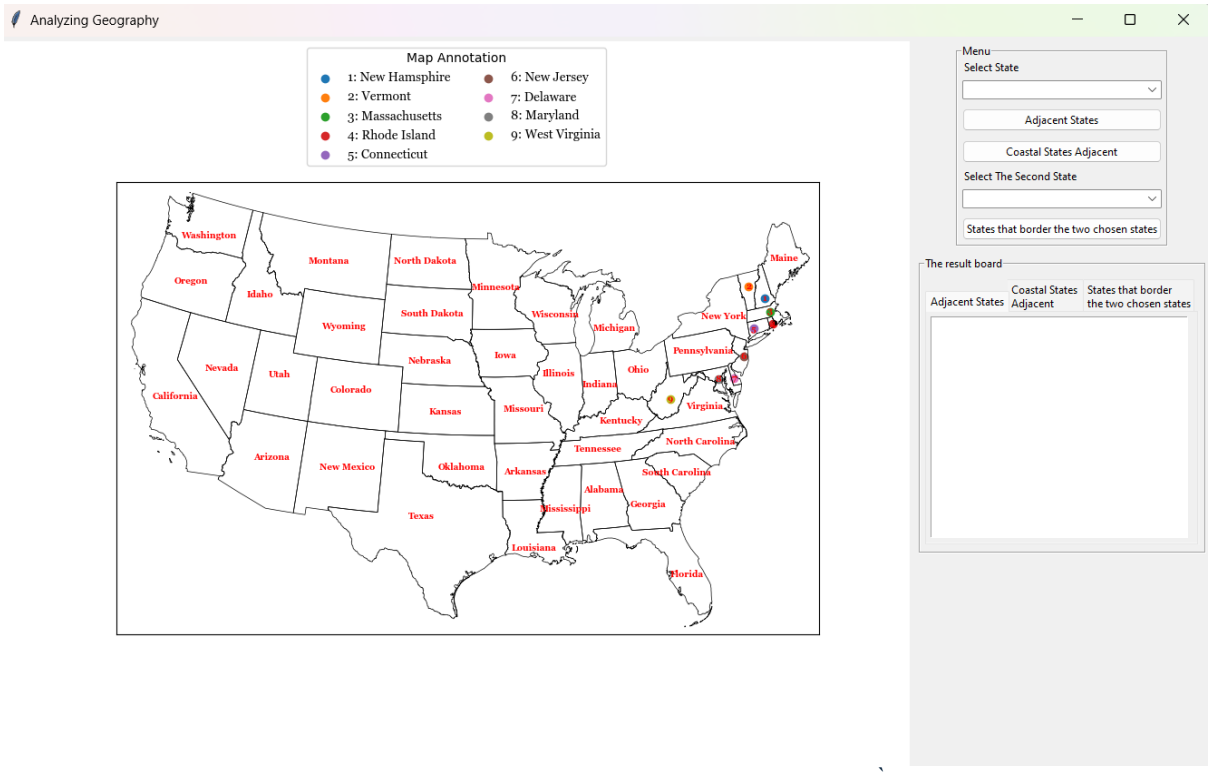
        fig.legend(title = 'Map Annotation' , loc='upper center', ncol=2, prop={'family': 'Georgia'})
        canvas = FigureCanvasTkAgg(fig, master=self)  # Nhúng figure vào cửa sổ tkinter
        canvas.draw()
        canvas.get_tk_widget().grid(row=0, column=0, padx=5, pady=5)  # Đặt canvas vào cửa sổ tkinter
    def cbo_chon_bang_click(self, *args):
        self.chon_bang = self.cbo_chon_bang.get()
    def cbo_ke_2_bang_click(self, *args):
        self.ke_2_bang = self.cbo_ke_2_bang.get()
    def btn_bang_ke_click(self):
        states_to_fill = []
        self.text_widget_1.delete('1.0', tk.END)
        output = run(0, self.x, self.adjacent(self.chon_bang, self.x))
        for item in output:
            self.text_widget_1.insert(tk.END, item + '\n', 'center')
            states_to_fill.append(item)
        self.text_widget_1.grid(row=0, column=0, padx=5, pady=5, sticky = tk.EW)
        self.ve_ban_do(states_to_fill)

    def btn_ke_ven_bien_click(self):
        states_to_fill = []
        self.text_widget_2.delete('1.0', tk.END)
        output = run(0, self.x, self.adjacent(self.chon_bang, self.x), self.coastal(self.x))
```

```
        for item in output:
            self.text_widget_2.insert(tk.END, item + '\n', 'center')
            states_to_fill.append(item)
        self.ve_ban_do(states_to_fill)
        if len(states_to_fill) == 0:
            messagebox.showinfo("Notification", "There are no states adjacent to " +
self.chon_bang + " that are coastal")
            self.text_widget_2.delete('1.0', tk.END)
        self.text_widget_2.grid(row=0, column=0, padx=5, pady=5, sticky = tk.EW)
    def btn_ke_2_bang_click(self):
        states_to_fill = []
        self.text_widget_3.delete('1.0', tk.END)
        if self.cbo_ke_2_bang.get() == '':
            messagebox.showinfo("Notification", "Please select the second state")
            return 0
        output = run(0, self.x, self.adjacent(self.chon_bang, self.x),
self.adjacent(self.ke_2_bang, self.x))
        for item in output:
            states_to_fill.append(item)
            self.text_widget_3.insert(tk.END, item + '\n', 'center')
        self.ve_ban_do(states_to_fill)
        if len(states_to_fill) == 0:
            messagebox.showinfo("Notification", "There are no states that are adjacent to
" + self.chon_bang + " and " + self.ke_2_bang)
            self.text_widget_3.delete('1.0', tk.END)
        self.text_widget_3.grid(row=0, column=0, padx=5, pady=5, sticky = tk.EW)

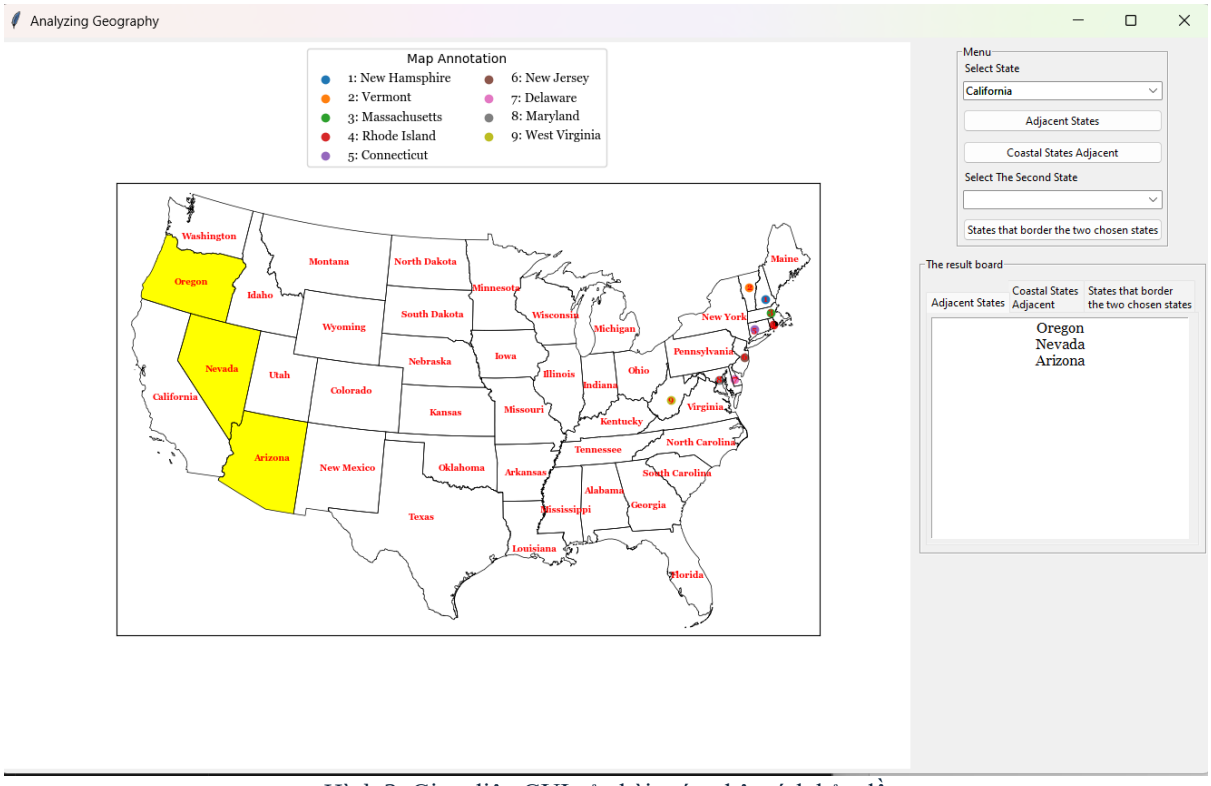
if __name__ == '__main__':
    app = App()
    app.mainloop()
```

Kết quả giao diện:



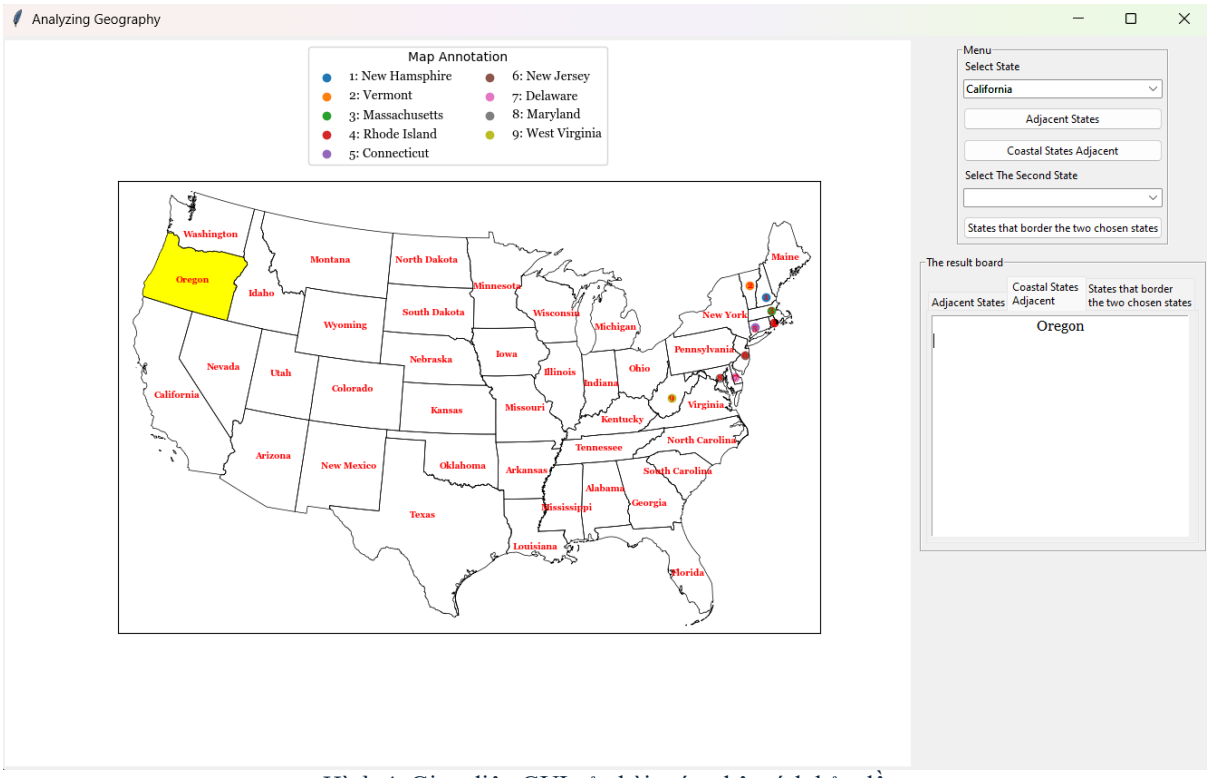
Hình 2. Giao diện GUI của bài toán phân tích bản đồ

Sau khi chọn 1 bang bất kỳ, ví dụ bang California, các bang kề sẽ là những bang được tô màu vàng và liệt kê ở bảng Result:



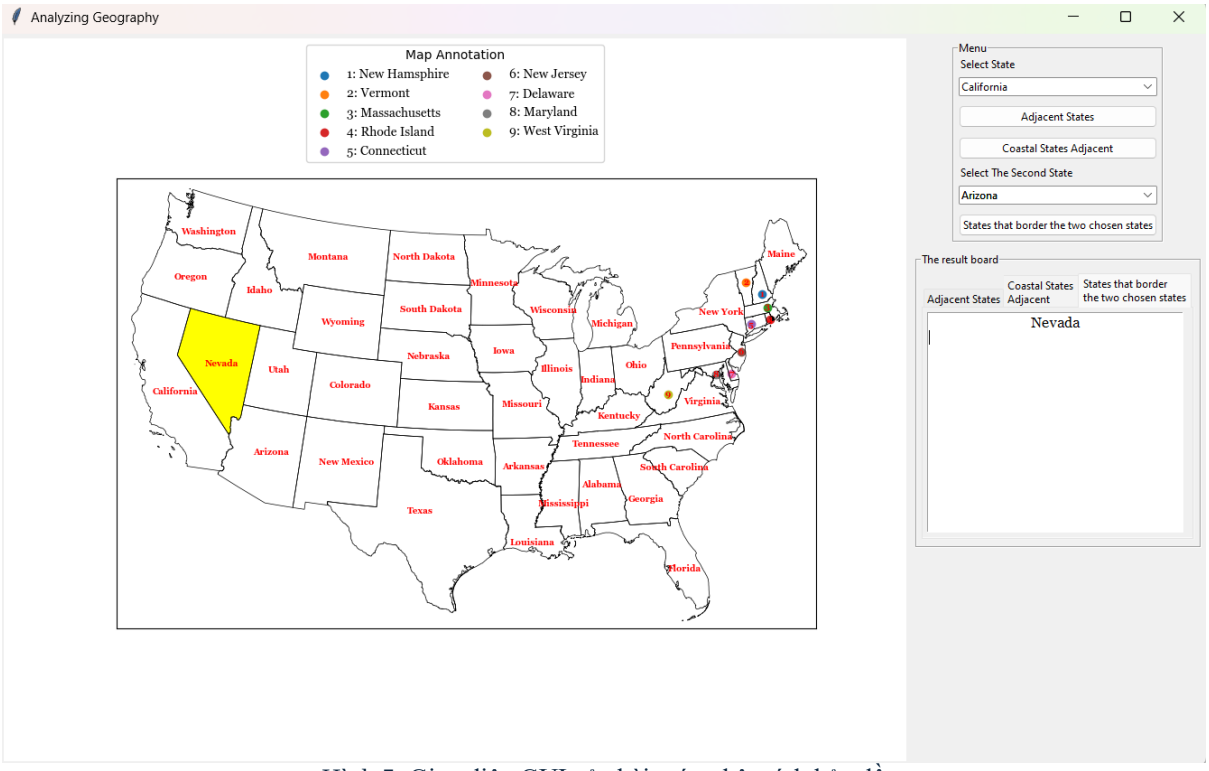
Hình 3. Giao diện GUI của bài toán phân tích bản đồ

Sau khi chọn Coastal States Adjacent, các bang kề bang California và kề biển sẽ được tô màu vàng, và liệt kê ở bảng Result:



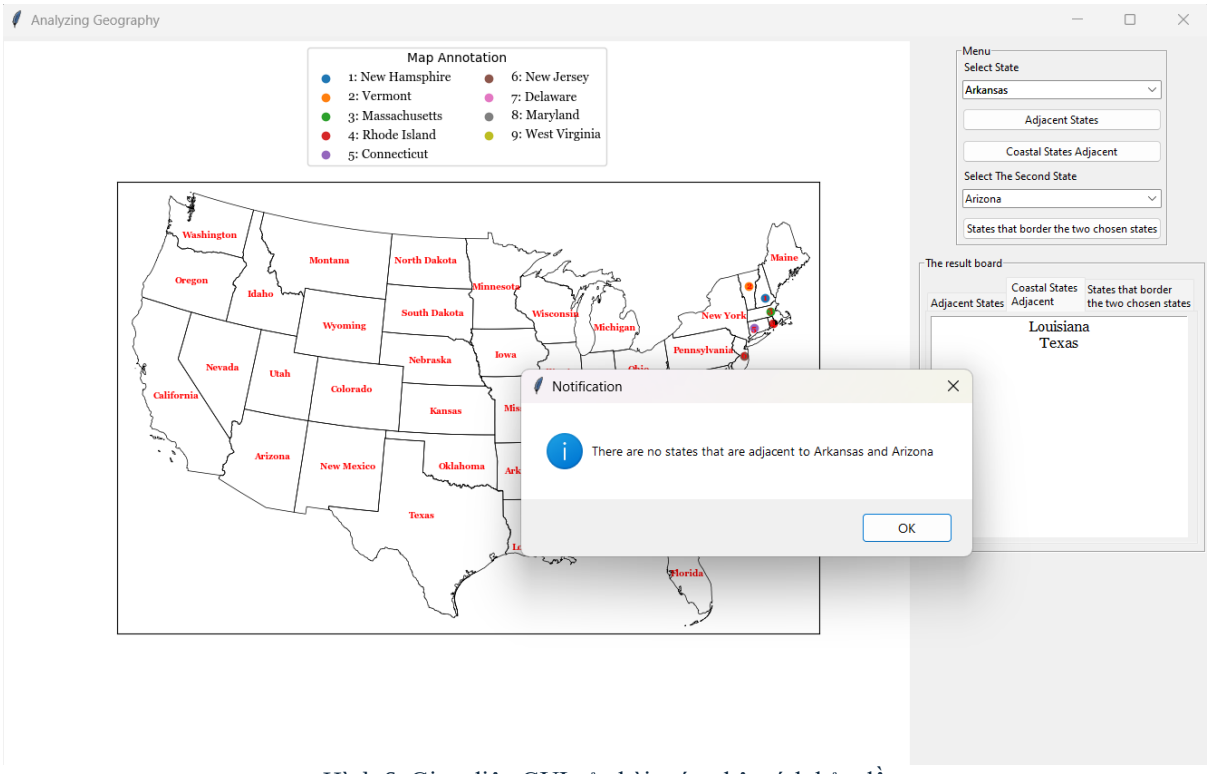
Hình 4. Giao diện GUI của bài toán phân tích bản đồ

Sau khi chọn bang thứ 2, bang nào kề với hai bang đó sẽ được tô màu, và liệt kê ở bảng Result:



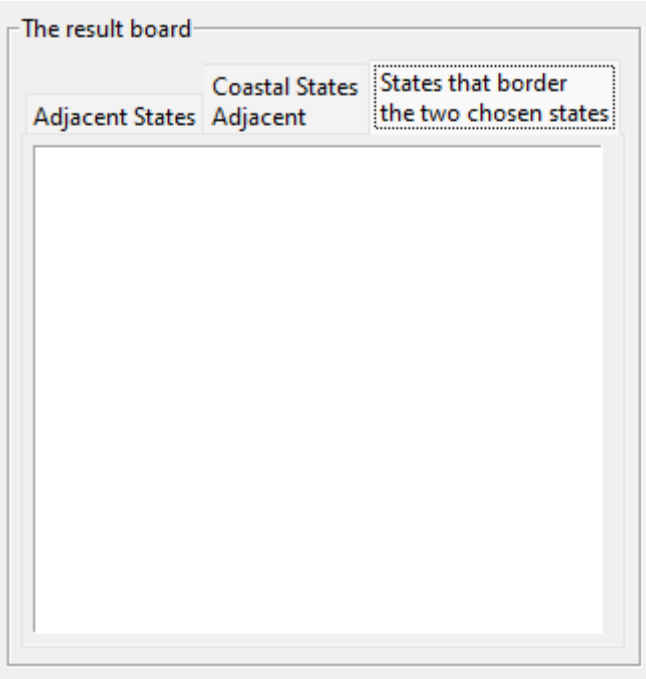
Hình 5. Giao diện GUI của bài toán phân tích bản đồ

Nếu không có bang nào kề với 2 bang được chọn, kết quả sẽ hiện ra một thông báo: "*There are no states...*"



Hình 6. Giao diện GUI của bài toán phân tích bản đồ

Bảng kết quả cũng sẽ được cập nhật: Tag “States that border the twon chosen states” rỗng:



Hình 7. Bảng kết quả

1.2. Code giao diện web

```
                                                    #---------
import tkinter as tk
import numpy as np
import matplotlib.pyplot as plt
import streamlit as st
from tkinter import ttk
from mpl_toolkits.basemap import Basemap
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.patches import Polygon
from tkinter import messagebox
from logpy import run, fact, eq, Relation, var
from streamlit import components
from matplotlib.animation import FuncAnimation

THEMES = [
    "light",
    "dark",
    "green",
    "blue",
]
GITHUB_OWNER = "streamlit"
GITHUB_REPO = "theming-showcase"
```

```
adjacent = Relation()
coastal = Relation()

file_coastal = 'coastal_states.txt'
file_adjacent = 'adjacent_states.txt'

with open(file_coastal, 'r') as f:
    line = f.read()
    coastal_states = line.split(',')

for state in coastal_states:
    fact(coastal, state)

with open(file_adjacent, 'r') as f:
    adjlist = [line.strip().split(',') for line in f if line and line[0].isalpha()]

for L in adjlist:
    head, tail = L[0], L[1:]
    for state in tail:
        fact(adjacent, head, state)

x = var()

st.markdown('<h1 style="text-align:center; color:Violet;">Analyzing
Geography</h1>', unsafe_allow_html=True)

st.lst_city = ["Alabama", "Arizona", "Arkansas",
    "California", "Colorado", "Connecticut", "Delaware",
    "Florida", "Georgia", "Idaho", "Illinois", "Indiana",
    "Iowa", "Kansas", "Kentucky", "Louisiana", "Maine", "Maryland",
"Massachusetts",
    "Michigan", "Minnesota", "Mississippi", "Missouri", "Montana", "Nebraska",
    "Nevada", "New Hampshire", "New Jersey", "New Mexico", "New York", "North
Carolina",
    "North Dakota", "Ohio", "Oklahoma", "Oregon",
    "Pennsylvania", "Rhode Island", "South Carolina", "South Dakota",
    "Tennessee", "Texas", "Utah", "Vermont", "Virginia",
    "Washington", "West Virginia", "Wisconsin", "Wyoming"]

def ve_ban_do(states_to_fill):
    state_coords = {
        'California': (-119.4179, 36.7783),
        'Texas': (-99.9018, 31.9686),
        'Florida': (-81.5158, 27.6648),
        'New York': (-75.3060, 42.7128),
        'Illinois': (-89.3985, 40.6331),
```

```
    'Pennsylvania': (-77.4945, 40.9033),
    'Ohio': (-82.9071, 40.4173),
    'Georgia': (-83.5389, 32.1656),
    'North Carolina': (-79.0193, 35.4596),
    'Michigan': (-84.5068, 43.1148),
    '6': (-74.4057, 40.0583),
    'Virginia': (-78.2494, 37.5407),
    'Washington': (-120.3321, 46.8062),
    'Arizona': (-111.2937, 34.4484),
    '3': (-71.4589, 42.3601),
    'Tennessee': (-86.7816, 35.8627),
    'Indiana': (-86.1349, 39.7684),
    'Missouri': (-92.3295, 38.5767),
    '8': (-76.7413, 39.0458),
    'Wisconsin': (-89.4012, 44.2731),
    'Colorado': (-105.9821, 39.2501),
    'Minnesota': (-94.4650, 46.0778),
    'South Carolina': (-81.1637, 33.8361),
    'Alabama': (-86.6023, 33.3182),
    'Louisiana': (-91.8749, 30.0843),
    'Kentucky': (-84.7700, 37.4393),
    'Oregon': (-120.6765, 43.8393),
    'Oklahoma': (-97.0929, 35.0078),
    '5': (-73.0877, 41.6032),
    'Iowa': (-93.6977, 41.8780),
    'Mississippi': (-89.4985, 32.3547),
    'Arkansas': (-92.3311, 34.7465),
    'Utah': (-111.8535, 39.5608),
    'Nevada': (-116.4398, 39.1699),
    'Kansas': (-98.4842, 38.4119),
    'New Mexico': (-105.6056, 34.5199),
    'Nebraska': (-99.9018, 41.4925),
    '9': (-80.7549, 38.3498),
    'Idaho': (-114.7420, 44.0682),
    'Maine': (-69.0455, 45.2538),
    '1': (-71.5724, 43.1939),
    '4': (-71.4828, 41.5801),
    'Montana': (-109.3626, 46.8797),
    '7': (-75.5277, 38.9108),
    'South Dakota': (-99.9018, 44.3683),
    'North Dakota': (-100.7837, 47.5515),
    '2': (-72.6778, 44.1588),
    'Wyoming': (-107.2903, 43.0760)
}
states_digit = {'1': 'New Hamsphire', '2': 'Vermont', '3': 'Massachusetts', '4': 'Rhode
Island',
```

```
            '5': 'Connecticut', '6': 'New Jersey', '7': 'Delaware', '8': 'Maryland',
               '9': 'West Virginia'}

   fig = plt.figure(figsize=(10,8))
   ax = fig.add_subplot(1,1,1)

   m = Basemap(llcrnrlon=-119,llcrnrlat=22,urcrnrlon=-64,urcrnrlat=49,
      projection='lcc',lat_1=33,lat_2=45,lon_0=-95, ax=ax)  # Tạo một bản đồ
   m.readshapefile('st99_d00', 'states')  #Đọc file shapefile và vẽ các state lên bản đồ
   for state, coords in state_coords.items():
      lon, lat = coords
      x, y = m(lon, lat)
      plt.text(x, y, state, fontsize=7, fontweight = 'bold', ha='center', va='center',
color='red', fontname = 'Georgia')
   for info, shape in zip(m.states_info, m.states):
      if info['NAME'] in states_to_fill:
         poly = Polygon(shape, facecolor='yellow')
         plt.gca().add_patch(poly)
   for i, name in states_digit.items():
      lon, lat = state_coords[i]
      x, y = m(lon, lat)
      plt.scatter(x, y, label=f'{i}: {name}')
   fig.legend(title = 'Map Annotation' , loc='upper center', ncol=2, prop={'family':
'Georgia'})
   return fig

st.sidebar.markdown('Menu')

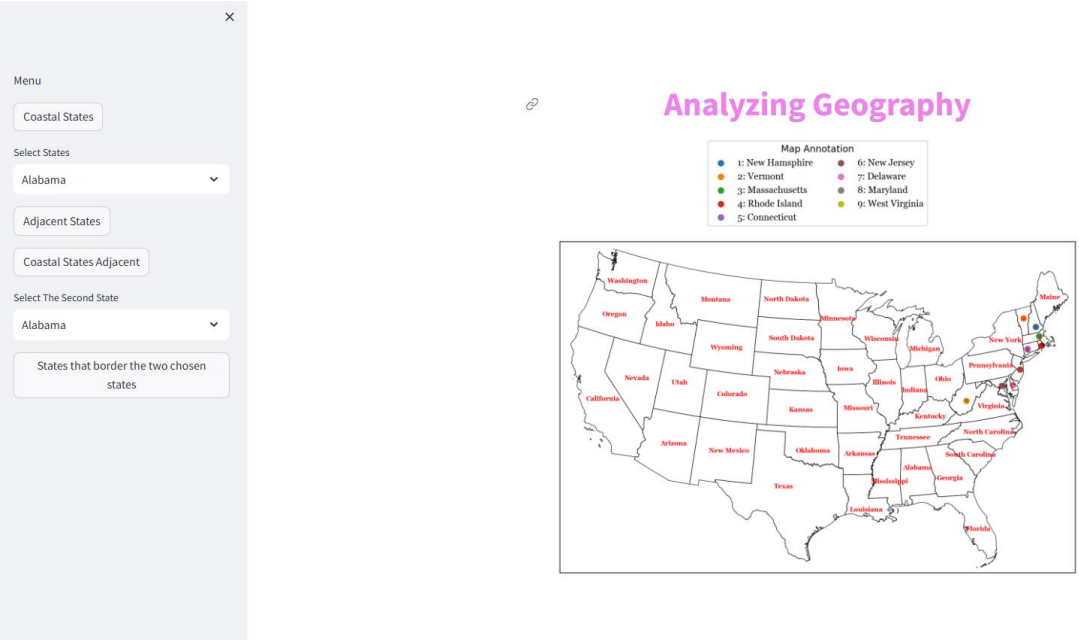
if "label" not in st.session_state:
   st.session_state["label"] = ''
if "flag_anima" not in st.session_state:
   st.session_state["flag_anima"] = False
if st.session_state["flag_anima"] == False:
   if "flag_ve_ban_do" not in st.session_state:
      st.session_state["flag_ve_ban_do"] = True
      fig = ve_ban_do([])
      st.session_state['fig'] = fig
      st.pyplot(fig)
      print (st.session_state["flag_ve_ban_do"])
      print ("Vẽ bản đồ lần đầu")
   else:
      if st.session_state["flag_ve_ban_do"] == False:
         st.session_state["flag_ve_ban_do"] = True
         fig = ve_ban_do([])
         st.session_state['fig'] = fig
         st.pyplot(fig)
```

```
        else:
            print("Đã ve bản đồ")
            st.pyplot(st.session_state['fig'])
    if st.sidebar.button('Coastal States'):
        state_to_fill = []
        output = run(0, x, coastal(x))
        for item in output:
            state_to_fill.append(item)
            states = ', '.join(state_to_fill)
        st.session_state['label'] = 'Coastal States: '
        st.session_state['label'] += states
        fig = ve_ban_do(state_to_fill)
        st.session_state['fig'] = fig
        st.rerun()
    #select_states = st.selectbox('Select States', st.lst_city)
    select_states = st.sidebar.selectbox('Select States', st.lst_city)
    if st.sidebar.button('Adjacent States'):
        state_to_fill = []
        output = run(0, x, adjacent(select_states, x))
        for item in output:
            state_to_fill.append(item)
            states = ', '.join(state_to_fill)
        st.session_state['label'] = 'Adjacent States: '
        st.session_state['label'] += states
        fig = ve_ban_do(state_to_fill)
        st.session_state['fig'] = fig
        st.rerun()
    if st.sidebar.button('Coastal States Adjacent'):
        state_to_fill = []
        states = ''
        output = run(0, x, adjacent(select_states, x), coastal(x))
        for item in output:
            state_to_fill.append(item)
            states = ', '.join(state_to_fill)
        st.session_state['label'] = 'Coastal States Adjacent: '
        st.session_state['label'] += states
        if len(state_to_fill) == 0:
            st.session_state['label'] = 'There are no states adjacent to ' + select_states + '
that are also coastal'
        fig = ve_ban_do(state_to_fill)
        st.session_state['fig'] = fig
        st.rerun()
    select_2_states = st.sidebar.selectbox('Select The Second State', st.lst_city)
    if st.sidebar.button('States that border the two chosen states'):
        state_to_fill = []
        states = ''
```

```
        output = run(0,x,adjacent(select_states, x), adjacent(select_2_states, x))
        for item in output:
            state_to_fill.append(item)
            states = ', '.join(state_to_fill)
        st.session_state['label'] = 'States that border the two chosen states: '
        st.session_state['label'] += states
        if len(state_to_fill) == 0:
            st.session_state['label'] = 'There are no states that are adjacent to ' +
select_states + ' and ' + select_2_states
        fig = ve_ban_do(state_to_fill)
        st.session_state['fig'] = fig
        st.rerun()
    if st.session_state["label"] != '':
        st.write(st.session_state["label"])

else:
    if st.session_state["flag_anim"] == True:
        components.html(st.session_state["anim"].to_jshtml(), height=550)
        _, _, col3, _, _ = st.columns(5)
        with col3:
            if col3.button('Reset'):
                st.session_state["flag_anim"] = False
                st.session_state["flag_ve_ban_do"] = False
                st.rerun()
```

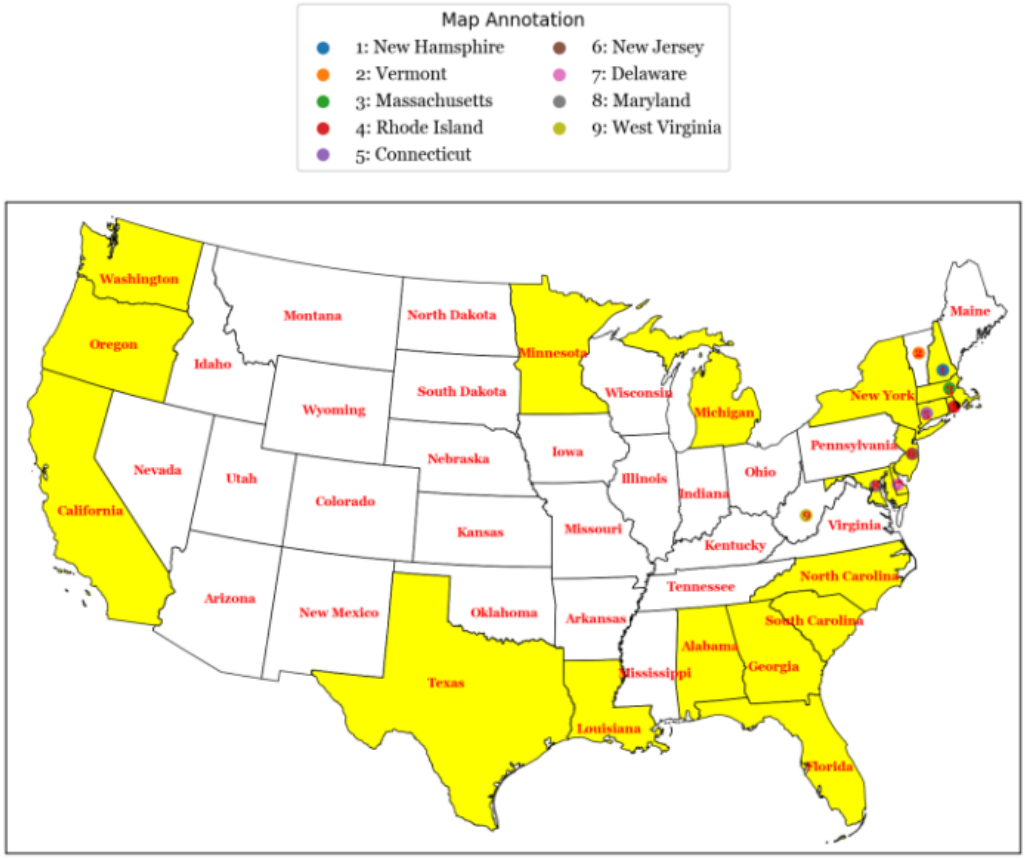
Kết quả giao diện: Giao diện bao gồm thanh Menu bên trái chứa những lựa chọn. bên phải là bản đồ Hoa Kỳ:



Hình 8. Giao diện web của bài toán phân tích bản đồ

Khi click vào nút Coastal States trên thanh Menu, những bang kề biển sẽ được tô màu và liệt kê bên dưới bản đồ:

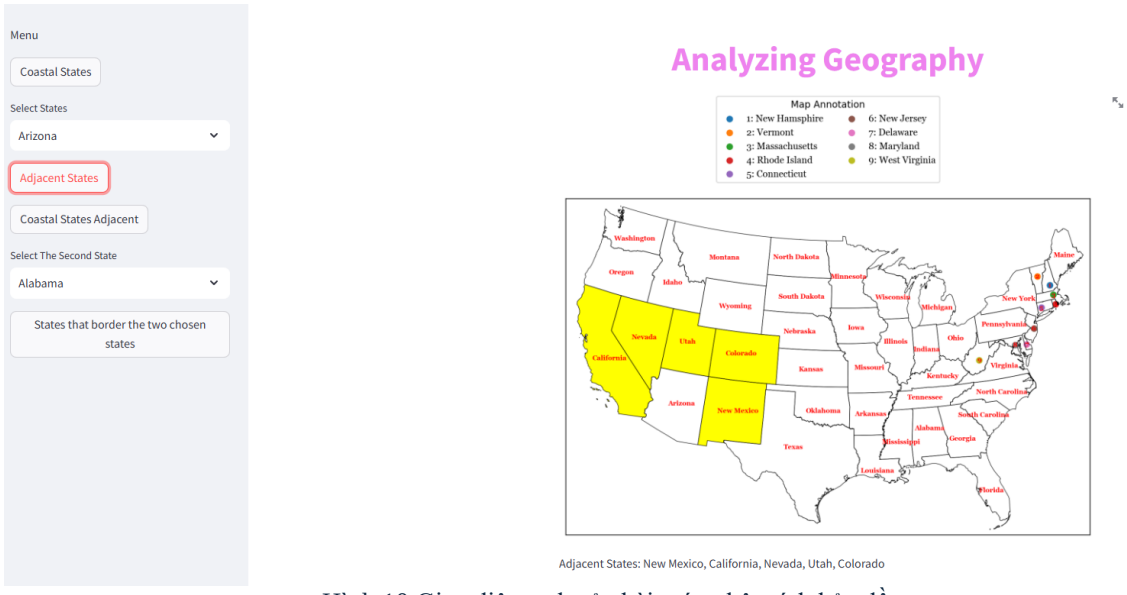
Analyzing Geography



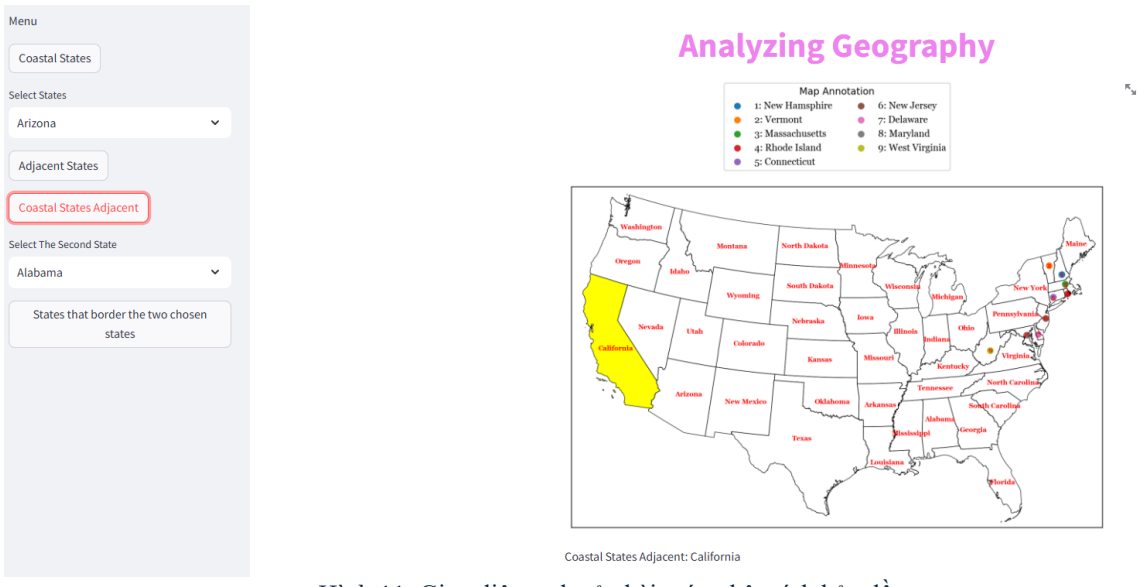
Coastal States: Michigan, Alabama, New York, New Hampshire, Delaware, Connecticut, Washington, New Jersey, Rhode Island, Texas, Maryland, Oregon, South Carolina, Massachusetts, California, Virgin Islands, Florida, Minnesota, Georgia, Louisiana, North Carolina

Hình 9. Giao diện web của bài toán phân tích bản đồ

Tương tự, khi click vào các nút Adjacent States, Coastal States Adjacent, những bang kề hay kề và kề biển sẽ được tô màu và liệt kê bên dưới bản đồ:

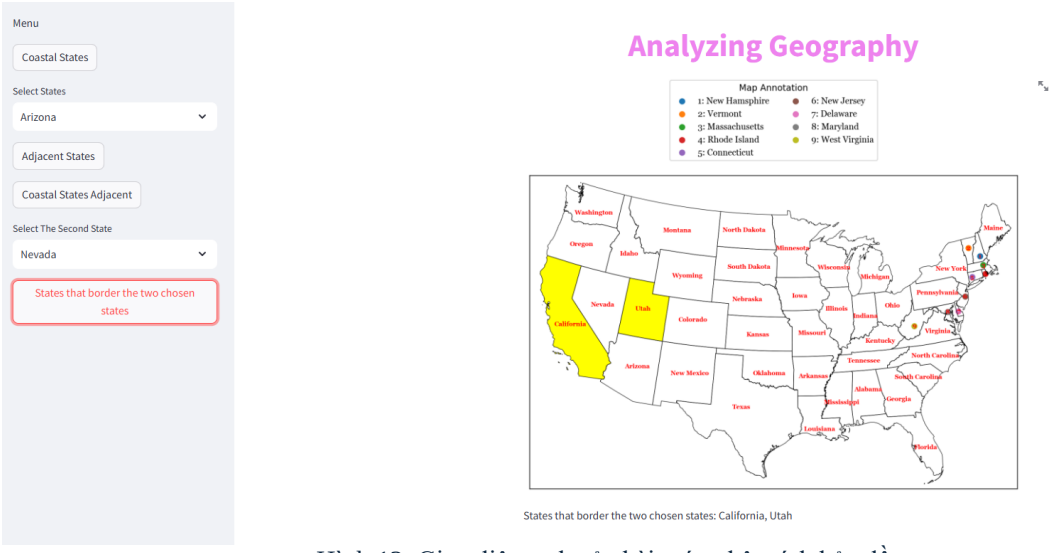


Hình 10.Giao diện web của bài toán phân tích bản đồ



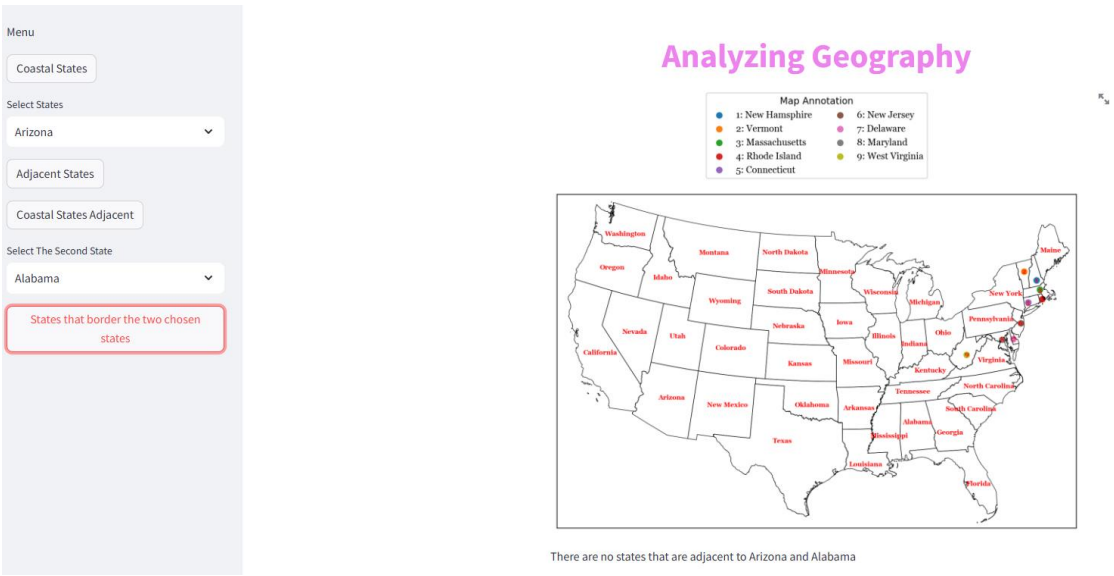
Hình 11. Giao diện web của bài toán phân tích bản đồ

Chọn bang kề cả hai bang được chọn:



Hình 12. Giao diện web của bài toán phân tích bản đồ

Nếu không có bang nào kề hai bang được chọn, hiện lên dòng “There are no states...” bên dưới bản đồ:



Hình 13. Giao diện web của bài toán phân tích bản đồ

KẾT LUẬN

Đề tài đã trình bày cơ sở lý thuyết của logic, bao gồm hai loại logic phổ biến là logic mệnh đề và logic vị từ trong các bài toán, cũng như những ứng dụng trong giải quyết bài toán phân tích bản đồ dựa trên logic và thư viện logic có sẵn logpy trong Python.

TÀI LIỆU THAM KHẢO

1. Alberto Artasanchez Prateek Joshi, Artificial Intelligence with Python, Packt Publishing Ltd, 2nd Edition, 2020.

2. Stuart J. Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, 4th Edition, Pearson Education ©, 2021.

3. Từ Minh Phương, Giáo trình Nhập môn trí tuệ nhân tạo, Học Viện Công nghệ Bưu chính Viễn thông, 2014.

4. George F. Luger, William A. Stubblefield – Albuquerque – Artificial Intelligence – Wesley Publishing Company, Inc – 1997.

5. Phạm Thọ Hoàn, Phạm Thị Anh Lê, Giáo trình Trí tuệ nhân tạo, Khoa Công nghệ thông tin, Trường Đại học Sư phạm Hà Nội, 2011.