**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**-----------------🙛🙙-----------------**

**Báo Cáo Nhóm Môn: Thực Hành Trí Tuệ Nhân Tạo**

**Đề Tài: Viết chương trình tìm lộ trình tối ưu để xe đưa khách tham quan đến các danh thắng lịch sử ở thành phố HCM Sử Dụng Thuật Toán GTS.**

**Sinh viên thực hiện: Nhóm 01**

1. 2001207012 - Trần Quang Nhật (Nhóm Trưởng)
2. 2001207133 - Nguyễn Thành Tài

**GVHD: Ngô Dương Hà**

**Thành phố Hồ Chí Minh, tháng 12 năm 2022**

**Lịch làm việc nhóm hàng tuần**

|  |  |  |
| --- | --- | --- |
| Tuần | Nguyễn Thành Tài | Trần Quang Nhật |
| 1. Phân công | 100% | 100% |
| 2. Lên kế hoạch | 100% | 100% |
| 3. Tìm tài liệu | 100% | 100% |
| 4. Lên ý tưởng | 100% | 100% |
| 5. Thống nhất ý tưởng | 100% | 100% |
| 6. Triển khai | 100% | 100% |
| 7. Thống kê những gì đã làm | 100% | 100% |
| 8. Thực hiện giao diện | 100% | 100% |
| 9. Thực hiện chức năng | 100% | 100% |
| 10. Tổng kết đóng gói | 100% | 100% |

**Công việc mỗi thành viên trong nhóm**

|  |  |  |
| --- | --- | --- |
|  | Nguyễn Thành Tài | Trần Quang Nhật |
| Công việc | Lý thuyết + Code + word | Lý thuyết + Code + word |

**MỤC LỤC**

[**GIỚI THIỆU** 1](#_Toc123322961)

[**Phạm vi đồ án:** 1](#_Toc123322962)

[**Mục tiêu:** 1](#_Toc123322963)

[**Sự cần thiết khi chọn đề tài:** 1](#_Toc123322964)

[**I.** **Phân tích đề tài:** 2](#_Toc123322965)

[**II.** **Thiết kế** 3](#_Toc123322966)

[**III.** **Thực Hiện: cài đặt ứng dụng bài toán** 3](#_Toc123322967)

[**IV.** **Hướng dẫn sử dụng code thuật toán GTS (Tham lam)** 5](#_Toc123322968)

[**V.** **Ưu điểm và nhược điểm thuật toán GTS (Tham lam)** 5](#_Toc123322969)

[**VI.** **Ứng dụng của thuật toán tham lam** 6](#_Toc123322970)

[**VII.** **Thực hiện: cài đặt ứng dụng bài toán** 7](#_Toc123322971)

[**VIII. Kết Luận:** 16](#_Toc123322972)

[**TÀI LIỆU THAM KHẢO** 17](#_Toc123322973)

# **GIỚI THIỆU**

## **Phạm vi đồ án:**

- Tìm hiểu bài toán tìm lộ trình tối ưu để xe đưa khách tham quan đến các danh thắng lịch sử ở thành phố HCM bằng thuật toán GTS.

## **Mục tiêu:**

- Hiểu rõ thuật toán GTS (Thuật toán tham lam).

- Ứng dụng thuật toán trong bài toán tối ưu để xe đưa khách tham quan đến các danh thắng lịch sử ở thành phố Hồ Chí Minh.

## **Sự cần thiết khi chọn đề tài:**

- Trong mọi khía cạnh của cuộc sống hằng ngày, công việc nghiên cứu con người luôn phải đối mặt với vấn đề. Trong số những vấn đề đó, có những vấn đề chỉ đơn giản là cần giải pháp nhưng có những vấn đề khác đòi hỏi khắt khe hơn không chỉ là giải pháp mà phải là giải pháp tốt nhất (best solution) nhanh nhất, hiệu quả nhất nó được gọi là những vấn đề cần được tối ưu.

- Ý tưởng Greedy (tham lam) là một trong những hướng suy nghĩ được sử dụng trong tình huống này. Các thuật toán sử dụng Greedy như nền tảng logic để tiếp cận, giải quyết những vấn đề được gọi là những Greedy Algorithm. Các thuật toán ứng dụng Greedy thường diễn ra, hoạt động quá nhiều chặng về không gian hay thời gian và tại mỗi chặng chúng ta thường không biết thông tin toàn bộ dữ liệu của quá trình mà chỉ biết tình trạng hiện tại và thông tin bước đi kế tiếp. Ý tưởng chính của Greedy chính là chúng ta không cần quan tâm đến dữ liệu tổng thể mà chỉ từ những dữ liệu mà từng chặng chọn ra giải pháp tối ưu tại mỗi chặng đó sẽ mang lại giải pháp tối ưu cho tổng thể. Từ ý tưởng đó cho thấy, kết quả có được từ Greedy Algorithm chỉ mang tính tương đối, thông thường giải pháp cuối cùng không phải là tối ưu mà là gần tối ưu. Tuy nhiên, với điều kiện, thông tin thường không mấy rõ ràng của đầu vào của mình và cách tiếp cận vấn đề khá trong sáng, Greedy thường đưa ra những giải pháp tương đối tốt trong giới hạn có thể chấp nhận và đặc biệt có tốc độ nhanh.

1. **Phân tích đề tài:**
2. **Phân tích yêu cầu**

- Yêu cầu của bài toán phải tìm được đường đi tối ưu nhất để xe tham quan đến các danh thắng lịch sử ở TP.HCM.

- Đây là bài toán tìm chu trình có trọng số nhỏ nhất trong một đơn đồ thị có hướng có trọng số. Thuật toán tham lam cho bài toán là chọn địa điểm có chi phí nhỏ nhất tính địa điểm hiện thời đến các địa chưa qua.

1. **Yêu cầu chức năng**

- Input: C=C(i,j)

- Output: Tour (hành trình tối ưu)

**- Mô tả:**

- Cost: Chi phí cho chuyến đi

- Khởi tạo: Tour=0, Cost=0, v = u.

- Mọi k: 1->n (thăm tất cả thắng cảnh)

- Chọn <v,w> là đoạn nối 2 địa điểm có chi phí nhỏ nhất từ thắng cảnh v đến thắng cảnh chưa

- Tour = Tour + <v,w>

- Cost = Cost + Cvw

**- Kết quả:**

- Tour = Tour + <v,u>

- Cost = Cost + Cvw

1. **Khái niệm:**

- Là một thuật toán giải quyết một bài toán theo kiểu metaheuristic để tìm kiếm lựa chọn tối ưu địa phương ở mỗi bước đi với hy vọng tìm được tối ưu toàn cục.

1. **Thiết kế**
2. **Đề xuất sử dụng thuật toán**

- Thuật toán GTS

1. **Cách thức giải quyết bài toán**

**GTS1**

**Bước 1: Khởi tạo**

- Tour = , Cost = 0, V= U (U là điểm khởi đầu)

**Bước 2: Thăm tất cả thắng cảnh tại TP.HCM**

- For k=1 to n do

**Bước 3: Chọn cung kế tiếp**

- Đặt (V,W) là đường kế tiếp có chi phí nhỏ nhất

- Tour = Tour + {(V,W)} , Cost = Cost + Cost (V,W)

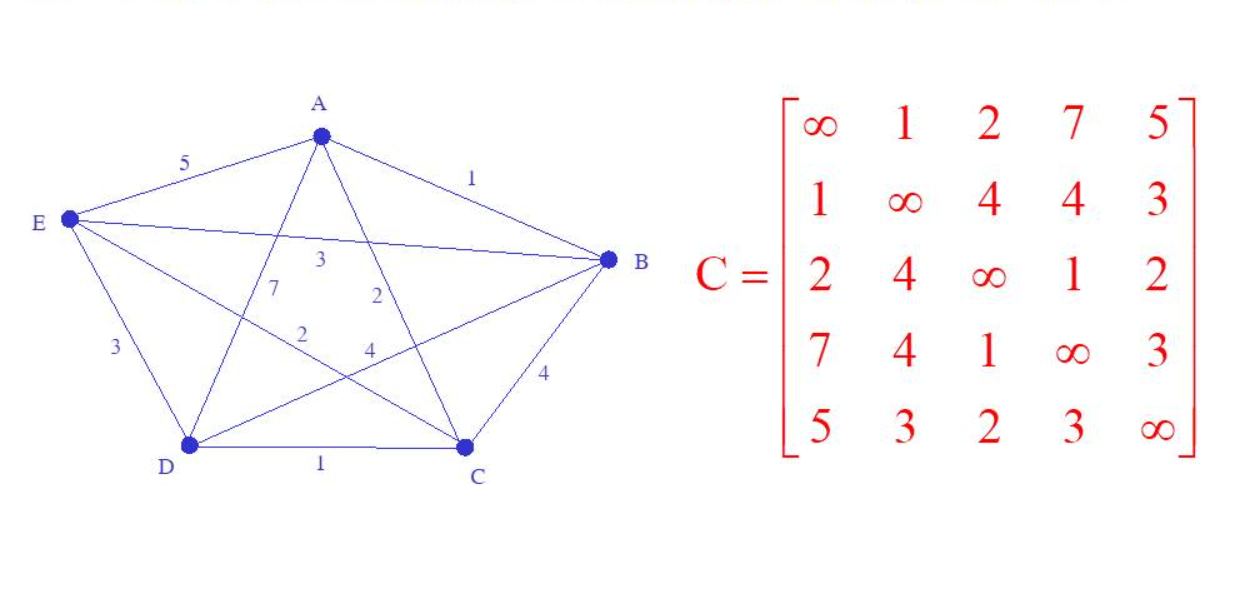
- Đặt V=W (gán để xét bước tiếp theo)

**Bước 4: Kết thúc chuyến đi**

- Đặt: Tour = Tour + {(V,W)}, Cost = Cost + Cost(V,U) -> Dừng.

1. **Thực Hiện: cài đặt ứng dụng bài toán**

Mô tả bài toán: Tìm đường đi tối ưu nhất để xe tham quan đến các danh thắng lịch sử ở TP.HCM.



A: Nhà Thờ Đức Bà

B: Bưu Điện Thành Phố

C: Dinh Độc Lập

D: Notre Dame Square

E: Nhà Thờ Tân Định

**Giải:**

Tour = , Cost = 0, V= A (A là điểm khởi đầu)

W={B,C,D,E}

Vì B có giá trị nhỏ nhất => W={B}

Tour = {(A,B)}

Cost = 0 +1=1, V = B

Xét tiếp điểm B ta có:

W={C,D,E}

ta có E là điểm có giá trị nhỏ nhất => W={B,E}

Tour = {(A,B),(B,E)}

Cost = 1+ 3 = 4, V = E

W = {C,D}

Xét C, D ta có C giá trị nhỏ nhất

Tour = {(A,B),(B,E),(E,C)}

Cost = 4 + 2 = 6, V = C

Tương tự xét điểm D ta có

Tour = {(A,B), (B,E),(E,C),(C,D)}

Cost = 6 + 1 =7, V = D

Sau khi đi hết các điểm ta trở về điểm A xuất phát

Tour = {(A,B),(B,E),(E,C),(C,D),(D,A)}

Cost = 7 + 7 = 14

=> Chuyến xe bắt đầu đi từ Nhà Thờ Đức Bà -> Bưu Điện Thành Phố -> Nhà Thờ Tân Định -> Dinh Độc Lập -> Notre Dame Square và trở về điểm bắt đầu là Nhà Thờ Đức Bà.

1. **Hướng dẫn sử dụng code thuật toán GTS (Tham lam)**

- Bước 1 : có dữ liệu ( ma trận 10x10 ) của các thành phố thể hiện chi phí từ 1 địa điểm đến những địa điểm khác khác

- Bước 2: lấy tọa độ của tất cả địa điểm , mỗi id sẽ có 2 giá trị là key và value

- Lat: vĩ độ , long: kinh độ được phân biệt từ 0-9

- Bước 3 chạy code trên visual studio Code với 2 file vừa nêu trên

1. **Ưu điểm và nhược điểm thuật toán GTS (Tham lam)**

* **Ưu điểm:**

- Dễ dàng đưa ra phương án tốt nhất. Thông thường yêu cầu các phương án sắp xếp.

- Thuật toán này có thể hoạt động tốt hơn các thuật toán khác (nhưng không phải trong mọi trường hợp).

* **Nhược điểm:**

- Khó thiết kế: Khi chúng ta đã tìm ra cách tiếp cận giải thuật tham lam phù hợp, việc thiết kế các thuật toán tham lam có thể dễ dàng. Tuy nhiên, việc tìm ra cách tiếp cận phù hợp có thể sẽ khó.

- Khó xác định: Việc biểu diễn một thuật toán tham lam phù hợp thường đòi hỏi một lập luận sắc bén.

1. **Ứng dụng của thuật toán tham lam**

- Sắp xếp lựa chọn.

- Bài toán về Knapsack.

- Cây khung nhỏ nhất.

- Single-Source Shortest Path.

- Sắp xếp lịch cho công việc.

- Thuật toán cây khung nhỏ nhất của Prim.

- Thuật toán cây khung nhỏ nhất của Kruskal.

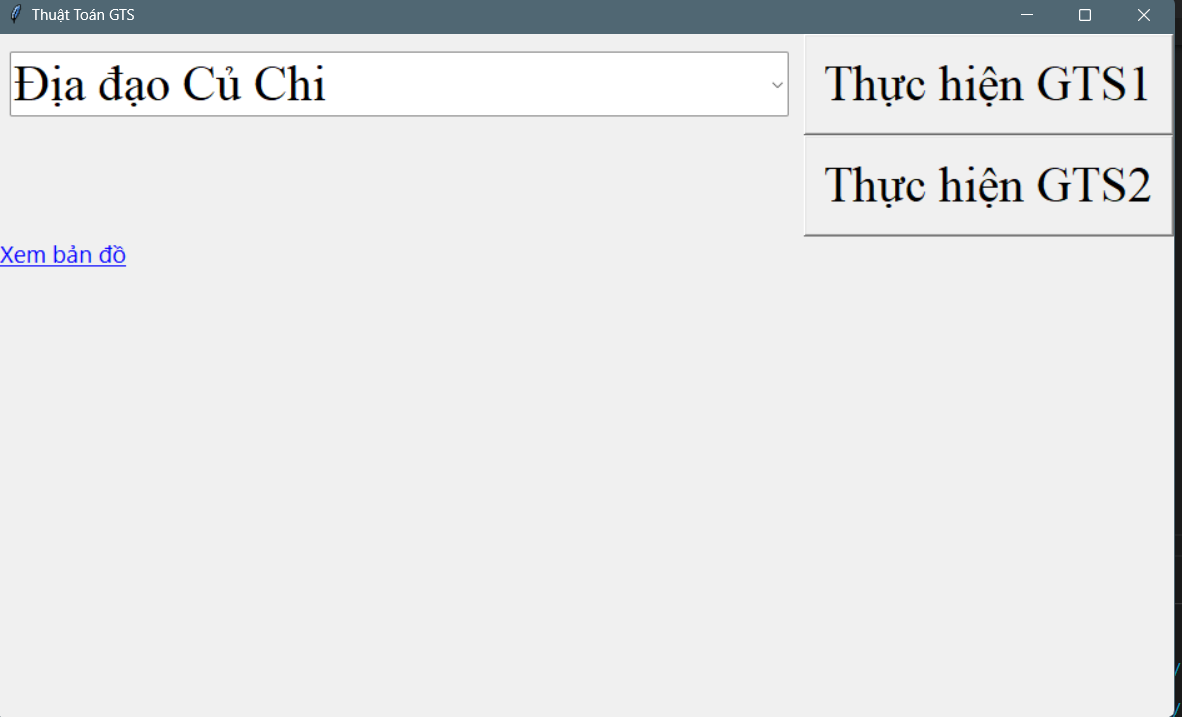
- Thuật toán cây khung nhỏ nhất của Dijkstra.

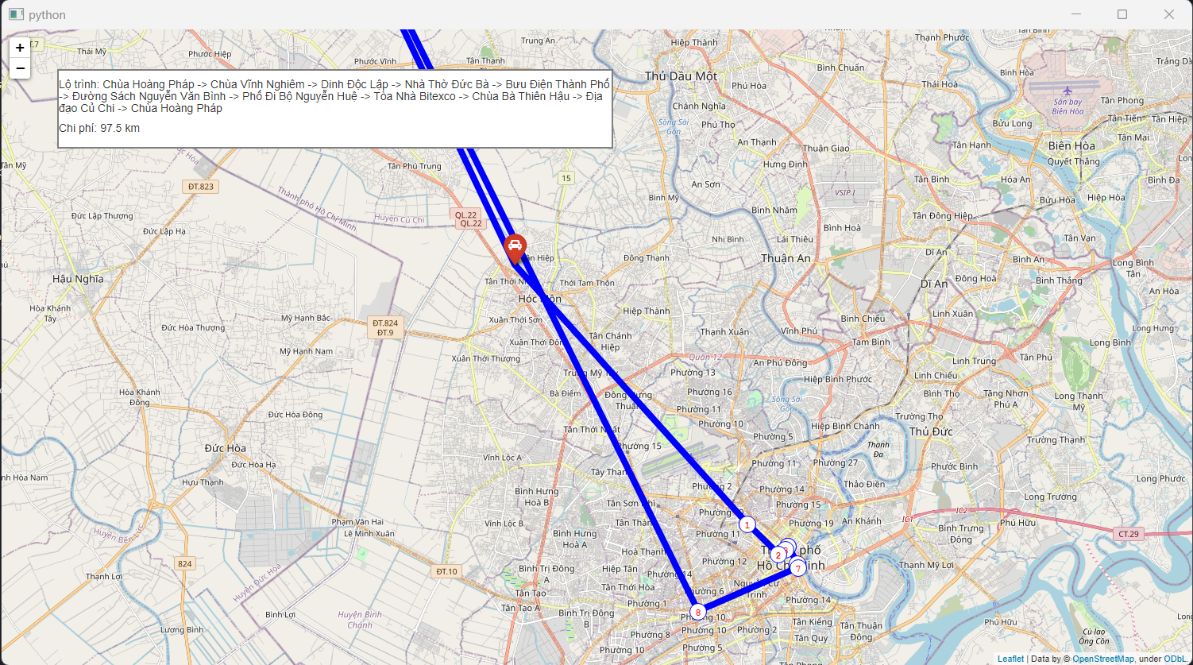
- Kỹ thuật nén dữ liệu Huffman.

- Thuật toán Ford-Fulkerson.

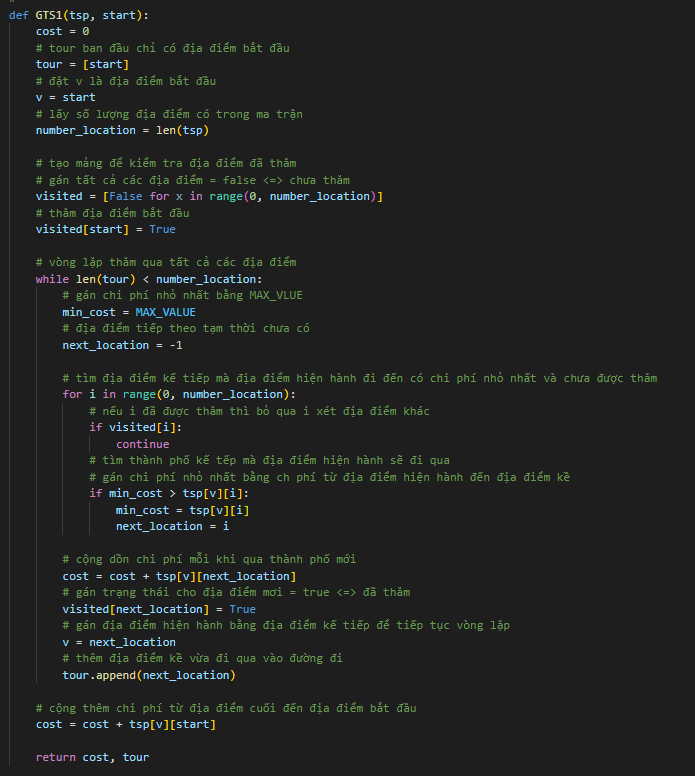
1. **Thực hiện: cài đặt ứng dụng bài toán**

**- Giao diện:**

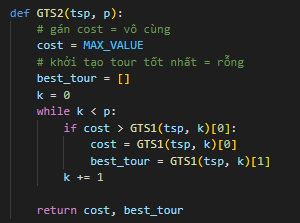
 Sử dụng giao diện tkintermapview của python để hiển thị tổng quan kết quả chạy thuật toán:

Ngoài ra còn có sự kết hợp giữa thư viện folium của và pyqt5 để giải quyết bài toán GTS2:

**GTS1 :**



Hàm GTS1 ta cần truyền vào ma trận chi phí của các thành phố và 1 điểm để bắt đầu (start) . Hàm sẽ trả về cho ta mảng tour (danh sách đường đi tới các thành phố tối ưu nhất với số chi phí thấp nhất) và cost (chi phí thấp nhất ứng với tour )

**GTS2:**

Hàm GTS2 ta cần truyền vào ma trận chi phí của các thành phố(tsp) và tổng các thành phố (p).Hàm sẽ gọi lại GTS1 và trả về cho ta mảng best\_tour (danh sách đường đi tới các thành phố tối ưu nhất trong p) và cost (chi phí thấp nhất ứng với best\_tour )

Cài đặt ứng dụng thuật toán GTS (Bao gồm GTS1 và GTS2): import tkinter

import json

import folium

from folium import plugins

from tkinter import \*

import tkinter.ttk as exTk

import tkinter as tk

from tkhtmlview import HTMLLabel

# tạo biến có giá trị cực lớn để so sánh

MAX\_VALUE = 999999

# đọc file

file = open('matrix.txt')

t = file.readline().split()

tsp = []

for i in file:

    x = list(map(float, i.split()))

    tsp.append(x)

#

def GTS1(tsp, start):

    cost = 0

    # tour ban đầu chỉ có địa điểm bắt đầu

    tour = [start]

    # đặt v là địa điểm bắt đầu

    v = start

    # lấy số lượng địa điểm có trong ma trận

    number\_location = len(tsp)

    # tạo mảng để kiểm tra địa điểm đã thăm

    # gán tất cả các địa điểm = false <=> chưa thăm

    visited = [False for x in range(0, number\_location)]

    # thăm địa điểm bắt đầu

    visited[start] = True

    # vòng lặp thăm qua tất cả các địa điểm

    while len(tour) < number\_location:

        # gán chi phí nhỏ nhất bằng MAX\_VLUE

        min\_cost = MAX\_VALUE

        # địa điểm tiếp theo tạm thời chưa có

        next\_location = -1

        # tìm địa điểm kế tiếp mà địa điểm hiện hành đi đến có chi phí nhỏ nhất và chưa được thăm

        for i in range(0, number\_location):

            # nếu i đã được thăm thì bỏ qua i xét địa điểm khác

            if visited[i]:

                continue

            # tìm thành phố kế tếp mà địa điểm hiện hành sẽ đi qua

            # gán chi phí nhỏ nhất bằng ch phí từ địa điểm hiện hành đến địa điểm kề

            if min\_cost > tsp[v][i]:

                min\_cost = tsp[v][i]

                next\_location = i

        # cộng dồn chi phí mỗi khi qua thành phố mới

        cost = cost + tsp[v][next\_location]

        # gán trạng thái cho địa điểm mơi = true <=> đã thăm

        visited[next\_location] = True

        # gán địa điểm hiện hành bằng địa điểm kế tiếp để tiếp tục vòng lặp

        v = next\_location

        # thêm địa điểm kề vừa đi qua vào đường đi

        tour.append(next\_location)

    # cộng thêm chi phí từ địa điểm cuối đến địa điểm bắt đầu

    cost = cost + tsp[v][start]

    return cost, tour

def GTS2(tsp, p):

    # gán cost = vô cùng

    cost = MAX\_VALUE

    # khởi tạo tour tốt nhất = rỗng

    best\_tour = []

    k = 0

    while k < p:

        if cost > GTS1(tsp, k)[0]:

            cost = GTS1(tsp, k)[0]

            best\_tour = GTS1(tsp, k)[1]

        k += 1

    return cost, best\_tour

# đọc tất cả tên địa điểm và tọa độ trong file locations.json

locations\_json = json.load(open('locations.json',  encoding='utf-8'))

locations = []

coordinates = {}

for i in locations\_json:

    location = locations\_json[i]['location']

    locations.append(location)

    coordinates[location] = (

        float(locations\_json[i]['lat']), float(locations\_json[i]['long']))

def getPoints(type):

    # lấy chi phí

    cost=0

    map = folium.Map(location=[10.7773895, 106.6885525], zoom\_start=12)

    # lấy giá trị combobox để chon địa điểm bắt đầu

    index = cbo.get()

    # nếu tên địa điểm vừa nhập có trong danh sách địa điểm thì thay tên địa điểm = vị trí để gọi thuật toán

    for i in range(len(locations)):

        if index == locations[i]:

            index = i

    # gọi lại thuật toán thêm vào tour

    tour = []

    if type == 1:

        for i in GTS1(tsp, index)[1]:

            tour.append(i)

        cost = GTS1(tsp, index)[0]

    if type == 2:

        for i in GTS2(tsp, len(locations))[1]:

            tour.append(i)

        cost = GTS2(tsp, len(locations))[0]

    # hiển thị tên địa điểm thay vì index

    for i in range(len(tour)):

        for j in range(len(locations)):

            # nếu ví trị j nào bằng với giá trị tour tại i

            # gán giá trị tour tại i bằng tên địa điểm tại j

            if j == tour[i]:

                tour[i] = locations[j]

    # tạo mảng chứa các tọa độ

    points = []

    for location in tour:

        points.append(coordinates[location])

    points.append(points[0])

    i = 1

    j = 1

    # tạo điểm đánh dấu từ tọa độ thứ 2 đến gần cuối

    while i < (len(tour)):

        while j < (len(points)-1):

            folium.Marker(points[j],

                          popup=tour[i],

                          tooltip=tour[i],

                          icon=plugins.BeautifyIcon(number=j,

                                                    border\_color='blue',

                                                    border\_width=1,

                                                    text\_color='red'

                                                    )

                          ).add\_to(map)

            i += 1

            j += 1

    # tạo điểm đánh dấu cho tọa độ xuất phát

    folium.Marker(points[0],

                  popup=tour[0],

                  tooltip=tour[0],

                  icon=folium.Icon(color='red', prefix='fa', icon='car')

                  ).add\_to(map)

    # vẽ đường đi cho các điểm tọa độ

    folium.PolyLine(points,

                    color='blue',

                    weight=8,

                    ).add\_to(map)

    # tạo chuỗi str để hiển thị lộ trình

    str = ''

    for i in range(len(tour)):

        str = str + ' ' + tour[i] + ' ->'

    str = str + ' ' + tour[0]

    # hiển thị lộ trình và chi phí lên trang html

    map.get\_root().html.add\_child(folium.Element("""

    <div style="position: fixed;

        top: 50px; left: 70px; width: 700px; height: 100px;

        background-color:white; border:2px solid grey;z-index: 900;">

        <h5>Lộ trình: {}</h5>""".format(str) + """

        <h5>Chi phí: {:.3f} km</h5>""".format(cost) + """

    </div>

    """))

    # cập nhật lại chi phí

    map.save('map.html')

# create tkinter window

root\_tk = tkinter.Tk()

root\_tk.title("Thuật Toán GTS")

cbo = exTk.Combobox(root\_tk, width=30, font='Times 30', state='readonly')

cbo['value'] = locations

cbo.current(0)

btnChonGTS1 = Button(root\_tk, text='Thực hiện GTS1', font='Times 30',

                 command=lambda:getPoints(1))

btnChonGTS2 = Button(root\_tk, text='Thực hiện GTS2', font='Times 30',

                 command=lambda:getPoints(2))

lblShowMap = HTMLLabel(root\_tk,  html='<a href="map.html">Xem bản đồ</a>')

cbo.grid(row=0, column=0)

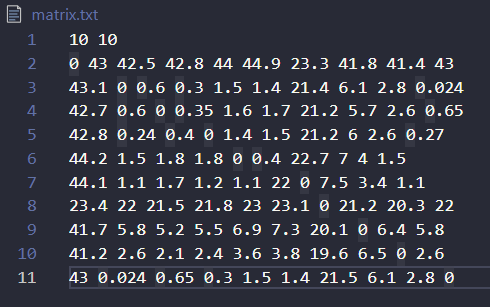
btnChonGTS1.grid(row=0, column=1)

btnChonGTS2.grid(row=1, column=1)

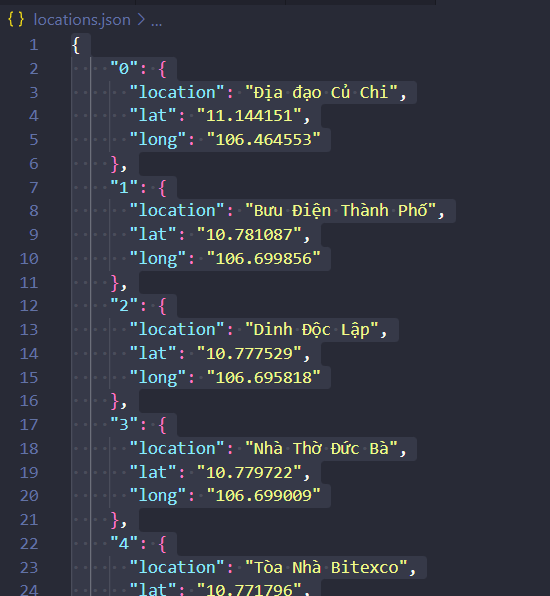
lblShowMap.grid(row=2, column=0)

root\_tk.mainloop()

**- File dữ liệu:**

File matrix.txt là ma trận chi phí để lưu dữ liệu khoảng cách từ 1 điểm đến các điểm còn lại:

File loctions.json để lưu các cặp key – value để lấy tên và tọa độ mỗi điểm



## **VIII. Kết Luận:**

- Thuật toán đơn giản, có thể tìm được đường đi có chi phí thấp nhưng không được tối ưu do đây là thuật toán “Tham lam”. Cứ hướng đi nào thấp thì đi bất chấp về sau.

# **TÀI LIỆU THAM KHẢO**

1. ỨNG DỤNG KỸ THUẬT KHAI PHÁ DỮ LIỆU ĐỂ TƯ VẤN HỌC TÂP CHO SINH VIÊN, <http://trituevietvn.com/chi-tiet/-UNG-DUNG-KY-THUAT-KHAI-PHA-DU-LIEU-DE-TU-VAN-HOC-TAP-CHO-SINH-VIEN-61?fbclid=IwAR1NblpzimBU20qX2KatPFxb007KvrZng6jU2gwKtLFURGLRqNmd-jShGIw>

2. Top 10 thắng cảnh ở Thành phố Hồ Chí Minh, <https://www.tripadvisor.com.vn/Attractions-g293925-Activities-c47-Ho_Chi_Minh_City.html>

3. <https://www.academia.edu/6451036/Ngbanhang?fbclid=IwAR3qmv_mOJpIf0KdKaGIPmOvRPt5JOVOnDgT5ktDlSZEGDXumNEQdClq1iA> , Bài toán người du lịch (TSP)