**Báo cáo project**

**Yêu cầu chương trình :**

* Chương trình sử dụng thuật toán tìm đường đi ngắn nhất giữa hai điểm, sau đó hiển thị trên cửa sổ phần mềm bằng ngôn ngữ Python
* Bên cạnh đó, chương trình có khả năng hiện thời gian chạy được thuật toán và quãng đường đi được

**Giao diện chương trình:**

* Sử dụng thư viện tkinter để vẽ những nút bấm, bản đồ và những thông tin liên quan
* window = tkinter.Tk()
* window.title("Shortest path GUI")
* window.geometry("1500x800")
* window['bg'] = 'white'

Khởi tạo cửa sổ giao diện

* Những nút bấm bao gồm : Tìm đường đi ngắn nhất, hiển thị những đường đi giữa các đỉnh với nhau, chuyển bản đồ (hiện nay có 2 bản đồ)
* Thông tin liên quan bao gồm quãng đường giữa 2 điểm đi được và thời gian thực thi thuật toán tìm đường đi ngắn nhất

Diagram

Description automatically generated

Toàn cảnh giao diện của chương trình

Diagram

Description automatically generated

Chương trình khi hiển thị tất cả các điểm

Diagram

Description automatically generated with medium confidence

Chương trình khi tìm được đường đi ngắn nhất và những thông số

**Xây dựng đồ thị:**

* Dùng 2 vòng lập lồng nhau để kiểm tra đường đi giữa tất cả các điểm, có thể kiểm tra nếu đường đi ấy cắt bản đồ bằng thư viện Shapely
* Để có 1 đồ thị được dùng trong việc tìm đường đi ngắn nhất, dạng đồ thị ở đây là danh sách kề
* Bên cạnh đó, những điểm trên canvas là trong hệ trục tọa độ, cho nên để đưa vào graph ta có thể đánh số nó từ 1 đến n để thuận tiện hơn. Khi gọi điểm đó ta có thể dùng coordinates[i] qua 1

**Thuật toán dijkstra:**

* Thuật toán được xây dựng để tìm đường đi ngắn nhất từ điểm bắt đầu (mặc định là chuột phải) đến những điểm còn lại. Qua đó, ta có distance[i] lưu lại được khoảng cách từ điểm bắt đầu tới điểm i
* Để vẽ được đường đi thì ta có thể truy xuất từ par[i] tức là điểm cần vẽ sau khi vẽ điểm i
* Toàn cảnh thuật toán dijkstra

start = n-2

end = n-1

        visited = []

        distance = []

        par = []

        for i in range(n):

            visited.append(False)

            distance.append(float('inf'))

            par.append(None)

        distance[start] = 0

        pq = PriorityQueue()

        pq.put((0, start))

        while not pq.empty():

            (d, v) = pq.get()

            visited[v] = True

            for i in range(len(graph[v])):

                if not visited[graph[v][i][0]] and distance[v] + graph[v][i][1] < distance[graph[v][i][0]]:

                    distance[graph[v][i][0]] = distance[v] + graph[v][i][1]

                    par[graph[v][i][0]] = v

                    pq.put((distance[graph[v][i][0]], graph[v][i][0]))

        print(distance[end])

        distancetoend = distance[end]

        coordinates.append((right\_point.x, right\_point.y))

        coordinates.append((left\_point.x, left\_point.y))

        while par[end] != None:

            tmp = canvas.create\_line(coordinates[par[end]][0], coordinates[par[end]][1], coordinates[end][0], coordinates[end][1], fill="purple", width=5)

            print(end, " ", par[end], " ", coordinates[end], " ", coordinates[par[end]])

            drawarray.append(tmp)

            end = par[end]

**Những tính năng thêm:**

* Chuyển đổi giữa bản đồ: bản đồ được lưu bởi mảng coordinates, mảng đó lưu lại tọa độ các đỉnh của bản đồ và được lưu trong file points.txt. Khi đổi sang bản đồ thứ 2, ta đọc file points2.txt và khởi tạo lại mảng coordinates để lưu lại những điểm của bản đồ mới là vẽ lên.
* Hiển thị tất cả đường đi : ta có biến graph là danh sách kề chứa đường đi giữa các điểm. Qua đó ta có thể dễ dàng vẽ được đường đi của đồ thị
* Hiển thị thời gian thực thi : ta có thư viện time để đo được thời gian giữa 1 thuật toán dijkstra bằng cách đưa

start\_time = time.time()

vào đầu function và ở cuối function ta lấy được thời gian thực thi bằng cách

elapsed\_time = time.time() - start\_time

**Thư viện sử dụng và tài liệu tham khảo:**

* Thư viện shapely (để kiểm tra xem 1 điểm có nằm trong bản đồ, 2 đường thẳng cắt nhau)
* Thư viện time (để đo thời gian thực hiện thuật toán)
* Thư viện Tkinter (để tạo giao diện chương trình)
* Nguồn tham khảo:
  + <https://gis.stackexchange.com/questions/82697/shapely-linestring-and-polygon-intersect>
  + https://stackoverflow.com/questions/36399381/whats-the-fastest-way-of-checking-if-a-point-is-inside-a-polygon-in-python