TRUY VẤN DỰA TRÊN DỮ LIỆU KHÔNG GIAN

Nội dung

I. Mô tả tổng quan

1. Cơ sở dữ liệu không gian

- Sử dụng dữ liệu không gian từ Openstreetmap (file osm)
- Phạm vi dữ liệu tải xuống: Việt Nam
- Phạm vi dữ liệu sử dụng trong project: Thành phố Hà Nội

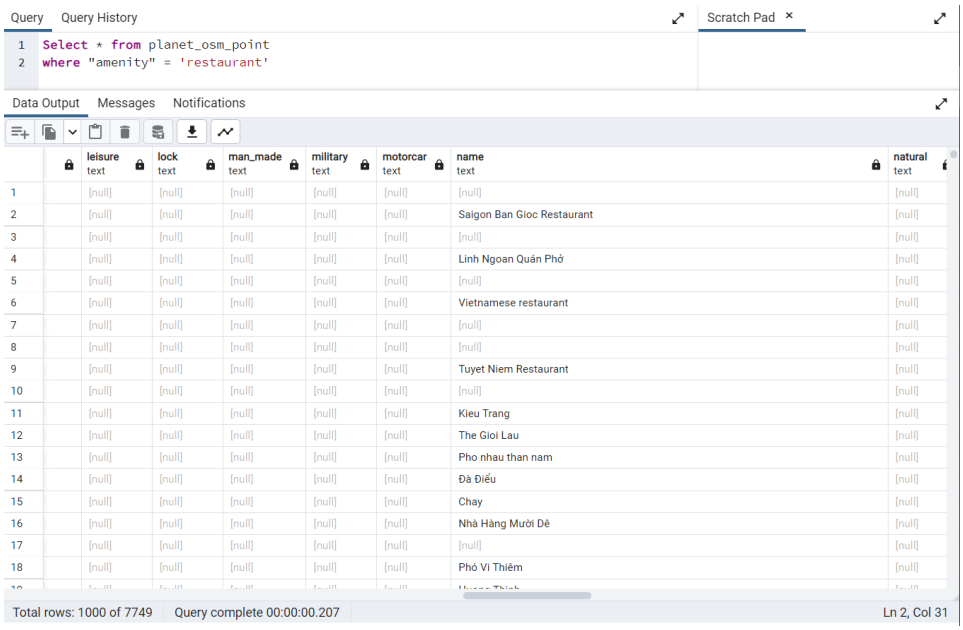
2. Thiết lập cơ sở dữ liệu

- Lưu trữ dữ liệu không gian trên PostgreSQL và PostGIS
- Các bảng dữ liệu được tải xuống sử OSM:
  + planet_osm_point: chứa tập dữ liệu các điểm trong Việt Nam
  + planet_osm_line: chứa tập dữ liệu tất cả các đường trong Việt Nam
  + planet_osm_roads: chứa tập dữ liệu các đường lớn trong Việt Nam (là tập con của planet_osm_line)
  + planet_osm_polygon: chứa tập dữ liệu các vùng trong Việt Nam

3. Xử lý dữ liệu

➢ Xác định dữ liệu không gian: "restaurant"

- Có 7749 bản ghi Nhà hàng trong tập dữ liệu.



➢ Các bảng dữ liệu sử dụng

a. Bảng geometries

- Bảng chứa tập dữ liệu "point" được tạo mới dựa trên bảng "planet_osm_point" tải xuống từ Openstreetmap và có tạo thêm các cột long, lat, longlat để phục vụ cho truy vấn.

```sql
SELECT planet_osm_point.*,
ST_X(planet_osm_point.way) AS X1, --point x
ST_Y(planet_osm_point.way) AS Y1, --point y
ST_X(ST_TRANSFORM(planet_osm_point.way,4674)) AS LONG, -- longitude point x SIRGAS 2000
ST_Y(ST_TRANSFORM(planet_osm_point.way,4674)) AS LAT, --latitude point y SIRGAS 2000
ST_ASTEXT(planet_osm_point.way) AS XY, --wkt point xy
ST_ASTEXT(ST_TRANSFORM(planet_osm_point.way,4674)) AS LongLat
INTO geometries
FROM
planet_osm_point
```

Truy vấn tạo bảng geometries

– Các cột sử dụng cho truy vấn:

	osm_id bigint	amenity text	name text	way geometry	long double precision	lat double precision	longlat text
1	660416724	[null]	Maritime delimitati...	0101000020110F0000C5E0C934F9026741A...	108.3791667	20.4013889003885	POINT(108.3791667 20.4013889003885)
2	7714070557	[null]	[null]	0101000020110F0000E4FC79178AA866417...	106.7154067	22.594979200397226	POINT(106.7154067 22.594979200397226)
3	7714109430	[null]	[null]	0101000020110F0000AC75F6AECAA86641...	106.72004859999998	22.585977200397465	POINT(106.72004859999998 22.585977200397465)
4	7714109433	[null]	下琅口岸	0101000020110F0000F856598AD9A86641C...	106.72111629999999	22.58512290039749	POINT(106.72111629999999 22.58512290039749)
5	7714109427	[null]	Cửa khẩu Bí Hà	0101000020110F00009E6B747EBEA866414...	106.71917259999998	22.58705140039744	POINT(106.71917259999998 22.58705140039744)
6	7623772102	[null]	[null]	0101000020110F00003EAB3B43B5A466414...	106.64491919999999	22.471636700400346	POINT(106.64491919999999 22.471636700400346)
7	7623772150	[null]	[null]	0101000020110F00002B470E95B7A466418...	106.64508590000001	22.471846800400343	POINT(106.64508590000001 22.471846800400343)
8	4627752711	[null]	那花	0101000020110F00003A481044B9A266411...	106.60841189999998	22.379864800402395	POINT(106.60841189999998 22.379864800402395)
9	4627752710	[null]	陇冬	0101000020110F00004D91D7D3E6A16641...	106.59328869999999	22.396897200402027	POINT(106.59328869999999 22.396897200402027)
10	1893363692	[null]	[null]	0101000020110F0000E76021DC6EA166413...	106.58466719999998	22.373735400402524	POINT(106.58466719999998 22.373735400402524)
11	2603721197	[null]	Cửa khẩu Nà Nưa	0101000020110F0000EBD49EF21BA16641B...	106.5787087	22.374343100402506	POINT(106.5787087 22.374343100402506)
12	4627752709	[null]	陇刀	0101000020110F000026B773237EA16641A...	106.5857652	22.409734600401748	POINT(106.5857652 22.409734600401748)
13	4627752707	[null]	江巷	0101000020110F00006E80056602A166419...	106.57687260000002	22.427541800401364	POINT(106.57687260000002 22.427541800401364)
14	4627752705	[null]	布局	0101000020110F0000DCCA4A4E62A16641...	106.583765	22.424577500401423	POINT(106.583765 22.424577500401423)
15	5371213741	[null]	[null]	0101000020110F0000179578D97EA066417...	106.5674188	22.429317000401316	POINT(106.5674188 22.429317000401316)
16	1890365764	[null]	[null]	0101000020110F00004D66CE396FA066418...	106.56629599999998	22.429717900401304	POINT(106.56629599999998 22.429717900401304)
17	1073848677(	ferry_termi...	[null]	0101000020110F0000B280AF2CEEA06641D...	106.5754192	22.469345100400396	POINT(106.5754192 22.469345100400396)
18	9694325963	[null]	水口街	0101000020110F000023C294F306A166414...	106.5771998	22.46750490040044	POINT(106.5771998 22.46750490040044)
19	9694344769	[null]	业秀园	0101000020110F000051EC36C818A16641C...	106.57848120000001	22.468242500400418	POINT(106.57848120000001 22.468242500400418)
20	1073853027(	ferry_termi...	[null]	0101000020110F0000650043C64A166419...	106.58390359999999	22.475021600400265	POINT(106.58390359999999 22.475021600400265)
21	4964032697	[null]	Cửa khẩu Tà Lùng	0101000020110F0000FA9E419029A16641A...	106.5796872	22.474160900400285	POINT(106.5796872 22.474160900400285)

osm_id: id của từng point
amenity: phân loại, thuộc tính của các point (vd: atm, restaurant, hospital, school, …)
way: tọa độ hình học của point dưới dạng mã HEXEWKB
long: kinh độ
lat: vĩ độ
longlat: kinh độ, vĩ độ của point

b. Bảng planet_osm_polygon

– Bảng dữ liệu chứa các “polygon” được tải xuống từ Openstreetmap.
– Các cột sử dụng cho truy vấn:

osm_id bigint	amenity text	boundary text	name text	way geometry
1079049760	community_centre	[null]	Nhà thiếu nhi Kim Đồng	0103000020110F000001000000090000001B1510AEAB8F6641B4FCAA9D8AC543413B2AA75
1079166328	police	[null]	Công an tỉnh Cao Bằng	0103000020110F000001000000050000000552F37E59F8F6641B3C920A09BC54341BD945EF5
-14358919	townhall	[null]	HĐND - UBND tỉnh Cao Bằng	0103000020110F000002000000090000000070B48246A98F6641A1FB0960E4C5434141B6416A
1079166334	school	[null]	Trường Tiểu học Hoà Chung	0103000020110F000001000000090000000F41337F5828F6641E6C7C8F608C5434172663C1D
1079166322	[null]	[null]	[null]	0103000020110F000001000000050000000C2B3E6AB8F664127185D3969C54341414E2CFF
1079166325	[null]	[null]	[null]	0103000020110F000001000000050000000B2B593EAF8F6641FD195B4E41C54341339DFF02
1079166327	police	[null]	Công an thành phố Cao Bằng	0103000020110F000001000000050000000031700BC7B48F6641FFD54B4A3EC54341F08148DF
1079166321	[null]	[null]	[null]	0103000020110F000001000000D0000000826290BAD8F6641975FCFDF55C5434158329B87
1079166324	townhall	[null]	Thành uỷ - HĐND - UBND thành phố Cao Bằng	0103000020110F000001000000A0000003335AD9DAB8F66414642C69868C5434151C6513C
1079166323	[null]	[null]	[null]	0103000020110F000001000000050000080F0B616B48F66412239B4916AC543410FFDDA7E
1079166326	police	[null]	Công an tỉnh Cao Bằng	0103000020110F000001000000050000024A40E68B58F6641C324E7655AC543414C1B7390
1079049763	school	[null]	Trường THCS Hợp Giang	0103000020110F000001000000070000001C53957FB88F6641E697171EF8C44341C4F7B2A5
741257927	[null]	[null]	[null]	0103000020110F000001000000D7000000FA47A7ECDE8E664130BB7740D6C74341099E61ED
1079166329	kindergarten	[null]	Trường Mầm non Hoà Chung	0103000020110F000001000000060000006494B11C8E8F66419E2B43D968C44341BCB3658F
1089629449	[null]	[null]	Bộ Chỉ huy Quân sự tỉnh Cao Bằng	0103000020110F000001000000012000000B962264AB88F66412C459D3CB0C34341C2945B9E
1068173329	hospital	[null]	Bệnh viện Y học cổ truyền Cao Bằng	0103000020110F000001000000F000000A5F1CCC9E78F6641A6F9DA7B92C343419ADF253C
-7100875	[null]	administrative	Thành phố Cao Bằng	0103000020110F000001000000DB010000B9AF674A0B8B6641954C160BFAC6434108FF6D32
1149062814	police	[null]	Công an tỉnh Cao Bằng	0103000020110F000001000000F000000024417A66AE8F66411936BE9143C343418509B3C3
1073138448	[null]	[null]	Mỏ sắt Nà Rụa	0103000020110F000001000000027000000D11A8042698F6641CC36781165C143418395168F
520782952	[null]	[null]	[null]	0103000020110F000001000000050000000A9EF651D7584664170169 24ACE9843415BACF56C
520785396	[null]	[null]	[null]	0103000020110F000001000000050000000060306E7E628466416BB743EBC2984341FA18D6A8

osm_id: id của từng polygon
amenity: phân loại, thuộc tính của các polygon (vd: atm, restaurant, hospital, school, …)
boudary: danh giới để lấy ra các polygon lớn như tỉnh, thành phố (vd: administrative, …)
way: tọa độ, hình dạng các đa giác (polygon) dưới dạng mã HEXEWKB

c. Bảng osm_2po_4pgr

– Bảng "osm_2po_4pgr" được tạo ra sau khi tiền xử lý dữ liệu Openstreetmap (osm) bằng công cụ "osm2po" để sử dụng trong pgrouting.
– Các cột sử dụng cho truy vấn:

source integer	target integer	km double precision	kmh integer	cost double precision	geom_way geometry
1194	406662	0.0657002	90	0.00073	0102000020E6100000020000090A4FF40942C5A40FC5CC87E710C2540F7C0D88C9D2C5A402019BCF9570
406662	406660	0.1421605	90	0.0015796	0102000020E6100000060000F7C0D88C9D2C5A402019BCF9570C2540882018 51A42C5A407E9C7AEE4E0
406660	406658	0.1200068	90	0.0013334	0102000020E6100000020000638C5940B22C5A40ECB47FAF320C2540BA69334EC32C5A403F1F65C4050
406658	1439231	0.0733742	90	0.0008153	0102000020E6100000020000BA69334EC32C5A403F1F65C4050C2540498C54CECD2C5A407BEF1417EC0
1439231	406653	0.0227366	90	0.0002526	0102000020E6100000020000498C54CECD2C5A407BEF1417EC0B25409BAC510FD12C5A40DFE75322E40
406653	1439233	0.1071466	90	0.0011905	0102000020E6100000020000009BAC510FD12C5A40DFE75322E40B25408D0F0E51E02C5A40948F38BFBC0
1439233	1500471	0.6416706	90	0.0071297	0102000020E610000006000008D0F0E51E02C5A40948F38BFBC0B25400EE25EF4F02C5A40BBFE6ECB910
1500471	1417238	0.1892335	90	0.0021026	0102000020E6100000040000000D00946CA3B2D5A404CEF2C8BD30A25405DBE9AA84A2D5A408A7A1C61AC
1417238	1417237	0.1024979	90	0.0011389	0102000020E6100000020000000B01EF7AD562D5A40B24EF0A88C0A2540BAC61D25652D5A402697B503640
1417237	406583	0.1242448	90	0.0013805	0102000020E6100000020000000BAC61D25652D5A402697B503640A2540BD74EED1762D5A402693AEF4350
406583	1417235	0.3833625	90	0.0042596	0102000020E6100000050000000BD74EED1762D5A402693AEF4350A2540354AF2B7982D5A4034CA445BDF0
1417235	1417233	0.2167515	90	0.0024084	0102000020E610000003000000 3CF14174AD2D5A401A55E12AAA092540D4CED5FBB22D5A409879BCDA9B
1417233	1417230	0.0680753	90	0.0007564	0102000020E6100000030000006 2EB634FCC2D5A40D22D85515A09254000A370E2D02D5A40D8158E7B4E0
1417230	1417226	0.0483053	90	0.0005367	0102000020E6100000020000057355200D62D5A4008B18E3E41092540E2A6ABE0DC2D5A40EE92DD712F0
1417226	1439611	0.3610257	90	0.0040114	0102000020E6100000050000 00E2A6ABE0DC2D5A40EE92DD712F092540ED65DB69EB2D5A401D1CEC4D0C
1439611	1443008	1.1790501	90	0.0131006	0102000020E6100000080000002157EA59102E5A40AEF4DA6CAC0825405A187D60222E5A40073CD5C67D0

source: thuộc tính cho nút nguồn
target: thuộc tính cho nút đích
km: khoảng cách
kmh: tốc độ
cost: chi phí

geom_way: tọa độ hình học của các đoạn đường dưới dạng mã HEXEWKB

4. Hiển thị kết quả

– Sử dụng Qgis để hiển thị các point và network routing tìm được từ các hàm tự tạo (các hàm dựa trên query và có thể sử dụng Python cho việc trả về kết quả).

II. Các chức năng

A, Sử dụng khoảng cách Euclide : ST_Distance

1. Tìm Nhà hàng gần nhất với 1 điểm xác định trước

Bài toán thực tế: Tìm kiếm nhà hàng gần nhất với vị trí hiện tại.

Cách giải quyết:

– Input: tọa độ vị trí cho trước gồm longtitude và latitude của điểm đó.

– Output: thông tin về nhà hàng gần nhất bao gồm tên, tọa độ longtitude và latitude,...

⇨ Sử dụng hàm ST_Distance để tính khoảng cách từ các nhà hàng đến điểm cho trước, sau đó sắp xếp các bản ghi theo giá trị tăng dần của khoảng cách này và lấy bản ghi đầu tiên.

Ví dụ: Tìm nhà hàng gần nhất với 1 điểm có kinh độ, vĩ độ cho trước

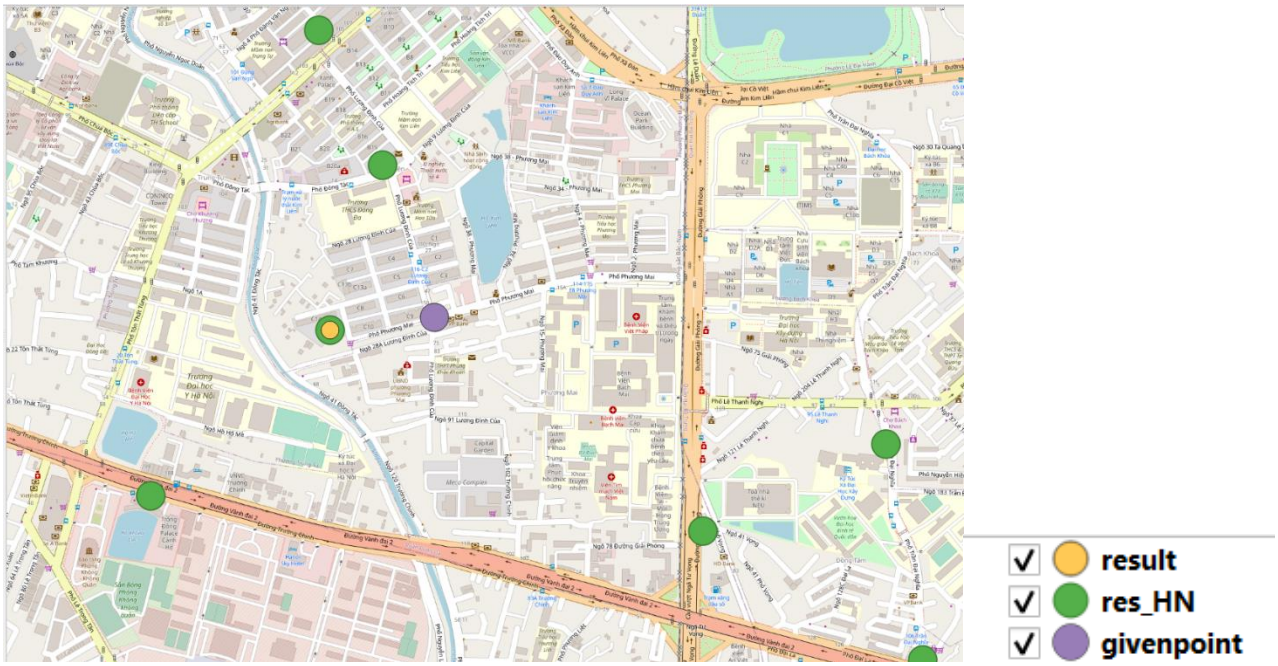
Query | Query History

```
1  ----find 1 restaurants that is nearest to given query point
2  SELECT *
3  FROM geometries
4  WHERE amenity = 'restaurant'
5  ORDER BY ST_Distance(
6    ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857),
7    ST_Transform(ST_SetSRID(ST_MakePoint(105.836111, 21.00342980040441), 4326), 3857) --given point
8  )
9  LIMIT 1;
10
```

Data Output | Messages | Notifications

	osm_id bigint	access text	addr:housename text	addr:housenumber text	addr:interpolation text	admin_level text	aerialway text	aeroway text	amenity text	ar te
1	6419220950	[null]	[null]	[null]	[null]	[null]	[null]	[null]	restaurant	[n

Kết quả:



2. Tìm Nhà hàng gần nhất với nhiều điểm xác định trước

Bài toán thực tế: Tìm kiếm nhà hàng gần nhất với nhiều vị trí của nhiều người tìm kiếm, trong trường hợp những người này muốn gặp nhau.

Cách giải quyết:

- Input: tọa độ longtitude và latitude của nhiều điểm cho trước.
- Ouput: thông tin về nhà hàng gần nhất, bao gồm tên, vị trí longtitude và latitude,...
  - ⇨ Sử dụng hàm ST_Distance để tính tổng khoảng cách từ cách nhà hàng đến nhiều điểm cho trước này, sau đó sắp xếp các bản ghi theo giá trị tăng dần của khoảng cách này và lấy bản ghi đầu tiên.

Ví dụ: Tìm Nhà hàng gần nhất với 2 điểm cho trước

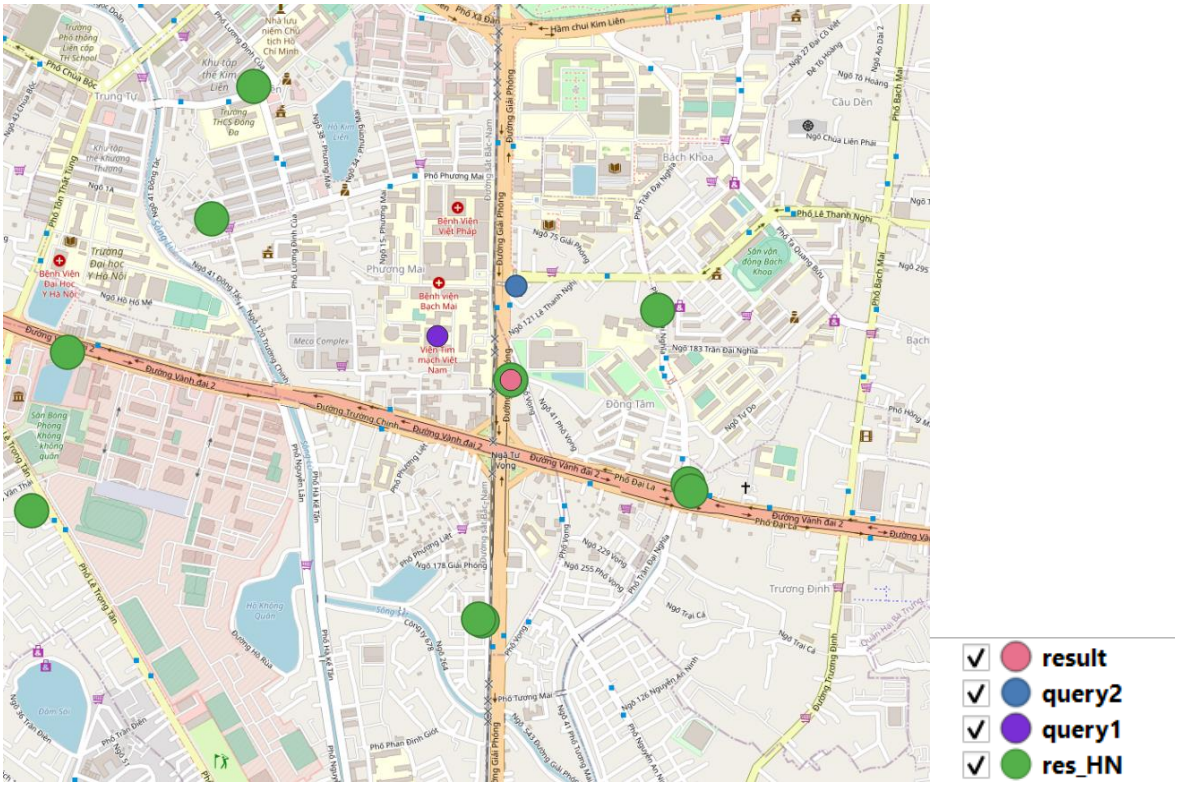
Query Query History

```
12  ----find 1 restaurant that is nearest to given 2 query points
13  SELECT *
14  FROM geometries
15  WHERE amenity = 'restaurant'
16  ORDER BY
17    ST_Distance(geometries.longlat::geography, ST_SetSRID(ST_MakePoint(105.83967809999999, 21.000431500404364), 4326)::geography) +
18    ST_Distance(geometries.longlat::geography, ST_SetSRID(ST_MakePoint(105.84166189999999, 21.00160820040438), 4326)::geography)
19  LIMIT 1;
```

Data Output Messages Notifications

	osm_id bigint	access text	addr:housename text	addr:housenumber text	addr:interpolation text	admin_level text	aerialway text	aeroway text	amenity text	area text	barrier text	bicycle text
1	9931534174	[null]	[null]	[null]	[null]	[null]	[null]	[null]	restaurant	[null]	[null]	[null]

Kết quả:



3. Tính mật độ nhà hàng trong một khu vực

Bài toán thực tế: Một doanh nghiệp muốn mở 1 chuỗi các nhà hàng và muốn tham khảo những chuỗi nhà hàng khác đã mở bao nhiêu quán trong 1 quận.

Cách giải quyết:

- Input: tên quận cần khảo sát
- Ouput: Mật độ nhà hàng trong quận đó, đơn vị: số nhà hàng trên km2
  - ⇨ Lấy Số nhà hàng nằm trong khu vực chỉ định (sử dụng hàm count để tính số nhà hàng và ST_Within để lọc những nhà hàng nằm trong khu vực chỉ định) chia cho Diện tích của khu vực này (sử dụng hàm ST_Area để trả về diện tích)

Ví dụ: Tính mật độ nhà hàng trong Quận Hai Bà Trưng

```
----density of restaurants in the area
WITH num AS(SELECT COUNT(*) AS numOf
            FROM geometries
            WHERE amenity = 'restaurant'
            AND ST_Within(way, (SELECT way
                                FROM planet_osm_polygon p
                                WHERE p.name LIKE '%Hai Bà Trưng%'
                                AND P.boundary = 'administrative'))),
S AS(
--find S of polygon
--HBT
SELECT ST_Area(way) /1000000 as square --km2
FROM planet_osm_polygon p
WHERE p.name LIKE '%Hai Bà Trưng%'
AND P.boundary = 'administrative')

SELECT numOf/square AS soNgtrenKm2 FROM num,S
```

Kết quả:

	songtrenkm2 double precision
1	4.939901735693913

4. Khoảng cách trung bình giữa 2 nhà hàng khác nhau trong khu vực

Bài toán thực tế: Một doanh nghiệp muốn mở 1 chuỗi các nhà hàng và muốn tham khảo những doanh nghiệp khác đã xây dựng chuỗi nhà hàng với khoảng cách là bao nhiêu.

Cách giải quyết:

- Input: tên quận cần khảo sát
- Ouput: khoảng cách trung bình giữa các nhà hàng
  - ⇨ Sử dụng ST_Distance để tính khoảng cách giữa 2 nhà hàng khác nhau trong khu vực chỉ định sau đó lấy trung bình(AVG) các khoảng cách này.

Ví dụ:

```
----find average distance between 2 diffirent restaurants in area
SELECT AVG(distance)
FROM  (SELECT h.osm_id, a.osm_id, ST_Distance(h.way, a.way) AS distance
FROM geometries h
     CROSS JOIN geometries a
WHERE h.amenity = 'restaurant'
      AND a.amenity = 'restaurant'
      AND ST_Within(a.way, (SELECT way
                            FROM planet_osm_polygon
                            WHERE name LIKE '%Quận Hai Bà Trưng'
                            AND boundary = 'administrative'))
      AND ST_Within(h.way, (SELECT way
                            FROM planet_osm_polygon
                            WHERE name LIKE '%Quận Hai Bà Trưng'
                            AND boundary = 'administrative'))
      AND h.osm_id <> a.osm_id
ORDER BY distance DESC) AS res
```

Kết quả:

	avg double precision
1	1218.9728012937996

B, Dựa trên khoảng cách network routing : pgrouting

Trong phần này chúng ta chỉ sử dụng giải thuật tìm đường Dijsktra trong Pgrouting để tìm đường đi (network routing) giữa 2 điểm hoặc giữa nhiều điểm.

5. Tìm đường đi và chi phí giữa 2 điểm

Bài toán thực tế: Tìm đường đi giữa vị trí hiện tại của 2 người.

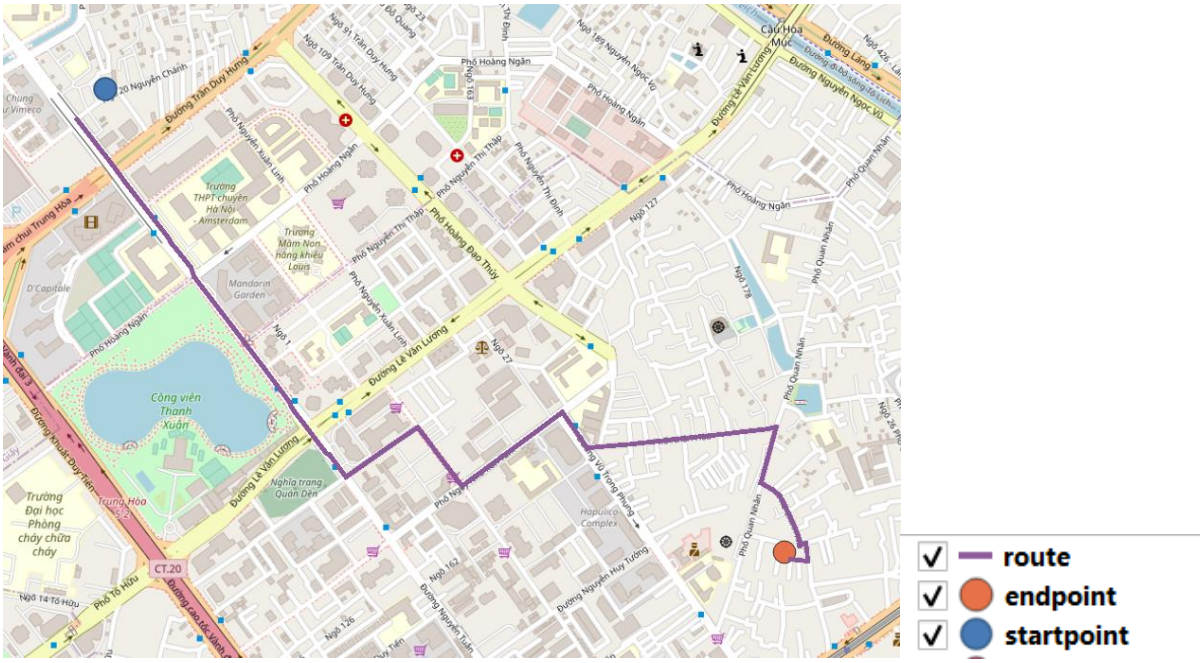
Cách giải quyết:

- Input: Vị trí của 2 điểm cho trước, bao gồm longtitude và latitude
- Output: Đường đi và chi phí
  - ⇨ Sử dụng Dijsktra trong pgrouting để tìm kiếm đường đi giữa 2 điểm. Tìm 2 điểm nằm trên đường mà gần nhất của 2 điểm cho trước(start and destination), sau đó sử dụng dijsktra để tìm đường đi ngắn nhất giữa 2 điểm vừa tìm này.

```
----find path&cost between 2 given point
WITH start AS (
  SELECT topo.source
  FROM osm_2po_4pgr as topo
  ORDER BY topo.geom_way <-> ST_SetSRID(
    ST_GeomFromText('POINT(105.795665 21.00939500040452)'),4326) --point(long lat)
  LIMIT 1
),
destination AS (
  SELECT topo.source
  FROM osm_2po_4pgr as topo
  ORDER BY topo.geom_way <-> ST_SetSRID(
    ST_GeomFromText('POINT(105.810844 20.99971330040434)'),4326) --point(long lat)
  LIMIT 1
)

select sum(fi.route_cost) , ST_Union(fi.route_geometry)
from
( SELECT (di.cost) AS route_cost, ST_Union(pt.geom_way) AS route_geometry
FROM pgr_dijkstra(
  'SELECT id, source, target, ST_Length(ST_Transform(geom_way, 3857)) AS cost FROM osm_2po_4pgr',
  ARRAY(SELECT source FROM start),
  ARRAY(SELECT source FROM destination),
  directed := false
) AS di
JOIN osm_2po_4pgr AS pt ON di.edge = pt.id
GROUP BY di.cost) as fi;
```

Kết quả:



6. Tìm đường đi network routing theo thuật toán dijkstra tới KNN được xác định bởi Euclide

Bài toán thực tế: Tìm đường đi và chi phí từ vị trí hiện tại đến K nhà hàng gần nhất.

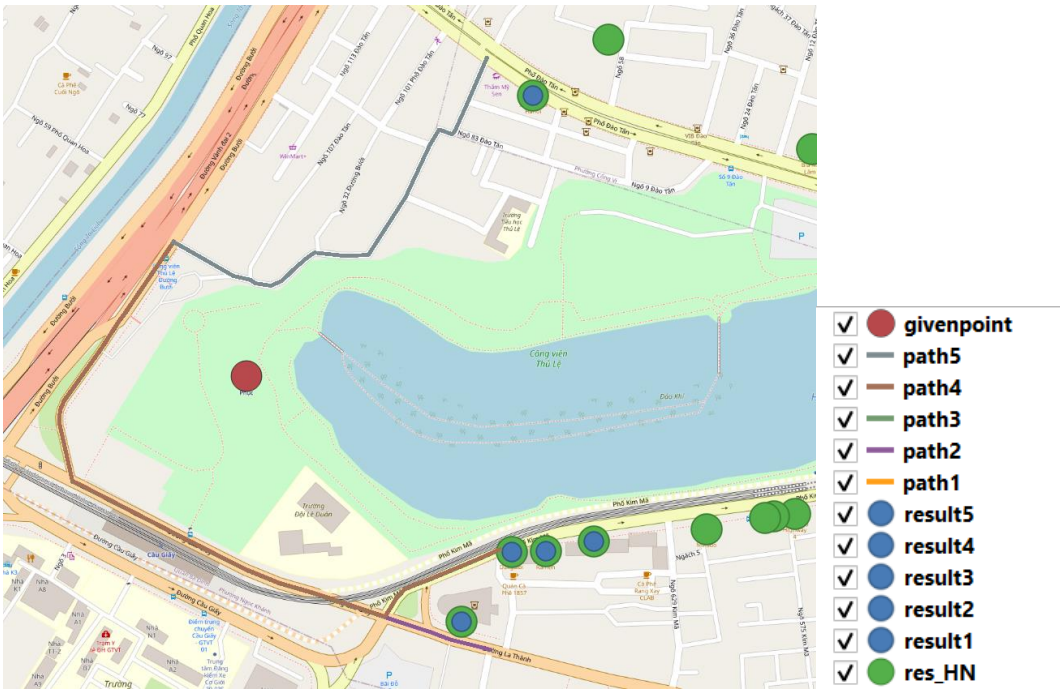
Cách giải quyết:

- Input: tọa độ vị trí điểm cho trước, bao gồm longtitude và latitude
- Output: đường đi và chi phí từ điểm đó đến K nhà hàng gần nhất
  - ⇨ Tìm K nhà hàng gần nhất với điểm cho trước (theo Euclid sử dụng St_Distance), sau đó sử dụng Dijsktra tìm đường đi đến K nhà hàng này từ điểm cho trước.

Ví dụ: Tìm đường đi thật tới 5NN được xác định bằng euclide

```
--find path pgRouting from a given query to KNN having shorest st_distance:
---------find point that haing the shortest Euclide distance and find road
WITH tenNN AS (
  SELECT long, lat
  FROM geometries
  WHERE amenity = 'restaurant'
  ORDER BY ST_Distance(
    ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857),
    ST_Transform(ST_SetSRID(ST_MakePoint(105.8042238, 21.030628300404896), 4326), 3857) --given point
  )
  LIMIT 5 --5 nearest points with euclidean distance to given point
)
SELECT
  point.long,point.lat,
  (
    SELECT ST_AsText(ST_Union(geom_way)) AS route  --network routing
    FROM pgr_dijkstra(
      'SELECT id, source, target,ST_Length(ST_Transform(geom_way, 3857)) AS cost FROM osm_2po_4pgr',
      (SELECT source
        FROM osm_2po_4pgr
        ORDER BY geom_way <-> ST_SetSRID(ST_MakePoint(105.8042238, 21.030628300404896), 4326) --given point
        LIMIT 1),
      ARRAY
      (SELECT topo.source
        FROM osm_2po_4pgr AS topo
        ORDER BY topo.geom_way <-> ST_SetSRID(ST_MakePoint(point.long, point.lat), 4326)
        LIMIT 1),
      directed := false
    ) AS di
    JOIN osm_2po_4pgr AS pt ON di.edge = pt.id
  ) AS route
FROM tenNN AS point
```

Kết quả:



7. Tìm điểm có network routing ngắn nhất với điểm cho trước (dựa trên bài toán 6).

Bài toán thực tế: Tìm điểm nhà hàng có đường đi ngắn nhất tới vị trí hiện tại.

Cách giải quyết:

- Input: tọa độ điểm cho trước, bao gồm longtitude và latitude
- Output: tọa điểm nhà hàng gần nhất và đường đi đến điểm này
  - ⇨ Từ 6, xác định điểm mà network routing có cost nhỏ nhất (sắp xếp KNN ở bước 6 theo giá trị chi phí cost tăng dần và chọn bản ghi đầu tiên).

```
--FILTER 5NN to 1NN having shorest lost by dijkstra
--find point, path that has the shortest route by Dijkstra from KNN nearest by Euclide
EXPLAIN(WITH tenNN AS (
  SELECT long, lat
  FROM geometries
  WHERE amenity = 'restaurant'
  ORDER BY ST_Distance(
    ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857),
    ST_Transform(ST_SetSRID(ST_MakePoint(105.8042238, 21.030628300404896), 4326), 3857)) -- given point
  LIMIT 5) --5 nearest points with euclidean distance to given point
 SELECT
  point.long,
  point.lat,
  lost_route.lost AS lost,
  lost_route.route_geometry
FROM tenNN AS point
CROSS JOIN LATERAL (
  SELECT
    sum(di.cost) AS lost,
    ST_Union(pt.geom_way) AS route_geometry
  FROM pgr_dijkstra(
    'SELECT id, source, target, ST_Length(ST_Transform(geom_way, 3857)) AS cost FROM osm_2po_4pgr',
    (
      SELECT source
      FROM osm_2po_4pgr
      ORDER BY geom_way <-> ST_SetSRID(ST_MakePoint(105.8042238, 21.030628300404896), 4326) -- given point
      LIMIT 1),
    ARRAY(
      SELECT topo.source
      FROM osm_2po_4pgr AS topo
      ORDER BY topo.geom_way <-> ST_SetSRID(ST_MakePoint(point.long, point.lat), 4326)
      LIMIT 1),
    directed := false) AS di
  JOIN osm_2po_4pgr AS pt ON di.edge = pt.id) AS lost_route
order by lost --sort nearest points based on total cost (lost) ascending
limit 1)
;
```

Kết quả:

⇨ Điểm này là điểm "tốt nhất" trong tập candidate KNN tìm được, nó có thể là điểm gần nhất theo đường đi thật nhưng không đảm bảo luôn đúng. Dù vậy, chi phí của bài toán là nhỏ.

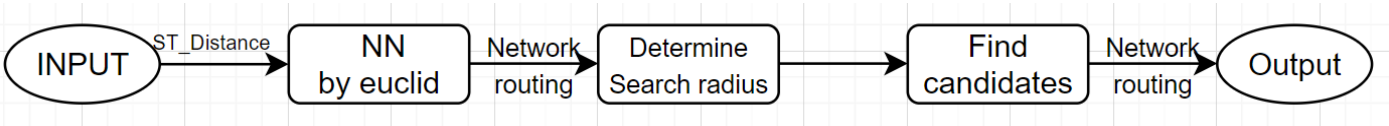
8. Tìm điểm có đường đi network routing thực sự ngắn nhất theo IER

Bài toán thực tế: Tìm điểm nhà hàng có đường đi ngắn nhất tới vị trí hiện tại.

Cách giải quyết:

- Input: tọa độ điểm cho trước, bao gồm longtitude và latitude
- Output: tọa điểm nhà hàng gần nhất và đường đi đến điểm này

Chuỗi hành động:



B1. Tìm điểm NN theo Euclid distance của điểm cho trước

```
#find euclide NN
db.execute(f"""select long, lat FROM geometries
            where amenity='restaurant'
            order by ST_Distance(ST_Transform(ST_SetSRID(ST_MakePoint({long}, {lat}), 4326), 3857),
                        ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857))
            limit 1""")
qPoint=db.fetchall()
```

B2. Tính khoảng cách network routing từ given point đến điểm NN trên

```
#find network routing
#qPoint[0][0],qPoint[0][1]: longtitude and latitude of NN point
db.execute(f"""
             WITH start AS (
           SELECT topo.source
           FROM osm_2po_4pgr as topo
           ORDER BY topo.geom_way <-> ST_SetSRID(
             ST_GeomFromText('POINT ({long} {lat})'),
                   4326)
                   LIMIT 1
                ),
             destination AS (
               SELECT topo.source
               FROM osm_2po_4pgr as topo
               ORDER BY topo.geom_way <-> ST_SetSRID(
             ST_GeomFromText('POINT ({qPoint[0][0]} {qPoint[0][1]})'),
                   4326)
               LIMIT 1
                 )
             SELECT sum(di.cost) as realcost,ST_Union(geom_way) as path
                 FROM pgr_dijkstra('
             SELECT id,
                  source,
                  target,
                  ST_Length(ST_Transform(geom_way, 3857)) AS cost
                     FROM osm_2po_4pgr',
                 array(SELECT source FROM start),
                 array(SELECT source FROM destination),
                 directed := false) AS di
                 JOIN   osm_2po_4pgr AS pt
                   ON   di.edge = pt.id
                   """)
result=db.fetchall()
DEmax= result[0][0] #network routing distance from NN point to given point
path= result[0][1]
```

B3. Tìm tập các candidat nằm trong khoảng cách tìm được ở trên (DEmax) hay nằm trong bán kính tìm kiếm DEmax với tâm là điểm cho trước

```
#DE max: network routing from given point to NN (NN by euclid)
db.execute(f"""select long, lat FROM geometries
            where amenity='restaurant'
            and ST_Distance(ST_Transform(ST_SetSRID(ST_MakePoint({long}, {lat}), 4326), 3857),
                        ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857))<=  {DEmax}
            """)
NN=db.fetchall()
```

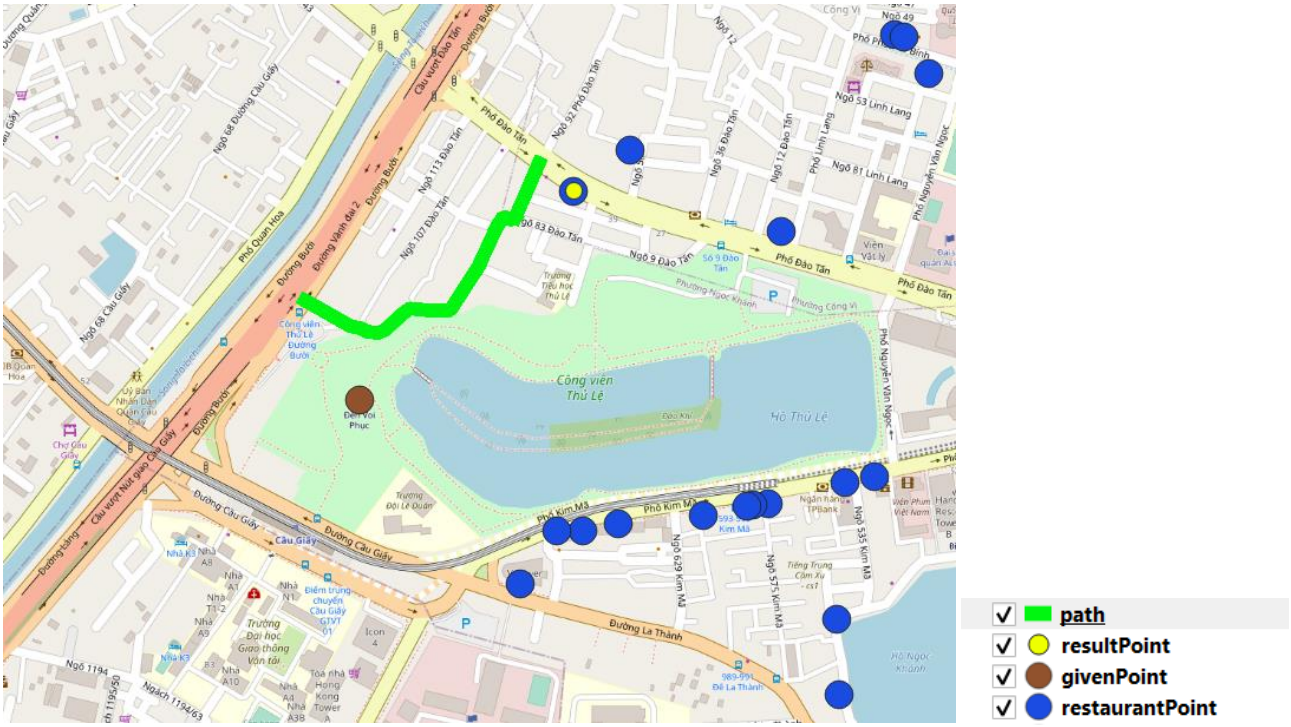
B4. Từ các candidate tìm được ở B3 (tập KNN), tính network routing distance đến điểm cho trước (khoảng cách DEmax), cập nhật khoảng cách này nếu gặp giá trị nhỏ hơn và cuối cùng chúng ta được giá trị nhỏ nhất của khoảng cách này cùng với điểm kết quả.

```
#find result
for i in NN:
    db.execute(f"""
            WITH start AS (
                  SELECT topo.source
                  FROM osm_2po_4pgr as topo
                  ORDER BY topo.geom_way <-> ST_SetSRID(
                            ST_GeomFromText('POINT ({long} {lat})'),4326)
                  LIMIT 1
                            ),
                destination AS (
                      SELECT topo.source
                      FROM osm_2po_4pgr as topo
                      ORDER BY topo.geom_way <-> ST_SetSRID(
                        ST_GeomFromText('POINT ({i[0]} {i[1]})'),4326)
                      LIMIT 1
                                  )
            SELECT ST_AsText(ST_Union(geom_way)), sum(di.cost) as realcost
                FROM pgr_dijkstra('
                        SELECT id,
                               source,
                               target,
                                ST_Length(ST_Transform(geom_way, 3857)) AS cost
                        FROM osm_2po_4pgr',
                                array(SELECT source FROM start),
                                array(SELECT source FROM destination),
                                directed := false) AS di
                JOIN   osm_2po_4pgr AS pt
                ON   di.edge = pt.id
                  """)
    x=db.fetchall()
    if(x[0][1]>DEmax) : continue
    else:
        path=x[0][0]
        DEmax =x[0][1]
        qPoint=i
```

Kết quả:

```
IER(105.8042238, 21.030628300404896)
```

```
((105.8068682, 21.033045300404943),
 471.871298337968,
 'MULTILINESTRING((105.8035522 21.0317839,105.8036057 21.031756,105.8037051 21.0317041,105.8041041 21.0314911,105.8042574 21.03
14584,105.8043078 21.0314273,105.8044495 21.031407,105.8045519 21.0314498,105.8046592 21.0315475,105.8047719 21.0316401,105.804
8496 21.0316927,105.8048891 21.0316887,105.8050964 21.0316676,105.8051715 21.0316651,105.8053484 21.0316726,105.805693 21.03217
33),(105.805693 21.0321733,105.8059144 21.0326189),(105.8059144 21.0326189,105.8060016 21.0327537,105.8061194 21.0327128,105.80
61471 21.0327422),(105.8061471 21.0327422,105.8062683 21.032994),(105.8062683 21.032994,105.8064193 21.0333239,105.806441 21.03
33701))')
```



- ⇨ Với bán kính tìm kiếm là network distance của điểm NN từ B1 của điểm cho trước được tìm thấy bởi IER, nếu không tìm thấy được candidat tốt hơn thì điểm NN trên chính là kết quả của bài toán. Mặt khác, với các nhà hàng nằm ngoài bán kính tìm kiếm thì chắc chắc khoảng cách network routing đến điểm cho trước sẽ lớn hơn bán kính tìm kiếm.
- ⇨ Vì vậy, kết quả cho ta được điểm thực sự có đường đi ngắn nhất, nhưng chi phí tính toán lớn hơn so với mục 7.

Nhận xét : Trong hàm tìm kiếm IER, chúng ta có sử dụng 1 vòng lặp for cho việc tính khoảng cách network routing từ điểm cho trước tới các candidat nằm trong bán kính tìm kiếm. Kết quả được trả về sau 10 vòng lặp (mật độ nhà hàng trong khu vực quanh điểm cho trước là lớn).

9. Tìm điểm Nhà hàng gần nhất với nhiều điểm cho trước

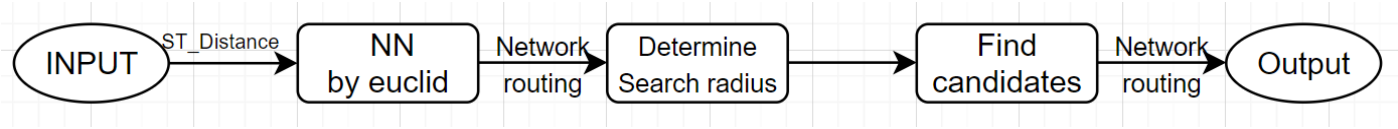
Bài toán thực tế: Nhiều người cùng muốn tìm 1 địa điểm nhà hàng mà gần với tất cả nhất. Để giới hạn lại không gian tính toán, ở đây chúng ta thực hiện với 2 điểm cho trước

Cách giải quyết:

- Input: tọa độ các điểm cho trước, bao gồm longtitude và latitude
- Output: tọa độ nhà hàng gần nhất với những điểm này và đường đi

Cách 1: Sử dụng khoảng cách Euclid để làm bán kính tìm kiếm candidat

Chuỗi hành động tương tự với 8, nhưng input lúc này là 2 điểm



B1. Tìm điểm NN theo euclide distance của 2 điểm cho trước và tính khoảng cách từ từng NN đến điểm cho trước của nó.

```
def euclid(long,lat):
    db.execute(f"""select long, lat,
    ST_Distance(ST_Transform(ST_SetSRID(ST_MakePoint({long}, {lat}), 4326), 3857),
                                ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857))
    FROM geometries
                where amenity='restaurant'
                order by ST_Distance(ST_Transform(ST_SetSRID(ST_MakePoint({long}, {lat}), 4326), 3857),
                                ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857))

                limit 1""")
    NN=db.fetchall()
    return NN
```

```
#POINT1(105.80145149999998 21.035779800404985)
x1=euclid(105.80145149999998, 21.035779800404985) #given point 1
x1 #NN của point1
```

```
[(105.8000055, 21.03867230040504, 380.69136656644343)]
```

```
#POINT2(105.78767219999999 21.036938200405007)
x2=euclid(105.78767219999999, 21.036938200405007) #given point 2
x2 #NN của point2
```

```
[(105.78914509999998, 21.035509500404984, 236.47278118702116)]
```

B2. So sánh và chọn khoảng cách lớn hơn để làm bán kính tìm kiếm candidat.

Ở ví dụ trên, ta chọn khoảng cách lớn hơn là khoảng cách từ x1 đến point 1: max1=380.69136656644343

B3. Tìm tất cả các điểm nhà hàng nằm trong bán kính này của từng điểm cho trước. Sau đó lấy giao hoặc hợp của 2 tập candidat này.

```
def find_NN(long,lat,max1):
    db.execute(f"""
    select long, lat FROM geometries
                where amenity='restaurant'
                AND  ST_Distance(ST_Transform(ST_SetSRID(ST_MakePoint({long}, {lat}), 4326), 3857),
                                 ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857)) <=  {max1}

                """)
    Point=db.fetchall()
    return Point
```

```
NN1=find_NN(105.80145149999998, 21.035779800404985,x1[0][2])  #x1[0][2] = max1
NN1
```

```
[(105.8000055, 21.03867230040504)]
```

```
NN2=find_NN(105.78767219999999, 21.036938200405007,x1[0][2])
NN2
```

```
[(105.7857867, 21.038758100405033),
 (105.7855911, 21.03890420040504),
 (105.78562479999998, 21.03890360040504),
 (105.78545609999999, 21.038906700405043),
 (105.78548989999999, 21.03890600040504),
 (105.7855236, 21.03890550040504),
 (105.7854859, 21.038763600405037),
 (105.7889278, 21.034573200404967),
 (105.78894929999998, 21.034648300404964),
 (105.78900139999999, 21.034937500404972),
 (105.78914509999998, 21.035509500404984),
 (105.78967349999999, 21.034375400404958),
 (105.7893302, 21.034508100404963)]
```

Chú ý: Sử dụng union hay intersection cho tập KNN của 2 điểm cho trước chỉ để thay đổi tập candidat và để giảm thời gian tính toán. Với union, tập candidat lớn hơn và có thể cho kết quả tốt hơn so với intersection.

Trong ví dụ này, chúng ta chỉ sử dụng cho phép hợp vì phép giao cho tập rỗng.

```
intersection = reduce(lambda acc, x: acc + [x] if x in NN1 and x not in acc else acc, NN2, [])
intersection
```

```
[]
```

```
union=set(NN1).union(NN2)
union
```

```
{(105.78545609999999, 21.038906700405043),
 (105.7854859, 21.038763600405037),
 (105.78548989999999, 21.03890600040504),
 (105.7855236, 21.03890550040504),
 (105.7855911, 21.03890420040504),
 (105.78562479999998, 21.03890360040504),
 (105.7857867, 21.038758100405033),
 (105.7889278, 21.034573200404967),
 (105.78894929999998, 21.034648300404964),
 (105.78900139999999, 21.034937500404972),
 (105.78914509999998, 21.035509500404984),
 (105.7893302, 21.034508100404963),
 (105.78967349999999, 21.034375400404958),
 (105.8000055, 21.03867230040504)}
```

B4. Với tập candidat mới, tìm network routing từ 2 điểm cho trước đến tập candidate này, sau đó sắp xếp theo chi phí cost tăng dần và chọn bản ghi có network routing nhỏ nhất.

```
result=[]
for i in intersection:
    db.execute(f"""
            WITH start1 AS (
                  SELECT topo.source
                  FROM osm_2po_4pgr as topo
                  ORDER BY topo.geom_way <-> ST_SetSRID(
                              ST_GeomFromText('POINT ({long1} {lat1})'),4326)
                  LIMIT 1
                              ),
                start2 AS (
                  SELECT topo.source
                  FROM osm_2po_4pgr as topo
                  ORDER BY topo.geom_way <-> ST_SetSRID(
                              ST_GeomFromText('POINT ({long2} {lat2})'),4326)
                  LIMIT 1
                              ),
                destination AS (
                      SELECT topo.source
                      FROM osm_2po_4pgr as topo
                      ORDER BY topo.geom_way <-> ST_SetSRID(
                        ST_GeomFromText('POINT ({i[0]} {i[1]})'),4326)
                      LIMIT 1
                                 )
            SELECT  {i} as point,(ST_Union(geom_way)), sum(di.cost) as realcost
                FROM pgr_dijkstra('
                        SELECT id,
                               source,
                               target,
                                ST_Length(ST_Transform(geom_way, 3857)) AS cost
                        FROM osm_2po_4pgr',
                                array(SELECT source FROM start1 UNION SELECT source FROM start2),
                                array(SELECT source FROM destination),
                                directed := false) AS di
                JOIN   osm_2po_4pgr AS pt
                ON   di.edge = pt.id
                  """)
    x=db.fetchall()
    if(x[0][-1] is not None):
        result.append(x)
```

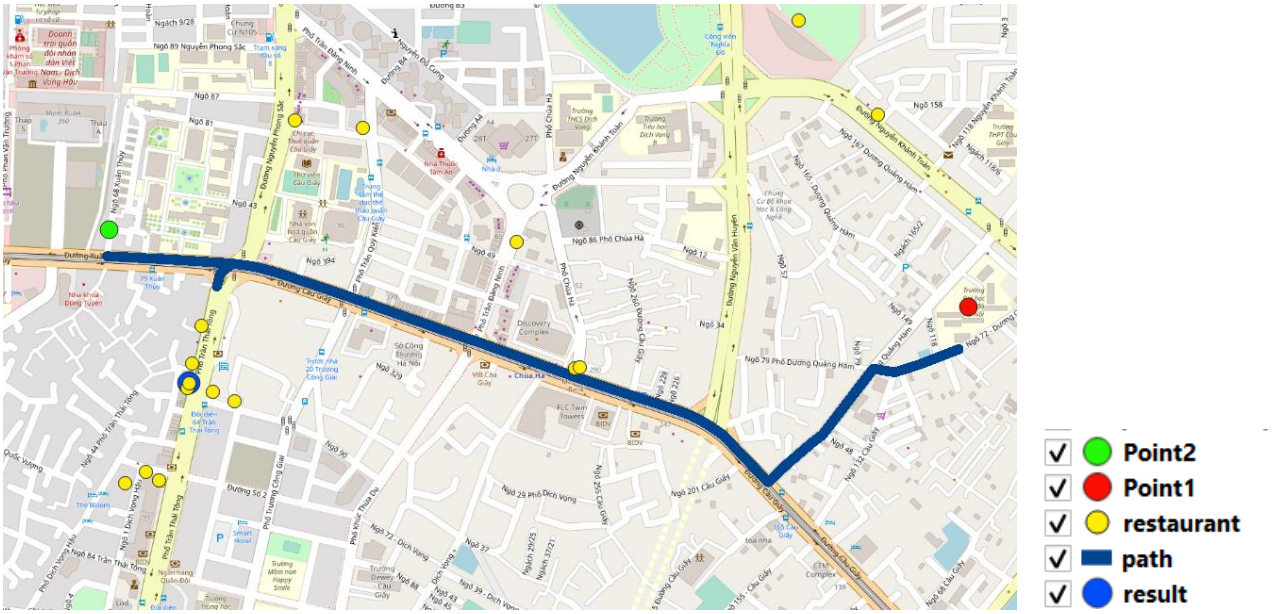
```
sorted_x= sorted(result, key=lambda x: x[0][-1])
return sorted_x[0] #result point, path, cost
```

Kết quả: tìm được kết quả với tổng chi phí cost=1813

```
#POINT1(105.80145149999998 21.035779800404985)
#POINT2(105.78767219999999 21.036938200405007)
result=problem9(long1=105.80145149999998,
lat1= 21.035779800404985,
long2=105.78767219999999,
lat2=21.036938200405007)
```

```
result  #union
```

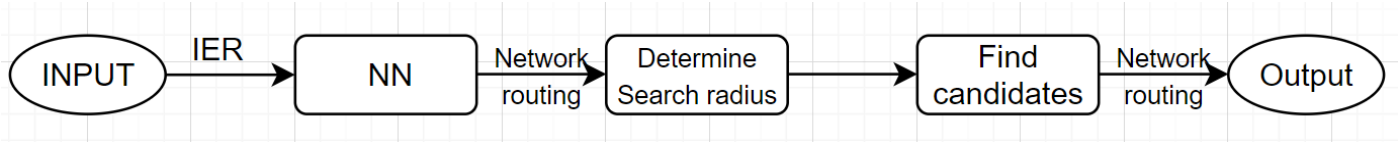

```
[('(105.78894929999998,21.034648300404964)',
  '0105000020E6100000210000001020000000200000079628CFE86725A40A90DF38A4C093540329FBD4486725A40D584A39A4809354001020000000200000329FBD4486725A40D584A39A4809354098C68A7585725A407517CD6F3D0935400102000000020000000770C4A489725A4088F0D4C85309354079628CFE86725A40A90DF38A4C0935400102000000020000000D60DCAEA8C725A409A33EC41520935400770C4A489725A4088F0D4C85309354001020000000500000007C105DAB98725A40A6D82CF246093540B9EF62AB95725A40578B998E4A0935400F42E50492725A40EA245B5D4E093540ECA2E8818F725A4012842BA050093540D60DCAEA8C725A409A33EC41520935400102000000020000004C3ED1D09A725A40BFB0A140440935407C105DAB98725A40A6D82CF24609354001020000000300000098C11891A8725A402DD21FF53209354004087A03A7725A402C561EEE340935404C3ED1D09A725A40BFB0A1404409354001020000000300000031 4C5C7DAB725A403BD400EF2E093540C123850DAA725A4064D7ACE93009354098C11891A8725A402DD21FF532093540010200000002000000 7D2E0906C6725A40E2D528DA0B093540314C5C7DAB725A403BD400EF2E0935400102000000030000004 6F5317AC9725A408569CEB007093540B84082E2C7725A4037F4609C090935407D2E0906C6725A40E2D528DA0B0935400102000000030000000CDE3D5DECC725A40F3BB9F09030935403384 08DDCA725A408C46E3F50509354046F5317AC9725A408569CEB007093540010200000003000000505B7E3BE4725A40A3CFA2D2E30835403423CD69E0725A40A74C20DBE8083540CDE3D5DECC725A40F3BB9F090309354001020000000300000001F02FD74F2725A4053848A60D20835408B5F67F9F0725A4070213427D4083540505B7E3BE4725A40A3CFA2D2E3083540010200000002000000999537D1F8725A40B0D3FEBDCA0835401F02FD74F2725A4053848A60D2083540010200000002000000085D61E51FC725A40BE79BB6FC6083540010200000003000000 6594C21701735A409812EE3AC00835405A7CAFC6FE725A4031F9556FC308354085D61E51FC725A40BE79BB6FC6083540010200000004000000F53115D06735A4050296508B6083540397CD28904735A40A12FBDFDB908354064A593B602735A4004E9BD8CBD0835406594C21701735A409812EE3AC0083540010200000002000000072B0EDA309735A401E34BBEEAD0835400F53115D06735A4050296508B6083540010200000004000000094D3E81F0E735A40098B8A389D083540AC63A6FE0B735A40B1B096F1A508354072DB638E0A735A40F017B325AB08354072B0EDA309735A401E34BBEEAD0835400102000000020000004879C5816735A407EFCA5457D08354094D3E81F0E735A40098B8A389D083540010200000040000008109DCBA1B735A4081762C5B900835405803948 61A735A4035199F138C083540E2F5AADF16735A40E38DCC237F08354048799C5816735A407EFCA5457D0835400102000000020000000EF3614E321735A401135D1E7A30835408109DCBA1B735A4081762C5B900835400102000000020000008 8AA4EBD24735A40DE1C531CAC083540EF3614E321735A401135D1E7A3083540010200000002000000051EADCFE26735A40559EE51EB708354088AA4EBD24735A40DE1C531CAC0835400102000000030000000A16E2EC31735A403485CE6BEC08354034F7EBA930735A4084961A46E608354051EADCFE26735A40559EE51EB708354001020000000020000000A16E2EC31735A403485CE6BEC083540AE5978A837735A404DA5FA29E90835400102000000020000000AE5978A837735A404DA5FA29E90835406631B1F938735A40E15E99B7EA0835400102000000030000006631B1F938735A40E15E99B7EA08354087DEE2E13D735A4049FDAB7DF0083540DE42B2DB42735A400466CFAFF70835400102000000020000000DE42B2DB42735A400466CFAFF7083540AD4214DD44735A4021730A97FA0835400102000000020000000AD4214DD44735A4021730A97FA083540A6D590B847735A40D7E2AEB9FE0835400102000000020000000A6D590B847735A40D7E2AEB9FE0835403A7BC26D48735A403C6068BFFF0835400102000000050000000770C4A489725A4088F0D4C853093540C7BAB88D86725A40527BB6B354093540C8050C3785725A40ABDA24F5540935403E1350977B725A40DF9FF76B57093540434EA95C72725A409D34684359093540010200000002000000434EA95C72725A409D34684359093540F4E967A068725A4001C68E215B093540',
  1813.2061564787077)]
```



⇨ Kết quả là điểm tốt nhất trong tập candidat tìm được nhưng không luôn thực sự là điểm có đường đi ngắn nhất.

<u>Cách 2:</u> Sử dụng khoảng cách network routing để làm bán kính tìm kiếm candidat

Chuỗi hành động tương tự cách 1:



B1. Với 2 điểm cho trước, tìm điểm gần nhất theo IER và tính tổng network routing lần lượt từ 2 điểm tìm được đến 2 điểm cho trước.

```
#POINT1(105.80145149999998 21.035779800404985)
#POINT2(105.78767219999999 21.036938200405007)
```

```
IER(105.80145149999998, 21.035779800404985)
##NN1=
```

```
((105.8000055, 21.03867230040504),
 606.024513451644,
 '0105000020E61000000A0000000102000000020000000A6D590B847735A40D7E2AEB9FE0835403A7BC26D48735A403C6068BFFF0835400102000000020000000AD4214DD44735A4021730A97FA083540A6D590B847735A40D7E2AEB9FE0835400102000000020000000DE42B2DB42735A400466CFAFF7083540AD4214DD445A4021730A97FA0835400102000000030000000DE42B2DB42735A400466CFAFF7083540F673AF3841735A40C6E5D3BE14093540F1434A5840735A4091FEA897290935400102000000020000000C029070E43735A403EC10F1835093540F1434A5840735A4091FEA897290935400102000000040000000C679EEF34E735A409D514E0F6509354016743 1BC48735A40F14927124C093540ED56E07547735A40D057DAF346093540C029070E43735A403EC10F1835093540010200000003000000A7CA9CD351735A408AE365F272093540C52BFF1051735A40383932456F093540C679EEF34E735A409D514E0F6509354001020000000200000004D37E4FA52735A4075A84B6B78093540A7CA9CD351735A408AE365F272093540010200000003000000 4D37E4FA52735A4075A84B6B78093540B4D918D51735A4043E4F4F57C09354032BA72AC41735A404DB21F73AF093540102000000020000003 2BA72AC41735A404DB21F73AF093540C21B2D6233735A4012B81A7EDC093540',
 [(105.8000055, 21.03867230040504), (105.79872529999999, 21.04008510040506)])
```

```
IER(105.78767219999999, 21.036938200405007)
#NN2=
```

```
((105.78914509999998, 21.035509500404984),
 276.80359196156576,
 '0105000020E6100000040000000102000000020000000434EA95C72725A409D3468435909354 0F4E967A068725A4001C68E215B0935400102000000050000000770C4A489725A4088F0D4C853093540C7BAB88D86725A40527BB6B354093540C8050C3785725A40ABDA24F5540935403E1350977B725A40DF9FF76B57093540434EA95C72725A409D3468435909354001020000000200000000770C4A489725A4088F0D4C85309354079628CFE86725A40A90DF38A4C093540102000000030000007 9628CFE86725A40A90DF38A4C093540329FBD4486725A40D584A39A4809354098C68A7585725A407517CD6F3D093540',
 [(105.78914509999998, 21.035509500404984)])
```

```
#calculating the cost by network routing from NN point(follow by IER) to 2 given point
def cost2point(long, lat, long1, lat1, long2, lat2):
    db.execute(f"""
               WITH start1 AS (
            SELECT topo.source
            FROM osm_2po_4pgr as topo
            ORDER BY topo.geom_way <-> ST_SetSRID(
              ST_GeomFromText('POINT ({long1} {lat1})'),
                    4326)
                    LIMIT 1
                ),
              start2 AS (
                SELECT topo.source
                FROM osm_2po_4pgr as topo
                ORDER BY topo.geom_way <-> ST_SetSRID(
              ST_GeomFromText('POINT ({long2} {lat2})'),
                    4326)
                LIMIT 1
                  ),
                 destination AS (
                SELECT topo.source
                FROM osm_2po_4pgr as topo
                ORDER BY topo.geom_way <-> ST_SetSRID(
              ST_GeomFromText('POINT ({long} {lat})'),
                    4326)
                LIMIT 1
                  )
              SELECT sum(di.cost) as realcost,ST_Union(geom_way) as path
                  FROM pgr_dijkstra('
              SELECT id,
                   source,
                   target,
                   ST_Length(ST_Transform(geom_way, 3857)) AS cost
                      FROM osm_2po_4pgr',
                  array(SELECT source FROM destination),
                  array(SELECT source FROM start1 UNION SELECT source from start2),
                  directed := false) AS di
                  JOIN   osm_2po_4pgr AS pt
                    ON   di.edge = pt.id
                    """)
    result=db.fetchall()
    cost= result[0][0]
    return cost
```

```
cost2point(105.8000055, 21.03867230040504,105.80145149999998 ,21.035779800404985,105.78767219999999, 21.036938200405007)
#sum cost of NN1
```

```
2262.4613884916093
```

```
cost2point(105.78914509999998, 21.035509500404984,105.80145149999998, 21.035779800404985,105.78767219999999, 21.036938200405007)
#sum_cost NN2
```

```
1813.2061564787077
```

B2. So sánh và chọn tổng khoảng cách nhỏ nhất trong 2 khoảng cách trên.

Ở ví dụ này, ta chọn khoảng cách từ NN2 đến 2 điểm cho trước, max1=1813.2061564787077

B3. Tìm tập candidat KNN nhà hàng của 2 điểm cho trước nằm trong bán kính tìm kiếm max1 này. Sau đó lấy giao của 2 tập này ta được 1 tập candidat mới.

```
db.execute(f"""
select long, lat FROM geometries
            where amenity='restaurant'
            AND  ST_Distance(ST_Transform(ST_SetSRID(ST_MakePoint({long2}, {lat2}), 4326), 3857),
                             ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857)) <=  {max1}

            """)
qPoint1=db.fetchall()
db.execute(f"""
select long, lat FROM geometries
            where amenity='restaurant'
            AND  ST_Distance(ST_Transform(ST_SetSRID(ST_MakePoint({long1}, {lat1}), 4326), 3857),
                             ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857)) <=  {max1}

            """)
qPoint2=db.fetchall()
intersection = reduce(lambda acc, x: acc + [x] if x in qPoint1 and x not in acc else acc, qPoint2, [])
union = set(NN1).union(NN2)
```

B4. Với tập candidat ở trên, tìm và tính tổng chi phí của network routing từ 2 điểm cho trước đến từng điểm trong tập candidate và chọn bản ghi có cost nhỏ nhất .

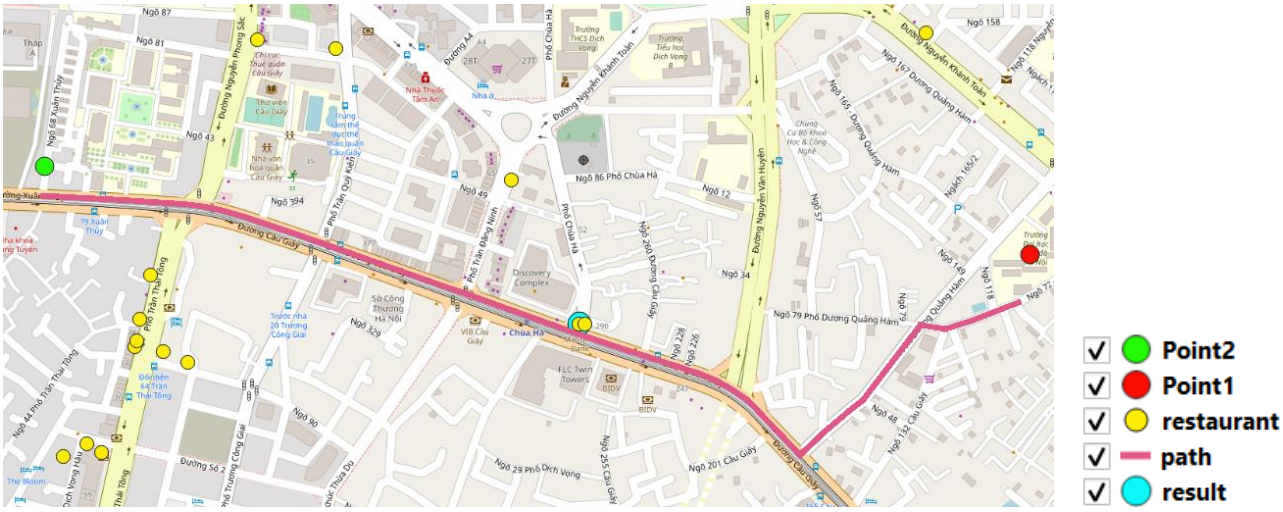
```
for i in intersection:
    db.execute(f"""
            WITH start1 AS (
                  SELECT topo.source
                  FROM osm_2po_4pgr as topo
                  ORDER BY topo.geom_way <-> ST_SetSRID(
                              ST_GeomFromText('POINT ({long1} {lat1})'),4326)
                  LIMIT 1
                            ),
                start2 AS (
                  SELECT topo.source
                  FROM osm_2po_4pgr as topo
                  ORDER BY topo.geom_way <-> ST_SetSRID(
                              ST_GeomFromText('POINT ({long2} {lat2})'),4326)
                  LIMIT 1
                            ),
                destination AS (
                      SELECT topo.source
                      FROM osm_2po_4pgr as topo
                      ORDER BY topo.geom_way <-> ST_SetSRID(
                        ST_GeomFromText('POINT ({i[0]} {i[1]})'),4326)
                      LIMIT 1
                                )
            SELECT {i} as point ,(ST_Union(geom_way)), sum(di.cost) as realcost
                FROM pgr_dijkstra('
                        SELECT id,
                               source,
                               target,
                                ST_Length(ST_Transform(geom_way, 3857)) AS cost
                        FROM osm_2po_4pgr',
                                array(SELECT source FROM start1 UNION SELECT source FROM start2),
                                array(SELECT source FROM destination),
                                directed := false) AS di
                JOIN   osm_2po_4pgr AS pt
                ON   di.edge = pt.id
                  """)
    x=db.fetchall()
    end.append(x)
sorted_x= sorted(end, key=lambda x: x[0][-1])
return sorted_x[0] #result point, path, cost
```

Kết quả: tìm được kết quả mới với chi phí nhỏ hơn so với cách 1, với cost=1709

```
def path2point(long1, lat1, long2,lat2):
    NN1=IER(long1,lat1)
    NN2=IER(long2,lat2)
    x1=cost2point(NN1[0][0],NN1[0][1],long1, lat1, long2,lat2)
    x2=cost2point(NN2[0][0],NN2[0][1],long1, lat1, long2,lat2)
    if x1>=x2:
        result=candidate(long1, lat1, long2,lat2,x2)
    else: result=candidate(long1, lat1, long2,lat2,x1)
    return result
```

```
path2point(105.80145149999998 ,21.035779800404985,105.78767219999999, 21.036938200405007)
```


```
[('(105.79514919999998,21.03487360040497)',
  '0105000020E61000001E0000000102000000020000006A6D590B847735A40D7E2AEB9FE0835403A7BC26D48735A403C6068BFFF0835400102000000020000
00AD4214DD44735A4021730A97FA083540A6D590B847735A40D7E2AEB9FE0835400102000000020000000DE42B2DB42735A4004066CFAFF7083540AD4214DD447
35A4021730A97FA0835400102000000030000006631B1F938735A40E15E99B7EA08354087DEE2E13D735A4049FDAB7DF0083540DE42B2DB42735A400466CFAF
F708354001020000000200000AE5978A837735A404DA5FA29E90835406631B1F938735A40E15E99B7EA083540010200000002000000A16E2EC31735A40348
5CE6BEC083540AE5978A837735A404DA5FA29E90835400102000000030000000A16E2EC31735A403485CE6BEC08354034F7EBA930735A4084961A46E6083540
51EADCFE26735A40559EE51EB708354001020000000200000051EADCFE26735A40559EE51EB708354088AA4EBD24735A40DE1C531CAC0835400102000000020
0000088AA4EBD24735A40DE1C531CAC083540EF3614E321735A401135D1E7A30835400102000000020000000EF3614E321735A401135D1E7A30835408109DCBA
1B735A4081762C5B900835400102000000040000008109DCBA1B735A4081762C5B900835405805394861A735A4035199F138C083540E2F5AADF16735A40E38DC
C237F08354048799C5816735A407EFCA5457D0835400102000000020000048799C5816735A407EFCA5457D08354094D3E81F0E735A40098B8A389D083540010
0200000040000094D3E81F0E735A40098B8A389D083540AC63A6FE0B735A40B1B096F1A508354072DB638E0A735A40F017B325AB08354072B0EDA309735A4
01E34BBEEAD0835400102000000020000072B0EDA309735A401E34BBEEAD0835400F53115D06735A4050296508B6083540010200000004000000F53115D06
735A4050296508B6083540397CD28904735A40A12FBDFDB908354064A593B602735A4004E9BD8CBD0835406594C21701735A409812EE3AC0083540010200000
0030000006594C21701735A409812EE3AC00835405A7CAFC6FE725A4031F9556FC308354085D61E51FC725A40BE79BB6FC6083540010200000002000000085D6
1E51FC725A40BE79BB6FC6083540999537D1F8725A40B0D3FEBDCA0835400102000000020000009999537D1F8725A40B0D3FEBDCA0835401F02FD74F2725A405
3848A60D2083540010200000003000001F02FD74F2725A4053848A60D20835408B5F67F9F0725A4070213427D4083540505B7E3BE4725A40A3CFA2D2E30835
4001020000000200000043 4EA95C72725A409D34684359093540F4E967A068725A4001C68E215B0935400102000000050000000770C4A489725A4088F0D4C85
3093540C7BAB88D86725A40527BB6B354093540C8050C3785725A40ABDA24F5540935403E1350977B725A40DF9FF76B57093540434EA95C72725A409D346843
5909354001020000000200000D60DCAEA8C725A409A33EC4152093540077 0C4A489725A4088F0D4C8530935400102000000050000007C105DAB98725A40A6D
82CF246093540B9EF62AB95725A40578B998E4A0935400F42E50492725A40EA245B5D4E093540ECA2E8818F725A4012842BA050093540D60DCAEA8C725A409A
33EC4152093540010200000002000000 4C3ED1D09A725A40BFB0A140440935407C105DAB98725A40A6D82CF2460935400102000000030000000 98C11891A8725
A402DD21FF5320935400 4087A03A7725A402C561EEE340935404C3ED1D09A725A40BFB0A140440935400102000000030000000 314C5C7DAB725A403BD400EF2E
093540C123850DAA725A4064D7ACE93009354098C11891A8725A402DD21FF532093540010200000002000000 7D2E0906C6725A40E2D528DA0B093540314C5C7
DAB725A403BD400EF2E093540010200000003000000 46F5317AC9725A408569CEB007093540B84082E2C7725A4037F4609C090935407D2E0906C6725A40E2D5
28DA0B093540010200000003000000 CDE3D5DECC725A40F3BB9F0903093540338408DDCA725A408C46E3F50509354046F5317AC9725A408569CEB0070935400
1020000000300000 0505B7E3BE4725A40A3CFA2D2E30835403423CD69E0725A40A74C20DBE8083540CDE3D5DECC725A40F3BB9F0903093540',
  1709.078164072508)]
```



⇨ Với bán kính tìm kiếm là tổng network distance của điểm NN(theo từng điểm cho trước) tới 2 điểm cho trước được tìm bởi IER, nếu trong bán kính tìm kiếm này không có candidat nào tốt hơn thì điểm NN ở trên chính là kết quả của bài toán. Mặt khác, với những điểm nhà hàng nằm ngoài bán kính tìm kiếm này, dễ thấy chắc chắn tổng đường đi của chúng đến 2 điểm cho trước sẽ lớn hơn bán kính này.

⇨ Vì vậy kết quả là điểm thực sự có tổng đường đi tới 2 điểm cho trước là ngắn nhất.

Nhận xét :

- Sử dụng tổng network distance làm bán kính tìm kiếm với mục đích là mở rộng tập candidat trong giới hạn chắc chắn tìm được điểm gần nhất theo đường đi, và điều này đúng trên lý thuyết.
- Kết quả trong cách 2 thực sự tốt hơn cách 1, nhưng thời gian và khối lượng tính toán lớn hơn so với cách 1.

10. Tìm Nhà hàng(NH) gần với cả bệnh viện(BV) và ATM nhất

Bài toán thực tế: Sử dụng tương tự như trong trường hợp một sinh viên tìm trọ và mong muốn trọ của mình vừa gần cả chợ và cả bệnh viện.

Cách giải quyết: Để có thể thu hẹp lại phạm vi tính toán, chúng ta chỉ xét trong khu vực quận Hoàn Kiếm với số BV là 2.

- Input: (tên khu vực cụ thể)
- Output: tọa độ của NH, BV và ATM thỏa mãn yêu cầu bài toán, đường đi từ NH đến BV và ATM với chi phí là nhỏ nhất

Các bước thực hiện:

B1. Do BV có mật độ là nhỏ nhất nên từ BV chúng ta đi tìm NH gần nhất theo Euclid.

(Tìm tất cả các BV trong khu vực chỉ định, sau đó từ mỗi BV tìm NH gần nhất với khoảng cách Euclid).

```
    db.execute(f"""
    select long, lat from geometries
    where amenity ='hospital'
    AND ST_Within(way, (SELECT way FROM planet_osm_polygon p WHERE osm_id='-9421131'))
""")
    h=db.fetchall() #candidat of Hospital
    for b in h:
        b=b
        db.execute(f"""
    select long, lat,
    ST_Distance(ST_Transform(ST_SetSRID(ST_MakePoint(105.8548261, 21.024432000404786), 4326), 3857),
                            ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857))
    FROM geometries
                where amenity='restaurant'
                AND ST_Within(way, (SELECT way FROM planet_osm_polygon p WHERE  osm_id='-9421131'))
                order by ST_Distance(ST_Transform(ST_SetSRID(ST_MakePoint(105.8548261, 21.024432000404786), 4326), 3857),
                            ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857))
                limit 1

    """)

        r= db.fetchall()  #candidat of Restaurant
```

B2. Từ NH này tìm ATM gần nhất theo Euclid.

Với mỗi NH tìm được ở B1, tìm ATM gần nhất với mỗi NH này theo khoảng cách Euclid.

```
#r[0][0], r[0][1]: long lat of each Restaurant point
    db.execute(f"""select long, lat,
ST_Distance(ST_Transform(ST_SetSRID(ST_MakePoint({r[0][0]}, {r[0][1]}), 4326), 3857),
                            ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857))
FROM geometries
            where amenity='atm'
            AND ST_Within(way, (SELECT way FROM planet_osm_polygon p WHERE  osm_id='-9421131'))
            order by ST_Distance(ST_Transform(ST_SetSRID(ST_MakePoint({r[0][0]}, {r[0][1]}), 4326), 3857),
                            ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857))

            limit 1""")
    a= db.fetchall()   #candidat of ATM
```

B3. Với 2 khoảng cách ở trên, chọn khoảng cách lớn hơn làm bán kính tìm kiếm.

```
if r[0][2]> a[0][2]:  # r[0][2]: distance from Restaurant to Hospital
    dis= r[0][2]
else:
    dis= a[0][2]   ## a[0][2]: distance from Restaurant to ATM
```

B4. Tìm tập các BV và ATM trong phạm vi này theo NH.

```
    db.execute(f"""
select long, lat
FROM geometries
            where amenity='hospital'
            AND ST_Within(way, (SELECT way FROM planet_osm_polygon p WHERE  osm_id='-9421131'))
            and ST_Distance(ST_Transform(ST_SetSRID(ST_MakePoint({r[0][0]}, {r[0][1]}), 4326), 3857),
                            ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857))<={dis}
""")
    h_nn= db.fetchall() #new candidat of Hospital

    db.execute(f"""
select long, lat
FROM geometries
            where amenity='atm'
            AND ST_Within(way, (SELECT way FROM planet_osm_polygon p WHERE  osm_id='-9421131'))
            and ST_Distance(ST_Transform(ST_SetSRID(ST_MakePoint({r[0][0]}, {r[0][1]}), 4326), 3857),
                            ST_Transform(ST_SetSRID(ST_MakePoint(long, lat), 4326), 3857))<={dis}
""")
    a_nn= db.fetchall() #new candidat of ATM
```

B5. Tính tổng và chọn network routing nhỏ nhất từ NH đến BV và ATM. Sau đó, lặp lại cho các BV trong khu vực đã chọn và chọn tổng network routing nhỏ nhất.

Ở đây ta chỉ cần tính tổng cost và network routing cho từng NH đến BV và ATM, sắp xếp chúng và sau đó lấy bản ghi có tổng cost là nhỏ nhất.

```
for i, z in itertools.product(h_nn,a_nn):
    db.execute(f"""
    WITH start1 AS (
  SELECT topo.source
  FROM osm_2po_4pgr as topo
  ORDER BY topo.geom_way <-> ST_SetSRID(
ST_GeomFromText('POINT({r[0][0]} {r[0][1]})'),4326)
  LIMIT 1)

,start2 as (
SELECT topo.source
  FROM osm_2po_4pgr as topo
  ORDER BY topo.geom_way <-> ST_SetSRID(
    ST_GeomFromText('POINT({z[0]} {z[1]})'),4326)
  LIMIT 1)
,destination AS (
  SELECT topo.source
  FROM osm_2po_4pgr as topo
  ORDER BY topo.geom_way <-> ST_SetSRID(
ST_GeomFromText('POINT({i[0]} {i[1]})'),4326)
  LIMIT 1
)
SELECT ST_Union(geom_way), sum(di.cost) as realcost, {i}, {z},{r[0]}
            FROM pgr_dijkstra('
                    SELECT id,
                           source,
                           target,
                            ST_Length(ST_Transform(geom_way, 3857)) AS cost
                    FROM osm_2po_4pgr',
                            array(SELECT source FROM start1 ),
                            array(SELECT source FROM destination UNION SELECT source FROM start2),
                            directed := false) AS di
            JOIN   osm_2po_4pgr AS pt
            ON   di.edge = pt.id
              """)
    c= db.fetchall() #candidate of 3 point and cost
  Some point has not path and cost, it means they are not reach to others
  if c[0][0] is not None:
    candidat.append(c)
end= sorted(candidat, key=lambda x: x[0][1])[0] #result for the smallest cost
```

Kết quả:

```
bePoint() #path, cost, hospital, atm, restaurant
```

```
[('0102000020E6100000040000004BBAB68BD765A40126E8D637F0635406A6CAF05BD765A40BBAAFDE77A063540CA07F30DBB765A40A314BE0864063540BE
45CC91BA765A400BF20E4B5E063540',
  63.24046506662777,
  '(105.8548261,21.024432000404786)',
  '(105.8553356,21.024990100404796)',
  '(105.8549087,21.024559900404793,17.810343901958767)')]
```

⇨ Được tập điểm NH, BV và ATM mà BV và ATM có đường đi đến NH là nhỏ nhất. Do chúng ta đang dùng khoảng cách Euclid làm bán kính tìm kiếm nên kết quả chỉ đảm bảo là tốt nhất trong tập candidat tìm được chứ không thực sự là kết quả tốt nhất cho bài toán.

Nhận xét : Với cách giải quyết bài toán như trên, chúng ta không thực sự tìm được kết quả tốt nhất nhưng với này đảm bảo bài toán của chúng ta luôn luôn tìm được kết quả.

C, Nhận xét

- Việc sử dụng khoảng cách Euclide làm bán kính tìm kiếm cho tập candidat không thực sự là tốt nhất nhưng giúp giảm thời gian tính toán.
- Việc sử dụng khoảng cách network routing làm bán kính tìm kiếm cho tập candidat thực sự tốt (không tính đến sự sai khác trong việc sử dụng Dijsktra trong Pgrouting) nhưng mất thời gian và khối lượng tính toán lớn.

III. Thêm index và so sánh thời gian truy vấn

1. Các bài toán sử dụng khoảng cách Euclide: ST_Distance

Tạo index GIST trên trường way (geometry) của bảng geometries

```
CREATE INDEX idx_vn_label ON geometries USING gist (way);
```

a. Tìm Nhà hàng gần nhất với 1 điểm xác định trước

Index

	osm_id bigint	access text	addr:housename text	addr:housenu text
1	6658017286	[null]	[null]	66
2	3684899299	[null]	[null]	23
3	5662890235	[null]	[null]	[null]

Total rows: 5 of 5 Query complete 00:00:00.150

0.150s

No index

	osm_id bigint	access text	addr:housename text	addr:h text
1	6658017286	[null]	[null]	66
2	9931534174	[null]	[null]	[null]
3	3684899299	[null]	[null]	23

Total rows: 5 of 5 Query complete 00:00:00.582

0.582s

b. Tìm Nhà hàng gần nhất với 2 điểm xác định trước

Index

	osm_id bigint	access text	addr:housename text
1	9931534174	[null]	[null]

Total rows: 1 of 1 Query complete 00:00:00.156

0.156s

No index

	osm_id bigint	access text	addr:housename text
1	9931534174	[null]	[null]

Total rows: 1 of 1 Query complete 00:00:00.201

0.201s

c. Tính mật độ nhà hàng trong một khu vực (quận Hai Bà Trưng)

Index

	sonhtrenkm2 double precision
1	3.747511661560899

Total rows: 1 of 1 Query complete 00:00:00.595

0.595s

No index

	sonhtrenkm2 double precision
1	4.939901735693913

Total rows: 1 of 1 Query complete 00:00:00.989

0.989s

d. Khoảng cách trung bình giữa 2 nhà hàng khác nhau trong khu vực

Index

	avg double precision
1	1268.0218714382922

Total rows: 1 of 1 Query complete 00:00:12.148

No index

	avg double precision
1	1218.9728012937967

Total rows: 1 of 1 Query complete 00:00:17.289

12.148s	**17.289s**

2. Các bài toán sử dụng khoảng cách network routing : pgrouting

Tạo index GIST trên trường geom_way (geometry) của bảng osm_2po_4pgr

```
CREATE INDEX new_index ON osm_2po_4pgr USING gist(geom_way);
```

a. Tìm đường đi và chi phí giữa 2 điểm

Index

	sum double precision	st_union geometry
1	7394.064779331535	0105000020E610000066000000010

Total rows: 1 of 1 | Query complete 00:00:15.285

15.285s

No index

	sum double precision	st_union geometry
1	7394.064779331532	0105000020E6100000660000000102000000C

Total rows: 1 of 1 | Query complete 00:00:22.666

22.666s

b. Tìm đường đi network routing theo thuật toán dijkstra tới KNN được xác định bởi Euclide

Index

	long double precision	lat double precision	route text
1	105.80478542334376	21.03010025642198	MULTILINESTRING((10
2	105.8051000240784	21.030102359922665	MULTILINESTRING((10

Total rows: 5 of 5 | Query complete 00:01:21.498

1p 21.498s

No index

	long double precision	lat double precision	route text
1	105.80666819999999	21.02911480040487	MULTILINESTRIN
2	105.806208	21.02851110040486	MULTILINESTRIN
3	105.80698279999999	21.029116900404865	MULTILINESTRIN

Total rows: 5 of 5 | Query complete 00:01:27.779

1p 27.779s

c. Tìm điểm có network routing ngắn nhất với điểm cho trước (dựa trên bài toán 6).

Index

	long double precision	lat double precision	lost double precision
1	105.80478542334376	21.03010025642198	554.6924656556598

Total rows: 1 of 1 | Query complete 00:01:23.798

1p 23.798s

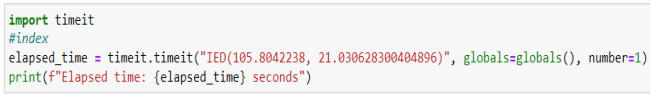
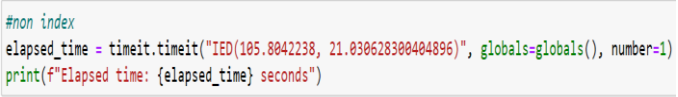
No index

	long double precision	lat double precision	lost double precision	route_geometry geometry
1	105.8068682	21.033045300404943	471.871298337968	0105000020E6100000(

Total rows: 1 of 1 | Query complete 00:01:36.063

1p 36.063s

d. Tìm điểm có đường đi network routing thực sự ngắn nhất theo IER

Index	No index
 **303.026s**	 **404.2516s**

e. Tìm điểm Nhà hàng gần nhất với nhiều điểm cho trước

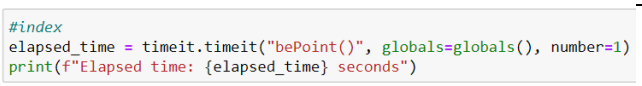
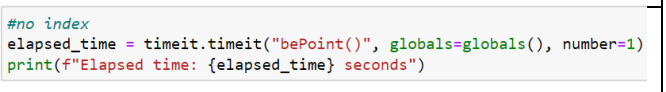
Cách 1:

Index	No index
 **253.188s**	 **406.25s**

Cách 2:

Index	No index
 **765.855s**	 **1275.387s**

f, Tìm nhà hàng (NH) gần với cả bệnh viện (BV) và ATM nhất

Index	No index
 **49.919s**	 **72.125s**

Nhận xét: Khi sử dụng index cho các trường dữ liệu không gian geometry, thời gian truy vấn giảm đi nhiều.

IV, Kết luận

Chú ý: Khoảng cách network distance và đường đi giữa các điểm được tìm thấy trong các function của chúng ta dựa trên giải thuật Dijkstra trong Pgrouting, với một số bước tiền tìm kiếm như sau:

- Tìm điểm (node) thuộc đường gần nhất với điểm bắt đầu
- Tìm điểm (node) thuộc đường gần nhất với điểm kết thúc
- Sử dụng Dijkstra tìm đường đi ngắn nhất giữa 2 node trên

⇨ Do vậy, đường đi và khoảng cách được tìm thấy không thực sự là đường đi và khoảng cách thực tế giữa 2 điểm ban đầu.

Qua việc so sánh giữa truy vấn sử dụng khoảng cách Euclide với việc sử dụng khoảng cách network routing (đường đi) giữa các điểm làm bán kính tìm kiếm, chúng ta có thể thấy tính toán trên khoảng cách Euclide giúp truy vấn nhanh hơn và có kết quả khá tốt, mặc dù trong lý thuyết thì không phải kết quả đúng. Với cách còn lại, chúng ta mất nhiều thời gian hơn, không gian tính toán lớn hơn, và kết quả trả về không quá chênh lệch so với sử dụng khoảng cách Euclide, nhưng nó đúng trên lý thuyết.

Tóm lại, qua đây chúng ta đã học được cách tổ chức, lưu trữ và thao tác với dữ liệu không gian, đã có một cái nhìn tổng quan về truy vấn dữ liệu không gian và đã tìm hiểu về một số bài toán về tìm điểm và tìm đường.