12:30 pm - definició d'objectius + anàlisi dels recursos.

Objectiu: anàlisi de xarxes terrestres multicapa per vehícle autònom. Cal buscar la ruta de menor cost possible.

Xarxa multicapa → part interessant del repte. S'ha de conviar de capa a capa tenint en compte costos.

punts a tenir en compte
1. Transicions entre capes (carreteres urbanes, autopistes, camins rurals).
2. Restriccions dinàmiques (costos variables en xarxes i punts d'accés limitats)
3. Complexitat computacional

Estructura de les dades

imatge de docker: PostgreSQL
- capes de dades
- punts de transició de les capes

+

visor de dades: visualització de capes
- permet programació en python
- creació d'extensions.
(QGIS)

dades necessàries
1. Punts aleatoris per a cada equip → un punt d'origen i destí
2. Costos de xarxes preestablertes. → xarxes tenen cost diferent, pel que el camí més curt no és l'òptim

Criteris d'avaluació

1. Eficiència (temps + recursos)
2. Qualitat solució (camí de menor cost + capacitat d'adaptar-se a canvis).
3. Creativitat i originalitat (utilitzar algorismes d'optimització, IA o metaheurístiques)
4. Documentació y claritat.

14:30 - Què recursos / fitxers se'ns dona?

* Dockerfile → conté instruccions per creació d'imatge docker. Definir entorn i dependències necessàries.

* Project.qgz → conté dades geoespacials i configuracions associades per interacció i visualització.
  - QGIS = software de sistemes d'info geogràfica.

* backup_eps.sql → recolzament de base de dades en SQL.

* docker-compose.yml → se usa para definir y ejecutar apps multicontenidor amb docker. complementan a dockerfile. (vull que em generi un contenidor amb postgres de non Dockerfile)

Problema a solucionar → no sabem fer servir docker : solució = anem a aprendre. Coses importants:

- non contenidos = repte_inicio
- port = "5432:5432"
- per veure → docker-compose ~~app~~ up
- per correr-ho, cal tenir docker desktop obert
- Avantatge: podriem exportar la imatge.

] → un cop fas això farà un 'build'.

Docker es com crear una VM o un "miniordinador" per poder accedir.
Pot ser, amb DBeaver podrem accedir a la base de dades

Pregunta: de què ens serveix fer servir docker? → per connectarse a base de dades.

Problema → como abrir fichero .qgs: Solució?
1. Instal·lar qgis
2. Abrir qgis
3. Proyecto → Abrir → Proyecto.qgs

- user, pswd? → literal user i password.

ens servirà per visualitzar la base de dades

S'ens obre un mapa de xarxes amb punts.
important! hem hagut de canviar on hi havia un espai ficar un =

?& → per passar d'una paret a un altre, com ho fas? fent servir els punts?
? 0 → com agafes dades del qgis per fer-les servir amb python.? ↓ no cal però seria

Hem guardat les taules a csvs per facilitar l'accés.

16:30 - Hem aconseguit visualitzar les dades. Ara què? ~~?~~ pesos? costos? COST PER MET

Project.qgs
- red1_puntos / red2_puntos / red3_puntos → Vèrtex ← això són taules
- red1, red2, red3 → Línies
- inicio, final → observació: no coincideixen amb les carreteres de vegades.

Problema → sql són 64k línies → visualitzar al dbeaver es molt complicat; s'ha creat perque qgis ho veu.

Solució → simplement hem de canviar el nom 'postgres' per nom 'database', per obrir-ho en DBeaver.

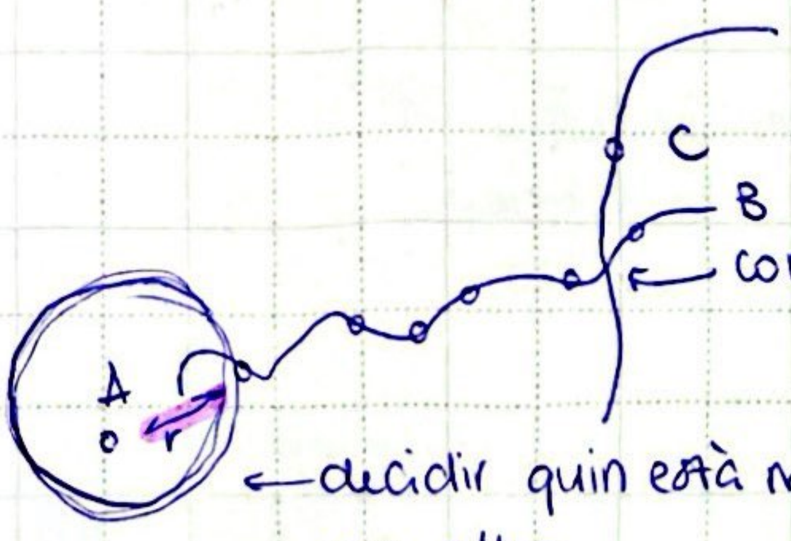
~~A5:41 mirem les dades.~~

punts inici → 5
punts final → 6

x3 → 10
x2 → pes = 5
x1 → pes = 3

MULTILINESTRING? → coordenades → si fas doble click, et surt la ruta. (x2 sobre un 'caml')

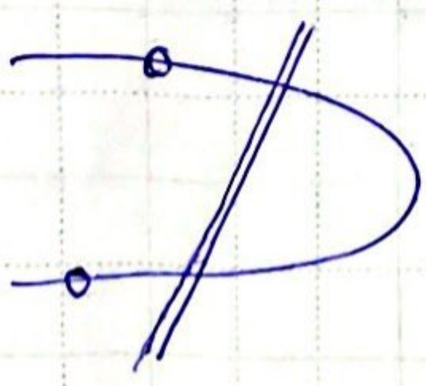
pesos = [NULL] en molts (tots) → això és un error: ara no solucionen.



com salles de carretera a carretera?

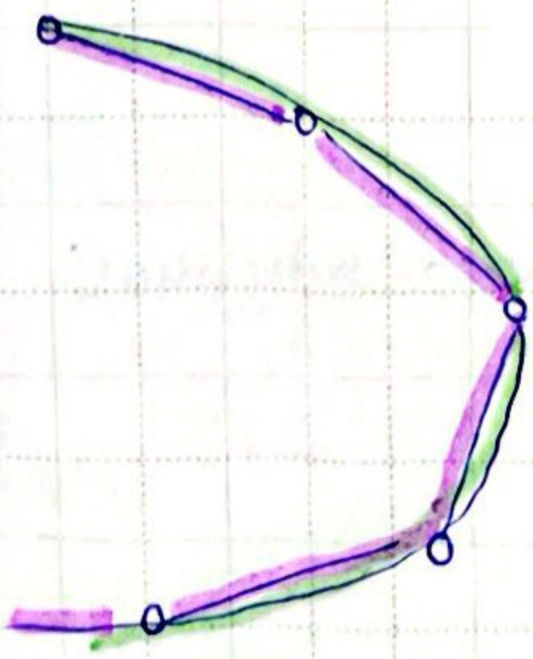
← decidir quin està més aprop amb un radi: a partir d'aquí arem a una xarxa i una altra.

Entre punt i punt NO hi ha una línia recta → per tant distància no es línia recta



Dubte! → no sabem com calcular els costos trobats.

Es legal "passar per on vulgui" o "fer canvi de carretera a carretera en qualsevol intersecció.



← com es mira? amb — o ⌒? Tots els girs estan documentats

16:20 - Com ho connectem amb python?

- Greedy? → NO, es quedaria amb millor
- Dynamic Programming → problema: massa memoria calcula
- Beamsearch → probabilitats serien els costos. → a cada pas escullis millors
- Topk-sampling
- Simulated annealing.
- Hill Climbing.
- A*

→ calcular distància de arestes

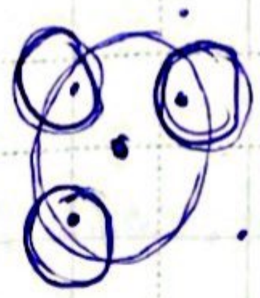
```
SELECT SUM (ST_Length (geom)) FROM eps.red1;

SELECT SUM (ST_Length (geom)) FROM eps.red2;

SELECT id, peso FROM eps.red2;

SELECT id, peso FROM eps.red3;
```

Beam search amb k = 3 ⟶ manera òptima de calcular k?



de cada punt, eroluim els 3 amb radi menor.

PgRouting te funcions com
- pgr-dijkstra
- pgr-astar ⟶ calcular directamente en la base de dades.
- pgr-withPoints
- pgr-ksp

multistring = minilínies

transformar multistring en una sola línia.

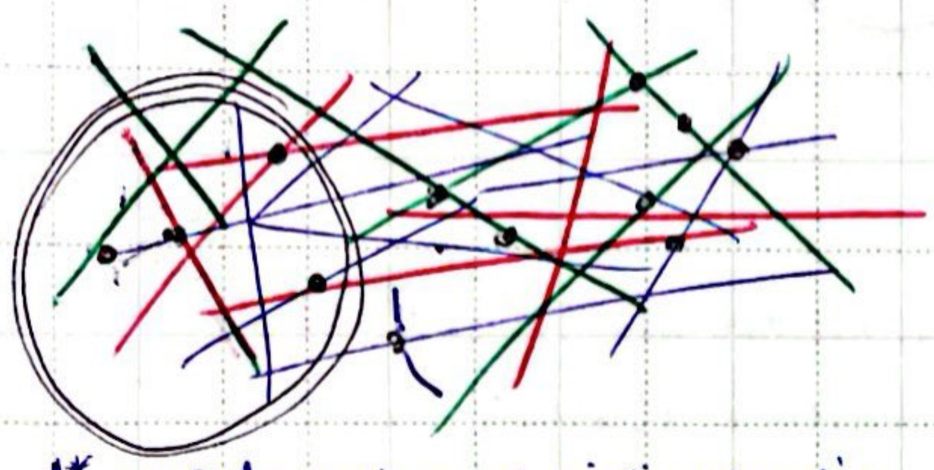
qgis → moltes formules q treballen amb dades espacials: et troben relacions entre punts...

- si hi ha interseccions entre línies.

web: postgis st: Topological Relationships.

18 - Descobrint PgRouting + pensant algorismes. → https://docs.pgrouting.org/latest/es/pgr-a

problema: puc fer 3 A* però es multicapa
puc fer 1 A* però la funció de cost es igual. 100% convé fer servir pgRouting.



A* en cada punt per decidir on vaig.

* pgr_aStarCost

pgr_aStarCost(SQL de arestas, salida, destino, [opciones])
opcionales [directed, heuristic, factor, epsilon]

} mejor no pq calcula coste y no ruta. no hace falta

* pgr_aStar → ruta + corta usando A*

pgr_aStar(SQL de arestas, salida, destino, k, [opciones])
opcionales [directed, heuristic, factor, epsilon])

pgadmin extension postgres per visualitzar.

- ST_intersects
- ST_3DClosestPoint →
- ST_3DShortestLine