ENUNCIAT PROJECTE LP curs 2023-24

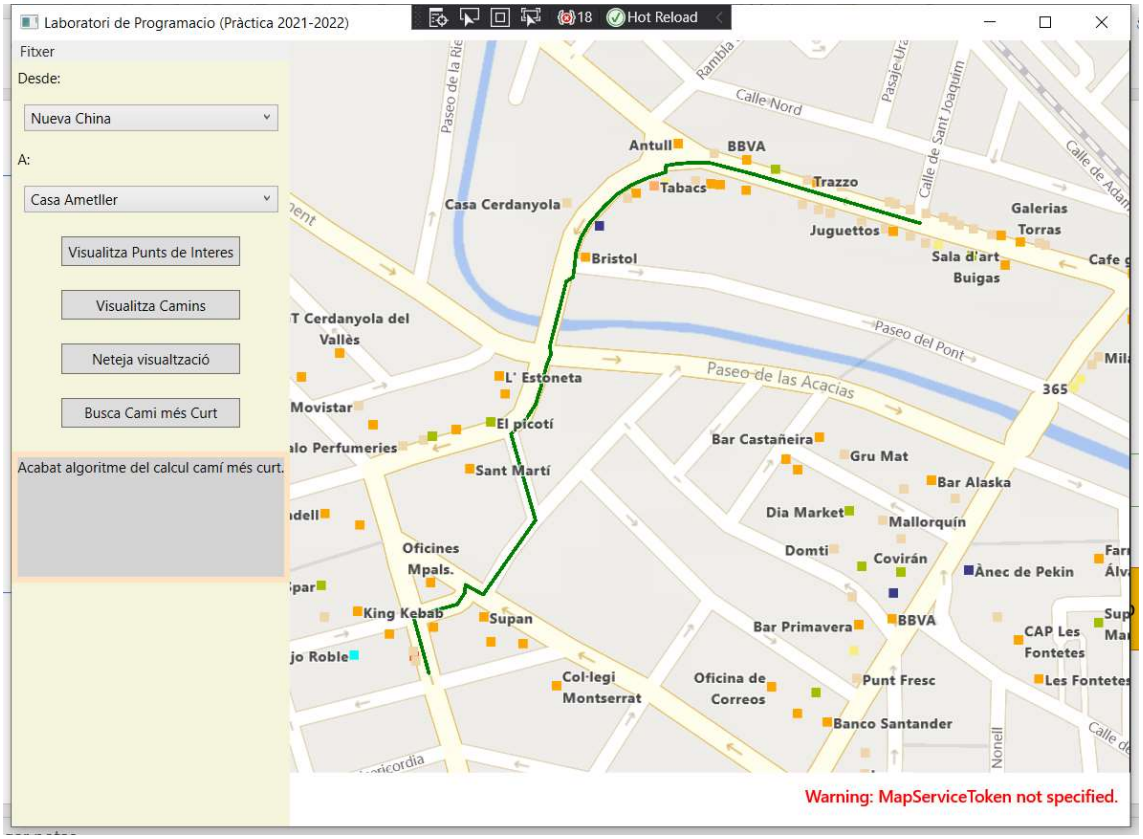
1. PROJECTE PART 2

La segona part de la pràctica haureu de trobar el camí més curt entre dos punts d'interés. Per aconseguir-ho, necessitareu definir noves estructures de dades que permetin tenir l'informació de la posició a l'espai dels nodes, i la seva relació amb la resta de nodes.

Aquestes estructures les heu vist, o veureu, a classe: són els grafs i els arbres.

El vostre resultat, serà quelcom semblant a la següent imatge:

(camí per anar des del Nueva China fins al Casa Ametller)



El procediment per l'ús de l'interfície gràfica per buscar-ne el camí més curt, haurà d'ésser el següent:

1. Obrir el programari "Laboratori de Programació (Pràctica 2023-2024)" que esteu utilitzant pel desenvolupament del projecte.
2. Descarregar un mapa compatible amb el format OSM, per exemple el Cerdanyola.osm.xml que us donem de la carpeta mapesOSM.
3. Des de l'interífice gràfica, fer Fitxer -> Obrir, i seleccionar el mapa descarregat en el anterior pas.
4. Això us carregarà el mapa, tal com heu implementat en el parsejaXmlElements, on guardeu la vostra informació dins de MapaSolucio.
5. Representareu els punts d'interès amb "Visualitza Punts de Interes"
6. Representareu tots els camins amb "Visualitza Camins".
7. Seleccionareu, del menú "Desde:", el punt d'interès del que voleu partir.
8. Seleccionareu, del menú "A:", el punt d'interès que voleu anar.
9. Premereu "Busca Camí més curt". Aquest cridarà el vostre mètode que heu implementat en la DLL, i transferirà les dades a l'interfíce gràfica, que se us detallarà més endavant.
10. Per poder representar el camí, el que haureu de fer primer és netejar la visualització amb "Neteja visualització".
11. Torneu a representar els punts d'interès amb "Visualitza Punts de Interes".
12. Finalment, per veure el camí, fareu "Visualitza Camins", que, representarà el punt camí més curt d'anar desde el node d'origen fins al node destí.

Tal com hem fet en la primera part de la pràctica, se us proposarà un seguit de Tasques que haureu d'implementar per assòlir els objectius de la pràctica. Us recomanem que seguiu l'ordre de manera seqüèncial. Se us facilitarà també un banc de proves que, aquest cop, coincidirà directament amb el que serà l'avaluador final.

1.2 Construcció del Graf

La primera tasca que heu de realitzar és la construcció d'un graf a on els nodes seràn els nodes de camí que heu llegit i desat a Camisolucio, i les arestes seran les connexions entre aquests nodes. Podria semblar que els punts d'interès també estarien lligats amb aquests nodes de camí, però tal com us vam informar en la primera part de la pràctica, tant els punts d'interès com els nodes de camí, **estàn desacoplats**.

Què significa? Doncs que el graf a construir **no** tindrà els nodes de punts d'interès, sino, tots els nodes de camí, de tots els camins, que heu carregat per un mapa.

Donat que heu implementat per la primera part un CamiSolucio per cada camí, on teniu les coordenades d'aquest camí per getCamiCoords(), i que a MapaSolucio teniu el conjunt de camins, podeu construir el graf mitjançant aquesta informació.

Ja hem determinat quins seràn els nodes del graf, però, ara mateix ens podem preguntar, com en realitzem les connexions.

El graf que implementareu serà un graf ponderat simètric (no dirigit) on el cost per anar d'un node a un altre ve determinat per la ***distància Haversine*** d'aquests dos. Aquesta distància es defineix com la distància angular donat dos punts representats per la seva latitud i la longitud.

A la **classe Util**, disposeu de dos mètodes per calcular-ne aquesta distància:

```
static double DistanciaHaversine(double lat1, double lon1, double lat2, double lon2);
static double DistanciaHaversine(Coordinate px1, Coordinate px2);
```

Aquest mètode és un mètode estàtic, on un exemple del seu d'ús pel mètode sobrecarregat que utilitza l'estructura Coordinate:

```
Coordinate origen = { 41.4904503, 2.1416965 };
Coordinate desti = { 41.4904870, 2.1406742 };
double distancia = Util::DistanciaHaversine(origen, desti);
```

Determinats quins seràn els nodes del graf, així com el cost entre nodes, podem determinar la primera tasca.

Tasca 1: Construir el Graf on els nodes que el formen són les coordenades dels camins que carregueu a través de **MapaSolucio**.

Consells

1. Sou lliures d'implementar el graf que realment necessiteu, però recomanem que implementeu els següents mètodes de la **classe Graf: afegirNode i afegirAresta**.

2. Podeu utilitzar les estructures de dades que voleu per representar el graf (matriu d'adjacència o matriu amb llistes).

3. Recordeu que **un punt pot estar en més d'un camí**. Al construir el graf, assegureu-vos que el node-cami del mapa es **correspon a només un node del graf.**

Avisos

1. Aquesta tasca s'avaluarà amb la vostra implementació de la **Tasca 4.**

1.3 Quin és el camí més proper al punt d'interés?

Recordem que el vostre objectiu per aquesta segona part és buscar el camí més curt entre dos punts d'interés.

Tal com hem dit en el punt anterior, el Graf el construirem a través de nodes de camí, i a més, cal tenir en compte que l'informació que ens donen de OpenStreetMaps, no ens diu a quin carrer pertany el punt d'interés.

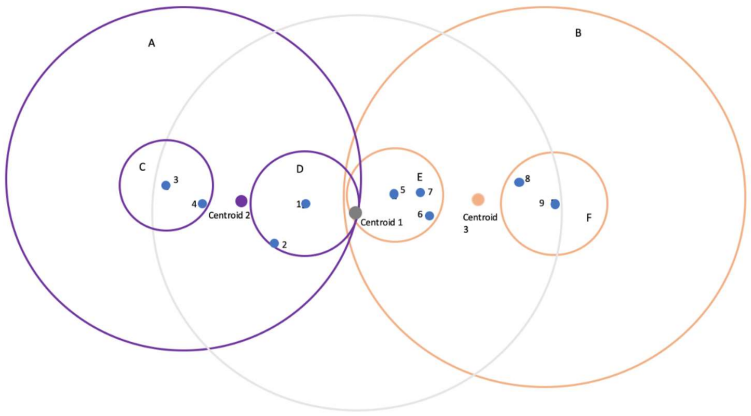
La pregunta que ens podem fer és, com aconseguim saber quin és el node-camí més proper al punt d'interés.

Per poder respondre això, hem de presentar una nova estructura de dades, anomenada BallTree, que és un cas concret d'arbre binari, així com un algorisme de cerca que s'aplicarà sobre aquesta estructura.

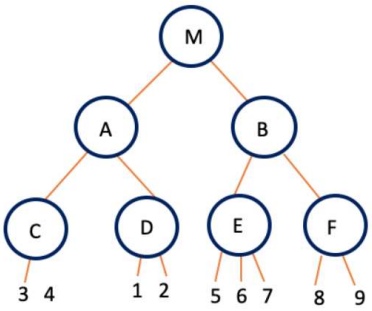
1.3.1 BallTree

El BallTree és un arbre binari que ens permet separar l'espai per distàncies, on aquestes distàncies són "radis" de cercles que inclouen els punts que el formen. A mesura que anem dividint l'arbre, podem imaginar-nos que els radis són més petits, i que els punts que el formen per cada cercle cada vegada es van disminuint, de manera que a les fulles de l'arbre hi tindreu un sol node, que és el node que finalment trobarem com el més proper a un punt d'interès.

Seguidament facilitem dos captures, extretes de [Tree algorithms explained: Ball Tree Algorithm vs. KD Tree vs. Brute Force | by Hucker Marius | Towards Data Science] on ajuda entendre la seva visualització:

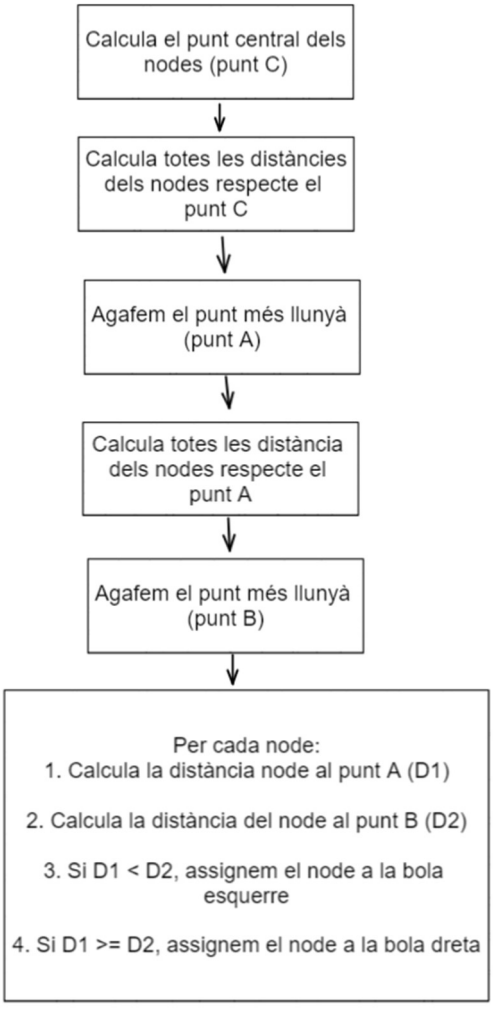


L'arbre finalment representat, seria el següent, on ens fixem que els nodes fulles són finalment els nodes camí, i que l'arbre ens separa l'espai per distàncies de radis (els nodes que estàn a la dreta, estàn més allunyats del seu centroide, que els que estàn a l'esquerre).



L'ús d'aquesta estructura de dades és fonamental per poder aplicar l'algorisme descrit seguidament, i ens serveix per respondre a la pregunta "quin és el node-camí més proper al punt d'interés", on, ens servirà per referenciar-lo al graf, i llavors aplicar-hi els algorismes pertinents per l'objectiu final (búsqueda del camí més curt).

Per poder construir el BallTree, heu de seguir el següent diagrama de flux, i el podeu implementar de manera recursiva. Recordeu que les dades que formen el BallTree són els vostres nodes de camí, i que és una estructura de dades diferent a la que heu fet al graf.



Aquesta sèrie de passos s'aplicarà per cada de bola del arbre que s'estigui computant, i la condició d'aturada serà quan la bola només contingui un node.

Per cada bola, li haurem de definir el seu pivot (que serà el punt central C), així com el seu radi (distància respecte el punt A), a part de tots els seus punts que el formen.

Això us determina la segona tasca que heu d'implementar per la segona part del projecte, que és construir el BallTree amb els nodes cami.

Nota: Si ho veieu oportú, podeu visitar la part de "k-d Construction" de l'entrada de la Wikipedia. Us facilitem l'enllaç aquí: https://en.wikipedia.org/wiki/Ball_tree.

Tasca 2: Construir el BallTree amb els nodes camins, i que disposi dels següents mètodes.

Consells

1. Heu de tenir en compte que el BallTree no pot tenir cap Ball amb dues coordenades iguals, d'altra manera no podreu acabar mai la condició recursiva. Heu de pensar alguna manera per assegurar-vos que no afegiu coordenades repetides al BallTree.

2. Utilitzeu les funcions de la classe Util que us proporcioneu i que creieu pertinets per resoldre el diagrama de flux.

Avisos

1. Per l'avaluació d'aquesta Tasca 2, haureu d'implementar en el vostre BallTree els següents mètodes:

- void inOrdre(std::vector<std::list<Coordinate>>& out): Recòrre l'arbre BallTree seguint el recorregut inordre. En retorna la llista de coordenades que conté cadascuna de les boles, del camí realitzat, a través del paràmetre de referència out.

- void postOrdre(std::vector<std::list<Coordinate>>& out): Recòrre l'arbre BallTree seguint el recorregut postordre. En retorna la llista de coordenades que conté cadascuna de les boles, del camí realitzat, a través del paràmetre de referència out.

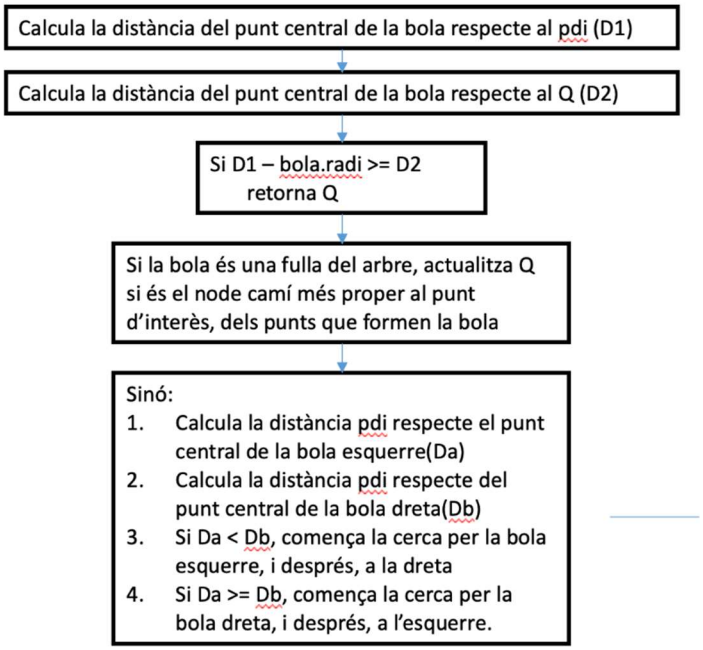
- void preOrdre(std::vector<std::list<Coordinate>>& out): Recòrre l'arbre BallTree seguint el recorregut preordre. En retorna la llista de coordenades que conté cadascuna de les boles, del camí realitzat, a través del paràmetre de referència out.

1.3.2 L'algorisme 1NN (node més proper)

Un cop separat l'espai dels nodes-camí per distàncies radi, podem buscar el node camí que correspont al punt d'interés que volem. Per aconseguir-ho, hem d'implementar l'algorisme del veí més proper (KNN, en anglès, K nearest neighbor, on el nostre cas, serà el primer node més proper) sobre el BallTree.

En aquest algorisme, li facilitarem les coordenades del punt d'interés (pdi), el primer cercle del Ball Tree (bola), i ens ha de retornar la coordenada del node camí com a resultat (Q).

L'algorisme, també de forma recursiva, té el següent diagrama de flux. Tingueu present quan parar la recursivitat.



Nota: Si ho veieu oportú, podeu visitar la part de "Nearest-neighbor search" de l'entrada de la Wikipedia. Tingueu present que l'algorisme de la Wikipedia busca els k-primers veins i utilitza una cua de prioritat. En el nostre cas, com que busquem un sol veí, no ens faria falta aquesta estructura de dades addicional. Us facilitem l'enllaç aquí: https://en.wikipedia.org/wiki/Ball_tree.

Tasca 3: Implementar la cerca del node-camí més proper del punt d'interés, a través de la construcció del BallTree.

1. Per l'avaluació d'aquesta **Tasca 3**, haureu d'implementar en el vostre BallTree els següent mètode:

- Coordinate nodeMesProper(Coordinate pdi, Coordinate& Q, Ball* bola_arrel)

El mètode nodeMesProper haurà de retornar la Coordinate node-camí, emmagatzemat a bola_arrel, més propera al pdi que es proporciona.

1.4 Camí més curt

Finalment per l'últimes tasca, ja tenim l'informació necessària per poder calcular finalment el camí més curt.

Haureu d'afegir a MapaBase la següent declaració:

```
virtual CamiBase* buscaCamiMesCurt(PuntDeInteresBase* desde, PuntDeInteresBase* a) = 0;
```

I haureu d'implementar-lo a MapaSolucio.

Tasca 4: Implementar el camí més curt, donat dos punts d'interés, en el MapaSolucio.

Avisos

1. Aquest és el únic mètode que s'utilitza a la l'interfície gràfica per l'obtenció i representació del camí més curt.

2. Seguiu els passos indicats a la Introducció per fer la crida d'aquest mètode desde l'interfície gràfica.