Laboratori de Programació

Guió de la 2ona Part Projecte 2023-24

Objectius

1. Implementar els algoritmes i estructures de dades necessaris per la cerca del camí més curt entre dos punts d'interès especificats.

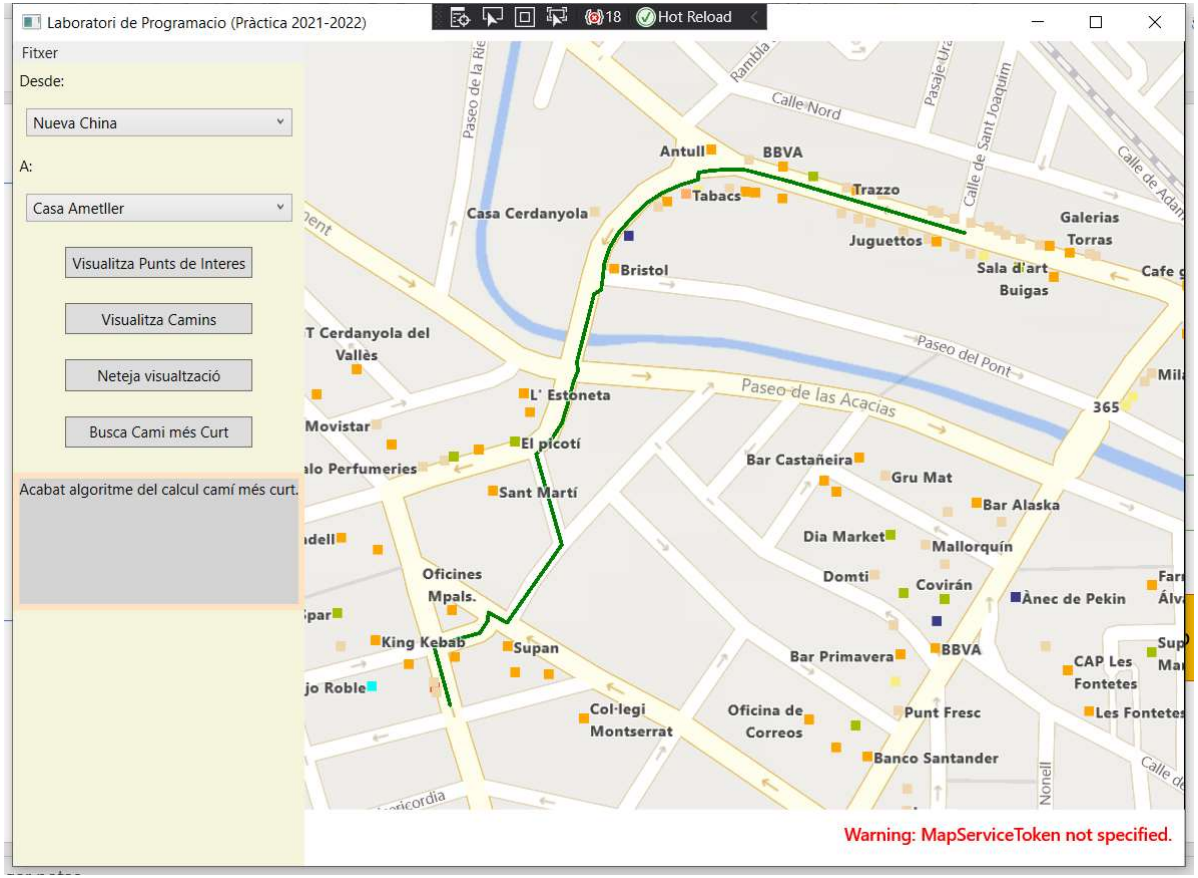
Parts del Projecte

Segona Part

Un cop tinguem tota la informació recopilada voldrem trobar el camí més curt entre dos punts d'interès. Per trobar-ho primer haurem de posar en correspondència el punt d'interès amb el node de camí més proper que tinguem per després trobar el camí més curt. Per fer tot això necessitarem arbres i grafs a més d'altres estructures de dades.

Entorn de desenvolupament: Visual Studio vs 2019, compilació C++14

https://visualstudio.microsoft.com/es/thank-you-downloading-visual-studio/?sku=Community&rel=16



Tasca 1

Afegir els fitxers: **GrafSolucio.h i GrafSolucio.cpp** a on construirem un graf que tindrà com a nodes les coordenades dels camins que carregueu a través de **MapaSolucio** i com arestes les connexions entre aquests nodes donades pels camins i ponderades per la **distància Haversine** entre els nodes.

Consideracions:

- **Important:** Un punt pot estar en més d'un camí, però al mapa es correspon a només un node del graf.
- Recomanem tenir els mètodes de la **classe Graf: afegirNode i afegirAresta**.
- Escolliu implementar com a matriu d'adjacència o a través de vectors amb llista tal com hem vist a classe.
- Penseu a on haureu de tenir aquesta estructura.
- A la **classe Util**, disposeu de dos mètodes per calcular-ne la distància Haversine:
  - `static double DistanciaHaversine(double lat1, double lon1, double lat2, double lon2);`
  - `static double DistanciaHaversine(Coordinate px1, Coordinate px2);`

Tasca 2

Utilitzar els fitxers **BallTree.h i BallTree.cpp** per definir la classe BallTree, la informació d'aquesta classe la obtindreu a partir dels nodes de camí:

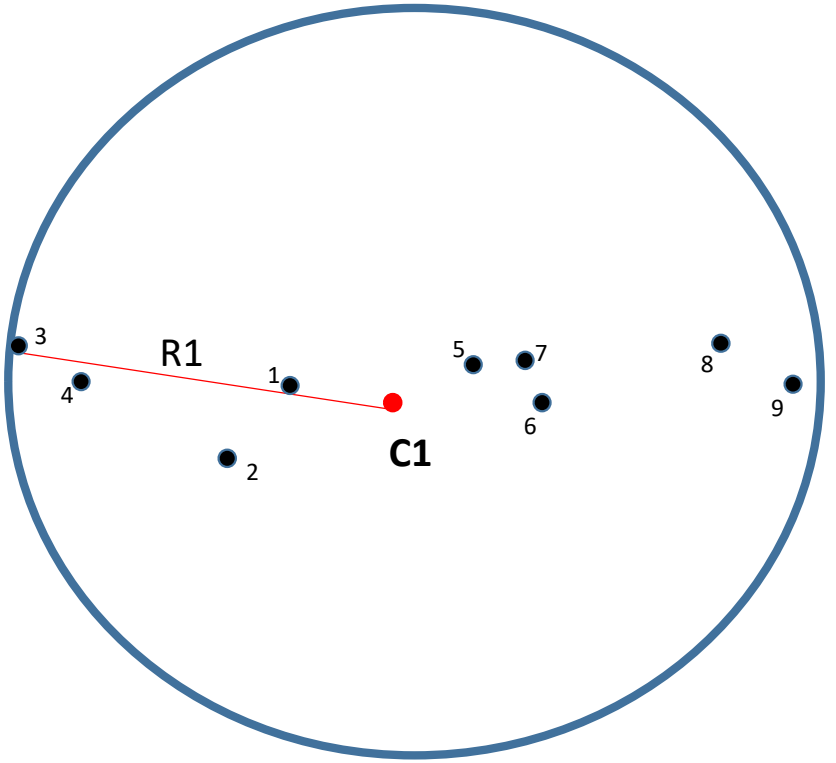
Un **BallTree** és un arbre binari a on cada node és un **BallTree**

Idea general tasques 2 i 3: Com els punts d'interès estan separats dels nodes de camí necessitem trobar quin és el node de camí més proper a un punt d'interès per poder connectar-lo amb la resta de nodes del graf. Per fer això primer construirem un **BallTree** amb tots els nodes de camí i després per cada punt d'interès cercarem al **BallTree** quin és el node de camí més proper a ell.

Tasca 2: Ball Tree

Què és un BallTree:

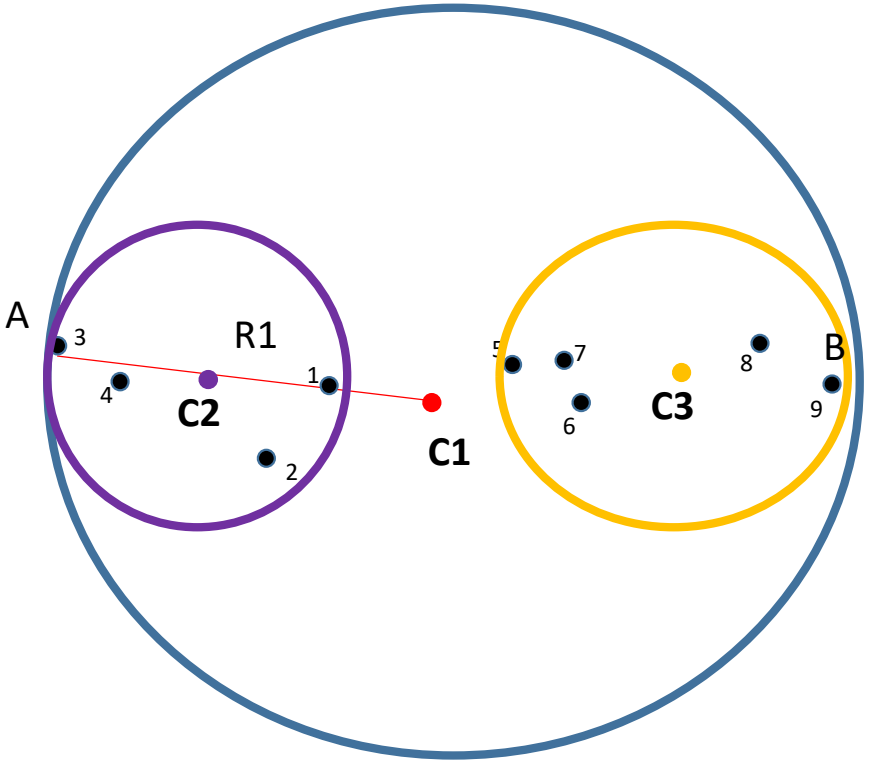
- Arbre binari a on cada node és altre un BallTree. Es tracta doncs d'una estructura de dades recursiva.
- Durant aquesta presentació, utilitzarem Ball com BallTree indistinctivament.
- Un BallTree és un conjunt de punts amb un centre C1 i un radi R1.



Tasca 2: Ball Tree

Què és un BallTree:

- Sobre el punt central C1 del Ball arrel calculem el punt A més llunyà i el punt B més llunyà a l'A.
- Dividim els punts entre els que estan més propers a B (subarbre dret) i els que estan més propers a A (subarbre esquerre).

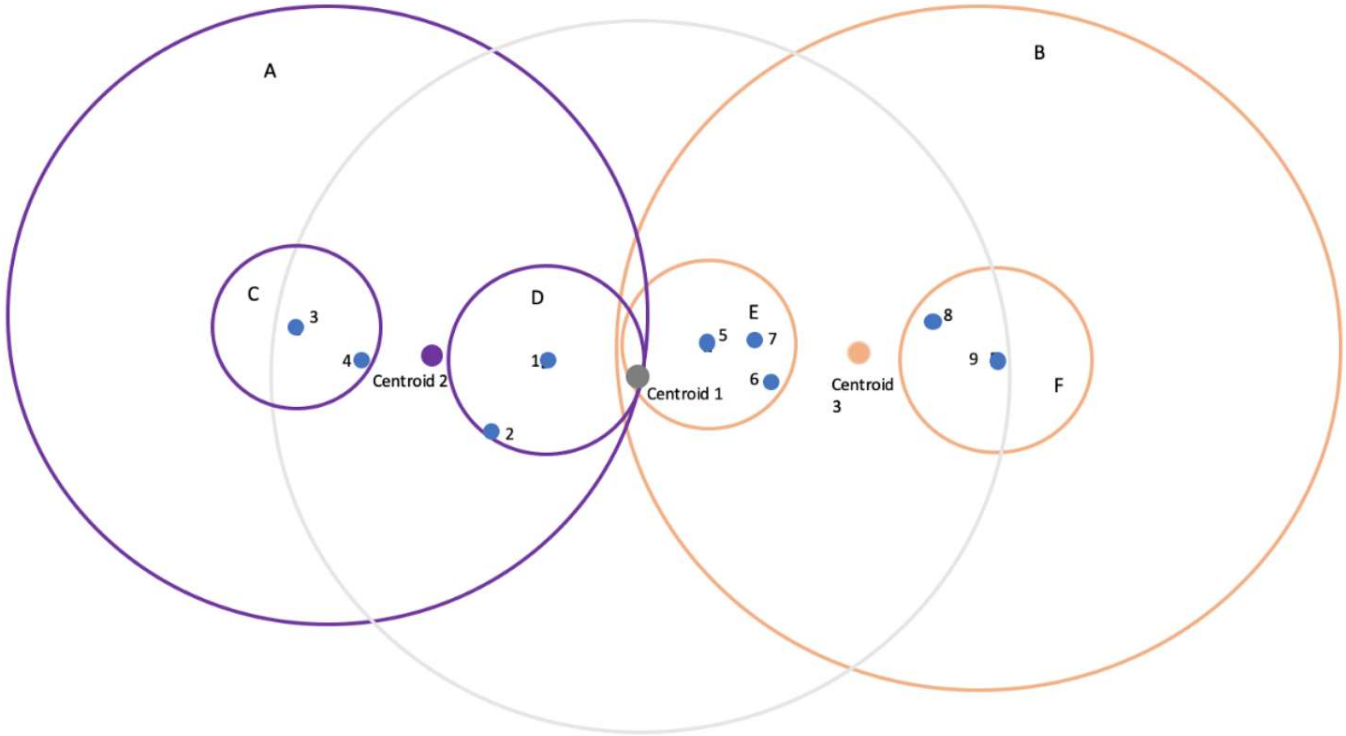


Tasca 2: Ball Tree

Referència: https://towardsdatascience.com/tree-algorithms-explained-ball-tree-algorithm-vs-kd-tree-vs-brute-force-9746debcd940

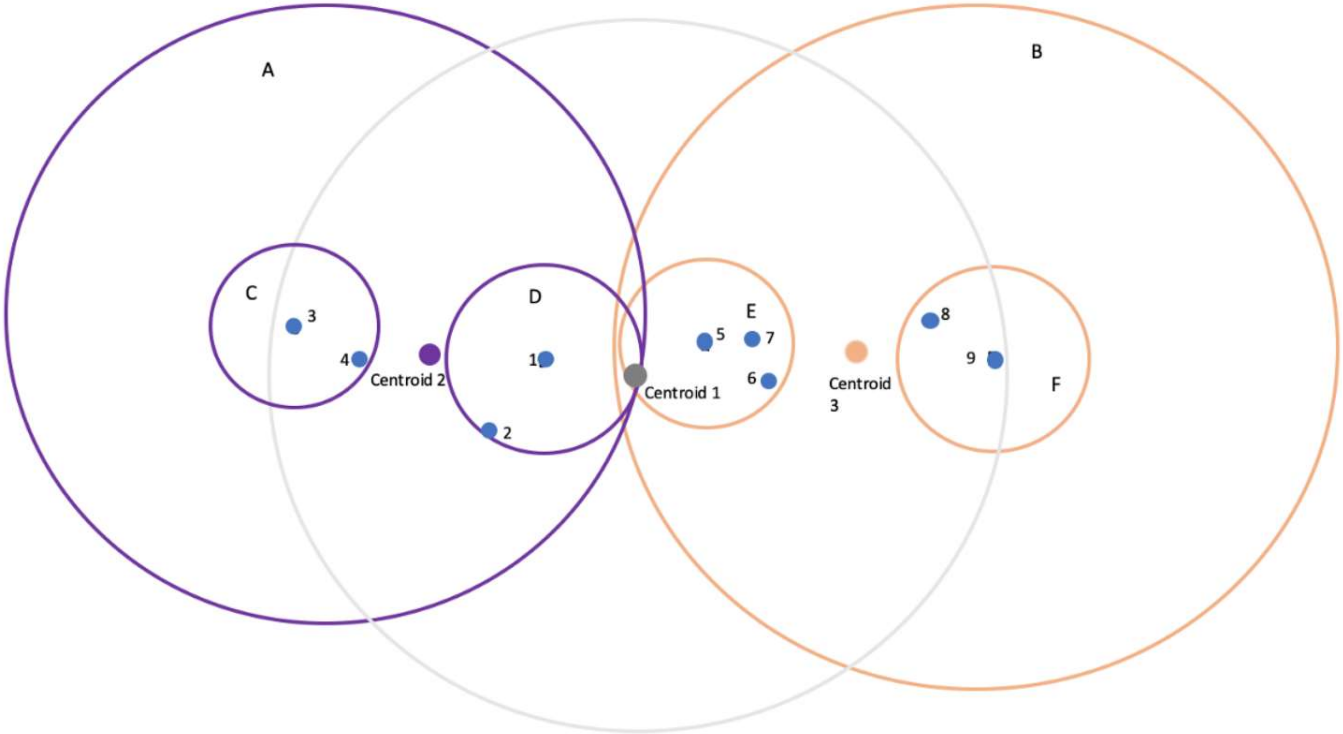
Què és un BallTree:

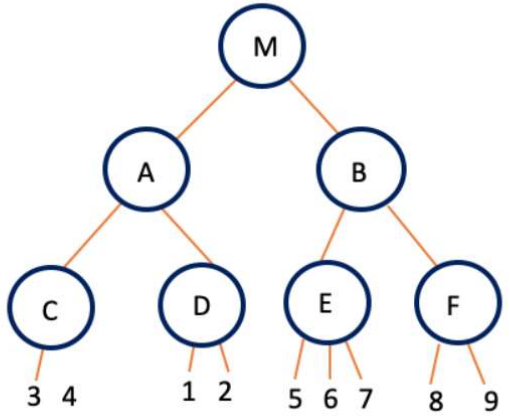
- Fem aquest procés de dividir els Balls fins arribar a tenir un sol punt per fulla.
- Això ens permet dividir l'espai per distàncies a on aquestes distàncies són "radis" de cercles que inclouen els punts que el formen.



Tasca 2: Ball Tree

Referència: https://towardsdatascience.com/tree-algorithms-explained-ball-tree-algorithm-vs-kd-tree-vs-brute-force-9746debcd940





Tasca 2: Ball Tree

BallTree: Per cada bola:

- El seu pivot (que serà el punt central C)
- El seu radi (distància respecte el punt A)
- Un apuntador a la bola BallTree filla dreta
- Un apuntador a la bola BallTree filla esquerre
- Tots els punts que la formen.

```
Coordinate getPivot() {
    return m_pivot;
}

double getRadi() {
    return m_radi;
}

Ball* getEsquerre() {
    return m_right;
}

Ball* getDreta() {
    return m_left;
}

std::vector<Coordinate>& getCoordenades() {
    return m_coordenades;
}
```

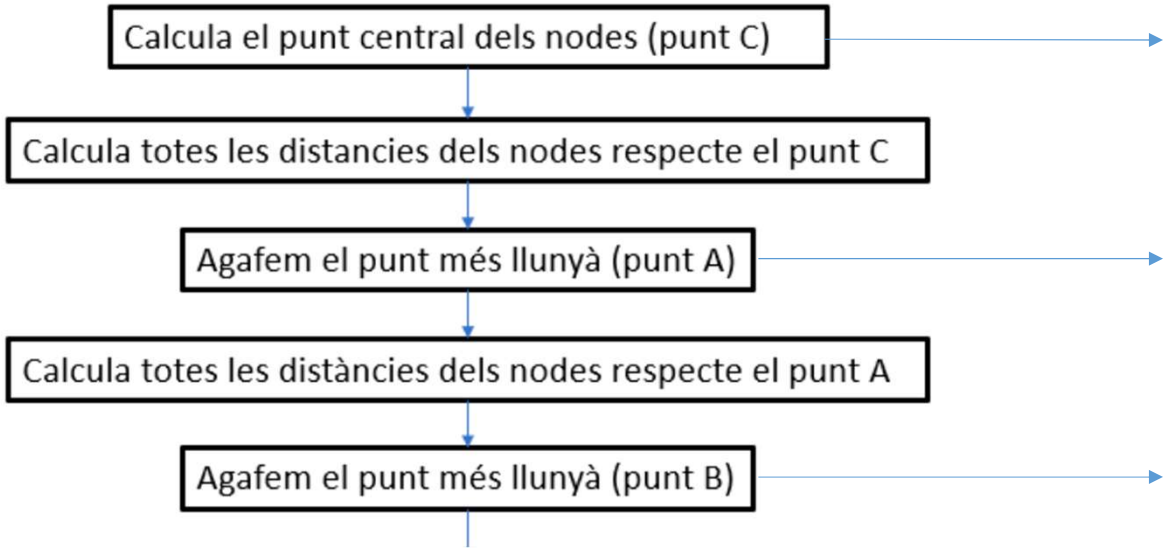
Tasca 2:

Utilitzar els fitxers **BallTree.h i BallTree.cpp** per definir la classe BallTree: Arbre binari format per "Balls" que contindrà els nodes camins

- **No** poden haver coordenades repetides a les fulles del BallTree (però diferents camins poden tenir les mateixes coordenades).
- Podeu utilitzar funcions de la classe Util
- Heu d'implementar:
  - **void inOrdre(std::vector<std::*list*<Coordinate>>& out):** Recòrre arbre en inordre. Retorna llista coordenades de cada Ball, del camí realitzat, a través del paràmetre de referència out.
  - **void postOrdre(std::vector<std::list<Coordinate>>& out):** Recòrre arbre en postordre. 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 arbre en preordre. Retorna la llista de coordenades que conté cadascuna de les boles, del camí realitzat, a través del paràmetre out

Tasca 2:

Algorisme construcció BallTree



C = punt central de bola.getCoordenades()

A = coordenada del qual és la distància màxima respecte al punt C

B = coordenada del qual és la distància màxima respecte al punt A

Consell: Veure funcions de la classe Util pel càlcul de distancies i certs punts.

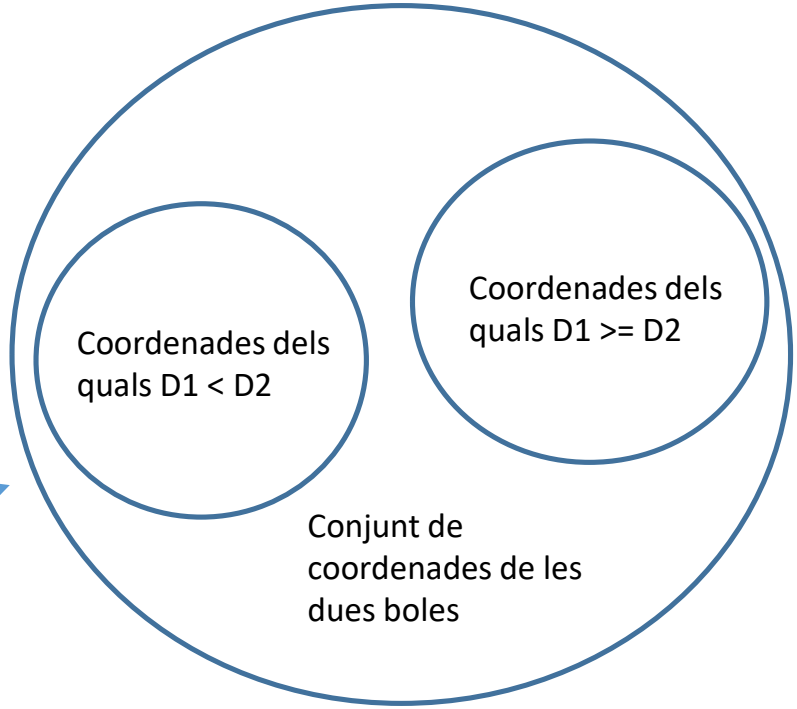
Tasca 2:

Condició d'aturada: quan la bola "gran" disposi d'una sola coordenada (de manera que no té sentit assignar aquesta a cap de les seves boles filles). Estarem doncs, a una fulla.

Iterem pels nodes (node-i):

1. Calcula la distància del node-i al punt A (D1)
2. Calcula la distància del node-i al punt B (D2)
3. Si D1 < D2, assignem node-i a la bola esquerre
4. Si D1 >= D2, assignem a node-i a la bola dreta

Nova bola del qual té nova bola esquerre i una nova bola dreta amb les coordenades assignades.



Tasca 3:

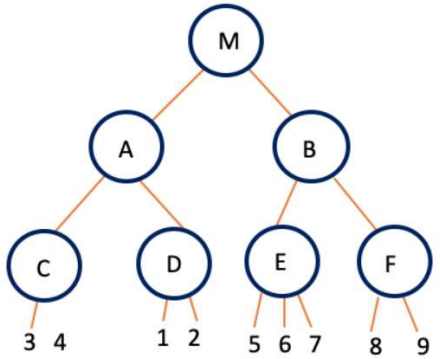
Tasca 3: Implementar la cerca del node-camí més proper del punt d'interès, a través del BallTree.

Per l'avaluació heu d'implementar al BallTree el següent mètode:

- **Coordinate nodeMesProper(Coordinate *pdi*, Coordinate& Q, BallTree* *bola*)**

  Que retornarà la Coordinate node-camí, emmagatzemada a ***bola***, més propera al ***pdi*** que es proporciona utilitzant el diagrama de flux descrit seguidament.

Tasca 3:



Calcula la distància del punt central de la bola respecte al pdi (D1)

Calcula la distància del punt central de la bola respecte al Q (D2)

Si D1 – bola.radi >= D2
retorna Q

Si la bola és una fulla de l'arbre, actualitza Q si és el node camí més proper al punt d'interès, dels punts que formen la bola

Sinó:

1. Calcula la distància pdi respecte el punt central de la bola esquerre (Da)
2. Calcula la distància pdi respecte del punt central de la bola dreta (Db)
3. Si Da < Db, comença la cerca per la bola esquerre, i després, a la dreta
4. Si Da >= Db, comença la cerca per la bola dreta, i després, a l'esquerre.

Q és la coordenada del arbre del qual és la distància més curta respecte pdi.

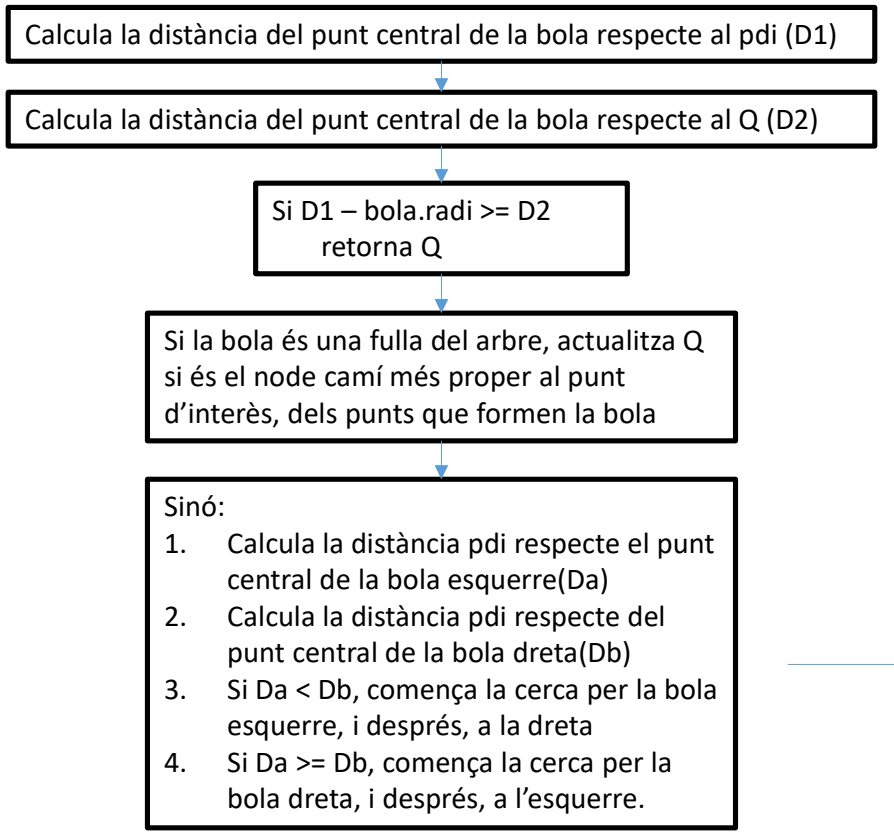
És a dir, s'ha d'actualitzar Q si s'ha trobat una coordenada a la bola del qual està més proper al pdi:

Distancia(pdi, coordenada) < Distancia(pdi, Q_actual)

Per la primera iteració, podem forçar que l'algorisme actualitzi Q indicant-li una coordenada del qual és segur que és molt llunyà a qualsevol punt dels que tractem (per exemple, podem considerar el valor inicial de Q com una coordenada amb latitud i longitud iguals a 0.0).

Un cop actualitzat Q per paràmetre, en retorna la coordenada Q també per la funció.

Tasca 3:

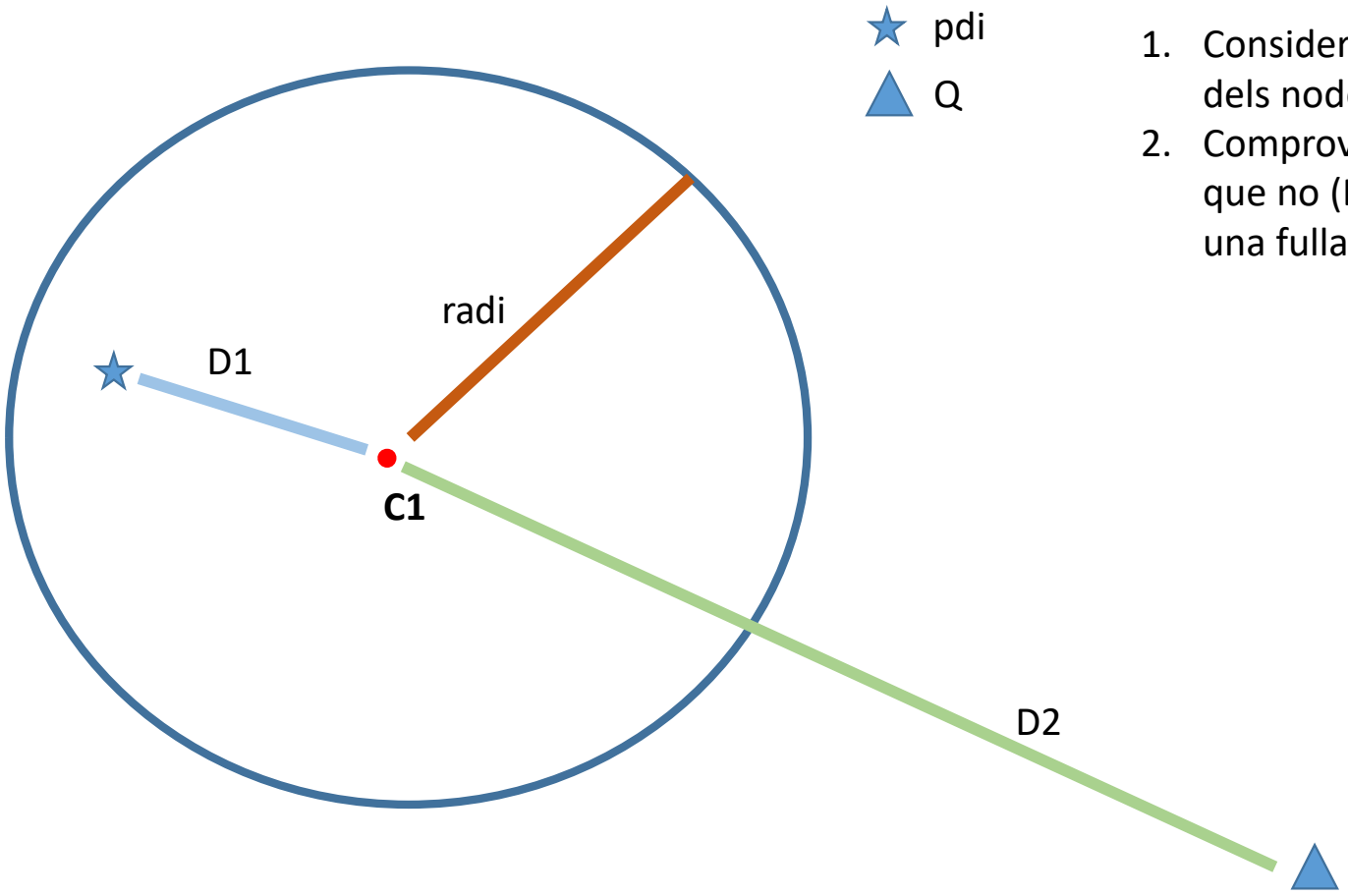


Calculem les distàncies del pdi (punt de interès) respecte el pivot (punt central de la bola) per les boles fill esquerre i dreta.

Comencem altra cop la cerca per la bola fill més propera al pdi.

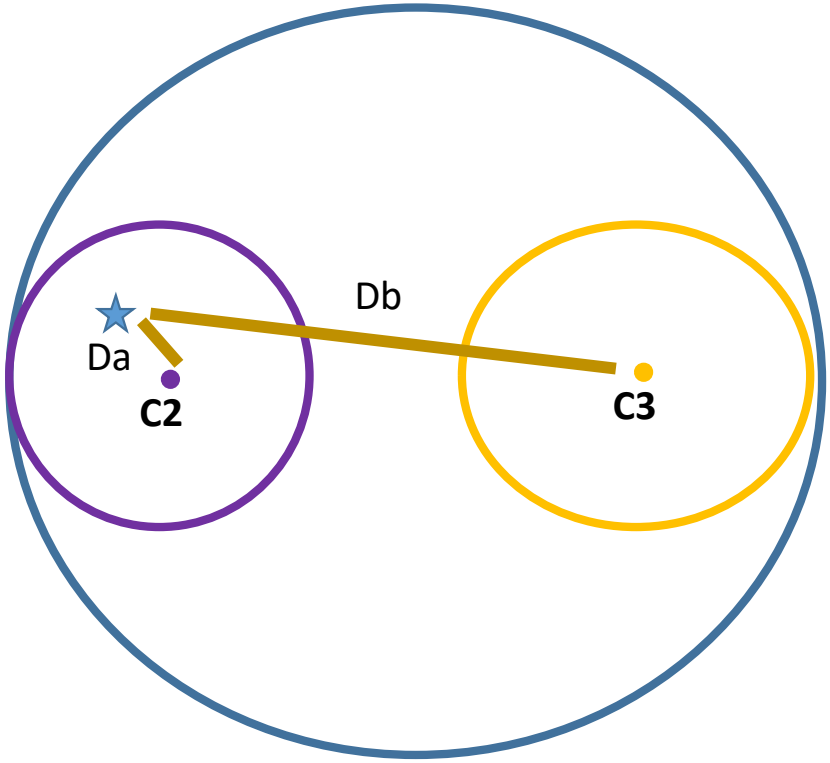
Seguidament, ho fem per la bola més llunyana.

Tasca 3:



1. Considerem Q com a una coordenada molt llunyana dels nodes del arbre.
2. Comprovem si Q és el nostre valor desitjat, i en cas que no (D2 > D1 – radi), comprovem que no sigui una fulla.

Tasca 3:

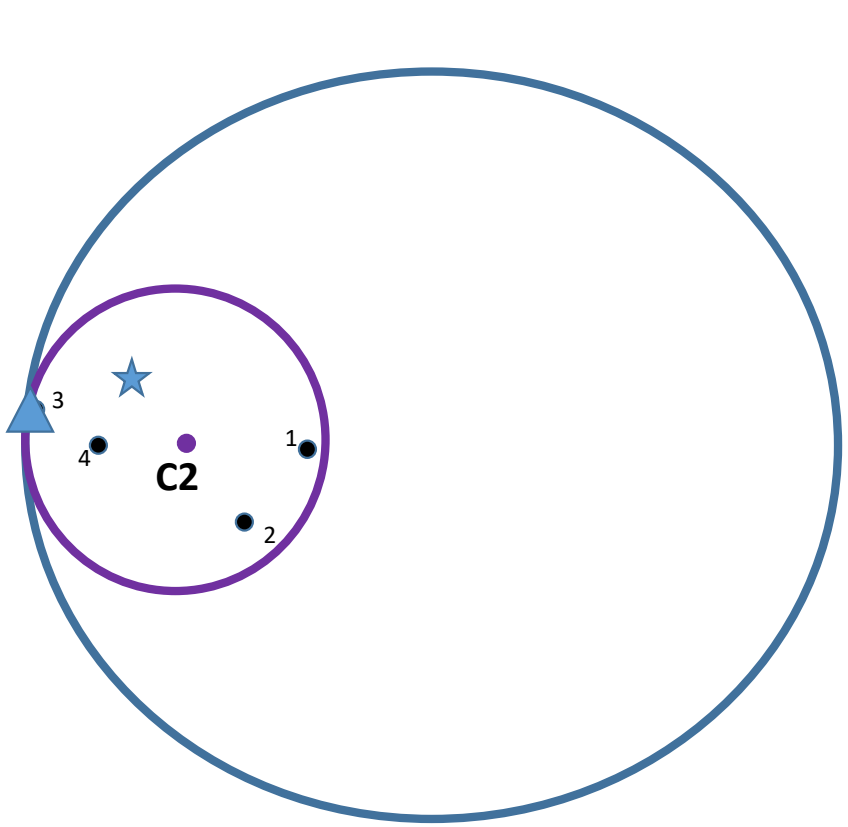


pdi

Q

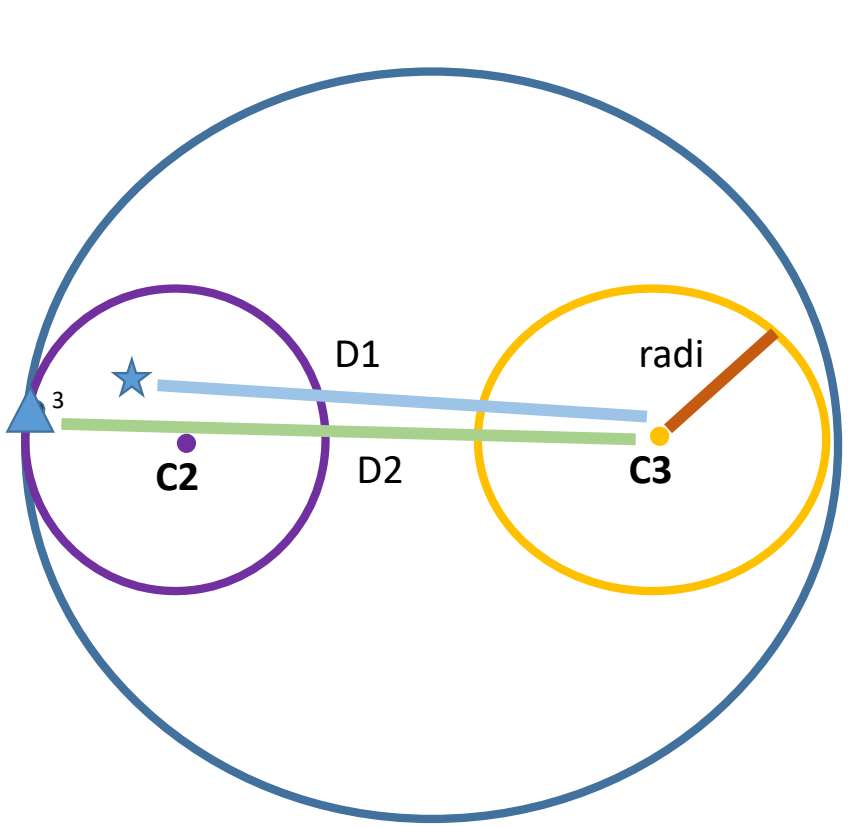
1. Considerem Q com a una coordenada molt llunyana dels nodes del arbre.
2. Comprovem si Q és el nostre valor desitjat, i en cas que no (D2 > D1 – radi), comprovem que no sigui una fulla.
3. Decidim a quina bola cerquem primer mitjançant el càlcul de distàncies entre pdi i els centres C2 i C3.
4. Fet que Da < Db, continuem la cerca primer per la bola C2.

Tasca 3:



1. Considerem Q com a una coordenada molt llunyana dels nodes del arbre.
2. Comprovem si Q és el nostre valor desitjat, i en cas que no (D2 > D1 – radi), comprovem que no sigui una fulla.
3. Decidim a quina bola cerquem primer mitjançant el càlcul de distàncies entre pdi i els centres C2 i C3.
4. Fet que Da < Db, continuem la cerca primer per la bola C2.
5. Per la facilitat en l'animació, considerem que hem arribat a una fulla. Busquem de la bola, quin és el node-camí més proper al pdi, i actualitzem Q.
6. En aquest cas, la coordenada més propera és 3.

Tasca 3:



1. Considerem Q com a una coordenada molt llunyana dels nodes del arbre.
2. Comprovem si Q és el nostre valor desitjat, i en cas que no (D2 > D1 – radi), comprovem que no sigui una fulla.
3. Decidim a quina bola cerquem primer mitjançant el càlcul de distàncies entre pdi i els centres C2 i C3.
4. Fet que Da < Db, continuem la cerca primer per la bola C2.
5. Per la facilitat en l'animació, considerem que hem arribat a una fulla. Busquem de la bola, quin és el node-camí més proper al pdi, i actualitzem Q.
6. En aquest cas, la coordenada més propera és 3.
7. Seguint amb la cerca per la bola C3, comprovem si aquesta bola pot contenir una Q millor que la que tenim. En aquest cas, no, ja que D1 – radi < D2, de manera que retornem Q.

Tasca 4:

Tasca 4: Implementar a la classe **MapaSolucio** el càlcul del camí més curt, donats dos punts d'interès.

Assegureu-vos de tenir a **MapaBase** la següent declaració:

- `virtual CamiBase* buscaCamiMesCurt(PuntDeInteresBase* desde, PuntDeInteresBase* a) = 0;`
- I haureu d'implementar aquest mètode a **MapaSolucio**.
- **Avisos**

1. Aquest és l'únic mètode que s'utilitza a la 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 la interfície gràfica.