Universitat Autònoma de Barcelona

escola d'enginyeria

(6052: RESOLUCIÓ DEL PROBLEMA DEL CARTER XINÈS EN UN MAPA GOOGLE)

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Jordi Cantó Estany
i dirigit per
Joaquim Borges Ayats
Bellaterra, 18 de juny de 2015

Universitat Autònoma de Barcelona

escola d'enginyeria

El sotasignat, Joaquim Borges Ayats

professor/a de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en/na Jordi Cantó Estany.

I per a que consti firma la present.

Signat: Joaquim Borges Ayats

Bellaterra, 18 de juny de 2015

La meva sincera gratitud a tothom qui ha col·laborat en la realització d'aquest Projecte Fi de Carrera perquè, sense vosaltres, mai hauria estat possible.

A na Mari-Llanos, als meus pares, i a tota la gent que ha empès, moltes gràcies.

Índex de continguts

Introducció

Les motivacions

El desconeixement i la curiositat per treballar amb les eines desenvolupades per Google és un repte potent i atractiu, i fou la principal raó per decantar-me per un projecte d'aquestes característiques.

També és interessant el fet d'aplicar, sobre un entorn tan habitual a dia d'avui com és el servei Maps de Google, el coneixement de la Teoria de Grafs, un àrea que sovint és molt present a la vida quotidiana malgrat que passi desapercebuda.

En afegit, el treball m'ha permès prendre contacte amb l'entorn web i JavaScript, que abans d'aquest projecte era escàs sinó nul, i essent entorns que gaudeixen d'una àmplia accessibilitat avui dia, el feien un repte molt temptador.

Així, no seria descabellat considerar-ne l'extensió a altres plataformes com Android o iOS, i més tenint en compte el gran ventall d'eines disponibles al nostre abast.

L'estat de l'art

El problema del carter xinès [1] és un problema d'optimització de distàncies. Així, podem definir el recorregut del carter com un recorregut tancat que travessa, almenys una vegada, cadascuna de les arestes d'un multi-graf ponderat, connex i no eulerià [2] (la darrera condició

no és indispensable, però si el multi-graf és eulerià, el circuit eulerià que conté serà la resposta al problema). Aleshores, les característiques del graf per tractar són:

- Ponderat, perquè cada aresta tindrà un pes associat;
- Connex, perquè per a cada parell de vèrtexs existeix una trajectòria que els uneix, i
- No eulerià perquè, en cas de contenir-ne un circuit, aquesta serà la solució del problema.

Durant l'avaluació dels diferents punts del projecte, no s'ha trobat res de semblant en les fonts consultades. Sí que és cert de que hi ha disponibles diverses funcionalitats, implementacions en certs llenguatges d'alguns dels algorismes nombrats, però no s'ha trobat res que resolgui el problema del carter xinès mitjançant codi obert, i molt menys aplicant el problema sobre un mapa de Google.

Un mètode que resol el problema del carter xinès és l'algorisme d'Edmonds que, cercant els vèrtexs de grau senar del multi-graf, en troba les arestes tals que la seva duplicació faci que aquests vèrtexs tinguin grau parell complint amb el teorema d'Euler. Aquestes arestes per duplicar formaran part del camí mínim entre aquests vèrtexs de grau senar, i obtindrem aquest camí mínim mitjançant la reconstrucció de camins de l'algorisme de Hierholzer, que troba un circuit eulerià.

Però, abans d'arribar a aquest punt, cal aconseguir crear un recorregut sobre un mapa Google, que es pugui extrapolar a un graf de les característiques adequades sobre el que aplicarem l'algorisme d'Edmonds.

Per això, la interacció amb el mapa Google ha d'assegurar els següents punts:

- La selecció de carrers es farà tram a tram: un clic sobre un carrer seleccionarà el tram entre la cruïlla més propera a banda i banda.

- No pot haver-hi cap tram seleccionat aïllat de la resta, és a dir, no es pot sortir dels carrers seleccionats per tal d'arribar-ne a un altre.
- A més a més dels trams desitjats, l'usuari seleccionarà un sol punt d'origen i final pel recorregut, és a dir, el recorregut ha d'acabar al mateix punt on comença.

El procés de construcció del graf esdevindrà mimètic: les arestes seran els trams de carrer que formen el recorregut, i els vèrtexs seran les cruïlles o interseccions que els uneixin.

Estudi de viabilitat

Les diverses eines desenvolupades per Google permeten interactuar amb el seu entorn amb garanties, fins i tot a nivell d'usuari.

D'altra banda, el problema del carter xinès és un àmbit d'estudi consolidat dins la Teoria de Grafs, de manera que el material relacionat disponible és prou extens com per permetre l'aprofundiment desitjat en els seus mètodes de resolució.

Finalment, cal tenir en compte el conjunt d'entorns de treball que hem pogut anar tastant al llarg de l'enginyeria, que poden adequar-se a les característiques del nostre projecte.

Així, be sigui per la complexitat del problema, per la manca de material relacionat, o pel volum de dades per processar que han de manegar, és factible que el projecte gaudeixi d'una viabilitat com pocs d'altres poden gaudir-ne.

Els objectius del projecte

L'objectiu principal que es vol assolir mitjançant aquest projecte és el d'obtenir un lloc web que permeti trobar, seleccionats un nombre aleatori de trams (seccions de carrer, entre cantonades) connexos entre sí, la ruta mínima a seguir per passar-hi per tots, tot començant i acabant al mateix punt (origen).

D'altra banda, seria bo obtenir, per a la resolució del problema, uns algorismes fàcilment escalables, de manera que donades les condicions de rendiment adequades pugui obtenir-se un resultat pel problema del carter xinès donat qualsevol nombre d'arestes.

La distribució del temps

La distribució del temps plantejada a l'informe previ no té res a veure amb la que s'ha aplicat realment, que es mostra a continuació.

El plantejament inicial, que incloïa la defensa del projecte al gener amb tots els canvis que comporta, es va haver de descartar per imprevistos i contratemps totalment aliens a la realització del projecte, que van reduir la meva dedicació fins la pràctica nul·litat.

Aleshores, la distribució real que es planteja a continuació és visiblement més curta en el temps, amb un impàs de mesos d'inactivitat, de manera que si els objectius s'han pogut dur a terme ha estat gràcies a una dedicació diària totalment intensiva.

Tasca	Data d'inici	Data de fi
Avaluació del problema	16/10/2014	31/10/2014
Contacte amb l'API de Google	3/11/2014	21/11/2014
Realització i entrega de l'informe previ	9/12/2014	12/12/2014
Introducció i aprofundiment en les eines	20/04/2015	28/04/2015
Desenvolupament dels algorismes	29/04/2015	17/05/2015
Implementació dels algorismes desenvolupats	18/05/2015	7/06/2015
Proves i correccions al codi implementat	8/06/2015	14/06/2015
Confecció i entrega de la memòria	15/06/2015	18/06/2015
Preparació de la presentació i entrega del disc	19/06/2015	22/06/2015

L'organització de la memòria

El plantejament seguit per confeccionar el present document és el següent:

Al següent apartat, es revisaran els fonaments teòrics sobre els que s'ha basat la resolució del present projecte.

A continuació, es descriu el procés que seguirà l'aplicació per tal d'assolir els objectius marcats, així com la problemàtica que hi ha sorgit i com s'ha solucionat.

Finalment, conté un informe tècnic on s'analitzen, per una banda, les eines que s'han fet servir per la resolució del projecte i, per l'altra, el funcionament dels diversos elements desenvolupats.

Per acabar, s'exposaran les conclusions obtingudes i es citaran les referències emprades.

Fonaments teòrics

La teoria de grafs

La teoria de grafs és una branca de les matemàtiques i la informàtica que es dedica a l'estudi dels grafs, estructures matemàtiques utilitzades per a modelitzar relacions entre parelles d'objectes [3].

En aquest context [4], un graf consisteix en una col·lecció de vèrtexs o nodes connectats per línies anomenades arestes. Els grafs poden ser no dirigits, és a dir sense fer distinció entre l'ordre dels dos vèrtexs associats a cada aresta, o dirigits, les arestes dels quals van d'un vèrtex a un altre; en aquest cas, les arestes s'anomenen arcs. Els grafs se solen representar gràficament amb un punt o cercle per cada vèrtex, i una línia entre vèrtexs connectats. Si el graf és dirigit, els arcs se simbolitzen amb fletxes, indicant la direcció de la relació entre els dos vèrtexs.

Els grafs són un dels principals objectes d'estudi de la matemàtica discreta. Les aplicacions de la teoria de grafs giren al voltant d'estructures que poden ser sistematitzades amb grafs, com per exemple l'estructura de llocs web, l'anàlisi de xarxes, l'estudi de molècules en química i física, o altres camps com els estudis sociològics. El precursor de la teoria de grafs fou Leonhard Euler, que la va iniciar intentant resoldre el problema dels set ponts de Königsberg.

En teoria de grafs, el grau [5] o valència d'un vèrtex és el nombre d'arestes que hi incideixen, amb els bucles comptats dues vegades, i el teorema d'Euler [6] exposa que, donat un graf o multi-graf simètric i connex, té circuit eulerià si i només si tots els vèrtexs tenen grau parell, i

té camí eulerià si i només si el nombre de vèrtexs amb grau senar és 0 o 2. Aleshores, donat un graf simètric i connex, és eulerià si i només si es pot fer una partició de les arestes del graf en circuits disjunts d'arestes.

Finalment, el lema de graus enuncia que un graf sempre hi ha un nombre parell de vèrtexs amb grau senar, i es defineix com un graf eulerià aquell graf connex i no dirigit en el que tots els seus vèrtexs tenen grau parell. Aleshores, Un camí eulerià és un camí que passa per cada aresta del graf només una única vegada i, un circuit eulerià, és un camí tancat que passa per cada aresta del graf només una única vegada.

Com a curiositat, Euler també va trobar la correlació existent en els grafs planaris (aquells que poden ser dibuixats sense que cap aresta intersequi) entre el nombre de cares, el d'arestes i el de vèrtexs, de la següent manera:

$$Vèrtex - Arestes + Cares = 2$$

El problema del carter xinés

El problema del carter xinès és un problema d'optimització de distàncies, que fou formulat de la següent manera:

Un carter surt de l'oficina de correus, reparteix la correspondència a cadascuna de les illes de cases del seu districte i retorna al punt de partida. Quina és la ruta que ha de seguir per a que la distància total recorreguda sigui mínima?

Aleshores, definirem el recorregut de carter d'un graf ponderat connex i no eulerià, com un recorregut tancat que travessa cadascuna de les arestes del graf almenys un cop. És a dir, el graf haurà de complir els següents requisits:

- Haurà de ser ponderat, perquè cada aresta tindrà un pes associat;
- Haurà de ser connex, és a dir, perquè per cada parell de vèrtexs existirà una trajectòria que els uneix, i
- No haurà de ser eulerià perquè, en cas de contenir un circuit eulerià, aquesta serà la solució del problema.

Per tant, el que cal trobar és la manera de que sí existeixi un circuit eulerià a partir del graf proposat.

L'algorisme d'Edmonds

Un mètode que resol el problema del carter xinès és l'algorisme d'Edmonds [7] que, cercant els vèrtexs de grau senar del graf, en troba les arestes tals que la seva duplicació faci que aquests vèrtexs tinguin grau parell complint amb el teorema d'Euler.

Aquestes arestes per duplicar formaran part del camí mínim entre aquests vèrtexs de grau senar, i obtindrem aquest camí mínim mitjançant la reconstrucció de camins amb algorismes com el de Fleury o el de Hierholzer.

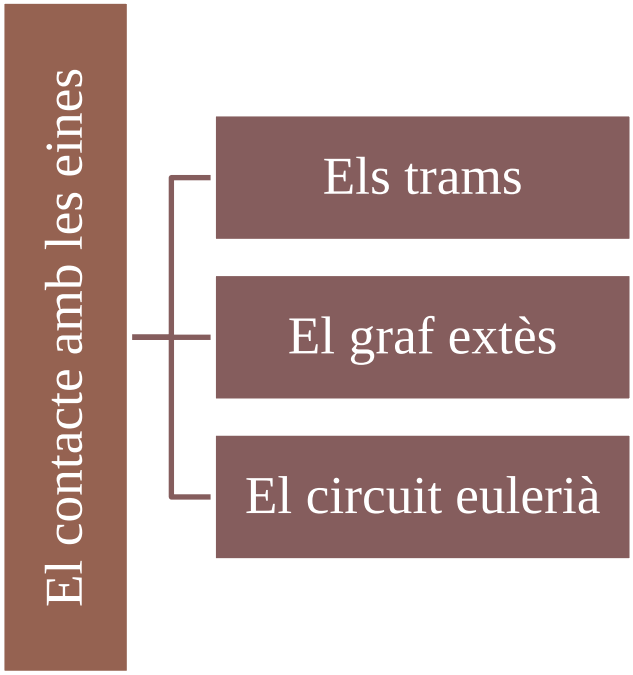
L’algorisme de Hierholzer

Un mètode per la reconstrucció de camins és l’algorisme de Hierholzer [8], que consisteix en crear un circuit eulerià partint de la concatenació de circuits disjunts respecte de les arestes.

El funcionament és el següent, iniciem la construcció d'un circuit en el graf *G*, a partir d'un vèrtex que té grau parell. Un cop construït aquell circuit, eliminem les arestes del graf i concatenem el circuit amb el circuit eulerià que estem construint. Buscarem un nou vèrtex de grau parell entre els que ja hem visitat i com el graf és eulerià i compleix el teorema d'Euler, llavors segur que el trobarem ja que el graf encara té arestes.

Llavors, construirem un nou circuit i el concatenarem al circuit eulerià que estem construint de la següent manera, agafarem el primer vèrtex del circuit i substituirem tot el circuit a la primera ocurrència d'aquest vèrtex en el circuit eulerià. Aquest procés de cerca d'un vèrtex de grau parell, construcció d'un circuit i concatenació ho farem mentre hi hagin arestes en el graf.

Descripció del procés



El contacte amb les eines

Prèviament a qualsevol anàlisi de resolució del problema, cal conèixer mínimament les eines amb que es treballa per tal d'esbrinar els límits de cadascun, i les avantatges i els inconvenients en cas d'haver-hi més d'una alternativa.

Com que l'aplicació ha de treballar basant-se en el servei de mapes de Google, es va optar per començar mirant que pot oferir l'API de Google Maps, i de seguida es va veure que treballa sobre JavaScript, de manera que seria el pas següent. Pel que fa a l'API, es va comprovar que obtenir un mapa pel que navegar i afegir-hi marcadors seria senzill, i el principal problema radicaria en la gestió de les rutes tal i com ho implementa la versió actual.

Posteriorment, es va començar a treballar amb el llenguatge JavaScript, que amb les nocions de programació adquirides no semblava complicat, tot i que realment poc a veure té amb el llenguatge Java malgrat el nom. Des d'aquí, afegir-ho a una carcassa HTML fou trivial: la carcassa només cal que contingui el mapa per seleccionar els trams, i un botó per indicar que la selecció ha acabat i es pot procedir a calcular la ruta.

L'obtenció dels trams

Per resoldre el problema, el primer que cal és un mapa, i la possibilitat de triar els trams a voluntat de l'usuari. Obtenir el mapa on triar els trams, així com col·locar-hi els marcadors segons cada clic, és trivial, però aquí es troba la primera i principal problemàtica, que xoca frontalment amb el plantejament inicial del projecte: Google només permet la selecció de punts als seus mapes.

És a dir, el plantejament de seleccionar un tram sencer entre dues interseccions amb un sol clic, queda descartat. La solució trobada dona més protagonisme a l'usuari, que és qui haurà de triar les parelles de punts que seran els extrems d'un tram, que ara ja no seran específicament interseccions: ara, és l'usuari qui tria un conjunt de parelles origen-destí.

L'ampliació del graf

Una vegada obtinguts dos conjunts amb el mateix nombre d'elements, un d'orígens i l'altre de destins, cal verificar que es tracta d'un graf connex i aleshores centrar-se només en els de grau

senar, és a dir, els que apareixen un nombre senar de vegades en el macro-conjunt format per orígens i destins.

Donat el conjunt de nodes amb grau senar, que ha de ser parell pel lema de graus, aleshores cal definir-ne la matriu de distàncies mínimes, i trobar-ne el conjunt de parells amb cost total mínim i, segons l'algorisme d'Edmonds, són els que s'afegiran als conjunts originals d'orígens i destins.

L'obtenció del circuit eulerià: la ruta a seguir

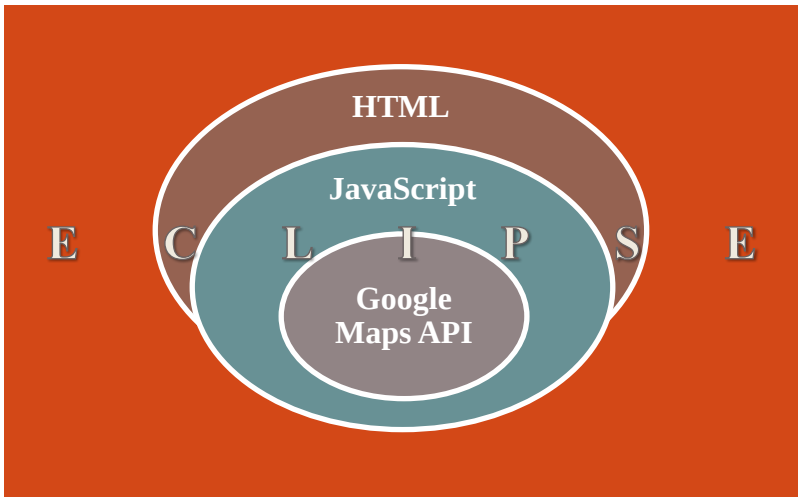
Finalment, ara disposem d'un conjunt de trams que contenen un circuit eulerià, i només resta trobar la ruta a seguir mitjançant el mètode de reconstrucció de camins de l'algorisme de Hierholzer.

Informe tècnic

En aquest apartat s'explicarà el perquè s'han escollit les eines emprades, i el com s'han desenvolupat els documents, procediments i algorismes que s'han implementat en la resolució d'aquest projecte.

Les eines

En apartats anteriors, s'ha pogut veure que les eines emprades han estat l'aplicació Eclipse, i els llenguatges HTML i JavaScript, i l'API de Google Maps.



Eclipse

L'Eclipse [9] és un entorn integrat de desenvolupament de codi obert, programada principalment en Java i per desenvolupar projectes en diversos llenguatges sempre que es disposi dels connectors corresponents per a cadascun.

S’ha triat aquesta eina per la seva versatilitat, i perquè la seva utilització ha estat recurrent al llarg dels estudis, de manera que només calia reprendre el contacte, fet que és un avantatge cabdal. En concret, s’ha emprat el paquet Eclipse Java EE IDE for Web Developers, en la seva versió Luna Service Release 2 (4.4.2), la darrera disponible.

HTML

L’HTML [10] és un llenguatge de marcat que deriva de l’SGML dissenyat per estructurar textos i relacionar-los en forma d’hipertext. Gràcies a la Internet i als navegadors web ha esdevingut en un dels formats més populars que existeixen per a la construcció de documents per la web.

L’HTML 5 [11] és la cinquena gran revisió del llenguatge bàsic, i és la que s’ha fet servir per estructurar la pàgina web que carregarà el mapa amb el que interaccionarà l’usuari, una vegada vist que el nucli de l’aplicació s’implementaria mitjançant JavaScript.

JavaScript

El llenguatge JavaScript [12] és un llenguatge script basat en el concepte d’herència per delegació, implementat originàriament per Netscape Communications Corporation, i que va derivar en l'estàndard ECMAScript. És conegut sobretot pel seu ús en pàgines web, però també s'utilitza en altres aplicacions. Malgrat el nom, JavaScript no deriva del llenguatge de programació Java, però ambdós comparteixen una sintaxi similar basada en el llenguatge C.

El principal motiu de la tria de JavaScript, en detriment de llenguatges més robustos, és que l’API de Google Maps hi està basada. A més a més, tot i que ni de bon tros es pot considerar rellevant, l’ús del llenguatge C ha estat recurrent al llarg dels estudis, i JavaScript hi té algunes similituds.

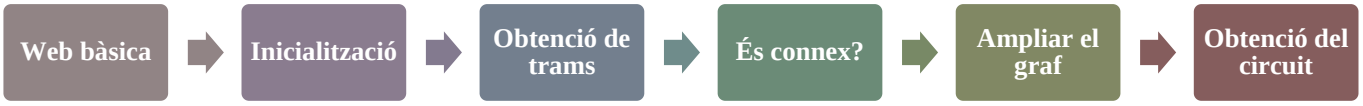
API de Google Maps

L'API [13] de Google Maps és una interfície de programació d'aplicacions basada en el llenguatge script JavaScript que el propi servei Google Maps de Google té a disposició dels desenvolupadors.

Donades les seves característiques, esdevé una eina bàsica i primordial a l'hora de treballar amb el servei de mapes de Google. Aquesta eina corporativa garanteix una fiabilitat i una accessibilitat que, obtenir-les de manera individual, implicaria una despesa de temps i esforç que no es podrien assumir, i encara menys garantir-ne els resultats.

Pel projecte hagués estat preferible emprar la versió 2 de l'API, que és més ràpida i intuïtiva per treballar sobre una funcionalitat concreta, però malauradament és obsoleta i s'ha hagut de fer servir la versió 3 [14], molt més orientada a funcionalitats múltiples amb múltiples capes, fet que l'alenteix.

La implementació



La pàgina web bàsica

La pàgina web bàsica està implementada mitjançant HTML5 i és una mera carcassa pels tres elements que conté: el mapa de Google sobre el que interactuarà l'usuari i sobre el que es

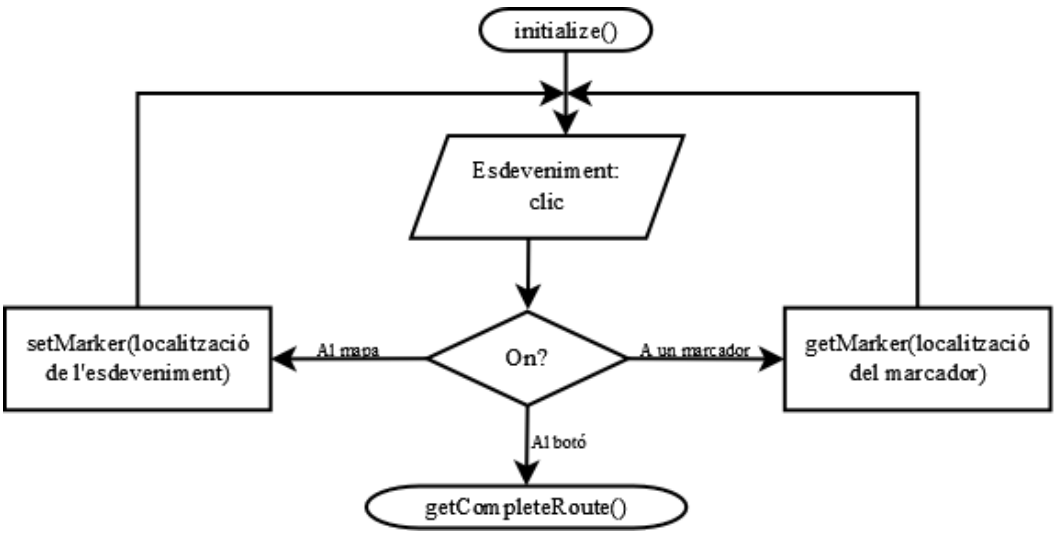
mostrarà la resposta; un botó perquè l'usuari pugui indicar quan ha acabat de seleccionar els trams, i un espai on es mostrarà mitjançant text la ruta calculada.

Perquè la sensació d'espera per l'usuari sigui menor, el que es carrega durant la càrrega de la pàgina és, d'una banda, el mapa en sí i, d'altra banda, les funcions per l'obtenció dels trams. Així, la resta de funcions, que es faran servir una vegada s'hagi de calcular el recorregut, es carreguen amb el cos de la pàgina web, mentre que el que s'ha carregat a la capçalera és accessible amb visible anterioritat.

Pel que fa als scripts, a banda de les variables globals de l'aplicació, les úniques funcions detallades a la pròpia web són la que inicialitza el procés de captura de trams i la que inicia el càlcul de ruta, a més a més de la línia de codi que vincula el servei de mapes de Google a l'esdeveniment de càrrega de la pàgina i la funció d'inicialització.

La inicialització

Al script dins la capçalera de la pàgina web, es poden trobar les variables globals de l'aplicació, la funció d'inicialització i la línia de codi que vincula el servei de mapes de Google a l'esdeveniment de càrrega de la pàgina i la funció d'inicialització.

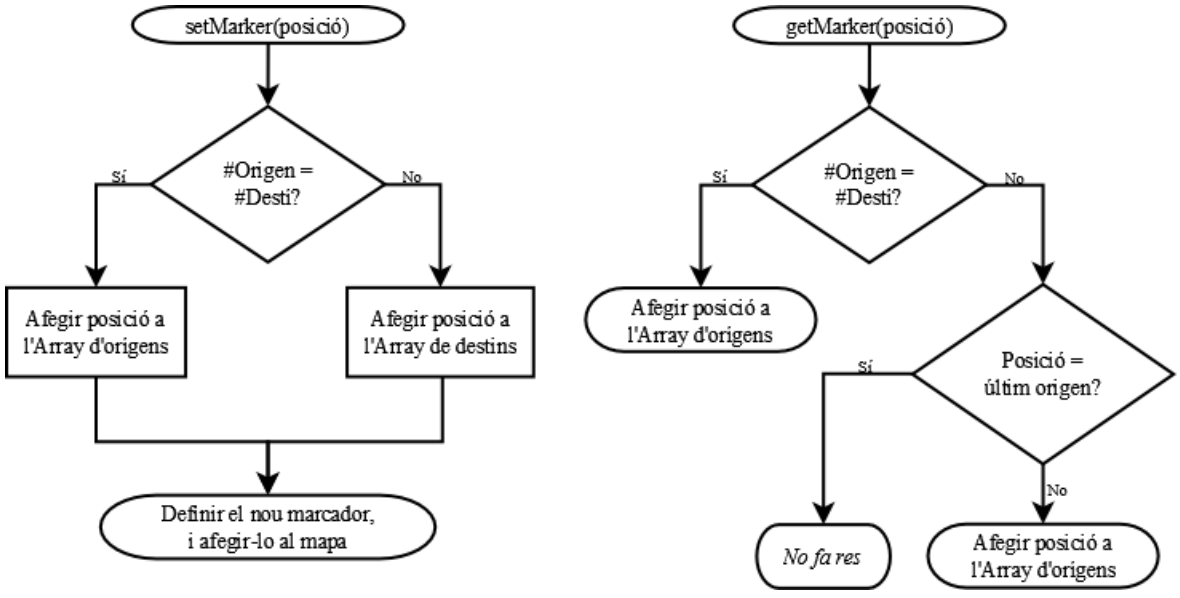


La funció d'inicialització és simple: es defineixen les propietats del mapa amb el que es treballarà, es creen els objectes necessaris per treballar-hi (el propi mapa, i un marcador de Google), i s'escolta indefinidament els esdeveniments introduïts per l'usuari que es volen tenir en compte.

L'obtenció dels trams

L'obtenció dels trams es realitza mitjançant una parella de coordenades origen-destí, coordenades del tipus *LatLng* facilitat per l'API de Goolge Maps. Aquestes coordenades s'emmagatzemen en dos Array, l'un d'orígens i l'altre de destins on, donada una posició *i* dins els Array, l'origen del tram *i* es correspon amb la destinació del tram *i*.

Aleshores, podem distingir dos casos: quan l'usuari clica sobre el mapa, que es resol amb la crida del procediment *setMarker(localització)*, i quan ho fa sobre un marcador ja existent, que es resol amb la crida del procediment *getMarker(localització)*.



El procediment *setMarker(localització)* inicialment avalua si la localització correspon a un origen o a un destí, fet que és totalment indiferent a l'hora de calcular el traçat, però no a

l'hora de gestionar-lo. És a dir, de cara al funcionament del programa, és indispensable que l'Array d'orígens i el de destins tinguin la mateixa longitud i, aleshores, el que fa el programa és desar la localització del paràmetre com origen si tots dos Array tenen la mateixa longitud, i com a destí en cas contrari.

Una vegada emmagatzemada la localització, es crea un nou marcador de Google definint-ne la localització al mapa, i es col·loca al mapa. Es va considerar l'opció d'avaluar si es triava el mateix punt com origen i destí del mateix tram, però ràpidament es va descartar: d'una banda, clicar dues vegades exactament al mateix punt del mapa és difícil i, en tot cas, al crear el marcador el més factible és que s'hi cliqui a sobre, i per tant és allà on cal valorar aquesta casuística.

El procediment *getMarker(localització)* és molt similar a l'anterior, però s'obvia la creació d'un nou marcador a una localització ja marcada. En canvi, si els Array són de diferent longitud, es comprova que la localització no coincideix amb la de la darrera posició de l'Array d'orígens abans d'afegir-la al de destins, per evitar trams on origen i destí facin referència al mateix punt del mapa.

La comprovació de la connectivitat

La comprovació de la connectivitat és duu a terme mitjançant la funció *isConnected(),* que retornarà *true* si el conjunt d'orígens i destins és connex, i *false* en cas contrari.

L'algorisme implementat és un algorisme BFS (*Breadth First Search*, "cerca primer en amplada"), és a dir, es comença a l'arrel (l'origen en el nostre cas) i s'exploren tots els destinataris d'aquest punt per, a continuació, explorar els respectius veïns adjacents fins haver recorregut tot el graf.

La funció, que només pot ser cridada si la longitud del Array d'orígens és la mateixa que la del de destins, d'entrada retorna *true* en cas que la longitud sigui igual a 1: un graf amb un sol origen i un sol destí serà connex per definició.

En cas contrari, apareix el problema de la tipologia dels objectes: A JavaScript, pel que fa als Array, les eines per trobar-ne elements es limiten a uns pocs tipus d'objecte bàsics. Així, el primer que es fa és crear dos Array paral·lels d'orígens i destins amb les coordenades desades en tipus String, aplicant el mètode facilitat a l'API. A més a més, es crea un tercer Array de la mateixa longitud amb tots els elements inicialitzats a *false*, que indicaran que el tram ha resultat accessible.

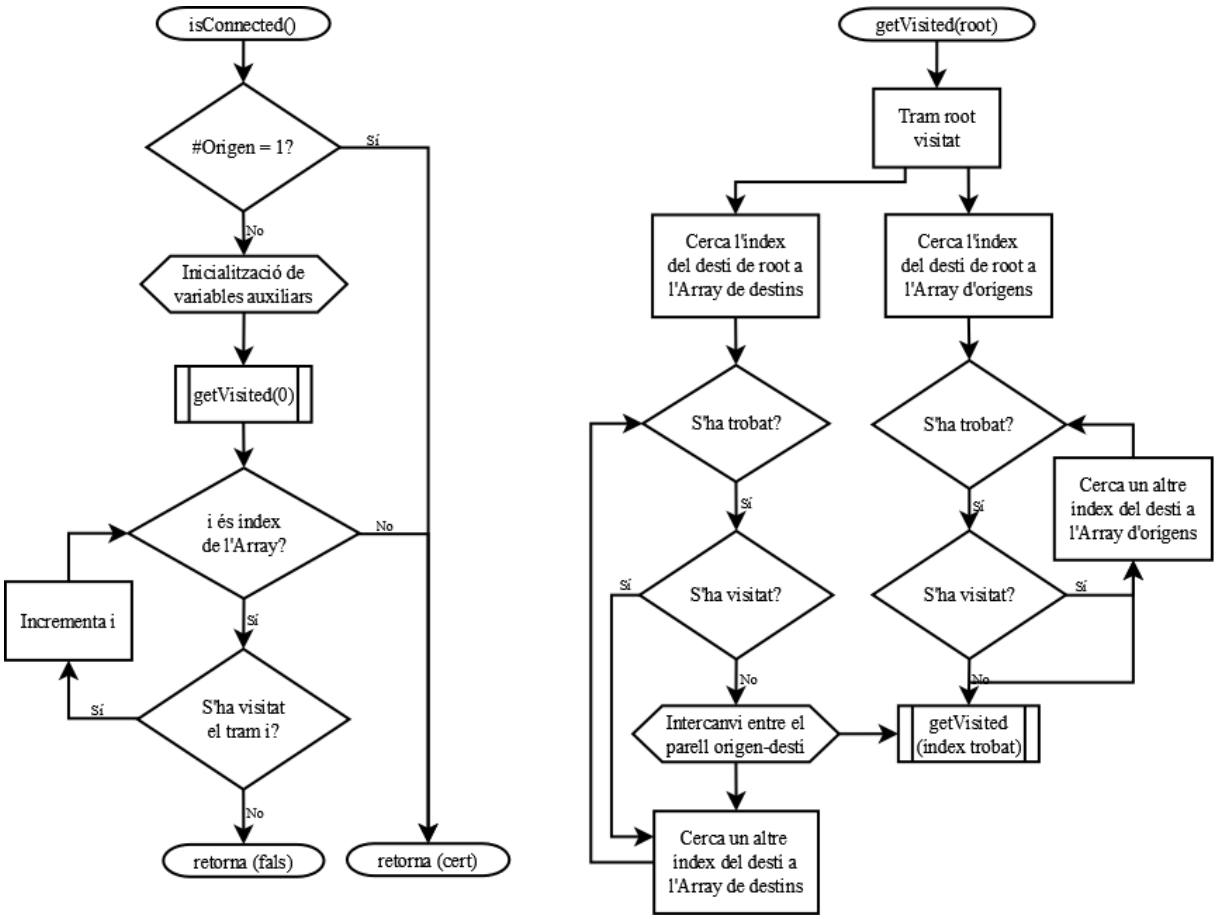
Una vegada creats els Array necessaris per procedir, es realitza el procediment *getVisited(base)*, que anirà marcant recursivament els trams que vagin resultant accessibles al recorregut. Una vegada cridada la funció, localment només li calen una variable auxiliar (per controlar el fet que sempre partim d'un origen) i dos índexs per recórrer individualment els Array modificats d'orígens i de destins.

El procediment es crida amb un paràmetre enter arrel, de manera que *getVisited(root)* crea les seves variables locals i la posició *root* a l'Array de trams visitats pren valor *true*. A continuació, es realitza una cerca circular de la següent aparició de *root* a l'Array d'orígens i, si s'hi troba una posició que no ha estat visitada, s'aplicarà el procediment a l'índex obtingut. Aquesta cerca es repeteix mentre l'índex trobat, si n'hi ha, sigui diferent a l'arrel.

Després de la primera cerca circular, se'n realitza una altra però aquesta vegada es cerca la següent aparició de *root* a l'Array de destins seguint el mateix procés. L'única diferència, a banda de l'Array objectiu de la cerca, és que cal intercanviar els valors d'origen i de destí

degut a que, per la definició del procediment, *root* fa referència a un origen i, l'índex trobat, a un de destí.

Una vegada finalitza l'execució del procediment recursiu *getVisited()*, només resta avaluar-ne el resultat: si hi ha un sol tram que mantingui el paràmetre a *false* dins l'Array dels trams visitats, voldrà dir que no s'hi pot arribar des d'enlloc, mentre que si tots han pres el valor *true*, la ruta serà connexa.



L'ampliació del graf

Una vegada s'ha garantit que la ruta és connexa, cal obtenir una ruta que contingui un circuit eulerià i, per aconseguir-ho, es defineix un procés de tres fases: localitzar els marcadors amb grau senar (és indiferent si són orígens o destins), definir la matriu de distàncies entre aquests

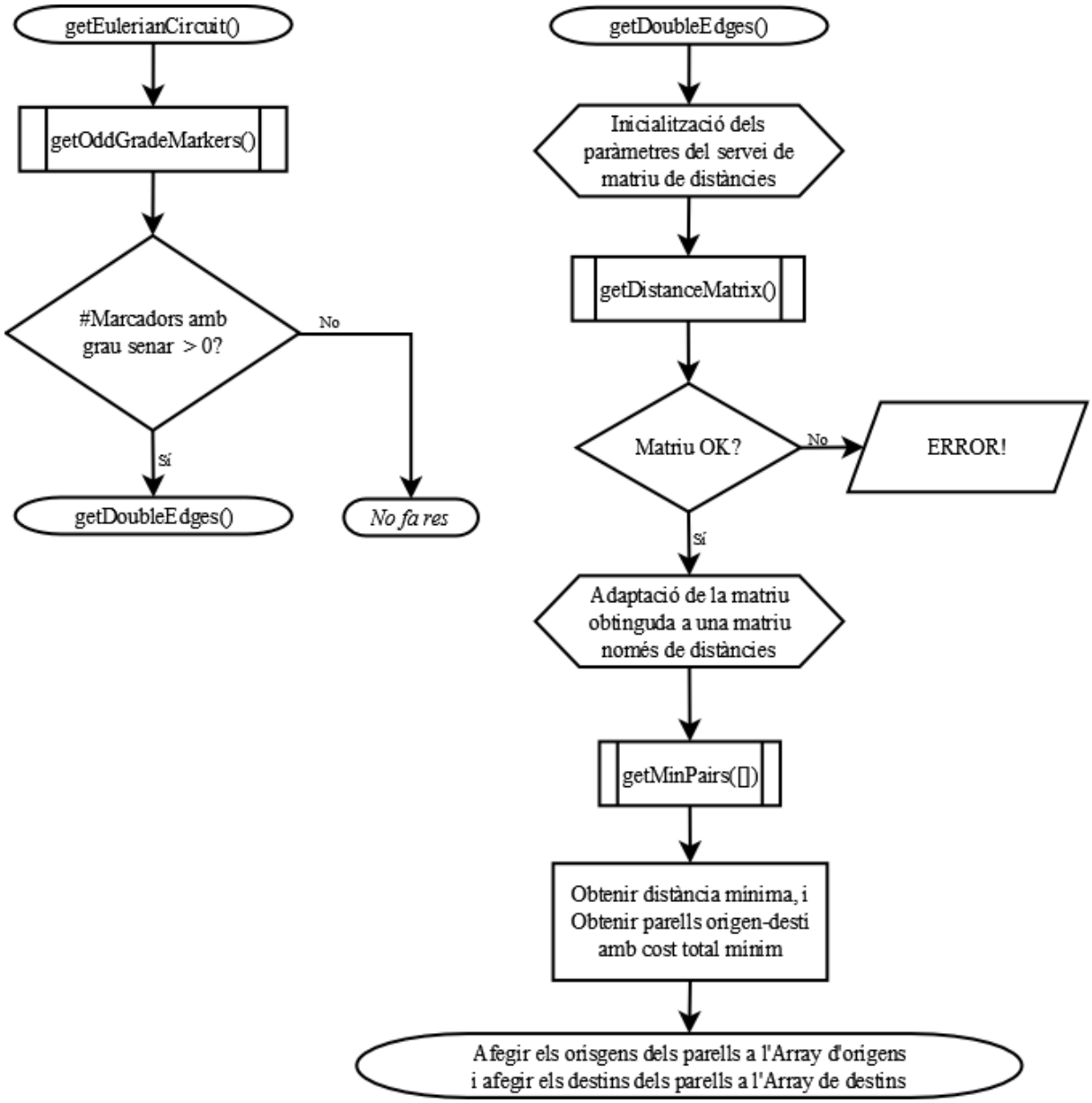
marcadors, i obtenir-ne els parells amb cost total mínim que definiran els trams per afegir a la ruta inicial.

Per obtenir els nodes de grau senar s'ha definit el procediment *getOddGradeMarkers()*, que agrupa tots els marcadors en un Array, i en declara un d'igual longitud pels graus. Per la definició de l'algorisme, tots els marcadors tenen com a mínim grau 1, i a mesura que se'n troben repeticions s'incrementa el grau en les dues posicions que coincideixen. Per descomptat, aquest primer pas del procediment comporta nodes repetits tantes vegades com grau tinguin, fet que no interessa per trobar els parells mínims.

Per resoldre-ho, es crea un Array de String i es recorre l'Array de tots els marcadors una sola vegada: si el grau és senar, es cerca el marcador en tipus String a l'Array auxiliar i, si no hi és, s'afegeix el marcador en sí mateix a l'Array de marcadors amb grau senar, i en tipus String a l'Array auxiliar. Així, quan finalitza el procediment, disposem d'un Array només amb els marcadors de grau senar.

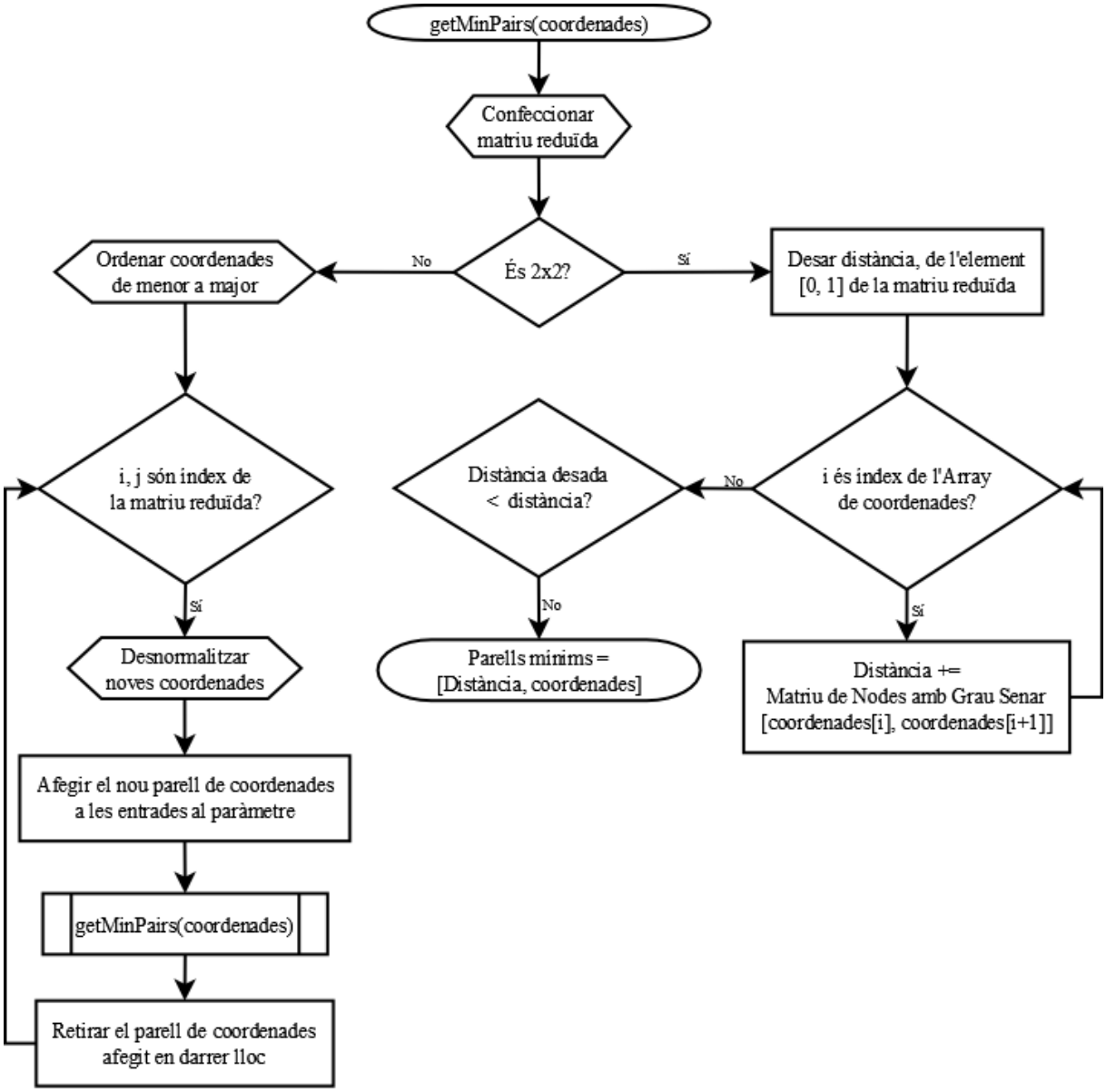
Una vegada s'han localitzat i agrupat els marcadors de grau senar, el procediment *getDoubleEdges()* fa servir el servei de matriu de distàncies de Google Maps per obtenir la matriu de distàncies només entre aquests marcadors, tots contra tots, indicant que el mètode de transport és caminant, que s'emprarà el sistema mètric, i que s'evitaran accidents com autopistes, peatges o ferris.

La matriu de distàncies que s'obté directament del servei de Google és poc manejable: enlloc de matriu, les dades obtingudes són un conjunt d'etiquetes que identifiquen un Array de conjunts d'etiquetes. Així, el primer que es fa amb la matriu de distàncies obtinguda és crear un Array de dues dimensions, una matriu, només amb les distàncies entre els marcadors, que és el que realment és necessari per al desenvolupament del programa en aquest punt.



A continuació, es crida el procediment *getMinPairs([])*, que rep per paràmetre un Array buit ja que d'entrada no es coneixen les coordenades de cap dels parells mínims a la matriu de distàncies obtinguda. Una vegada finalitza aquest procediment, s'haurà obtingut un Array de coordenades que es corresponen amb les posicions dels elements que s'afegiran a l'Array d'orígens (els elements de les posicions parells, des de la 0 en endavant) i al de destins (els elements de les posicions senars, des de la 1 en endavant).

El procediment *getMinPairs(coordenates)* comença simplificant la matriu original, ignorant totes les files i columnes que apareixen a l'Array de coordenades (al cas inicial, la matriu auxiliar serà igual que l'original), ja que són punts pivot que ja s'han avaluat i la idea és que cada element formi part només d'una sola parella.



Si la longitud ja s'ha reduït a 2, es captura l'única opció possible, i finalitza la present línia d'execució: s'afegeix l'única distància que conté a la suma de les distàncies que indiquen les coordenades acumulades i, si el resultat obtingut és inferior a l'obtingut en alguna iteració

anterior, es desen les coordenades obtingudes perquè seran un conjunt de parells amb cost total inferior a l'anterior. En canvi, si la longitud és més gran, cal continuar revisant.

Per continuar obtenint coordenades, es crea un Array auxiliar del mateix tipus on s'ordenen de més petita a més gran, que seran la referència al moment actual. Tot seguit, per a cadascuna de les distàncies que resten a la matriu reduïda, es desen les posicions per desnormalitzar-les (així fan referència a la matriu completa, enlloc de a la reduïda), s'afegeixen a l'Array de coordenades per cridar recursivament *getMinPairs(coordenates)* i, finalment, s'extreuen abans de passar a la següent iteració del mateix nivell de la recursivitat.

L'obtenció del circuit eulerià: la ruta a seguir

Abans d'aplicar l'algorisme de Hierholzer per trobar el camí a seguir, cal realitzar preparatius per adaptar la confecció del programa a les condicions de l'algorisme. Com ha ocorregut en punts anteriors del projecte, el fet de tractar les arestes enlloc dels nodes provoca contratemps, igual que el fet d'haver de treballar amb índex cercant Array de String enlloc de cercar directament els objectes de tipus LatLng.

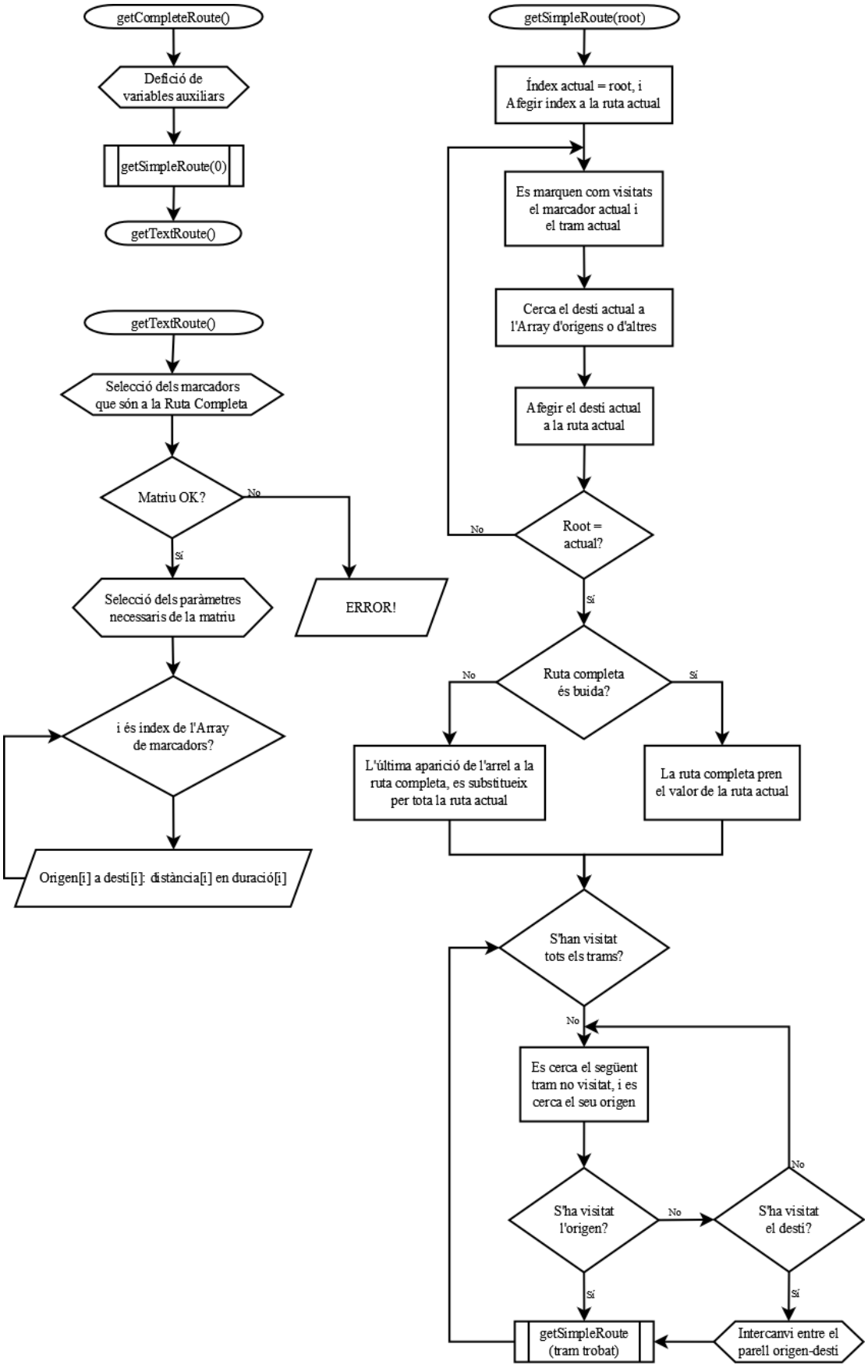
Així, per trobar el camí, al procediment *getCompleteRoute()* es creen diversos Array: a banda dels habituals d'orígens i destins, cal afegir-hi els homònims de tipus String, dos de tipus booleà per identificar els nodes i les arestes visitats, i un de marcadors, on es desa el conjunt dels punts en coordenades, tal qual, sense repeticions. Una vegada preparat el terreny, es procedeix a obtenir el/s circuit/s mitjançant el procediment *getSimpleRoute(base)*, i a exposar la ruta a seguir en format text mitjançant el *getTextRoute()*.

El procediment *getSimpleRoute(root)* és l'aplicació directa de l'algorisme de Hierholzer: es segueix un camí, punt a punt pels trams introduïts, fins que es torna a l'origen. Inicialment, es

desen en ordre els índex corresponen a cada marcador dins l'Array que conté el conjunt de punts. Aleshores, es cerquen arestes pendents de visitar i que tinguin algun dels extrems ja visitat, i aquest extrem passa a ser el paràmetre *root* en la següent crida recursiva del procediment.

En acabar, es cerca la darrera aparició del nou *root* a l'Array d'índex, i l'element se substitueix pel nou circuit trobat. Cal tenir en compte que, mentre que la crida recursiva és recurrent sempre que quedin arestes per visitar, dins de la cerca de cada circuit s'intenta allargar al màxim cadascun d'ells, tot evitant nodes visitats i sobretot arestes visitades mentre no sigui inevitable.

Per acabar l'execució, el procediment *getTextRoute()* fa servir el servei de la matriu de distàncies de Google Maps. Prèviament, amb l'Array d'índexs obtingut, s'han construït dos Array de coordenades de tal manera que el destí d'un origen és l'origen del destí següent i, una vegada obtinguda la matriu de distàncies, el procés és senzill: totes les dades que calia obviar en la crida durant la cerca de parells mínims, ara ens permeten obtenir adreces detallades, així com la distància i el temps que es trigarà entre punt i punt de la ruta.


getCompleteRoute()
Definició de variables auxiliars
getSimpleRoute(0)
getTextRoute()
getTextRoute()
Selecció dels marcadors que són a la Ruta Completa
Matriu OK?
No
Sí
ERROR!
Selecció dels paràmetres necessaris de la matriu
i és index de l'Array de marcadors?
Origen[i] a desti[i]: distància[i] en duració[i]
getSimpleRoute(root)
Índex actual = root, i Afegir index a la ruta actual
Es marquen com visitats el marcador actual i el tram actual
Cerca el desti actual a l'Array d'origens o d'altres
Afegir el desti actual a la ruta actual
Root = actual?
No
Sí
Ruta completa és buida?
No
Sí
L'última aparició de l'arrel a la ruta completa, es substitueix per tota la ruta actual
La ruta completa pren el valor de la ruta actual
S'han visitat tots els trams?
No
Es cerca el següent tram no visitat, i es cerca el seu origen
S'ha visitat l'origen?
No
S'ha visitat el desti?
No
Sí
Sí
getSimpleRoute (tram trobat)
Intercanvi entre el parell origen-desti

Conclusions

Aquest projecte pretenia aconseguir resoldre un cas similar al problema del carter xinès donats una sèrie de trams de carrers seleccionats per l'usuari a un mapa de Google. Un cop finalitzat el procés, l'aplicació hauria de proporcionar a l'usuari un recorregut que li permetés recórrer tots els trams, finalitzant a l'origen, emprant la ruta més curta possible.

El procés compta amb l'avantatge de que el servei de mapes de Google permet disposar de rutes entre les coordenades que, mentre que en models teòrics s'haurien de limitar a desfer un camí realitzat per tornar a un node anterior, en aquest cas l'usuari té la possibilitat de trobar una ruta alternativa una vegada s'apliquen determinats algorismes.

Per realitzar la implementació del programa, ha calgut desenvolupar nous algorismes, implementar-ne de ja coneguts i aprofundir en coneixements adquirits durant els estudis sinó aprendre'ls des de zero. Finalment, s'ha aconseguit implementar l'aplicació, i s'han realitzat les proves que s'han considerat oportunes per verificar-ne el funcionament. El resultat ha estat una aplicació funcional, fàcilment escalable, i compatible amb qualsevol navegador web compatible amb JavaScript.

L'únic límit insalvable, obviant les limitacions de hardware, és que el servei de matriu de distàncies de Google Maps limita aquesta matriu a 25 orígens o 25 destins, i a un màxim de 100 elements (encreuaments d'orígens amb destins). Aquest fet, juntament amb la versatilitat de la pròpia aplicació, obren un ventall de possibilitats per perfeccionar el projecte.

Les més significatives a simple vista son, d'una banda, l'optimització de l'aplicació per cada dispositiu que pugui suportar un navegador web compatible amb JavaScript (telèfons

intel·ligents, tauletes...) i, d'altra banda, ampliar l'algorisme perquè sigui capaç de superar les limitacions imposades pel servei de matriu de distàncies de Google Maps, ja sigui fragmentant les crides, agrupant coordenades per proximitat, o mitjançant d'altres alternatives.

Referències i bibliografia

Referències

1. **Algorithms and Theory of Computation Handbook**, CRC Press LLC, 1999, "Chinese postman problem", in *Dictionary of Algorithms and Data Structures*, Vreda Pieterse and Paul E. Black, eds. 2 September 2014. (última modificació a dia 2 de setembre de 2014)
   http://www.nist.gov/dads/HTML/chinesePostman.html

2. **Applied Combinatorics**, Roberts F.S.; Tesman B., (2009, 2nd ed.), CRC Press

3. **Grafs: fonaments i algorismes**, Basart i Muñoz J.M., 2a ed., Barcelona: Servei de publicacions de la UAB, 1998

4. **Modern Graph Theory**, Bollobás B. (1998), New York: Springer-Verlag

5. **Graph Theory**, Diestel R., 3a ed., Berlin, New York: Springer-Verlag, 2005

6. **Capítol 5: Grafs eulerians i grafs hamiltonians**, Matemàtica Discreta (2015), Enginyeria Informàtica, Escola d'Enginyeria, Universitat Autònoma de Barcelona

7. **Grafs eulerians: El problema del carter xinès**, article a la Universitat de Lleida (accedit a 18 de juny de 2015)
   http://griho2.udl.es/josepma/eines/eulerians/Edmonds.htm

8. **Introduction of Graph Theory**, Jun-ichi Y. (accedit a 18 de juny de 2015)
   http://jwilson.coe.uga.edu/EMAT6680/Yamaguchi/emat6690/essay1/GT.html

9. **Eclipse (software)**, article a la Wikipedia en castellà (última edició a dia 4 de juny de 2015)
http://es.wikipedia.org/wiki/Eclipse_(software)

10. **HTML**, article a la Wikipedia en castellà (última edició a dia 16 de juny de 2015)
http://es.wikipedia.org/wiki/HTML

11. **HTML5**, article a la Wikipedia en castellà (última edició a 17 de juny de 2015)
http://es.wikipedia.org/wiki/HTML5

12. **JavaScript**, article a la Viquipèdia en català (última modificació a dia 7 de maig de 2015)
https://ca.wikipedia.org/wiki/JavaScript

13. **Interfície de programació d'aplicacions**, article a la Viquipèdia en català (última modificació a dia 10 de juliol de 2014)
https://ca.wikipedia.org/wiki/Interf%C3%ADcie_de_programaci%C3%B3_d%27aplicacions

14. **Google Maps JavaScript API V3 Reference**, referència per Google Developers (darrera actualització a 12 de juny de 2015)
https://developers.google.com/maps/documentation/javascript/reference?hl=es

Bibliografia

Solutio problematis ad geometriam situs pertinentis, Euler L., Comment. Academiae Sci. I. Petropolitanae 8 (1736)

Ueber die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren, Hierholzer C. (1873), Mathematische Annalen

JavaScript and HTML DOM Reference, referència al lloc w3schools.com (accedit a 18 de juny de 2015)
http://www.w3schools.com/jsref/default.asp

Google API Tutorial, referència al web w3schools.com (accedit a 18 de juny de 2015)
http://www.w3schools.com/googleapi/default.asp

JavaScript reference, referència al lloc web Mozilla Developer Network (darrera actualització a 14 de gener de 2015)
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

Manual de algorítmica, Borrego Ropero R., Recio Domíngez D., Proyecto fin de carrera, ETS. Ingeniería Informática, Departamento Matemáticas Aplicadas I, Universidad de Sevilla (accedit a 18 de juny de 2015)
http://forja.rediris.es/frs/download.php/87/AlgoritmosDeGrafos.pdf

Sinopsi

Resolució del problema del carter xinès en un mapa Google:

En aquest Projecte Fi de Carrera s'implementa una aplicació web capaç de resoldre el problema del carter xinès sobre un mapa de Google en el que l'usuari ha introduït una selecció de trams de carrers connexos, mostrant-li la ruta més curta per recórrer tots aquests trams i retornar a l'origen. N'hi ha prou amb un navegador web compatible amb JavaScript i amb no excedir les limitacions d'alguns dels serveis que proporciona Google Maps.

Resolución del problema del cartero chino en un mapa Google:

En este Proyecto Fin de Carrera se implementa una aplicación web capaz de resolver el problema del cartero chino sobre un mapa de Google en el que el usuario ha introducido una selección de tramos de calles conexos, mostrándole la ruta más corta para recorrer todos estos tramos y retornar al origen. Hay suficiente con un navegador web compatible con JavaScript y con no exceder las limitaciones de los servicios que proporciona Google Maps.

Resolution of the Chinese postman problem on a Google map:

In this End of Career Project, the application able to solve the problem of Chinese postman on a Google map is implemented on which the user has introduced a range of connected sections of streets, showing him the shortest route to go all these sections and return to the source. It is enough with a web browser compatible with JavaScript and not to exceed the limitations of some of the services provided by Google Maps.