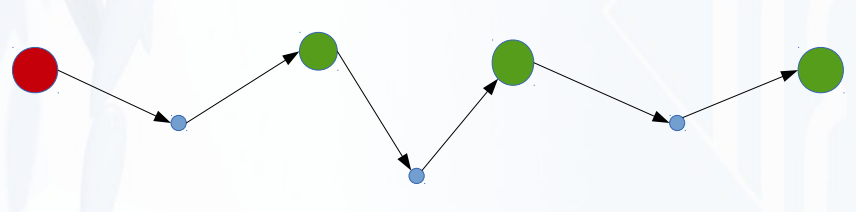
***1. Introducción al problema y dominio: Openstreetmap y  problema de búsqueda de estados.***

En esta práctica vamos a implementar un programa capaz de establecer una ruta para salir de un punto de una calle y visitar un conjunto de lugares.



Para ello seguiremos varias estrategias de búsqueda sobre un espacio de estados para conseguir un recorrido más corto u óptimo. Las estrategias que usaremos son:

* Profundidad
* Coste Uniforme
* Anchura
* Voraz
* A\*

Posteriormente compararemos los resultados de cada estrategia para ver la complejidad de cada una y saber cuáles son las más óptimas

El espacio de estados sobre el que se delimitará la búsqueda es un trozo de mapa. Como por ejemplo un trozo de Ciudad real:

Para poder trabajar sobre este trozo de mapa utilizaremos la herramienta de la página OpenStreetMap OSM. Con esta herramienta conseguiremos un archivo osm, que contiene todas las vías y nodos de ese mapa.

Para manejar este archivo osm utilizaremos la librería osm que nos permite seleccionar los nodos y vías que queramos.

Una vez hecha la selección crearemos un grafo, que es el espacio de estados, a partir del cual haremos las correspondientes búsquedas.

**2.Resolución del problema y estrategias implementadas.**

Definido ya el problema pasamos a explicar su resolucion, para resolver el problema usamos una busqueda basica, la cual nos sirve de tronco para implementar las distintas estrategias, esta se apolla en una busqueda incremental para poder disminuir las complejidades de las soluciones, haciendo que la solucion, si se encuentra en una parte mas superficial del arbol de busqueda se pueda encontrar sin recorrer todo el arbol.

La resolucion de nuestro problema se basa en:

- Un arbol de busqueda, en el cual se introduciran los nodos de busqueda

- Un grafo en el que se encuentran todos los nodos de OSM con lso que trabajaremos

- Una frontera en al que se guardaran los nodos de busqueda aun sin explorar

- Un espacio de estados en el que se encuentran todos los posibles estados de nuestro sistema

- Una funcion heuristica que asigna valor a los nodos de las estrategias informadas, esta calcula el camino en linea recta desde el nodo hasta el objetivo, de forma que nos “predice” cuanto cuesta llegar al objetivo de forma ideal. Como en nuestro sistema podemos tener mas de un objetivo, selecciona el coste ideal al objetivo mas lejano

- Un nodo incial en el que comenzar el problema y a partir del cual construir el arbol de busqueda.

Para resolver el problema seguiremos los pasos:

- Conseguimos mediante unas coordenadas fijas en el programa el mapa de OSM con el que trabajar

- Creamos el grafo en funcion de las vias y nodos de ese mapa de OSM

- Preguntamos la complejidad para crear el estado incial y conseguir los objetivos

- Creamos el espacio de estados en funcion de las coordenadas fijas del programa

- Creamos el problema en funcion del estado inicial y el espacio de estados

- Preguntamos la estrategia a usar, la profundidad maxima y el incremento de la profundidad para la busqueda incremental

- Tomamos un dato de tiempo para marcar el inicio de la busqueda

- Ejecutamos la busqueda con todos los datos recogidos y creados

- Se crea la frontera y se le inserta el nodo incial

- Se expande un nodo de la frontera, se insertan sus hijos y se comprueba si es solucion, mediante un while que comprueba si la frotnera esta vacia y si hemos lelgado a la profundidad maxima, para, en caso de que no haya, poder parar si no hay solucion.

- Se repite el paso anterior hasta llegar a la profundidad maxima.

- Si no se ha encontrado solucions e incrementa la profundidad, si se llega a la maxima, se decidira que no hay solucion

- Se crea la solucion, en caso de que haya, haciendo el camino desde el objetivo hasta el padre.

- Se toma un dato de tiempo para marcar el final de la busqueda de forma que podamos calcular la compleidad temporal de la busqueda

- Se toma el dato contador de la clase problema para saber cuantos nodos hemos explorado para encontrar la solucion y poder expresar la complejidad espacial

- Se escribe la solucion en un txt con todos lso nodos ha recorrer y sus valroaciones y costes.

- Se escribe la solucion en un gpx para pdoer cargarlo y visualizar la ruta.

La principal unica diferencia en las estrategias ses el valro que se les da a los nodos de busqueda, dado que nuestra frontera es una cola con prioridad, esta se ordenara en funcion del valor de los nodos de busqueda.

Las distintas estrategias implementadas son:

- Busqueda en anchura, la cual se caracteriza por valorar lso nodos según su profundidad, es decir, todos los nodos de una misma profundidad tienen el mismo valor.

- Busqueda de coste uniforme, que asigna valro a lso nodos según cuesta lelgar a ellos, es decir, no tiene coste por si solo u nodos, si noq ue el coste de todos es el coste inherente a esa accion y estado.

- Busqueda en profundidad, asigna el valor a los nodos en base a su profundidad, como en la anchura pero con distinto coste, en la anchura su objetivo es visitar a los nodos por niveles, aquí se le asgina mas valor a los nodos mas profundos.

- Busqueda voraz, asgina valro a los nodos en funcion de la heuristica, y solo ella, por lo que prioriza los nodos idealmente mas cercanos al objetivo.

- Busqueda A\*, asigna valor a los nodos en funcion de la heuristica mas el coste de alcanzar dicho nodo, ya que un nodo puede ser idealmente cercano al objetivo pero muy costoso de alcanzar, de esta forma siempre encontramso el camino mas corto al objetivo

***3.Estructuras y clases definidas su justificación.***

Clase principal

En la clase principal(main) realizaremos la lectura del archivo osm. A partir de la lectura crearemos una tabla-hash con los nodos seleccionados con su respectiva longitud y latitud.

Una vez hecho esto, crearemos el grafo a partir de la tabla-hash, que nos servirá también para ponerle peso a las aristas.

En la clase principal es donde haremos el algoritmo de búsqueda para obtener la solución. Una vez obtenida la solución, creamos una función para obtener la función en un archivo gpx.

Clase EspacioEstados

En esta clase estableceremos como atributos la longitudes y latitudes máximas y mínimas del espacio de estados.

En esta clase podremos saber si un estado es válido , si se ha llegado al objetivo del problema y también podemos obtener los sucesores de un nodo. Para saber los sucesores utilizaremos el grafo utilizado en la clase principal.

Clase Estado

En esta clase únicamente crearemos los atributos del objeto estado: localización, objetivos, latitud y longitud.

Clase nodoBusqueda

En esta clase definimos los atributos del objeto nodo: id, padre, estado, costo, acción, profundidad, valor.

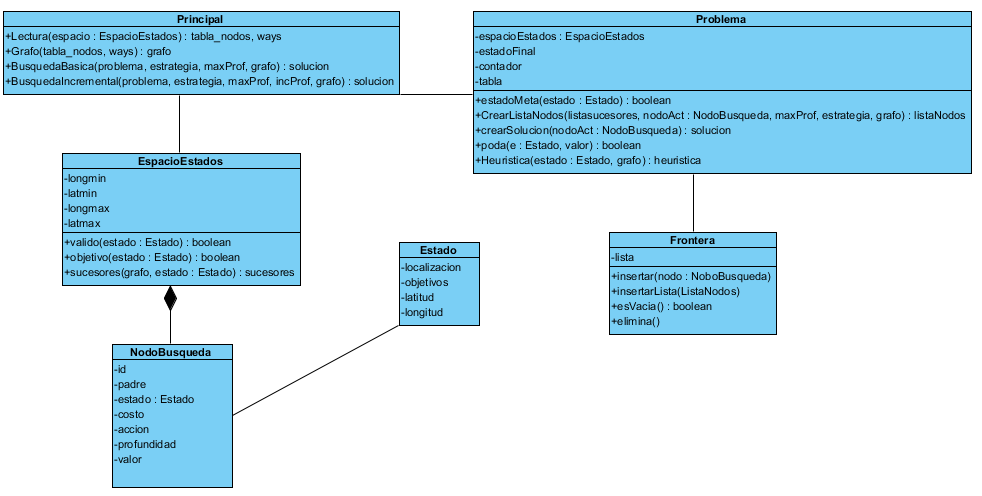
Clase Frontera

En esta clase creamos la lista Frontera e implementamos las funciones para crearla y modificarla.

Clase Problema

En esta clase definimos los atributos del problema: Espacio de estados y estado inicial.

Como funciones, podemos saber si un estado es el estado objetivo del problema, podemos crear la lista de nodos que añadiremos a la frontera, creamos la función para crear la solución, implementamos la función poda y la función para calcular la heurística.



**4. Elementos especialmente importantes de nuestra implementación.**

Aparte de los elementos principales y propios de las busquedas y sus estrategias, tambien tenemos otros elementos importantes como:

- Una tabla hash en la que guardamos los nodos del OSM para poder crear el grafo con las coordenadas de cada nodo de OSM, la cual es borrada tras la creacion del grafo

- Una funcion lectura que dadas unas coordenadas de unas esquinas, nos descarga un mapa de OSM que contiene los nodos y calles con las que trabajaremos

- Una funcion grafo que apollandose en las vias y lso nodos que contienen, crea el grafo que simbolizara nuestro mapa, recorriendo con un for la lsita de vias y buscando lso nodos que contiene cada via en la tabla hash para insertar los nodos unidos por la via.

- Una funcion menu que nos permite interactuar con el usuario pidiendole la complejidad de problema que quiere (selección de nodos de partida y objetivos), la profundidad maxima que desea, el incremento con el que alcanzar dicha profundidad maxima durante la busqueda, y la estrategia con la que desea encontrar solucion al problema.

- Una funcion poda la cual nos permite acortar las busquedas ya que evita la exploracion de nodos ya explorados, basandose en una tabal hash cuya clave el el estado del nodo y contiene el valro del nodo, esta funcion evitara al exploracion de nodos ya repetidos y peor valorados.

- Un contador en la clas eproblema que nos cuenta uantos nodos han sido explorados en la busqueda y asi poder calcular la complejidad espacial

El elemnto mas importante de nuestr apractica es el uso de las librerias para python networkx y gpxpy, las cuales debera descargar e instalar en su computador para pdoer ejecutar la practica, dado que usamos python 3, debera instalarlas con pip3 y el nombre de la librería.

**6.** **Opinión personal de la práctica.**

La practica de forma grupal nos ha parecido interesante, dado que nos permite crear un util y algo increible como es un GPS, por otro lado es dura de crear y completar a que es clara con los objetivos pero poco clara con como conseguirlos.

Conforme a teoria pienso que nos enseña e implementa de una forma bastante buena la teoria del primer parcial.

**7.** **Manual de usuario**

Nuestro programa no dispone de interfaz grafica por loq ue se ejecutara e ineractura mediante consola

Dado que es un programa en python la forma de ejecutarlo e inciarlo es mediante la ejecucin del scrip Principal situandonos en la capeta donde se ubica con cd <ruta de carpeta> y una vez situados usar el comando:

python3 Principal

Tras ello el unico uso que podreos hacer de ella se notificara por la pantalla loq ue debe hacer, ya que debera interactuar con el programa para indicar la complejidad que desea, la estrategia a usar, la profundidad maxima y el incrmento de esta para la busqueda incremental.