Un dibujo de una cara feliz

Descripción generada automáticamente con confianza bajaInterfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Un dibujo de una persona

Descripción generada automáticamente con confianza baja

**Universidad Politécnica de Madrid**

Escuela Técnica Superior de Ingenieros Informáticos

Proyecto de diseño de una aplicación que halle el trayecto óptimo entre dos estaciones de metro.

Caso particular: estaciones de metro de Lyon

CURSO 2023-2024

GRUPO 8:

* Jaime Perán De Montes-Jovellar 200274
* Marcos Ayuso Camarena 200369
* Alberto González López 140332
* Lucas Panfil Balmaseda 180342
* Francisco Chicote Martín 200256

[Descripción general del proyecto. 3](#_Toc152610526)

[Estructura de información. Datos. 4](#_Toc152610527)

[Primer conjunto de datos. Ubicación. 4](#_Toc152610528)

[Segundo conjunto de datos. Trasbordos 4](#_Toc152610529)

[Tercer conjunto de datos. Horas. 4](#_Toc152610530)

[Cuarto conjunto de datos. Distancias. 4](#_Toc152610531)

[Quinto conjunto. Líneas. 4](#_Toc152610532)

[Algoritmo utilizado, A\* 5](#_Toc152610533)

[Cálculos efectuados -> f\*(n) = g\*(n) + h\*(n) 5](#_Toc152610534)

# Descripción general del proyecto.

El objetivo del proyecto ha consistido en el diseño de la aplicación que permita hallar el trayecto óptimo entre dos estaciones del metro de Lyon teniendo en cuenta que, para llegar de una estación A a una estación B, pueden concurrir diferentes circunstancias que modifiquen ese trayecto óptimo como pueden ser, la existencia de un trasbordo entre estaciones, el horario en el que se esté realizando el trayecto, y la heurística calculada para esos dos puntos del trayecto.

Imagen que contiene Mapa

Descripción generada automáticamente

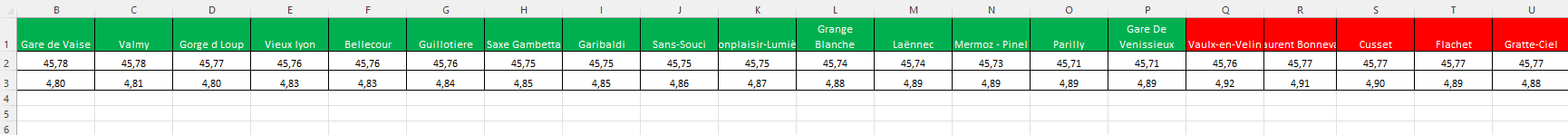
# Estructura de información. Datos.

Para la realización del trabajo se han tenido en cuenta diversos parámetros que han constituido la base de hechos de información de la que se ha servido la implementación para el cálculo y la evolución de los grafos.

Se ha utilizado un formato de tabla en la que hemos recogido la información estructurándola en pestañas (conjuntos de datos homogéneos), filas y columnas.

### Primer conjunto de datos. Ubicación.

Se han recogido las ubicaciones en coordenadas de grados decimales(DD), ej. Gare de Vaise (45,7835154061337 ; 4,80483329334771).



### Segundo conjunto de datos. Trasbordos

Se han identificado y, otorgado un valor numérico que identifica la penalización de uso, los trasbordos posibles en función de los requisitos marcados en el mapa del metro de Lyon

Tabla, Calendario

Descripción generada automáticamente

### Tercer conjunto de datos. Horas.

Se ha creado un mapa horario en el que se reflejan los retrasos producidos por tiempo en hora punta con una penalización determinada en función de la estación de metro, de tal forma que en una estación concurrida puede tener una confluencia mayor en hora punta y por tanto mayos penalización que otra estación menos transitada.

Calendario

Descripción generada automáticamente

### Cuarto conjunto de datos. Distancias.

En este caso se ha registrado el valor de todas las distancias entre las estaciones contiguas y aquellas a las que se puede acceder mediante trasbordo directo. En el caso de no ser contiguas o no tener trasbordo los valores se han informado a cero.

Calendario

Descripción generada automáticamente con confianza baja

### Quinto conjunto. Líneas.

Se establecen el conjunto de paradas que tiene cada línea y se identifican con su color del mapa.

Tabla

Descripción generada automáticamente

# Algoritmo utilizado, A\*

Se escoge el primer nodo origen. A continuación, “*El mejor el primero”,* se escoge el nodo mas prometedor, de acuerdo a la función heurística asignada para cada una de las estaciones.

Se utiliza una función de reordenamiento de los nodos de la llista abierta

f\*(n) será el coste real de un camino óptimo desde el nodo origen a un nodo meta, restringido a que dicho nodo pase por el nodo n.

g será el coste del camino que va desde el nodo inicial origen al nodo meta en el árbol de búsqueda, este camino es el de coste más bajo encontrado hasta el momento.

h será nuestra función heurística.

Buscamos con esta búsqueda A\*:

COMPLETITUD.

Un algoritmo es completo si termina con una solución si ésta existe.

ADMISIBILIDAD.

Un algoritmo es admisible si asegura encontrar una solución óptima, si esta existe. A\* es admisible.

EFICIENCIA.

Un algoritmo A1 es más eficiente que otro A2 si A1 está más informado que A2, así cada nodo desarrollado por A1 también es desarrollado por A2. A2 desarrolla, al menos tantos nodos como A1

OPTIMIDAD.

Un procedimiento es óptimo, dentro de un conjunto de procedimientos, si es más eficiente que todos los elementos de dicho conjunto.

## Cálculos efectuados -> f\*(n) = g\*(n) + h\*(n)

Ésta establece su valor heurístico h(n) de acuerdo a las coordenadas de la estación (DD:X,Y) y la velocidad media del metro de Lyon y cualquier otro punto con conexión.

g(n) se calcula en función del recorrido, el destino seleccionado, las distancias por estaciones e incrementado su valor con h(n) para obtener su f(n).

Se utilizan dos estructuras de ordenamiento de los nodos de acuerdo con el algoritmo A\*, una lista abierta, para nosotros nodosAbiertos [] y lista cerrada, nodosCerrados.

Cálculo del camino más corto

En base a las estructuras anteriormente mencionadas, se utiliza una estructura recursiva que irá construyendo el camino a recorrer tratando la pila de nodos abiertos y cerrados. El procedimiento en la implementación sería el que sigue:

1. recibe un nodo como entrada y lo trata según este en el grafo (la primera vez recibe el nodo inicial)
2. si no está en nodos abiertos no cerrados se añade a la lista de nodos abiertos
3. Se calcula el coste de la arista, entendida ésta como el recorrido al siguiente nodo, y se guarda el valor como CosteActual
4. A continuación se calcula el coste al siguiente nodo. Si el coste es cero o mayor que el coste calculado anteriormente nos quedamos con los valores almacenados en CosteActual.
5. Mientras haya elementos en nuestra estructura de nodosAbiertos seguimos iterando. El siguiente nodo de nodosAbiertos lo metemos en la lista de nodosCerrados e iteramos.

La lista de nodos abiertos está vacía, se ha creado el camino en función de los elementos de la lista de nodos cerrada, y se ha construido el path en pathAux para poder “pintar” el recorrido en la interfaz del usuario.

# Caso de uso

En este apartado se presenta un caso de uso de funcionamiento