Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Departamento de Inteligencia Artificial

(Inteligencia Artificial)

CAMPUS
DE EXCELENCIA
INTERNACIONAL

“Cálculo del trayecto óptimo entre dos estaciones del metro Toshkent Metropoliteni”

“Cálculo del trayecto óptimo entre dos estaciones del metro Toshkent Metropoliteni”

Autores *(Grupo 7)*:

Víctor Nieves Sánchez	*X150375*
Daniel Morgera Pérez	*X150284*
Alejandro Carmona Ayllón	*X150251*
Raúl Prieto Acedo	*X150216*

Resumen:

La finalidad de este proyecto es diseñar una App para hallar el trayecto óptimo entre dos estaciones del metro Toshkent Metropoliteni.

Para el cálculo del trayecto mas corto entre dos estaciones utilizamos el algoritmo A*, el cual nos proporciona una optimización de caminos de coste mínimo en grafos de decisión.

Fecha:

17 de Diciembre de 2018, Campus Montegancedo (Boadilla del Monte), Universidad Politécnica de Madrid.

Índice:

1. Objetivos:

Diseñar una aplicación para hallar el trayecto óptimo entre dos estaciones del metro de *Taskent* (solo se contemplaran las líneas y estaciones del mapa anexo), teniendo en cuenta distintos parámetros como pueden ser el número de transbordos, longitud de los mismos, la hora a la que se realiza el trayecto, el número de tramos de escaleras que tenemos que utilizar, etc.

Para el cálculo del mejor camino entre dos estaciones se utilizarán algoritmos de búsqueda en la optimización de caminos de coste mínimo en grafos de decisión. Utilice alguno de los algoritmos *A**, *IDA** o *BIDA**.

La elección de cualquiera de los algoritmos explica que en su función de evaluación se pueden representar las distintas características y particularidades de cada estación (si tiene o no transbordo, longitud del mismo, tramos de escaleras, horario en el que se realiza el transbordo, etc.).

2. Metodología de trabajo:

Los cuatro miembros se se reunieron para decidir que algoritmo usar (al final se eligió el *A**), así como el lenguaje a usar y la interfaz gráfica y el diseño de la aplicación.

Una vez todo contemplado se volvió a reunir el grupo para configurar el mapa, es decir, decidir el tiempo entre estaciones, los transbordos, así como la forma de codificar el mismo.

Teniendo todo lo anterior, se asignaron una serie de días los cuales se volvería a juntar el grupo y trabajarían en la aplicación en sí.

3. Introducción:

El algoritmo de búsqueda *A** se clasifica dentro de los algoritmos de búsqueda en grafos. Presentado por primera vez en 1968 por *Peter E. Hart*, *Nils J. Nilsson y Bertram Raphael*, el algoritmo *A** encuentra, siempre y cuando se cumplan unas determinadas condiciones, el camino de menor coste entre un nodo origen y uno objetivo.

*A** es un algoritmo completo: en caso de existir una solución, siempre dará con ella.

Si para todo nodo n del grafo se cumple $g(n)=0$, nos encontramos ante una búsqueda voraz. Si para todo nodo n del grafo se cumple $h(n)=0$, A* pasa a ser una búsqueda de coste uniforme no informada.

Para garantizar la optimización del algoritmo, la función $h(n)$ debe ser heurística admisible, esto es, que no sobrestime el coste real de alcanzar el nodo objetivo.

De no cumplirse dicha condición, el algoritmo pasa a denominarse simplemente *A*, y a pesar de seguir siendo completo, no se asegura que el resultado obtenido sea el camino de coste mínimo. Asimismo, si garantizamos que $h(n)$ es consistente (o monótona), es decir, que para cualquier nodo n y cualquiera de sus sucesores, el coste estimado de alcanzar el objetivo desde n no es mayor que el de alcanzar el sucesor más el coste de alcanzar el objetivo desde el sucesor.

4. Desarrollo de la práctica:

4.1. Lenguaje usado:

Ya que durante nuestra carrera universitaria el lenguaje que hemos aprendido desde el principio es Java, hemos decidido que seria la manera más cómoda de implementar este proyecto.

A parte de las librerías básicas proporcionadas directamente por java, usamos las siguientes, que se podrán encontrar ya en la carpeta *lib* del proyecto.

- *hamcrest-core-1.3.jar*
- *javabdd_repackaged-1.0b2.jar*
- *junit-4.12.jar*
- *net-datastructures-4-0.jar*

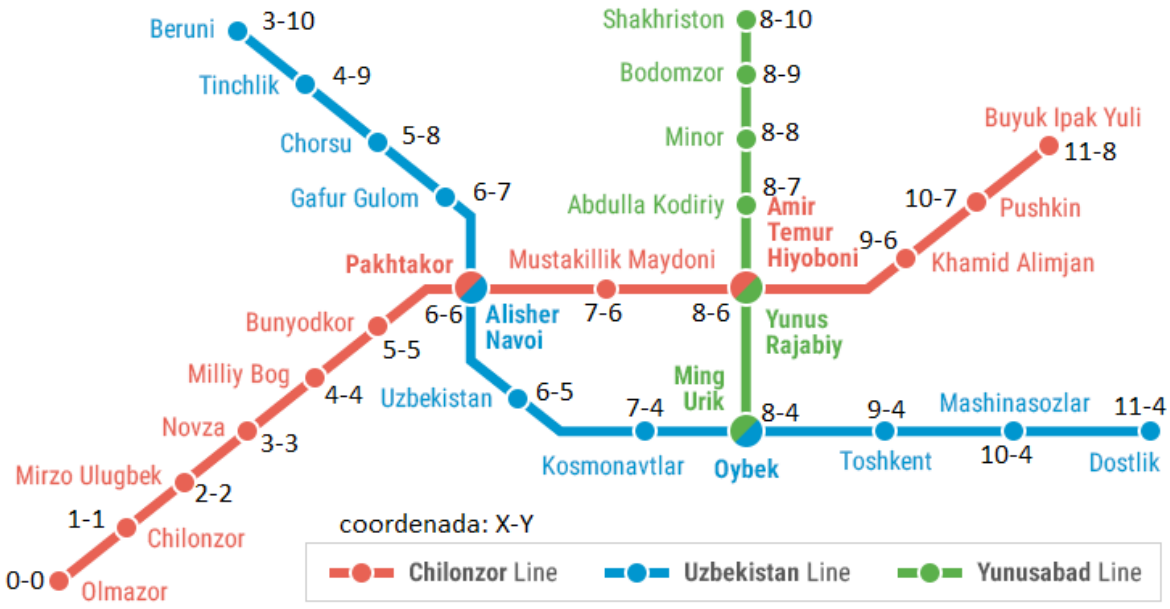
4.2. Algoritmo utilizado:

Como ya se ha nombrado en anteriormente, se ha utilizado el algoritmo *A**.

4.3. Características:

Para implementar el área de búsqueda hemos dividido el mapa en cuadrantes de modo que cada estación tenga sus propias coordenadas cartesianas *x* e *y*. En el siguiente mapa, podemos ver el mapa sobre el que se realiza el ejercicio así como las coordenadas asignadas a cada estación.

Hay que marcar que, las estaciones con transbordos, que tienen dos nombres, se ha optado por tener un nombre conjunto, así como que la *entrada* y *salida* de esas estaciones están asignadas solo a una línea. Por ejemplo, la estación *Pakhtakor* y la estación *Alisher Navoi* se han considerado como una sola con nombre *Pakhtakor-Alisher Navoi*, y la *entrada* y *salida* de esa estación está asignada a la línea *Roja (Chilonzor Line)*.



4.4. Tiempos entre estaciones:

Consideremos los tiempos de transbordo duraran 1 minuto, y los tiempos entre estaciones los consideramos en función de la distancia que hay entre ambas estaciones, usando Google Maps para ver estas distancias, y decidiendo Km por minuto, es decir, una distancia menor de 1 Km equivale a 1 minuto de trayecto, de 1 Km a 2 Km equivale a dos minutos, de 2 Km a 5 Km equivale a 4 minutos.

4.5. Pasos dados para la ejecución:

Para que la ejecución sea más sencilla, se ha creado un ejecutable *.jar*, el cual solo es necesario ejecutarlo en un terminal mediante el mandato:

java -jar MetroToshkent.jar

También puedes hacerlo desde el código fuente ejecutando la clase *Ventana.java*. Teniendo en cuenta que deberás tener instaladas las mencionadas librerías para que todo funcione correctamente.

5. Resultados:

Con este trabajo, hemos aprendido a como implementar un algoritmo de búsqueda entre dos puntos de un grafo y también se ha mejorado en cuanto a temas de programación orientada a objetos, diseño y programación de interfaces gráficas.

6. Dificultades encontradas:

Una de las dificultades que encontramos fue incluir los transbordos tanto en el mapa como en los tiempos para ir de una estación a otra. De hecho, tuvimos un problema con un transbordo en concreto, el cual resolvimos fácilmente porque descubrimos que nos faltaba incluir una línea de las coordenadas implementadas. Por culpa de este fallo uno de los recorridos no detectaba el transbordo y tenia que avanzar a la siguiente estación que si tenía implementada la dirección del transbordo para que llegara a su destino, provocando que hubiera 3 movimientos adicionales. Este error ya fue corregido y realiza todos los transbordos perfectamente.

Otra de las dificultades encontradas fue la realización de la interfaz gráfica y conectar los datos en tiempo de ejecución con lo calculado por el algoritmo.

7. Bibliografía:

https://www.google.com/maps

http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

https://www.codeproject.com/Articles/9880/%2FArticles%2F9880%2FVery-simple-A-algorithm-implementation

8. Anexo:

Mapa sobre el cual se ha realizado la práctica: