DOCUMENTACIÓN TAREAS 1 Y 2 SISTEMAS INTELIGENTES

Rubén Montero Martín, Pedro Antonio Ramírez de Verger, César Sánchez Jerez

Tarea 1

La resolución de la tarea 1 la hemos llevado a cabo por medio de la librería igraph.

Empezamos el desarrollo en java, pero ante la complejidad de trabajar con graphml en este lenguaje, decidimos hacer caso a las recomendaciones y cambiarnos a Python y buscar alguna librería.

La primera opción fue pygraphml, pero no nos permitía trabajar de la forma que requeríamos e investigamos igraph.

El código de esta tarea es:

Aclaración: los métodos ***vs*** y ***es*** son pertenecientes a la librería y se encargan de proporcionar la secuencia de vértices y aristas del grafo respectivamente, y lo utilizaremos con bastante frecuencia.

-El constructor al cual se le pasa un archivo graphml

-El método perteneceNodo en el que comprobamos si el nodo que se le pasa está dentro de todos los nodos del grafo.

-El método posicionNodo que se encarga de, tras comprobar si pertenece al grafo con perteneceNodo, accede a las etiquetas x e y del vértice que se le proporciona para saber su posición.

-El método adyacentesNodo, que tras comprobar si el nodo proporcionado pertenece al grafo, obtiene los ids, autogenerados por la librería, de las aristas adyacentes al nodo, y a partir de ahí obtenemos los vértices que se encuentran en el extremo opuesto de la arista y la distancia entre ellos.

Tarea 2

Para esta entrega hemos creado las clases Estado, EspacioEstados, Problema, NodoArbol y Frontera.

La clase Estado se encarga de obtener el nodo en el que nos encontramos en dado momento y los nodos que quedan por visitar. (Nos falta la codificación md5).

La clase EspacioEstados se encarga de generar los sucesores de un estado en concreto por medio del método sucesores, el cual obtiene las acciones posibles desde el estado actual, el coste de dicha acción y el destino que servirá para generar el próximo estado.

La clase Problema obtiene el estado y el espacio de estados del fichero json proporcionado y discierne si ese estado es objetivo.

La clase nodoArbol se encuentra en este momento incompleta.

La clase Frontera se encarga de generar una lista ordenada, según un criterio dado, de nodos a visitar.

https://github.com/PedroRamirezVerger/BC-07