ESTRUCTURAS UTILIZADAS

El programa se compone de las siguientes clases:

- -Principal.py
- -Problema.py
- -nodoBusqueda.py
- -frontera.py
- -Estado.py
- -EspaciodeEstados.py
- -distancia.py

Para crear el Espacio de Estados hemos tenido previamente que crear un objeto Estado en la clase Estado. El objeto Estado esta formado por los atributos localización, que son las coordenadas(longitud y latitud) del estado, y objetivo por alcanzar.

El Espacio de Estados está implementado en la clase EspaciodeEstados. Esta clase contiene como atributos:

- -El Espacio de Estados en sí, el cual es el grafo que creamos en la tarea anterior.
- -Las acciones que nos llevan de un estado a otro, las cuales las implementamos con una lista con las posibles opciones de movimiento: Norte(N),Sur(S),Este(E),Oeste(O),Noreste(NE), Noroeste(NO),Sureste(SE), Suroeste(SO).
- -El estado actual.
- -El estado final/objetivo.

El Espacio de Estado contiene como operaciones:

- -EsValido(): Esta operación sirve para saber si un estado pasado por parámetros es adyacente al estado actual.
- -EsObjetivo(): Esta operación sirve para saber si un estado pasado por parámetros es el estado Objetivo del Espacio de Estados.
- -Sucesores(): Esta operación la utilizamos para saber los sucesores a un estado dado. Cada sucesor esta formado por la acción para llegar a él , el estado adyacente y el costo para llegar a ese estado. Vamos añadiendo a una lista vacía inicialmente de sucesores, sucesor por sucesor. Para ello obtenemos los nodos adyacentes al nodo dado por parámetros y por cada nodo adyacente calculamos su acción para llegar a él, para la cual restamos las longitudes y

latitudes entre un nodo y otro, y dependiendo del resultado sabemos el movimiento(accion). Posteriormente obtenemos el peso de la arista para saber el coste y añadimos todo a la lista de sucesores.

Las estructuras del espacio de búsqueda son:

-El nodo de búsqueda: lo definimos como un objeto en la clase nodoBusqueda con los atributos: idNodo, padre, estado, costo, acción, profundidad y valor.

-La frontera: implementamos una lista ordenada de nodos. Utilizamos la librería Queue para implementar una cola con prioridad. Creamos un metodo CrearFrontera donde creamos la cola. También utilizamos las operaciones de Insertar(), donde metemos un nodo a la lista con la función "put()", Eliminar(), donde devolvemos el primer nodo de la lista y lo eliminamos con la funcion "pop()".Y la ultima operación es la de esVacia() donde comprobamos si la lista está vacia.

Finalmente, implementamos el problema en la clase Problema. Esta clase tiene como atributos el Espacio de Estados, el estado inicial y el estado objetivo.

COMPONENTES: -Eduardo Martín Izquierdo.
-Javier Oliver Diaz-Alejo.
-Eduardo Parra Mazuecos.