DOCUMENTACIÓN

PARTICIPANTES:
-Eduardo Martín Izquierdo
-Javier Oliver Diaz Alejo
-Eduardo Parra Mazuecos

ESTRUCTURAS UTILIZADAS Y CLASES

HEURÍSTICA

Para la estrategia 'A' y la estrategia 'voraz' hemos tenido que implementar la heurística. La heurística la hemos implementado en la clase “Problema” como una función, a la cual le pasamos el estado del nodo actual y el grafo del problema. En la función heurística calculamos la distancia del nodo actual a cada uno de sus estados objetivos, utilizando la clase distancia que se nos proporciona , guardando cada coste en una lista, de la cual obtenemos el máximo valor(heurística).

PODA

En la implementación de la poda, hemos incluido la función poda en la clase “Problema” también. A este clase le pasamos como parámetros el estado del nodo sucesor y el valor que tendrá este. En la clase “Problema” creamos una tabla-hash que tiene como “keys” cada estado y como contenido de cada estado su valor. Entonces cuando la función recibe ese estado, se verifica si ese estado está en la tabla-hash. En caso de que no esté en la tabla-hash lo incluimos con su valor y no podamos. En el caso de que esté en la tabla debemos comparar su valor con el valor que tenga en la tabla. Si el valor que se le pasa por parámetros es menor que el de la tabla-hash , no se poda y se actualiza el valor de la tabla-hash, en caso contrario se podaria.

Para podar el nodo, utilizamos esta función en la función “CrearListaNodos” en la que al elegir una estrategia u otra, calculamos el valor del nodo sucesor y utilizamos la función “poda”, que dependiendo de este valor, añadiremos el nodo a la frontera o no.

GPX

La solución del algoritmo la extraemos en un fichero GPX. Para implementar este fichero utilizamos la librería “gpxpy”. En este fichero añadimos el nombre de este, los “waypoints” de la ruta y los segmentos “track” para cada nodo de la solución.

COMPLEJIDAD ESPACIAL Y TEMPORAL

Para calcular la complejidad temporal, utilizamos la librería “time” para calcular el tiempo de ejecución con la función “time()”

Y en el calculo de la complejidad espacial, contamos todos los nodos que hemos añadido a la frontera, desde el nodo raiz hasta el nodo destino.