Prácticas de Robótica
Práctica 4 - Planificación.

Planificación y búsqueda de rutas a partir de un mapa del entorno.

Objetivos

1. Implementar las funciones necesarias para **cargar un mapa y navegar por él**. En concreto:
   a. Cargar un mapa y calcular la matriz de pesos/costes para ir de un punto a otro del mapa.
   b. Planificar camino óptimo para ir de un punto a otro utilizando la matriz de pesos.
   c. Recorrer dicho camino óptimo con el robot, paso a paso.
   d. Identificar nuevos obstáculos no representados en el mapa y *replanificar* en consecuencia.
2. Mantener en todo momento la **odometría del robot actualizada**

NOTAS:

- El formato de mapa será el mismo que se utilizará después en el trabajo de la asignatura, así que no lo cambieis si no quereis tener problemas de compatibilidad.
- En moodle encontrareis los ficheros base para esta práctica con los requisitos de las funciones a implementar. **Para evaluar las entregas se asume que el "signature" de las funciones requeridas es como en dichos ficheros.**

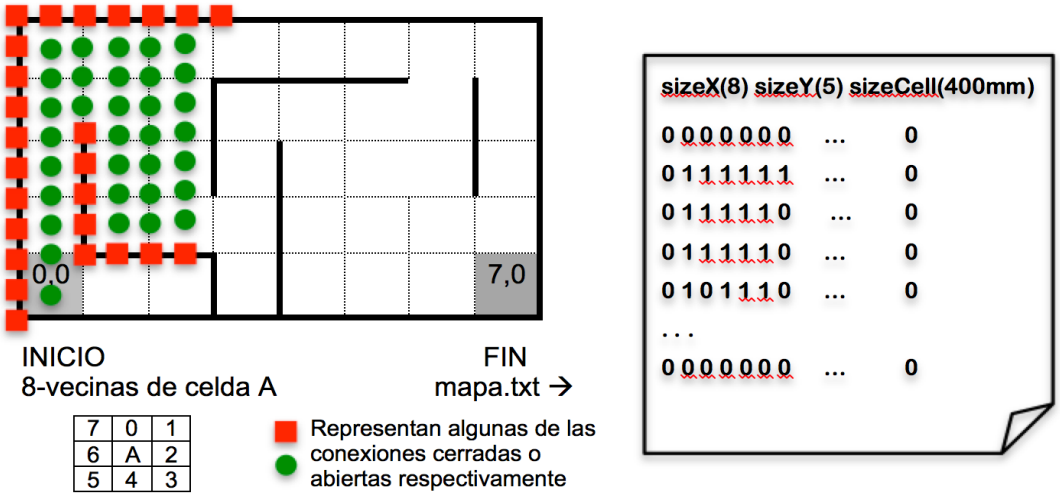
Descripción

1. Carga y visualización del mapa:

Probar example_map.py (moodle), que carga y visualiza un mapa y ejemplos de posiciones de un robot utilizando las funciones disponibles en **mapLib.py**. Los elementos básicos son:
- `connectionMatrix`: matriz de booleanos que representa el mapa. Indica qué conexiones están abiertas desde cada celda a cada una de sus 8 vecinas (una conexión puede estar anulada por distintos motivos: obstáculo, pared, límite del mapa).
- `isConnected(x,y,1)`: `True` si desde celda (x,y) se puede ir a su vecina 1.

Cualquier otra opción de representar las celdas y las conexiones entre ellas es válida, **solo si es totalmente compatible** con el formato de fichero de texto donde se almacena el mapa.

Ejemplo de un mapa y su fichero de configuración (datos básicos de las celdas, "grid" de 5x8 celdas de 400 mm, en la primera fila, y el resto son datos de la matriz de conexiones, matriz 11x17 en este ejemplo):



2. Planificar rutas y moverse por el entorno representado por el mapa:

Diseñar e implementar las siguientes funciones a partir de las plantillas que tenéis en moodle dentro de mapLib.py:

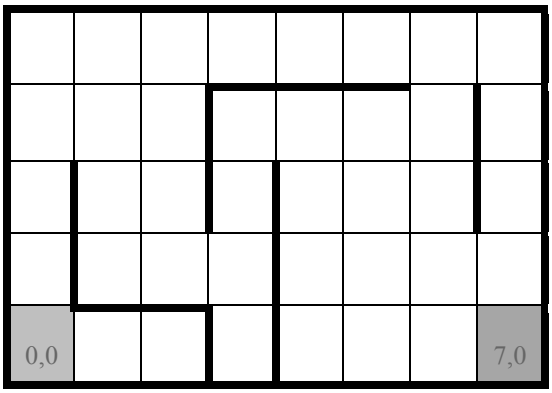
`path = planPath(x_ini, y_ini, x_goal, y_goal)`: implementar un algoritmo de planificación (NF1) para establecer la ruta (path) que debe seguir el robot por un mapa dado, desde la posición inicial (ini) a la salida (goal).
- A la función se le puede pasar cualquier casilla como origen o final.
- Dentro de **`planPath`** hay que implementar una función que calcule los pesos según el algoritmo NF1, llamada **`fillCostMatrix`**

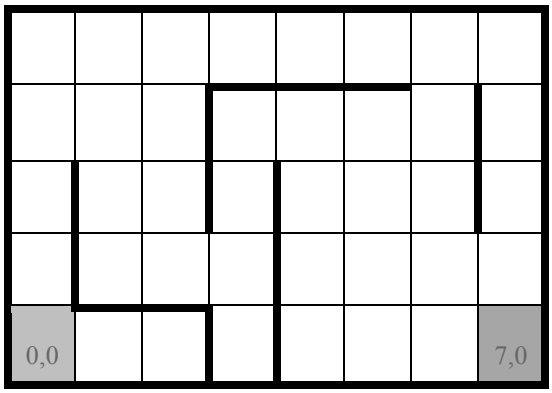
`go(x_goal, y_goal)`: implementar una función que mueva al robot desde la posición actual (posición marcada por la odometria) hasta la posición final (goal) que se le pide. **Recomendación:** esta función representa un movimiento "básico" (e.g., avanzar una celda), para recorrer todo el camino planeado, llamaremos varias veces a esta función dentro del bucle principal.

`detectObstacle()` & **`replanPath()`**: Una vez que lo anterior se ejecute correctamente, añadir la funcionalidad de detectar obstáculos y evitarlos implementando estas funciones.

Trabajo PREVIO (se evaluará al entrar a la práctica)

- Rellenar la matriz de pesos del algoritmo de planificación NF1 en este mapa para ir del (**x1, y1**) al (**x2, y2**).
  Utilizar distintos x, y según el número de vuestro grupo (el número del grupo está en moodle):
  G1: (0,4) (4,0); G2: (0,0)(7,0); G3: (1,4)(4,4); G4: (0,4)(4,1);
  G5: (7,0) (1,0); G6: (7,0)(4,3); G7: (1,1)(7,1); G8: (7,0)(1,1);
  Rellenar los dos casos si se os ocurren **diferentes soluciones**.





- Escribir el pseudocódigo/esquema/algoritmo en python para las funciones (podeis fijaros en los esquemas sugeridos en los ficheros de ayuda de moodle para arrancar):
  V0: `findPath(x1,y1, x2,y2), fillCostMatrix(), go(x,y)`
  V1: `detectObstacle(), replanPath()`

Evaluación

- **Entrega del código y LOG+Plot de odometría de trayectoria generada, en moodle, el día establecido para la entrega de la tarea.** Se valorará:
  - Claridad y legibilidad del código. Al menos se recomienda incluir un "docstring" en cada una de las funciones principales.
  - Funcionalidad correcta de las funciones requeridas
- **Demo** a los profesores de las tareas realizadas (V0: sin obstáculos añadidos y V1: con obstáculos desconocidos y re-planificación) en la fecha establecida