



# Laboratorio - Búsqueda con Robocode

José A. Montenegro

Dpto. Lenguajes y Ciencias de la Computación  
ETSI Informática. Universidad de Málaga  
[jmmontes@uma.es](mailto:jmmontes@uma.es)

10 de mayo de 2021



## Instalación

## Errores Comunes

## Conceptos Básicos de Robocode

## Algoritmos

## Heurísticas

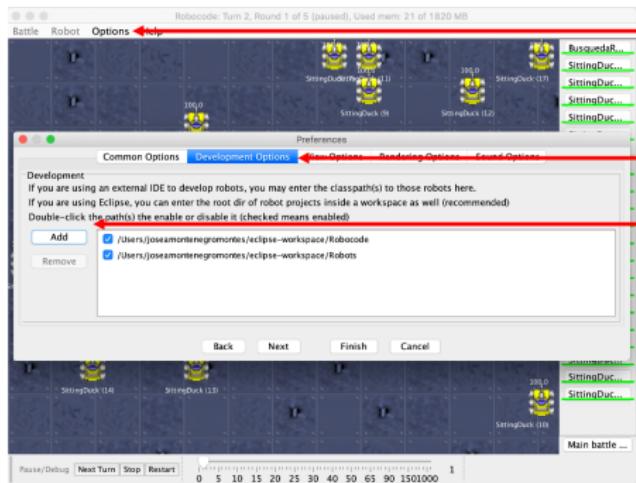
## Entrega

## Bugs



# Instalación

# Localización adicional Robots en Robocode



1. Options

2. Preferences

3. Development Options

4. Add

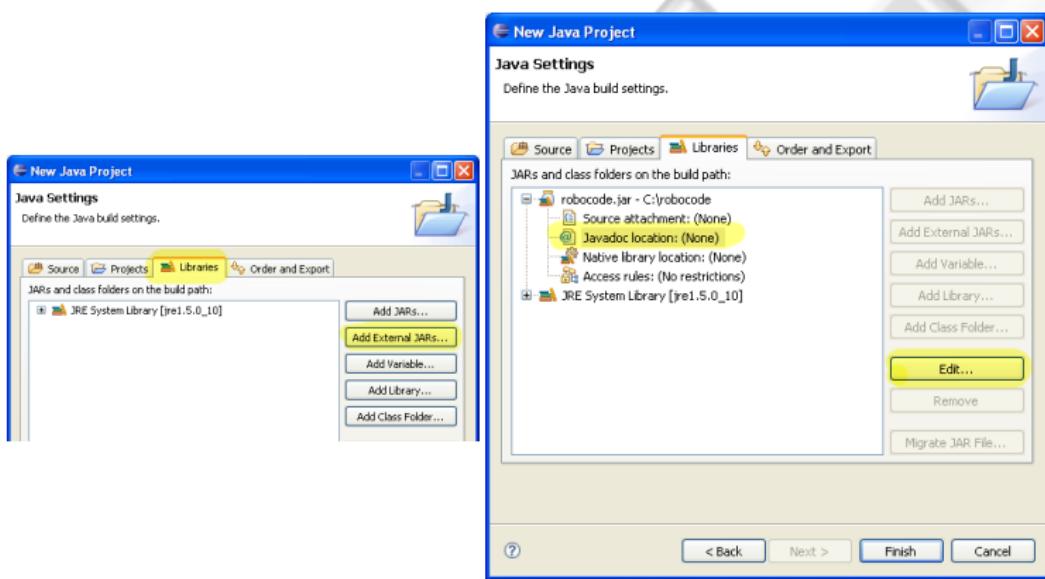
Añadimos la ruta de nuestro proyecto de Robot en eclipse

Si la configuración no es la adecuada obtendremos el error:

**Exception in thread "Application Thread"**  
**java.lang.ArrayIndexOutOfBoundsException: 1**  
**at Hito3.RouteFinder.main(RouteFinder.java:71)**

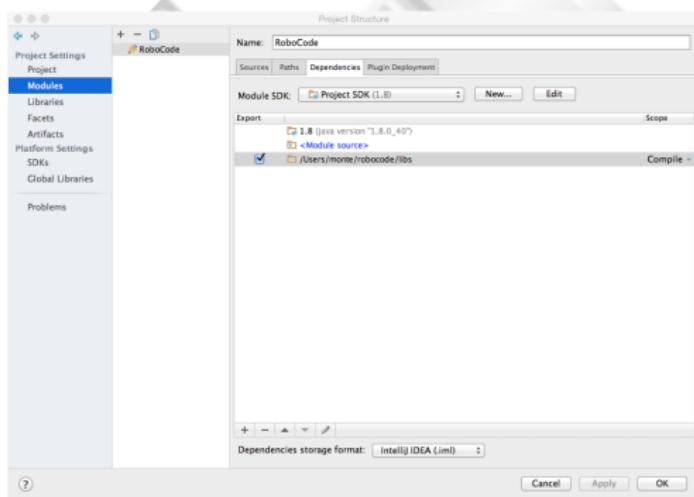
# Configuración proyecto en Eclipse

<http://robowiki.net/wiki/Robocode/Eclipse>



# Configuración proyecto en IntelliJ

- ▶ File
- ▶ Project Structure
- ▶ Modules
- ▶ Dependencies
- ▶ Añadir robocode/libs



# Errores Comunes

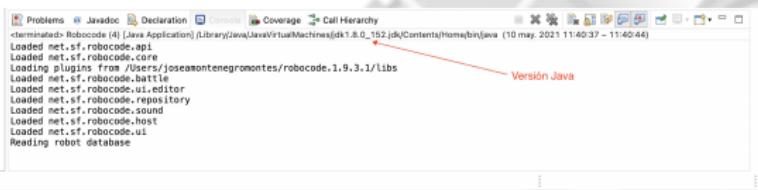
# Error `ArrayIndexOutOfBoundsException: 1`

El siguiente error:

```
Exception in thread "Application Thread"  
java.lang.ArrayIndexOutOfBoundsException: 1  
At RouteFinder.main(RouteFinder.java:71)
```

Se puede producir por dos situaciones:

1. No se ha incluido el directorio del robot en la configuración de Robocode (ver transparencia 4).
2. Estamos compilando el RobotBusqueda con una versión que no es java 1.8. Comprobar al ejecutar que estamos usando la versión correcta de Java en la consola de Eclipse como muestra la siguiente figura



La solución es instalar y configurar Java SE Development Kit 8 Downloads, <https://www.oracle.com/es/java/technologies/javase/javase-jdk8-downloads.html>

# Error version string ...

El siguiente error:

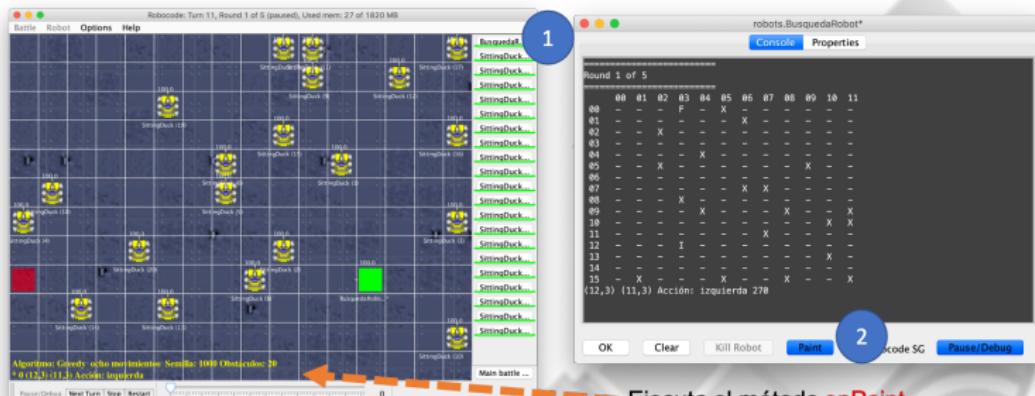
```
Exception in thread "Application Thread" java.lang.IllegalArgumentException: The format of the version
string is not a valid at net.sf.roboconf.version.Version.<init>(Version.java:32)
    at net.sf.roboconf.version.Version.compare(Version.java:167)
    at net.sf.roboconf.ui.dialog.RobocodeFrame.checkForNewVersion(RobocodeFrame.java:172)
    at net.sf.roboconf.ui.dialog.RobocodeFrame.checkUpdateOnStart(RobocodeFrame.java:158)
    at net.sf.roboconf.ui.WindowManager.showRobocodeFrame(WindowManager.java:155)
    at net.sf.roboconf.ui.WindowManager.setVisibleForRobotEngine(WindowManager.java:628)
    at roboconf.control.RobocodeEngine.setVisible(RobocodeEngine.java:207)
    at Main.RouteFinder.cfgRobocode(RouteFinder.java:60)
    at Main.RouteFinder.main(RouteFinder.java:152)
```

1. Ir al directorio config dentro de Robocode.
2. Abrir el fichero robocode.properties y cambiar la fecha de la propiedad:  
**robocode.versionchecked** por ejemplo poner un año más,  
**robocode.versionchecked=04/13/2022 20 16 22**

```
robocode.options.sound.enable=false
robocode.options.sound.enableBeep=true
robocode.options.common.notifyAboutNewVersion=false
robocode.options.common.notifyAboutNewVersionTime=10000
robocode.options.view.robotName=true
robocode.options.view.enableRobotName=true
robocode.options.hints.enableRobotName=false
robocode.options.hints.enableRobotNameTime=0
robocode.options.sound.availableOnDevmachines=true
robocode.options.devmachines=false
robocode.options.development.path=~/Users/jmmontenegro/eclipse-workspace/Maria
robocode.options.development.enabled=false
robocode.options.rendering.forceAbsoluteOrder=false
robocode.options.rendering.time=1000
robocode.options.view.enable=false
robocode.options.rendering.text.enable=false
robocode.options.rendering.image.enable=false
robocode.options.view.enableInitialization=true
robocode.options.view.enableInitializationTime=1000
robocode.options.view.printTextAsynchronously=false
robocode.options.desktopIcon.path=~/Desktop
robocode.options.common.disableWorkingEngine=true
robocode.options.common.disableWorkingEngineTime=1000
#date=versionchecked=04/13/2022 20 16 22
robocode.options.sound.enable=false
robocode.options.sound.enableBeep=true
robocode.options.sound.enableRobotName=true
robocode.options.sound.enableRobotNameTime=0
robocode.options.sound.availableOnDev=true
robocode.options.sound.availableOnDevTime=0
```

# Conceptos Básicos de Robocode

# Visualización información Depuración



Ejecuta el método `onPaint`

Dibuja información

```
public class BusquedaRobot extends Robot {
    ...
    public void onPaint(Graphics2D g) {
        // Información depuración
        // Casillas, punto inicial, final
        // Ejecución algoritmo
    }
}
```

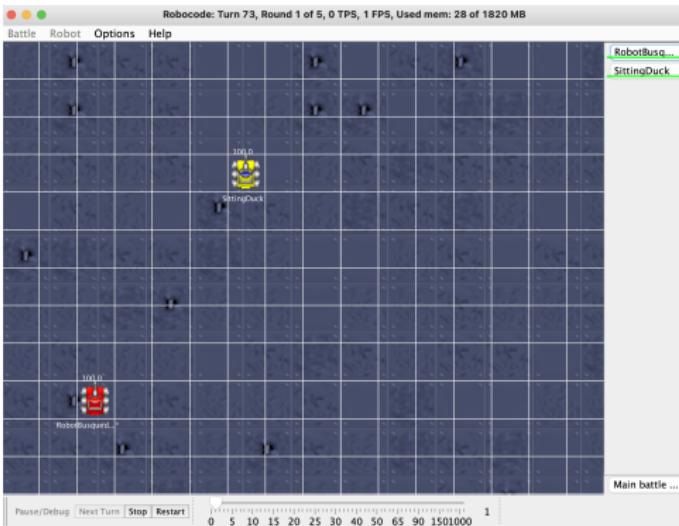
# Visualización información Depuración

[https://docs.oracle.com/javase/8/docs/api/java.awt.Graphics2D.html](https://docs.oracle.com/javase/8/docs/api/java.awt/Graphics2D.html)

```
public void onPaint(Graphics2D g) {  
    int x1, x2, y1, y2, width, height;  
    String cfgString="Hola";  
  
    g.setColor(Color.GREEN);  
    //Fills the specified rectangle  
    g.fillRect(x1,y1,width,height);  
  
    g.setPaint(Color.white); //Sets the Paint  
    //Draws a line, using the current color, between the points (x1,y1) (x2,y2)  
    g.drawLine(x1, y1,x2,y2);  
  
    //Sets this graphics context's font to the specified font.  
    g.setFont(new Font("Serif", Font.BOLD, 16));  
    //Draws the text given by the specified string  
    g.drawString(cfgString, x1, y1);  
}
```

# Ejercicio 1

Pintar las cuadrículas, utilizando la información de configuración del problema.

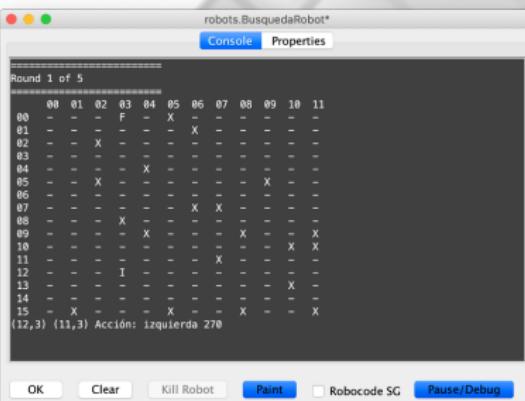


```
int fila = cfg.getNumColumna();
int columna = cfg.getNumFila();
int tamCelda = cfg.getTamCelda();
int filaPixels = cfg.getNumPixelFila();
```

## Ejercicio 2

- ▶ Genere problemas con 20 obstáculos, un estado inicial y final.
- ▶ Los problemas son generados de forma aleatoria, pero deben de ser reproducibles con el uso de la **semilla**.
- ▶ Imprimir el problema generado mediante una matriz, como muestra la figura, con la siguiente leyenda:

X Casilla con  
Obstáculo  
- Casilla Libre  
I Estado Inicial  
F Estado Final

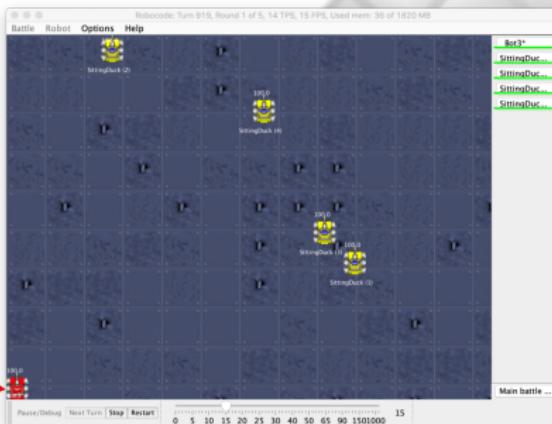


# Coordenadas Robot

<http://robocode.sourceforge.io/docs/robocode/robocode/control/RobotSetup.html>

```
// RobotSetup(Double x, Double y, Double heading)
double x = 0.0;                                //null means random.
double y = 0.0;                                //null means random.
double heading = 0.0;                            //null means random.
RobotSetup cfgRobot = new RobotSetup( x, y, heading);
```

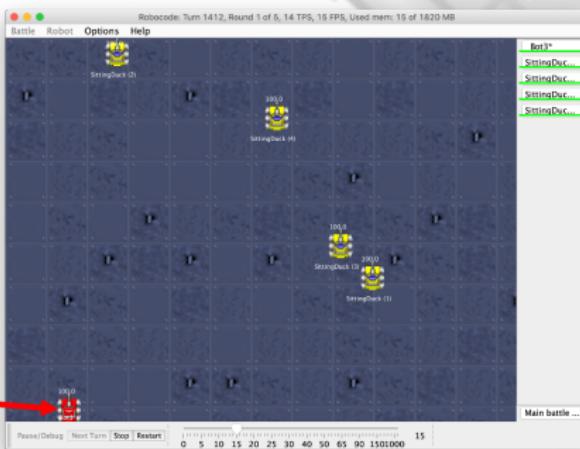
(0.0, 0.0)



# Coordenadas Robot

<http://robocode.sourceforge.io/docs/robocode/robocode/control/RobotSetup.html>

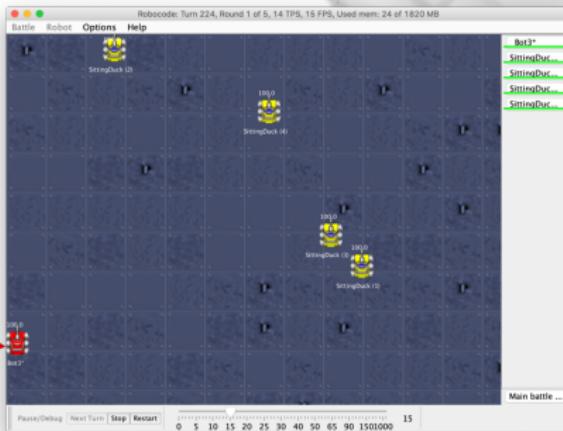
```
// RobotSetup(Double x, Double y, Double heading)
double x = 100.0;                                //null means random.
double y = 0.0;                                   //null means random.
double heading = 0.0;                             //null means random.
RobotSetup cfgRobot = new RobotSetup( x, y, heading);
```



# Coordenadas Robot

<http://robocode.sourceforge.io/docs/robocode/robocode/control/RobotSetup.html>

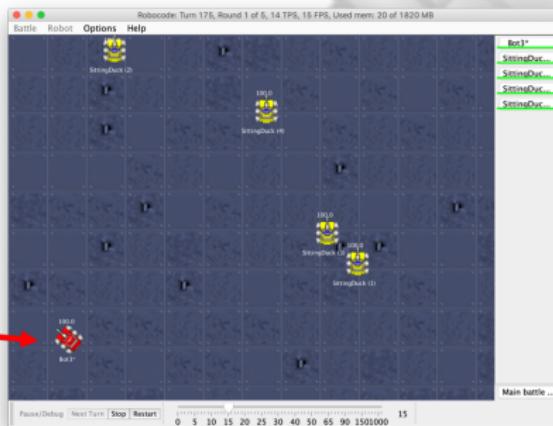
```
// RobotSetup(Double x, Double y, Double heading)
double x = 0.0;                                //null means random.
double y = 100.0;                               //null means random.
double heading = 0.0;                            //null means random.
RobotSetup cfgRobot = new RobotSetup( x, y, heading);
```



# Coordenadas Robot

<http://robocode.sourceforge.io/docs/robocode/robocode/control/RobotSetup.html>

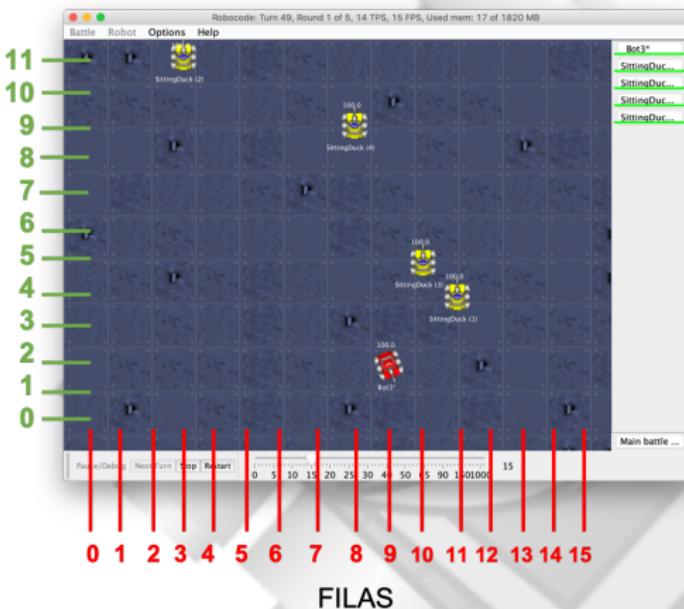
```
// RobotSetup(Double x, Double y, Double heading)
double x = 100.0;                                //null means random.
double y = 100.0;                                //null means random.
double heading = 315.0;                            //null means random.
RobotSetup cfgRobot = new RobotSetup( x, y, heading);
```



# Creación de Obstáculos

	COLUMNAS											
	00	01	02	03	04	05	06	07	08	09	10	11
00	-	-	-	-	-	-	-	-	-	-	-	-
01	-	-	-	-	-	-	-	-	-	-	-	-
02	-	-	-	-	-	-	-	-	-	-	-	-
03	-	-	-	-	-	-	-	-	-	-	-	X
04	-	-	-	-	-	-	-	-	-	-	-	-
05	-	-	-	-	-	-	-	-	-	-	-	F
06	-	-	-	-	-	-	-	-	-	-	-	-
07	-	-	-	-	-	-	-	-	-	-	-	-
08	-	-	-	-	-	-	-	-	-	-	-	X
09	-	-	I	-	-	-	-	-	-	-	-	-
10	-	-	-	-	-	-	-	-	-	-	-	X
11	-	-	-	-	-	-	-	-	-	-	-	-
12	-	-	-	-	-	-	-	-	-	-	-	-
13	-	-	-	-	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	-	-	-	-	-
15	-	-	-	-	-	-	-	-	-	-	-	-

800 x 600  
Fila x Columna

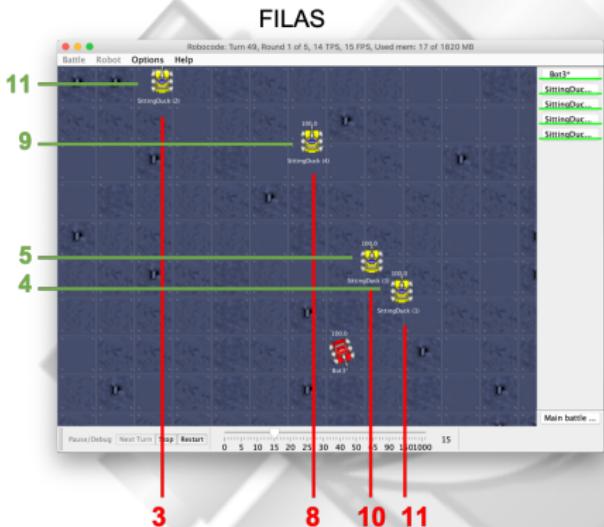


# Creación de Obstáculos

COLUMNAS											
00	-	-	-	-	-	-	-	-	-	-	-
01	-	-	-	-	-	-	-	-	-	-	-
02	-	-	-	-	-	-	-	-	-	-	-
03	-	-	-	-	-	-	-	-	-	-	X
F	05	-	-	-	-	-	-	-	-	-	-
I	06	-	-	-	-	-	-	-	-	-	-
L	07	-	-	-	-	-	-	-	-	-	-
A	08	-	-	-	-	-	-	-	-	-	X
S	09	-	I	-	-	-	-	-	-	-	-
10	-	-	-	-	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	-	-	-	X
12	-	-	-	-	-	-	-	-	-	-	-
13	-	-	-	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	-	-	-	-
15	-	-	-	-	-	-	-	-	-	-	-

800 x 600

FILA: 11 COLUMNA: 4 F: 575.0 C: 225.0  
 FILA: 3 COLUMNA: 11 F: 175.0 C: 575.0  
 FILA: 10 COLUMNA: 5 F: 525.0 C: 275.0  
 FILA: 8 COLUMNA: 9 F: 425.0 C: 475.0

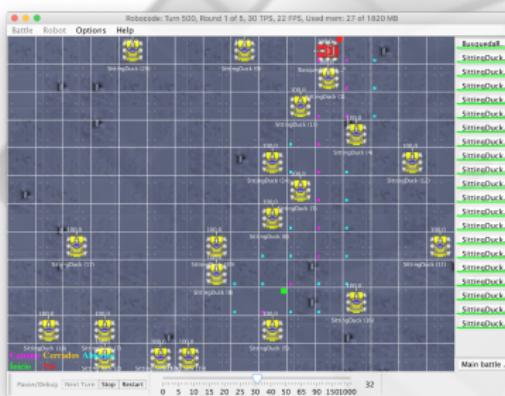


## Ejercicio 3

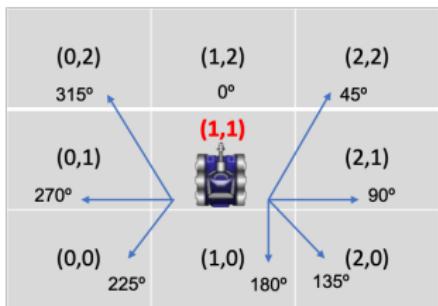
Utilizando los elementos generados en el Ejercicio 2

- ▶ Establezca el robot en el estado inicial.
- ▶ Establezca los obstáculos en sus posiciones.
- ▶ Dibuje un cuadro verde en el estado inicial.
- ▶ Dibuje un cuadro rojo en el estado final.

	00	01	02	03	04	05	06	07	08	09	10	11
00	-	-	-	-	-	-	-	-	-	-	-	-
01	-	X	-	-	-	-	-	-	-	-	-	-
02	-	-	-	-	X	-	-	-	-	-	-	-
03	X	X	-	-	-	-	-	-	-	-	-	-
04	-	-	-	-	-	-	-	-	-	-	-	X
05	X	-	-	-	-	-	-	-	-	-	-	-
06	X	-	-	-	-	-	-	-	-	-	-	-
07	-	-	-	X	X	-	-	-	-	-	-	-
08	-	-	-	-	-	-	-	-	-	-	-	X
09	-	X	I	-	-	X	-	-	-	-	-	-
10	-	-	-	-	-	-	X	-	-	X	-	-
11	-	-	-	-	-	-	-	-	-	-	X	F
12	-	-	X	-	-	-	-	-	X	-	-	-
13	-	-	-	-	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	X	-	-	-	-
15	-	-	-	-	X	-	-	-	-	-	-	-



# Orientación del Robot

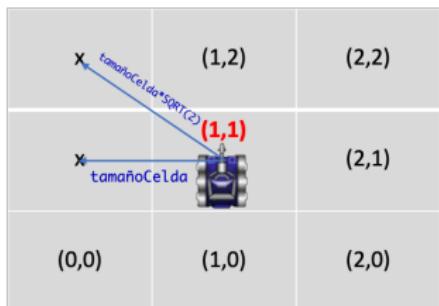


```
// 4 acciones
switch (action) {
    case arriba: grados = 0; break;
    case derecha: grados = 90; break;
    case abajo: grados = 180; break;
    case izquierda: grados = 270; break;
}

// 8 acciones
switch (action) {
    case arriba: grados = 0; break;
    case derechaNorte: grados = 45; break;
    case derecha: grados = 90; break;
    case derechaSur: grados = 135; break;
    case abajo: grados = 180; break;
    case izquierdaSur: grados = 225; break;
    case izquierda: grados = 270; break;
    case izquierdaNorte: grados = 315; break;
}
```

```
turnRight( normalRelativeAngleDegrees ( grados - getHeading() ) );
```

# Avance del Robot

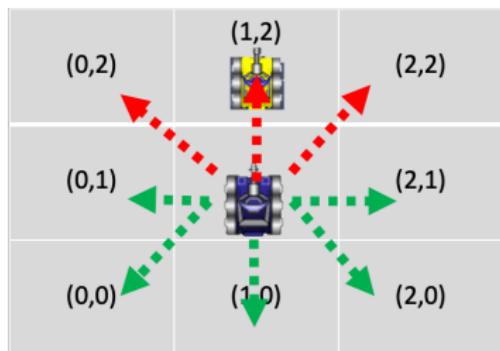


Ahead (avance);

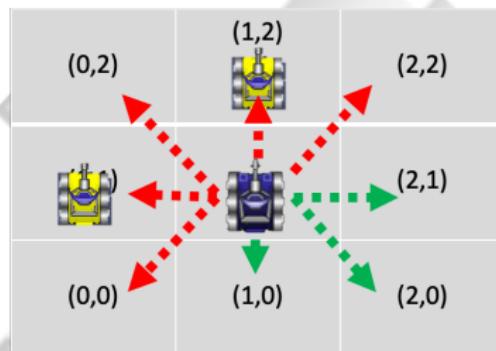
```
// 4 acciones
//arriba derecha abajo izquierda
avance = tamañoCelda;

//8 acciones
//arriba derecha abajo izquierda
avance = tamañoCelda;
//dNorte dSur iSur iNorte
avance = tamañoCelda * Math.sqrt(2.0);
```

# Movimientos Permitidos y No Permitidos del Robot



Sólo son posibles 5 movimientos



Sólo son posibles 3 movimientos

**Rojo:** Movimientos no permitidos, chocarían con el robot.

**Verde:** Movimientos permitidos.

## Ejercicio 4

Utilizando los elementos generados en el Ejercicio 2 y Ejercicio 3

- ▶ Mueva el robot desde un punto inicial a un punto final.
  1. Utilizando 4 movimientos
  2. Utilizando 8 movimientos
- ▶ Proporcionando el camino al robot sin utilizar un algoritmo.

# Algoritmos

# Algoritmo genérico

1. Crear un árbol de búsqueda A con raíz en s.
  - 1.1 Crear un conjunto de nodos ABIERTOS con s.
  - 1.2 Crear un conjunto de nodos CERRADOS vacío.
2. Mientras ABIERTOS no esté vacío
  - 2.1  $n \leftarrow \text{obtener}(\text{ABIERTOS})$ .
  - 2.2 Pasar  $n$  de ABIERTOS a CERRADOS.
  - 2.3 Si  $n$  es objetivo, entonces devolver el camino de  $s$  hasta  $n$  en A.
  - 2.4 Expandir  $n$ .  $M \leftarrow \text{sucesores}(n, G) - \text{antecesores}(n, A)$ .
  - 2.5 Para cada  $n_2$  en  $M$ ,
    - 2.5.1 Si  $n_2$  es nuevo ( $n_2$  no está ABIERTO ni CERRADO),  
Poner un puntero de  $n_2 \rightarrow n$ .  
Añadir  $n_2$  a ABIERTOS.
    - 2.5.2 en otro caso ( $n_2$  no es nuevo) decidir qué camino se conserva en el árbol.
3. Devolver FRACASO

# Amplitud

1. Crear un árbol de búsqueda A con raíz en s.
  - 1.1 Crear un conjunto de nodos ABIERTOS con s.
  - 1.2 Crear un conjunto de nodos CERRADOS vacío.
2. Mientras ABIERTOS no esté vacío
  - 2.1  $n \leftarrow \text{obtener}(\text{ABIERTOS})$ . (**Inicio Lista**)
  - 2.2 Pasar  $n$  de ABIERTOS a CERRADOS.
  - 2.3 Si  $n$  es objetivo, entonces devolver el camino de  $s$  hasta  $n$  en A.
  - 2.4 Expandir  $n$ .  $M \leftarrow \text{sucesores}(n, G) - \text{antecesores}(n, A)$ .
  - 2.5 Para cada  $n_2$  en  $M$ ,
    - 2.5.1 Si  $n_2$  es nuevo ( $n_2$  no está ABIERTO ni CERRADO), Poner un puntero de  $n_2 \rightarrow n$ . Añadir  $n_2$  a ABIERTOS. (**Final Lista**)
    - 2.5.2 en otro caso ( $n_2$  no es nuevo) decidir qué camino se conserva en el árbol.
3. Devolver FRACASO

**ABIERTO** es una cola (**FIFO**-First In, First Out): insertar al final de la cola y extraer del principio de la cola

# Profundidad

1. Crear un árbol de búsqueda A con raíz en s.
  - 1.1 Crear un conjunto de nodos ABIERTOS con s.
  - 1.2 Crear un conjunto de nodos CERRADOS vacío.
2. Mientras ABIERTOS no esté vacío
  - 2.1  $n \leftarrow \text{obtener}(\text{ABIERTOS})$ . (**Inicio Lista**)
  - 2.2 Pasar n de ABIERTOS a CERRADOS.
  - 2.3 Si n es objetivo, entonces devolver el camino de s hasta n en A.
  - 2.4 Expandir n.  $M \leftarrow \text{sucesores}(n, G) - \text{antecesores}(n, A)$ .
  - 2.5 Para cada  $n_2$  en M,
    - 2.5.1 Si  $n_2$  es nuevo ( $n_2$  no está ABIERTO ni CERRADO),  
Poner un puntero de  $n_2 \rightarrow n$ .  
Añadir  $n_2$  a ABIERTOS. (**Inicio Lista**)
    - 2.5.2 en otro caso ( $n_2$  no es nuevo) decidir qué camino se conserva en el árbol.
3. Devolver FRACASO

*ABIERTO* es una cola (**LIFO**-Last In, First Out): insertar al inicio de la cola y extraer del principio de la cola

# A\* y voráz

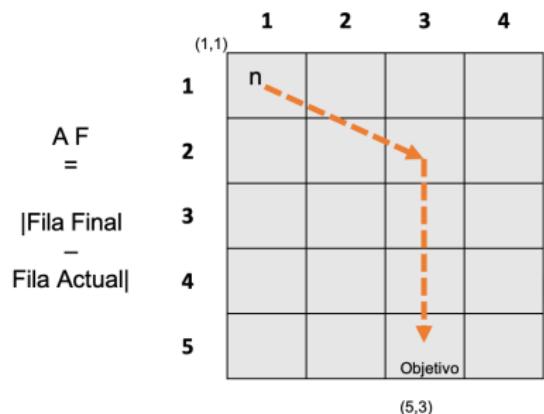
1. Crear un árbol de búsqueda A con raíz en s.
  - 1.1 Crear un conjunto de nodos ABIERTOS con s.
  - 1.2 Crear un conjunto de nodos CERRADOS vacío.
2. Mientras ABIERTOS no esté vacío
  - 2.1  $n \leftarrow \text{obtener}(\text{ABIERTOS})$ . (**Primer nodo, valor de f más bajo**)
  - 2.2 Pasar n de ABIERTOS a CERRADOS.
  - 2.3 Si n es objetivo, entonces devolver el camino de s hasta n en A.
  - 2.4 Expandir n.  $M \leftarrow \text{sucesores}(n, G) - \text{antecesores}(n, A)$ .
  - 2.5 Para cada  $n_2$  en M,
    - 2.5.1 Si  $n_2$  es nuevo ( $n_2$  no está ABIERTO ni CERRADO),  
Poner un puntero de  $n_2 \rightarrow n$ .  
Añadir  $n_2$  a ABIERTOS. (**Insertar ordenado según f**)
    - 2.5.2 en otro caso ( $n_2$  no es nuevo) decidir qué camino se conserva en el árbol.
3. Devolver FRACASO

*ABIERTO* es una lista ordenada por el valor f de cada estado. Menor valor al principio. Voraz:  $f = h$  y  $A^*$ :  $f = g + h$

# Heurísticas

## 8 Movimientos - Distancia octil

$$AC = | \text{Columnas Final} - \text{Columna Actual} |$$



Coste diagonal = 142  
Coste horizontal = 100

$$h(n) = \text{Coste diagonal} * \min(AC, AF) + \text{Coste horizontal} * [\max(AC, AF) - \min(AC, AF)]$$

$$h(n) = 142 * \min(2, 4) + 100 * [\max(2, 4) - \min(2, 4)]$$

AC =  $| 3 - 1 | = 2$   
AF =  $| 5 - 1 | = 4$

$$h(n) = 142 * 2 + 100 * [4 - 2] = 284 + 200 = 484$$

# Entrega

## Entrega Obligatoria

La entrega **obligatoria** será realizar un Problema con:

- ▶ Un estado inicial
- ▶ Un estado final
- ▶ 20 obstáculos

El robot deberá ir desde el estado inicial al estado final sin chocar con ningún obstáculo, utilizando el algoritmo de **amplitud**.

Consideremos como cuatro acciones disponibles: arriba, abajo, izquierda y derecha.

La ejecución deberá imprimir los estados abiertos, cerrados, el camino solución y el coste de la solución.

## Entrega Opcional puntuable, Búsqueda - Máximo 0,5 ptos

La entrega **puntuable** será realizar un Problema con:

- ▶ Un estado inicial
- ▶ Un estado final
- ▶ 20 obstáculos

El robot deberá ir desde el estado inicial al estado final sin chocar con ningún obstáculo, utilizando el algoritmo de A\*.

Si ejecutamos el algoritmo con cuatro acciones: arriba, abajo, izquierda y derecha, será puntuable con **0,25 ptos**.

Si ejecutamos el algoritmo con ocho acciones: 4 acciones y las diagonales, será puntuable con **0,5 ptos**.

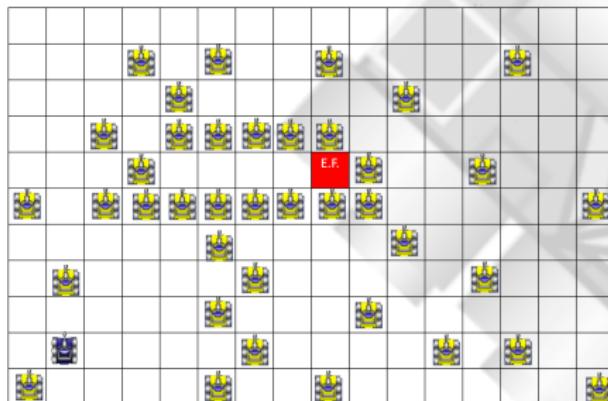
La ejecución deberá imprimir los estados abiertos, cerrados, el camino solución y el coste de la solución.

# Entrega Opcional puntuable, CSP - Máximo 0,5 ptos

La entrega **puntuable** será realizar un Problema con:

- ▶ Un estado inicial
- ▶ Un estado final
- ▶ Al menos 40 obstáculos

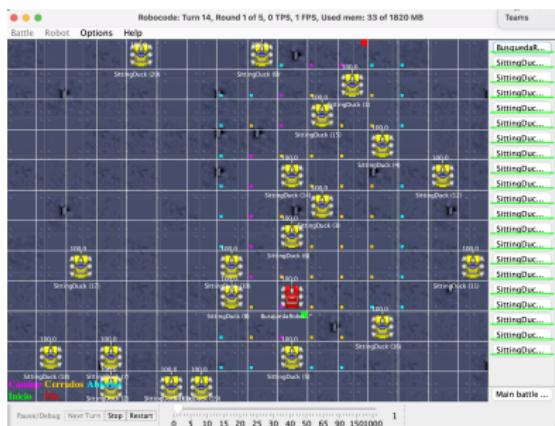
Para ellos formularemos la creación del mapa con obstáculos como un problema de **CSP (0,5 ptos)**, para que sea posible ir del estado inicial al estado final. Por ejemplo, el mapa mostrado en la imagen no sería un mapa con solución, ya que el robot no podría ir del estado inicial al estado final, con lo cual no tendría solución.



# Bugs

# Creación de Obstáculos

Ejecución de la rutina de creación de obstáculos en dos versiones de Robocode, en la versión 1.9.3.9 obtenemos un error en la posición de los obstáculos.



Robocode 1.9.3.1



Robocode 1.9.3.9

**José A. Montenegro Montes**

*Dpto. Lenguajes y Ciencias de la Computación  
ETSI Informática. Universidad de Málaga*

**jmmontes@uma.es**



UNIVERSIDAD  
DE MÁLAGA

