# Práctica 1. Búsqueda

**Objetivos:**

- Comprender el funcionamiento de la búsqueda heurística y en concreto del algoritmo A*.
- Implementar el algoritmo A* y saber cómo seleccionar una heurística apropiada al problema.
- Aplicar dicho algoritmo para implementar dos comportamientos diferentes: persecución y huida.
- Realizar un análisis cuantitativo respecto al número de nodos explorados con este algoritmo.

## *Introducción y entorno de trabajo*

En esta primera práctica de la asignatura se desarrollará un sistema para la resolución del problema de búsqueda del camino mínimo.

### *Netbeans y Java*

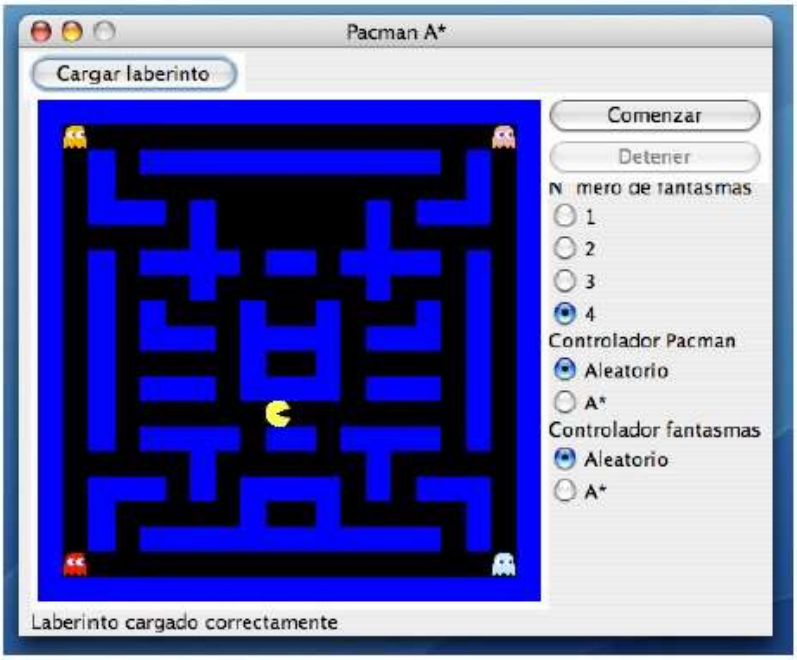
Para el desarrollo de la práctica se utilizará NetBeans y el lenguaje Java.

NetBeans es un entorno de desarrollo integrado, hecho principalmente para el lenguaje de programación Java, aunque extendido a otros lenguajes. NetBeans es un proyecto de código abierto, por lo tanto, lo podemos descargar libremente desde su página oficial:

https://netbeans.apache.org/download/nb110/nb110.html

### *Enunciado*

Se pretende aplicar el algoritmo para la creación del comportamiento de los personajes del cásico videojuego Pacman, tanto del personaje principal (Pacman) como de los enemigos (fantasmas). Pacman utilizará la información del camino más corto a los fantasmas para encontrar una ruta adecuada de huida y los fantasmas para desplazarse hacia Pacman. Se proporciona el código de un proyecto que deberá ser completado. El aspecto de dicha aplicación es el siguiente:




El botón **Cargar laberinto** permite leer un fichero de texto con la configuración del laberinto a utilizar. En el caso del laberinto de la figura anterior el fichero sería:

```
20 //Altura del laberinto, siempre es cuadrado
//Los 1s representan paredes, los 0s huecos
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1
1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
1 0 1 1 1 0 1 0 0 0 0 0 0 1 0 1 1 1 0 1
1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1
1 0 1 0 1 1 1 1 0 1 1 0 1 1 1 1 0 1 0 1
1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1
1 0 1 0 1 0 0 0 1 0 0 1 0 0 0 1 0 1 0 1
1 0 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 0 1
1 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1
1 0 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 0 1
1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
1 0 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 0 1
1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1
1 0 1 1 1 0 1 0 1 1 1 1 1 0 1 0 1 1 1 0 1
1 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1
1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 12  //Posición inicial de pacman
1 1 //Posición inicial del primer fantasma
18 18 //Posición inicial del segundo fantasma
18 1 //Posición inicial del tercer fantasma
1 18 //Posición inicial del cuarto fantasma
```

El grupo de botones de radio con el nombre **número de fantasmas** permite elegir el número de enemigos a los que deberá enfrentarse pacman durante la partida. Con respecto a los botones **controlador Pacman** y **controlador fantasmas,** permiten elegir tanto para uno como para los otros si se desea que se muevan al azar (comportamiento por defecto) o que se

muevan utilizando el algoritmo A*. El botón **Comenzar** permite iniciar la partida y el botón **Detener** interrumpirla pudiendo volver a empezar desde el principio al volver a pulsar el botón Comenzar.

## Diseño del algoritmo de búsqueda A*

Este algoritmo es uno de los más utilizados para encontrar un camino o ruta entre dos puntos, con la característica que, si se cumplen unas condiciones, el camino encontrado será el camino de menor coste entre los dos puntos y además, si existe, siempre encontrará ese camino (algoritmo completo).

A* es un algoritmo de búsqueda heurística, por lo tanto, utiliza una función heurística, que nos dará un valor para cada celda. La función heurística es una estimación optimista de cómo de lejos está el objetivo. Es decir, en el caso del fantasma, es una estimación optimista de lo que le costará llegar hasta la celda en la que se encuentra Pacman.

$$h(x,y) \sim <= \text{distancia al objetivo}$$

A continuación, se muestra el pseudocódigo del algoritmo A*.

### *Pseudocódigo A**

```
Alg A*
      listaInterior = vacio
      listaFrontera = inicio

      mientras listaFrontera no esté vacía
            n = obtener nodo de listaFrontera con menor f(n) = g(n) + h(n)
            listaFrontera.del(n)
            listaInterior.add(n)

            si n es meta
                  devolver
                  reconstruir camino desde la meta al inicio siguiendo los
      punteros
            fsi

            para cada hijo m de n que no esté en listaInterior
                  g'(m) = n·g + c(n, m)  //g del nodo a explorar m

                  si m no está en listaFrontera
                        almacenar la f, g y h del nodo en (m.f, m.g, m.h)
                        m.padre = n
                        listaFrontera.add(m)
                  sino  si  g'(m)  es  mejor  que  m.g   //Verificamos  si  el
nuevo camino es mejor
                        m.padre = n
                        recalcular f y g del nodo m
                  fsi
            fpara
      fmientras
      devolver no hay solución
falg
```

## implementación del algoritmo de búsqueda A*

El código viene preparado con dos configuraciones de ejecución:

- **Probar:** El objetivo de esta configuración es comprobar el correcto funcionamiento del algoritmo A*. Para ello, simplemente se carga un mundo y se aplica el algoritmo A* para que el fantasma (sólo se prueba con un fantasma) encuentre el mejor camino hacia Pacman.
- **Jugar:** Con esta configuración se inicia el juego que permite cargar un mundo y lanzar el juego. Es posible elegir el comportamiento (A* o aleatorio) tanto de Pacman como de los fantasmas.

El código a modificar se encuentra en la clase `Aestrella`. En dicha clase hay dos métodos que hay que completar:

```
int AestrellaFantasma(Laberinto laberinto)

int AestrellaPacman(Laberinto laberinto)
```

Ambos métodos deben devolver un movimiento que será el que realizará el personaje: Laberinto.IZQUIERDA, Laberinto.DERECHA, Laberinto.ARRIBA, Laberinto.ABAJO

En la clase A* se dispone de las siguientes variables:

**camino**: En esta matriz, que se inicializa toda a '.', debéis indicar las celdas que forman parte del camino con una 'X'.

**camino_expandido**: En esta matriz, que se inicializa toda a -1, debéis indicar el orden en el que se expanden las celdas.

**expandidos**: Es un entero donde debéis almacenar el número de nodos expandidos.

**coste_Total**: es un número donde debéis indicar el coste total del camino calculado por el A*.

```
//Camino
char camino[][];

//Casillas expandidas
int camino_expandido[][];

//Número de nodos expandidos
int expandidos;

//Coste del camino
double coste_total;
```

- No se pueden utilizar ficheros adicionales a la clase anterior
- No se debe modificar ninguna clase del entorno aparte de la anterior

**¡OJO! La implementación del algoritmo A* se debe realizar en la clase AEstrella.java. Es posible utilizar métodos y clases adicionales, pero siempre dentro del fichero AEstrella.java.**

En concreto, al final del A*, las variables deben contener:

- La variable **camino**, como se muestra en la imagen, debe contener una X en las celdas que forman parte del camino calculado por el A*.
- La variable **camino_expandido**, como se muestra en la imagen, debe contener el orden en el que se han ido expandiendo los nodos.
- La variable **expandidos**, debe contener el número total de nodos expandidos.
- La variable **coste_total**, debe contener el coste total del camino calculado por el A*.

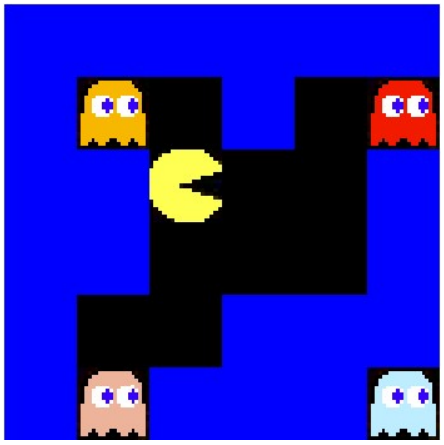
## Salida de la práctica en modo Probar

En este modo la ejecución debe mostrar la siguiente información:

- Coste del camino generado por A*
- Número de nodos expandidos
- Contenido de la matriz camino
- Contenido de la matriz camino explorado

Por ejemplo, la siguiente figura muestra en la parte izquierda el fichero de texto correspondiente a un mundo, en la parte central el mundo en modo gráfico y en la parte de la derecha la salida que debería proporcionar el A* para el fantasma 1 (esquina superior izquierda).

```
6
1 1 1 1 1 1
1 0 0 1 0 1
1 1 0 0 0 1
1 1 0 0 0 1
1 0 0 1 1 1
1 1 1 1 1 1
2 2
1 1
5 5
1 5
5 1
```



```
run:
NO MODIFICAR ESTE FORMATO DE SALIDA
Coste del camino: 2.0
Nodos expandidos: 3
Camino
. . . . . .
. X X . . .
. . X . . .
. . . . . .
. . . . . .
. . . . . .
Camino explorado
-1 -1 -1 -1 -1 -1
-1  0  1 -1 -1 -1
-1 -1  2 -1 -1 -1
-1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1
```

## Plan de entrega por hitos

Durante el periodo de ejecución de la práctica se realizarán tres hitos de entrega. Es obligatorio cumplir las fechas de las entregas correspondientes:

| Hito | Entrega | Fecha tope |
|---|---|---|
| 1 | Clase Aestrella con el algoritmo A* para el fantasma funcionando sin heurística (h=0) | 29 de septiembre |
| 2 | Clase Aestrella con la heurística para Pacman y la de los fantasmas más un fichero de documentación que explique la heurística utilizada tanto para Pacman como para los fantasmas | 13 de octubre |
| 3 (final) | Proyecto con varias heurísticas implementadas siguiendo la estructura del formato de entrega | 27 de octubre |

- La no entrega del hito 1 en la fecha prevista supone una penalización del 15%.
- La no entrega del hito 2 en la fecha prevista supone una penalización del 15%.
- Para la entrega final se dejará el programa empleando las heurísticas más adecuadas, tanto para Pacman como para los fantasmas. Las otras heurísticas analizadas aparecerán en el código, aunque no se utilicen.

**Formato de entrega del proyecto para el hito 3 (entrega final)**

La entrega debe consistir en un **fichero comprimido zip** con tres carpetas:

- /Fuente : clase Aestrella.java
- /Mundos: mundos de prueba utilizados
- /Doc: documentación en **pdf**

**La nota de la práctica sufrirá una penalización de dos puntos si no se cumple rigurosamente con los requisitos de la entrega (tanto en la estructura de los ficheros entregados como en la salida que debe generar la práctica)**

**Documentación del proyecto**

La documentación es una parte muy importante de la práctica. Como mínimo debe contener:

- Explicación detallada de la implementación del A* indicando su funcionamiento tanto en el caso del fantasma como en del pacman y **traza** de un problema pequeño donde se observe el funcionamiento del algoritmo A*
- Análisis comparativo de las **distintas heurísticas** implementadas analizando el número de nodos.
- Varios laberintos de prueba con diseños que muestren distintas casuísticas del problema.