

# Ingeniería del Conocimiento

## Algoritmo A\*



José María López Pulido

## Índice:

- Introducción.
- Implementación.
- Funcionamiento de la app.

## Introducción:

Este documento detalla la memoria de la práctica 1 de Ingeniería del Conocimiento, implementación del algoritmo A\*.

El algoritmo A\* es un algoritmo de búsqueda en grafos. Presentado por primera vez en 1968 por Peter E. Hart, Nils J. Nilsson y Bertram Raphael. El algoritmo encontrada dado un nodo origen y un final, el camino de coste mínimo entre los dos nodos, siempre y cuando ese camino exista.

Así, el algoritmo A\* utiliza una función de evaluación ,  $f(n) = g(n) + h'(n)$  donde  $h'(n)$  representa el valor heurístico del nodo a evaluar desde el actual, n, hasta el final, y  $g(n)$  , el coste real del camino recorrido para llegar a dicho nodo, n, desde el nodo inicial. A\* mantiene dos estructuras de datos auxiliares, que podemos denominar *abiertos*, implementado como una cola de prioridad (ordenada por el valor  $f(n)$  de cada nodo), y *cerrados*, donde se guarda la información de los nodos que ya han sido visitados. En cada paso del algoritmo, se expande el nodo que esté primero en abiertos, y en caso de que no sea un nodo objetivo, calcula la  $f(n)$  de todos sus hijos, los inserta en abiertos, y pasa el nodo evaluado a cerrados.

## Implementación:

El lenguaje utilizado para el desarrollo de la práctica es Java, consta de dos ventanas una para la generación del tablero y otra para la representación del propio tablero, utilizando la biblioteca Swing de Java.

El tablero es la representación de una matriz de nodos, cuya información contenida es la posición en el propio tablero, su distancia al nodo origen, su valor heurístico hasta el final, el nodo a través del cual se ha accedido a él.

El camino es representado con un `ArrayList<Nodo>` y representará el camino más corto desde el nodo inicial hasta el nodo final.

El mayor problema que ha tenido la implementación del algoritmo surge de la lista abierta y de la lista de nodos adyacentes al nodo seleccionado, dado que ambas listas si bien no deben de estar ordenadas, en ambas se debe coger el nodo de menor coste, para añadirlo a la lista cerrada.

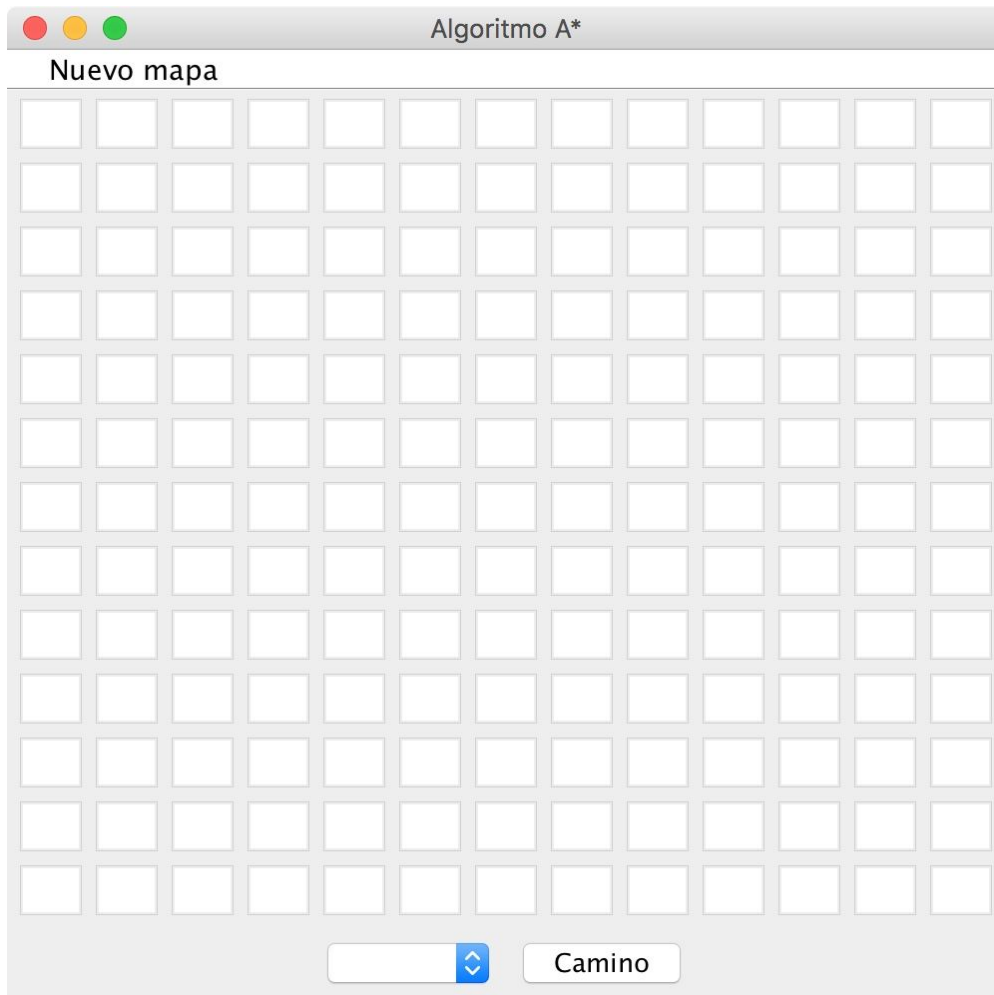
## Funcionamiento de la app:

Al ejecutar la aplicación podremos ver un pequeño formulario donde deberemos introducir el número de filas y columnas de la matriz.



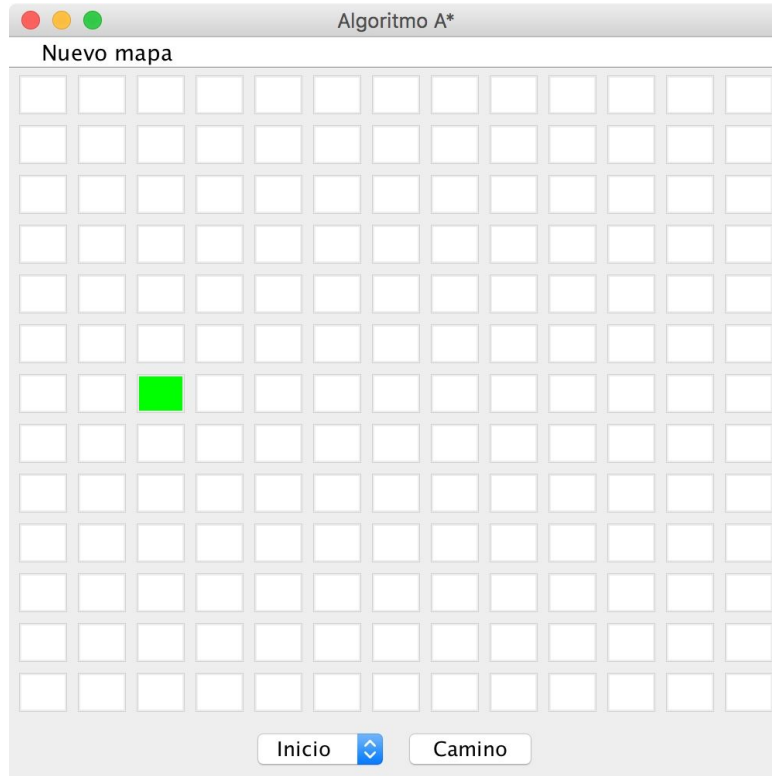
The screenshot shows a window titled "Algoritmo A\*" with a light gray background. On the left, the labels "Filas" and "Columnas" are positioned next to two empty text input fields. Below these fields, centered, is a button labeled "Aceptar".

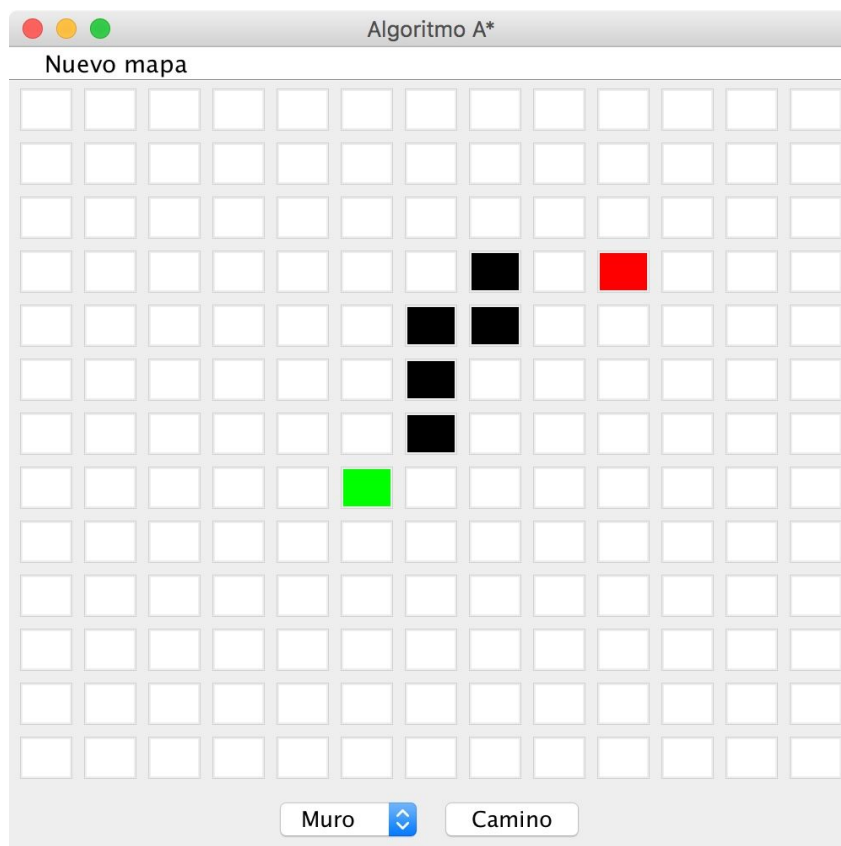
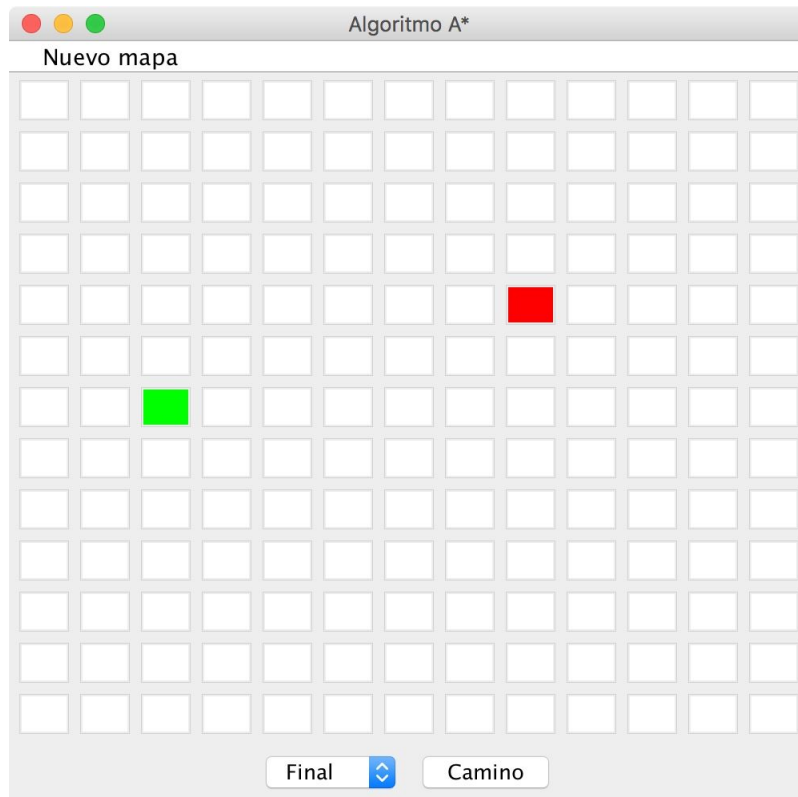
Una vez insertados unos valores correctos, y pulsado el botón de aceptar aparecerá otra ventana con un menú donde encontraremos un botón que nos servirá para introducir un nuevo número de filas y columnas.



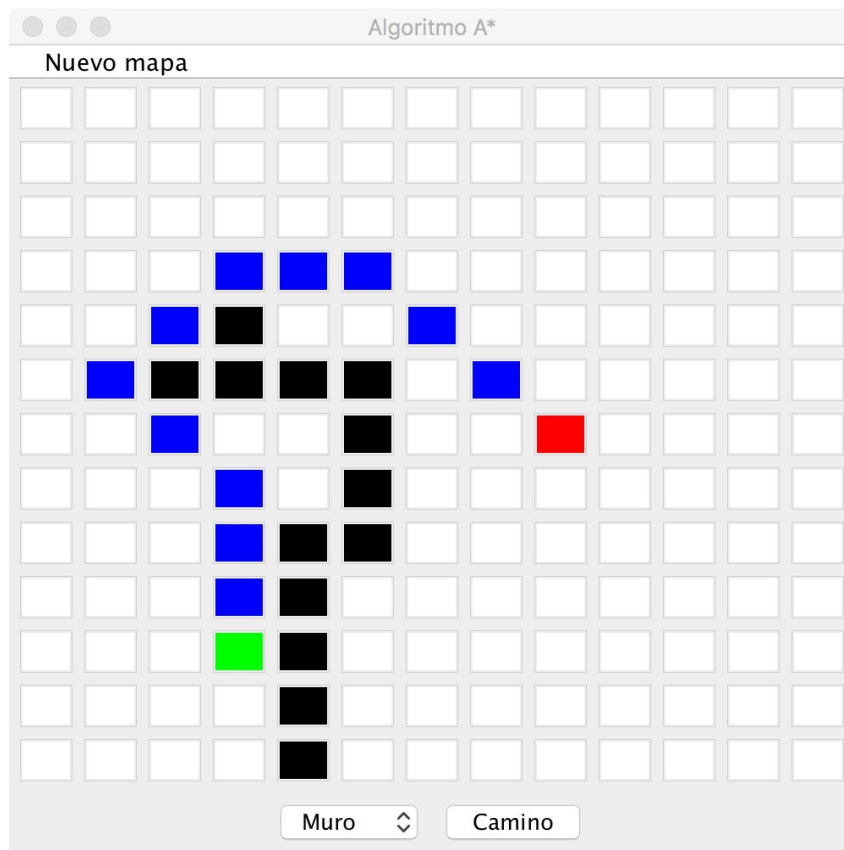
The screenshot shows a window titled "Algoritmo A\*" with a light gray background. Below the title bar, the text "Nuevo mapa" is displayed. Underneath is a large grid of 12 columns and 12 rows of small, empty square cells. At the bottom of the window, there is a small input field with a blue dropdown arrow, followed by a button labeled "Camino".

Según el elemento que tengamos seleccionado en nuestro desplegable podremos diseñar nuestro mapa, la opción vacía sirve para eliminar un elemento por ejemplo un muro. La opción inicio y/o final pondrán en la casilla seleccionada el inicio o final del recorrido, eliminando el punto anterior en caso de que existiera.





Una vez diseñado el mapa la pulsar sobre el botón “Camino”, obtendremos el camino desde el nodo origen hasta el nodo final, con reasignación de nodos.



Si queremos modificar un diseño ya creado, podremos añadir muros o mover inicio o final, sobre el resultado y volver a pulsar el botón de camino.

