Algoritmo A*

Sergio Pino Holgado

Universidad Complutense de Madrid

Ingeniería del Conocimiento

Índice

Índice de figuras

1. Introducción

El algoritmo de búsqueda A* (pronunciado .ª asterisco.º .ª estrella") se clasifica dentro de los algoritmos de búsqueda en grafos. Presentado por primera vez en 1968 por Peter E. Hart, Nils J. Nilsson y Bertram Raphael, el algoritmo A* encuentra, siempre y cuando se cumplan unas determinadas condiciones, el camino de menor coste entre un nodo origen y uno objetivo.

Mediante este algoritmo se pretende paliar la carencia de ciertos algoritmos como el voraz, que se guía exclusivamente por la función heurística, que puede no indicar el camino con el coste más bajo o por el coste real de moverse de un nodo a otro, pudiendo dar lugar a soluciones con un coste superior al mínimo. El algoritmo A* tiene en cuenta tanto el valor heurístico como el coste real de desplazamiento.

La función de evaluación que emplea A* es:

$$f(n) = g(n) + h'(n)$$

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 hasta el nodo n.

A* hace uso de dos estructuras auxiliares, una lista de nodos "abierta", que es implementada como una cola de prioridad cuyos elementos son ordenados según su valor $f(n)$; y otra lista "cerrada" donde se van almacenando los nodos que ya han sido visitados. En cada iteración del algoritmo se expande el nodo que se encuentra en la primera posición de la lista "abierta" y en caso de no ser un nodo objetivo calcula $f(n)$ para todos sus adyacentes, los añade a la lista "abierta" y convierte el nodo en "cerrado".

Una característica a tener en cuenta de este algoritmo es que mezcla la búsqueda del tipo primero en anchura con primero en profundidad, consiguiendo cambiar de camino siempre que se encuentren nodos más prometedores.

2. Detalles de implementación

El sistema ha sido desarrollado empleando el lenguaje Java (1.8) bajo el entorno IntelliJ IDEA (Ultimate 2016). El motivo de la elección tanto del lenguaje como del IDE ha sido la sencillez y potencia que ambos aportaban al presente proyecto, pues no requería ningún tipo de aprendizaje previo al desarrollo del mismo. La aplicación dispone de una interfaz por consola que es la encargada de recoger y presentar los datos al usuario, por lo que debe ser ejecutada en una terminal de comandos.

El código ha sido estructurado en dos paquetes principales:

- `negocio` donde reside la lógica del sistema: el algoritmo y las clases necesarias para su funcionamiento.
- `presentacion` la clase encargada de recoger y mostrar los datos al usuario.

Las funciones extra que se han implementado se definen a continuación:

- Posibilidad de añadir obstáculos al mapa y recalcular la ruta.
- Posibilidad de añadir un camino a seguir mediante waypoints.

3. Ejecución

A continuación se detallará brevemente como ha de ejecutarse la aplicación.

- Lo primero será ejecutar una terminal de comandos, *CMD* o *Power Shell* en Windows, *Terminal* en sistemas basados en `Unix`.
- Nos situaremos en el directorio donde se encuentra el fichero .jar ejecutable de la aplicación.
- Una vez en el terminal, ejecutaremos el siguiente comando: `$ java -jar AEstrella.jar`
- La aplicación solicitará los datos necesarios para la creación del mapa de datos. Los datos a introducir son los siguientes:

```
Introduce N° de filas:
5
Introduce N° de columnas:
5
Introduce punto inicial (X Y):
0 0
Introduce punto destino (X Y):
4 4
Introducir obstaculo (S/N):
s
Introduce obstaculo (X Y):
1 1
Introducir obstaculo (S/N):
n
Introducir waypoint (S/N):
s
Introduce waypoint (X Y):
3 1
Introducir waypoint (S/N):
n
```

Figura 1: Creación del mapa

  - Número de filas.
  - Número de columnas.
  - El punto de inicio del camino a recorrer.
  - El punto destino o fin del camino.
  - Las coordenadas de los obstáculos que se quieran introducir en el mapa.
  - Las coordenadas de los waypoints que se quieran introducir en el mapa.

- Una vez terminamos de introducir los datos necesarios se muestra un pequeño menú de opciones.

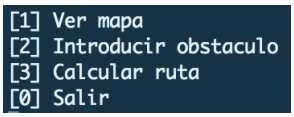

Figura 2: Menú de la aplicación.

- Se puede visualizar el mapa en su estado actual.

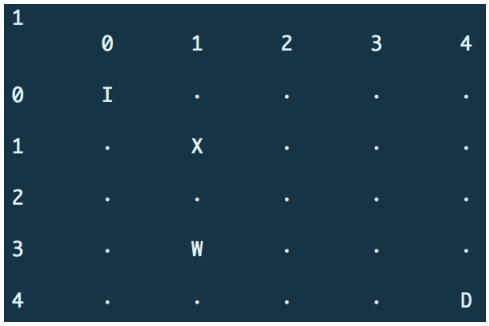

Figura 3: Estado actual del mapa.

- Se puede insertar un nuevo obstáculo aunque exista una ruta calculada.

Figura 4: Inserción de un nuevo obstáculo.

- Se puede solicitar el cálculo de una ruta y esta será mostrada en caso de existir.

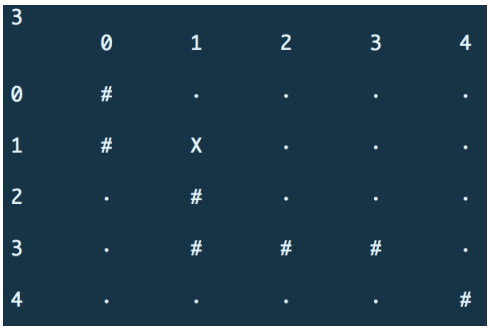

Figura 5: Camino encontrado.

4. Referencias

- **Wikipedia**, Algoritmo de búsqueda A*