

Proyecto 1 (15%)

Fecha de Entrega: 28 de setiembre 2023

Puzzle-N: Backtracking y A*

1 Objetivo

Con esta actividad vas a conseguir implementar la estrategia de búsqueda heurística **A*** para la resolución del problema del puzzle-N.

2 Descripción de la actividad

El puzzle-N es un juego de un único jugador que consiste en un tablero de (N^2) posiciones ($N \times N$), y ocho fichas numeradas del 1 al $(N^2) - 1$, situadas en el tablero, quedando una posición del tablero vacía.

El objetivo del juego es alcanzar una determinada disposición de las fichas a partir de una disposición inicial realizando solo los movimientos permitidos, que son mover una ficha adyacente a la posición vacía de forma horizontal o vertical, ocupando la posición vacía y quedando en su lugar vacía la casilla ocupada anteriormente por la ficha movida.

En esta actividad se debe utilizar **backtracking** y la estrategia de búsqueda heurística **A*** con el fin de resolver el problema del puzzle-N. Utiliza como heurística el número de fichas mal colocadas respecto al estado objetivo. Considera que el coste de cada movimiento es 1.

La aplicación debe ser capaz de procesar tableros de tamaño 3×3 hasta 20×20 , siendo este una opción que puede seleccionar el usuario a la hora de generar nuevos tableros. En los ejemplos que se le muestran en esta especificación se utiliza un tablero 3×3 como ilustración. Se deben utilizar imágenes en el tablero, con al menos seis opciones que el usuario puede elegir.

Un posible estado inicial del puzzle:solucion

Figura 1



El estado objetivo es el siguiente:

Figura 2



Debes desarrollar en el lenguaje de programación Javascript, una implementación de los algoritmos **Backtracking** y **A*** que desarrolle el árbol de búsqueda desde el estado inicial al estado objetivo.

En el caso de **A***, en cada iteración del algoritmo se deberá indicar claramente el nodo que ha sido expandido, el **contenido de la lista abierta** y de la **lista cerrada de nodos** (estados) con su valor de función de evaluación.

Para este problema, es común que se utilice la distancia de Manhattan para la estimación de coste de solución.

Se deberán crear tantas clases o estructuras de datos como sean necesarias para representar el espacio de estados y los nodos de exploración del árbol.

3 Interfaz gráfica

Se debe presentar el tablero en una página HTML desde donde se importen los archivos Javascript necesarios. Al final del procesamiento, se debe observar una animación de los pasos encontrados por el algoritmo para resolver el tablero y se debe indicar la secuencia de acciones a realizar para alcanzar el objetivo utilizando una notación sencilla. Por ejemplo: “mover 7 hacia la derecha” o “mover 5 hacia la izquierda” en un campo de texto adecuado para tal fin.

El programa debe ser capaz de resolver **problemas generados de forma aleatoria**, como pueden existir configuraciones de tablero sin solución, la aplicación debe ser capaz de informarlo.

Puede utilizar librerías externas para la representación visual, pero la implementación de los algoritmos debe ser un trabajo **totalmente original de los estudiantes**, si se comprueba que los estudiantes utilizaron código copiado de alguna fuente, perderían **todos** los puntos del proyecto.

El programa desarrollado debe ser un trabajo original del estudiante.

El código debe venir debidamente comentado, se debe utilizar **jsDoc** para cada función o clase utilizada.

4 Checklist

- Interfaz
 - Tablero de tamaño dinámico
 - Secuencia paso a paso de la búsqueda de solución (en el tablero y terminal), por ejemplo: video
 - Secuencia automática con la solución.

- Generar tablero de forma aleatoria
- Funcionalidades
 - Resolución por Backtracking
 - Resolución por A*
 - Además de todo lo dicho anteriormente

5 Nota importante

Existen mucho código en internet con soluciones al problema, por lo que durante la revisión se harán preguntas para comprobar el conocimiento del código por cada uno de los estudiantes. Aunque no es requisito, si los estudiantes utilizan Git para el trabajo colaborativo del código, se puede utilizar para comprobar el trabajo de cada uno.

Es probable que la ejecución de backtracking provoque que la computadora se quede sin recursos, por lo que es suficiente demostrar su correcto funcionamiento más que llegar a una solución de cada tablero.

6 Entregable

El proyecto deberá entregarse en el TecDigital, en un archivo comprimido que incluya los archivos necesarios para la ejecución. Se debe precisar en un archivo Readme.md cualquier iteración necesaria del usuario para la ejecución del programa.

Si la entrega se realiza después de la hora de entrega, se le penalizará con 5 puntos porcentuales que se acumulan cada 24 horas. Por ejemplo si entrega a las 10:05pm su evaluación tendrá una nota base de 95%, si entrega después de las 10:05 p.m. del siguiente día, su nota base será 90%, y así sucesivamente.

Se puede realizar en grupos de 4.

7 Evaluación

Actividad	Descripción	Peso%
Criterio 1	Implementación del algoritmo Backtracking	30%
Criterio 2	Implementación del algoritmo A*: el cálculo de la función de evaluación de los nodos $f(n)$ es correcta y el contenido de la lista abierta y de la lista cerrada para cada iteración es correcta	40%
Criterio 3	La interfaz gráfica cumple con lo solicitado	20%
Criterio 4	El resultado obtenido, es decir la secuencia de acciones a realizar para alcanzar el objetivo, es correcto	10%
		100 %