UNIDAD DE INGENIERÍA EN COMPUTACIÓN
Análisis de Algoritmos
Profesor: Kenneth Obando Rodríguez

TEC | Tecnológico de Costa Rica

Proyecto 1 (15%) Fecha de Entrega: Ver calendario

Traffic Jam: Backtracking y A*

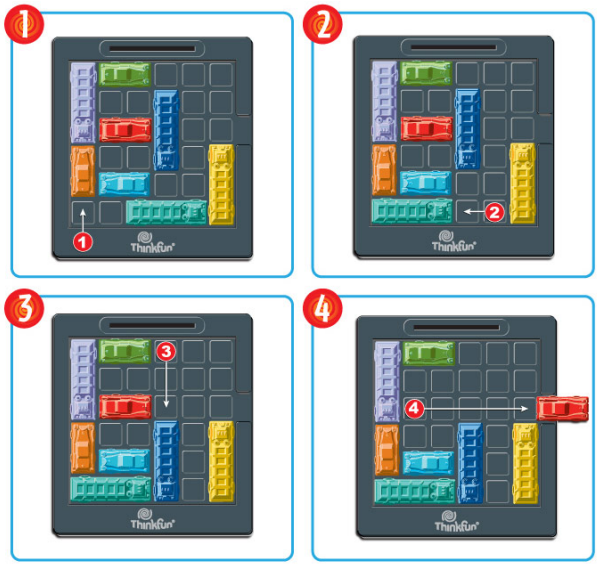
1 Objetivo

Desarrollar una aplicación en JavaScript que simule el juego Traffic Jam, aplicando algoritmos de búsqueda como Backtracking, DFS (Depth-First Search), BFS (Breadth-First Search) y A*. Este proyecto permitirá a los estudiantes integrar conocimientos teóricos y prácticos en el desarrollo de soluciones algorítmicas eficientes para problemas complejos.

2 Descripción del Juego

En Traffic Jam, el jugador se enfrenta a un tablero que simula un estacionamiento congestionado con varios carros estacionados. Estos carros pueden tener diferentes tamaños (generalmente de 2 a 3 espacios de longitud) y orientaciones (horizontal o vertical). Existe un carro objetivo, usualmente identificado por su color rojo, que necesita encontrar su camino hacia la salida del estacionamiento. La salida está ubicada en uno de los bordes del tablero, y solo el carro objetivo puede salir.

Los movimientos permitidos para cada carro son únicamente en línea recta y en la dirección a la que está orientado, y solo pueden moverse si tienen espacio libre (no bloqueado por otro carro). El objetivo del juego es mover los carros de manera que se libere el camino para que el carro objetivo pueda llegar a la salida con el menor número de movimientos posibles.



2.1 Especificaciones Técnicas

2.1.1 Tablero y Carros

- El tablero debe ser de tamaño variable, definido por el usuario, con un tamaño máximo de 12×12.
- Los carros son de tamaño y orientación variable, con al menos un carro objetivo.

2.1.2 Entrada del Usuario

- Los usuarios pueden definir tableros personalizados, incluyendo tamaño, posición y orientación de los carros, siguiendo el siguiente formato:

  El formato para el ingreso de tableros de prueba describe un tablero de juego para el Traffic Jam en una representación textual, donde cada símbolo tiene un significado específico:

  - `.` representa una casilla vacía en el tablero.
  - `-` indica una parte de un carro orientado horizontalmente.
  - `|` indica una parte de un carro orientado verticalmente.
  - `>` señala la parte frontal de un carro orientado hacia la derecha.
  - `v` marca la parte frontal de un carro orientado hacia abajo.
  - `B` marca la parte frontal del carro objetivo.
  - El espacio vacío ( ) se utiliza para separar las casillas y mejorar la legibilidad del tablero.
  - La salida se debe especificar con dos números correspondientes a las coordenadas de la salida (horizontal, vertical).

  Cada carro se representa mediante líneas (`-` o `|`) que indican su orientación (horizontal o vertical) y una flecha (`>` o `v`) que muestra hacia dónde está apuntando el frente del carro. La longitud de cada carro (número de casillas que ocupa) viene determinada por la cantidad de símbolos `-` o `|` seguidos, más el símbolo de la flecha.

  Ejemplo de Tablero

  El siguiente es un ejemplo de un tablero ingresado con este formato:

```
- - - - > . .
. . . . . . .
| . . - - - >
| . . . . . .
v . - - - B .
. . . . . . .
- - - - > . .
Salida: 0,5
```

  Salida

  Este tablero contiene varios carros, algunos orientados horizontalmente y otros verticalmente, distribuidos a lo largo de diversas filas y columnas.

  1. El primer carro está orientado horizontalmente en la primera fila, ocupando 5 casillas y apuntando hacia la derecha.

2. El segundo carro ocupa 4 casillas en la tercera fila, también orientado horizontalmente y apuntando hacia la derecha.
3. Un carro orientado verticalmente comienza en la tercera fila, ocupando dos casillas y apuntando hacia abajo.
4. Otro carro orientado horizontalmente se encuentra en la quinta fila, ocupando 4 casillas y apuntando hacia la derecha.

2.1.3 Algoritmos

- Implementar los algoritmos de Backtracking y A*.
- Permitir al usuario seleccionar el algoritmo a utilizar.

2.1.4 Métricas

- Medir tiempo de ejecución, número de movimientos y estados explorados.

3 Documentación

Los estudiantes deben entregar:

- Explicación de cada algoritmo implementado.
- Guía de uso de la aplicación.
- Análisis comparativo del rendimiento de los algoritmos.

4 Interfaz de Usuario

Desarrollar una interfaz gráfica para la interacción con el juego, selección de algoritmos y visualización de soluciones y métricas.

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 se debe indicar la secuencia de acciones a realizar para alcanzar el objetivo utilizando una notación sencilla. Por ejemplo: “mover carro 3 hacia la derecha” o “mover carro 5 hacia la izquierda” en un campo de texto adecuado para tal fin.

La aplicación debe ser capaz de informar si no encontró una solución.

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.

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 iteracción 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 %