**Diferentes algoritmos para salir de un laberinto**

Todo algoritmo para salida de un laberinto debe de funcionar y llevar a la salida bajo cualquier condición. Sin embargo algunos algoritmos pueden ser más extensos que otros o más específicos. A continuación se muestran tres tipos de algoritmos para laberintos:

* **Algoritmo de Tremaux**

El ingeniero Charles Tremaux (1859–1882) fue el inventor de un algoritmo sencillo y para salir de un laberinto. Pero este algoritmo dependiendo de la complejidad del laberinto puede tomar más tiempo debido a que valúa cada camino posible hasta encontrar una salida. Es importante resaltar el camino que se sigue o dejar rastro. Así de esa manera se sigue cualquier camino y si nos encontramos con nuevos cruces podemos tomar cualquiera y si regresamos a un cruce viejo se debe tomar el camino sin explorar y eventualmente se llegará a la salida.

* **Algoritmo de Backtracking(recursivo)**

El algoritmo genera secuencias completas para lograr llegar al objetivo. En el caso de un laberinto, se buscan todas las opciones de camino que se tienen, y si una es incorrecta, se retrocede al inicio y se utiliza una nueva vía. Esto se repite hasta llegar a la solución correcta. Gracias a que es un algoritmo recursivo, se puede retroceder hasta el punto de inicio, descartando el camino ya tomado y buscar la siguiente hasta encontrar la salida.

Este algoritmo no se escogió para el proyecto debido a que se busca un algoritmo que sea rápido, y el *backtracking* requiere repetir de nuevo el laberinto con un nuevo camino, lo que llevaría demasiado tiempo si es un laberinto muy grande.

**Explique las razones por las cuales seleccionó el algoritmo que implementará. Si desechó los algoritmos documentados y decidió hacer uno propio explique por qué. Recuerde que puede basarse en los análisis de complejidad para poder justificar sus elecciones. Tenga en cuenta que debe utilizar el mismo algoritmo que diseñó y probó en la segunda fase del proyecto.**

**Explique la estructura de datos a utilizar, y diseñe el diagrama de clases que utilizará.**

**Pseudocódigo:**

Las paredes son identificadas con '-'

si se encuentra la salida (\*) -> llegaste al final

si no se encuentra la salida (\*)

buscar espacio vacio ()

si hay espacio vacio()

moverse

si no hay () buscar en y + 1

si no hay () buscar buscar en x-1

si no hay () buscar en x + 1

si no hay () buscar en y-1

se busca y se mueve hasta encontrar la salida (\*)

**Pruebe que el algoritmo es útil para ayudar a un elemento a salir de cualquier algoritmo, para eso implemente el algoritmo en un entorno virtual. Puede cargar laberintos prediseñados, pero el algoritmo no se debe modificar, por lo que debe salir de cualquier laberinto.**

**Mida el tiempo en que se demora el elemento en encontrar la salida en laberintos de distinta complejidad, siempre usando el algoritmo seleccionado.**

**Suba todos los archivos a un repositorio de github, recuerde que debe haber varios commits de todos los miembros del equipo, durante todo el tiempo de disponibilidad de la tarea.**