Universidad del Valle de Guatemala     Algoritmos y estructuras de datos

Javier Carpio García - 17077 María Fernanda López - 17160

David Uriel Soto - 17551 Guatemala, 08/02/2018

Proyecto 1

1. **Algoritmos:**
   1. Algoritmo de Tremaux.

Algoritmo inventado por el ingeniero francés Charles Tremaux para encontrar la salida de un laberinto. Este consiste en marcar el camino que se va siguiendo, que también dependerá de cuantas veces se recorre un mismo camino, y realizar los siguientes pasos:

* No siga el mismo camino 2 veces.
* Si llega a un cruce nuevo, no importa qué camino siga
* Si un camino nuevo lo lleva a un cruce viejo, o a un callejón sin salida, retroceda hasta la entrada del camino.
* Si un camino viejo lo lleva a un cruce viejo, tome un camino nuevo, y si no lo hay tome cualquiera.

Este algoritmo puede llegar a tomar varias horas, pero si se sigue al pie de la letra este sistema siempre llevará a la salida del laberinto, siempre que exista una salida, o en caso de que no exista, llevará a la entrada.

* 1. Regla de la mano derecha/izquierda.

Es un método sencillo que se ha utilizado hace cientos de años para salir de diversos laberintos. Para realizar el algoritmo basta con definir qué mano se utilizará para realizar todo el algoritmo. Ponemos dicha mano escogida apoyada sobre la pared y se permanece con la mano así hasta salir del laberinto. De esta manera se estará seguro de tocar cada pared del laberinto hasta encontrar la salida.

Aunque bien es un algoritmo que siempre permitirá encontrar la salida de un laberinto, este puede llegar a llevar demasiado tiempo o muy poco, dependiendo de la forma en que se diseñó el laberinto y la posición del sujeto dentro del mismo. La otra gran desventaja es que, si el laberinto se forma de círculos concéntricos, puede llegar a hacer que el algoritmo no sirva y nunca se salga del laberinto.

* 1. Algoritmo de garantía

Una versión modificada del seguidor de paredes o del de la regla de la mano derecha, ya que este es capaz de resolver cualquier tipo de laberinto sin quedarse estancado, ya que cuenta los giros que se realizan para saber si se encuentra en un circuito cerrado o una isla y así poder salir de esta. Puede encontrar la solución de un laberinto desde su perímetro hasta el centro, pero no del centro al perímetro. Una ventaja de este algoritmo es que no necesita memorizar los lugares por donde ha pasado, pero aun así debe llevar la cuenta de sus giros.

* 1. Cadena

Este algoritmo busca dividir el laberinto completo en pequeños laberintos que están encadenados y este algoritmo los resuelve todos de poco en poco. Se basa en seguir la estructura del algoritmo seguidor de paredes, pero de una manera más eficiente ya que memoriza los lugares por los que ha pasado antes. Este algoritmo no encuentra el camino más corto, pero busca caminos alternativos basado en los lugares que ya ha pasado.

* 1. Vuelta atrás recursiva

Es muy similar al algoritmo de Tremaux, ya que lo que busca es no repetir el mismo camino dos veces, utilizando un mapeo en memoria para reconocer los caminos que ya ha recorrido. Lo que intenta es poder recordar el camino que siguió desde el inicio y volver al punto más lejano en donde había otro camino que seguir. Encuentra la solución como el resto de algoritmos, pero tampoco encuentra la más corta.

1. **Algoritmo a utilizer:**

*Algoritmo de Tremaux*

Se eligió este algoritmo ya que es un algoritmo sencillo de implementar con respecto a la complejidad de espacio que este puede llegar a ocupar con respecto a sus instrucciones, ya que este básicamente se rige por un pseudocodigo de 4 instrucciones. No es el algoritmo con la complejidad de espacio más pequeña, ya que el de la mano derecha basta con realizar una simple instrucción, pero el de la mano derecha no siempre llegaría a solucionar el laberinto. Sin embargo, el algoritmo de Tremaux siempre encuentra una solución para el laberinto. Ahora hablando de la complejidad de tiempo, es muy dificil saberlo ya que este depende mucho de la forma del laberinto o de cómo se posicione el objeto a resolver el laberinto dentro de este. Aunque bien se podría decir que por tener al menos 4 condiciones, sería más fácil que encontrara la solución, ya que la condiciones de este algoritmo establece, permiten reducir las opciones de recorrido hacia la salida haciendo que este haga recorridos innecesarios varias veces, aunque bien en algún momento puede hacer un par de veces un mismo recorrido. En el mejor de los casos su complejidad con respecto al tiempo podría ser n, pero en el peor de los casos podría ser n^3, ya que existen 3 condiciones que pueden ser ciclos. Es un algoritmo que no falla y es posible de simular a través de la programación virtual.

1. Pseudocódigo y/o diagrama de flujo

**Inicio**

Ingresar al laberinto

**Mientras** camina

              Mantenga su mano derecha apoyada a una pared

**Si** llega a un tope

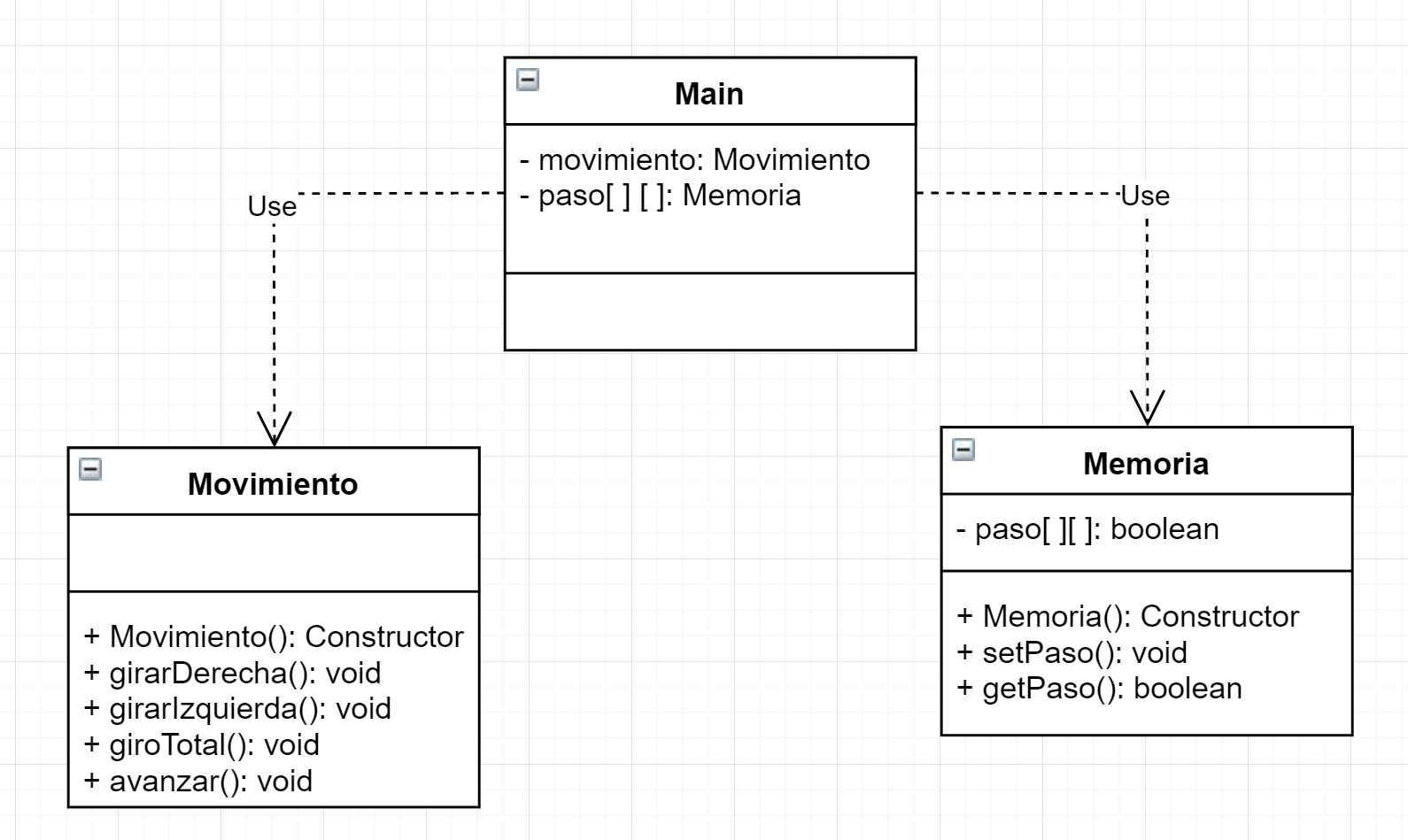
Gire a la derecha hasta que encuentre un camino en el que no haya pared

**FinSi**

**FinMientras**

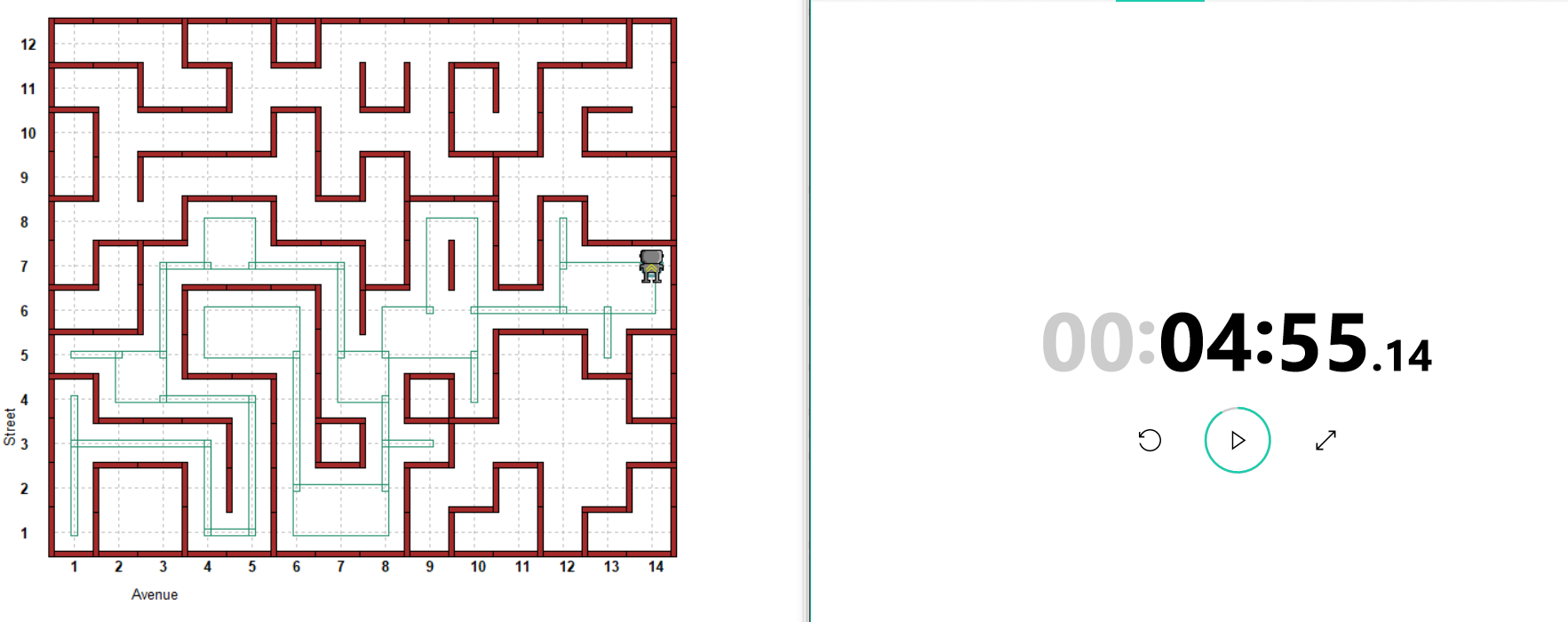
**Fin**

1. Diagrama de secuencias:

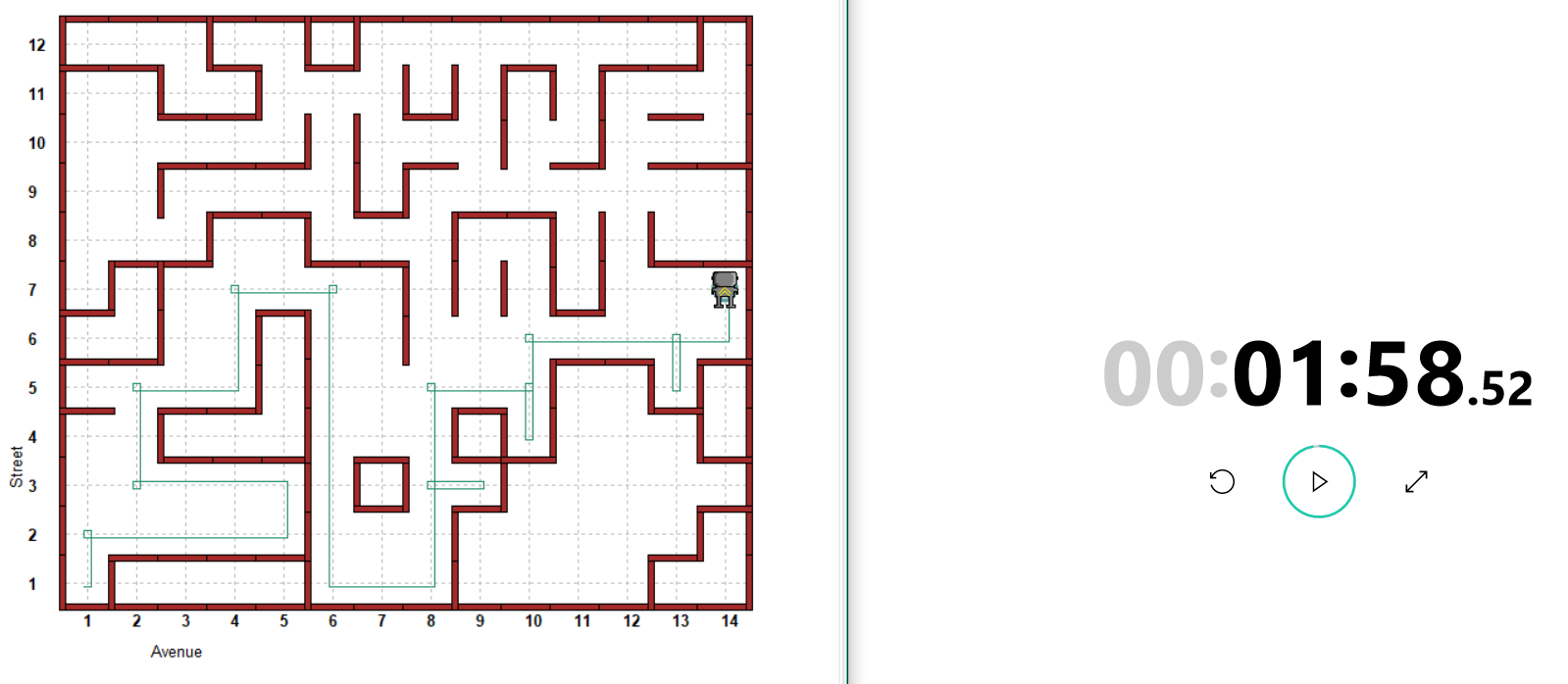
****

1. Tiempo que se tarda en ejecutar el algoritmo

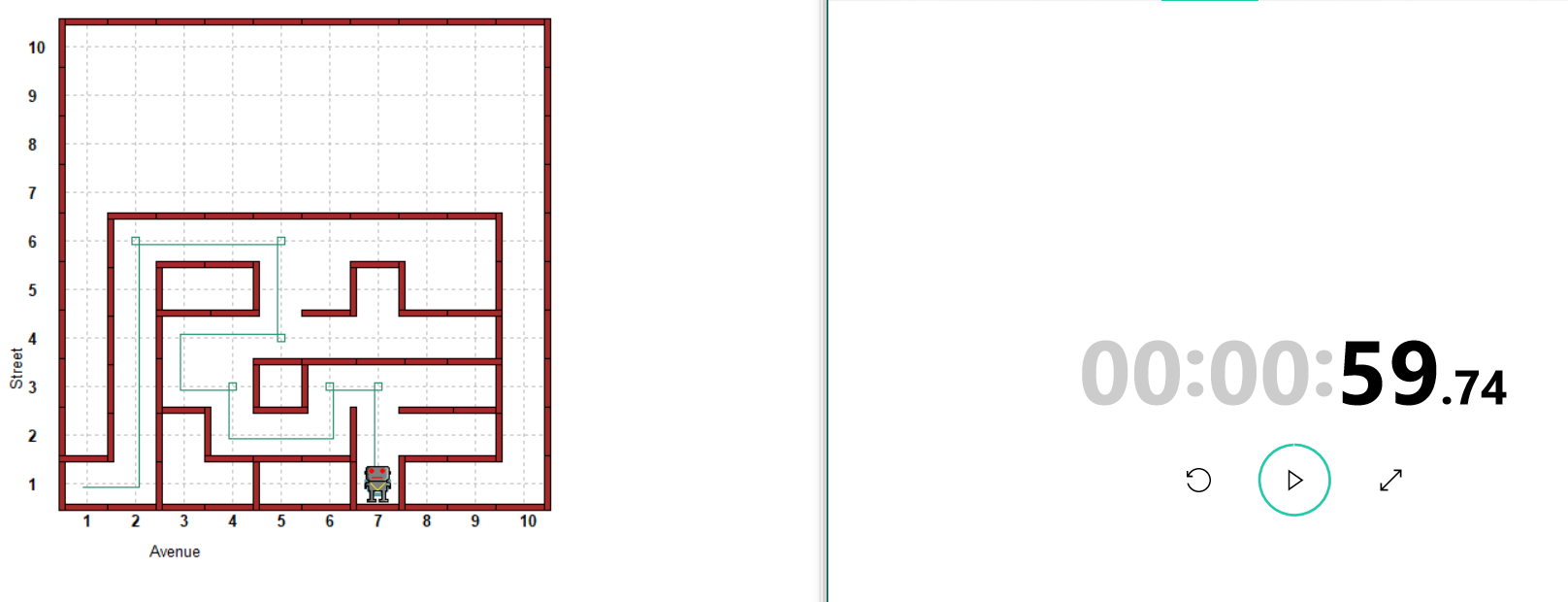
Tiempo No.1



Tiempo No.2



Tiempo No.3



Tiempo No.4

