Universidad del Valle de Guatemala

Algoritmos y Estructura de Datos

Ing. Oscar Robles

Sección 20

Raúl de León, 15112

Didier Salazar, 15487

Joseline Ortíz, 15039

Michelle Morales, 15322

**Algoritmos de resolución de laberintos**

1. **Algoritmo de azar:**

Este algoritmo consiste en que el robot toma arbitrariamente la dirección en cada uno de los nodos que corresponden a las celdas que tiene más de una posible dirección, de esta manera el robot tiene altas posibilidades de llegar a la meta entre otras ventajas como las que se enumeran a continuación:

* 1. No se requiere de mucha capacidad de procesamiento por parte del controlador
  2. Se simplifica el diseño del sistema electrónico

Por otro la implementación de este algoritmo tiene las siguientes desventajas:

* 1. Ausencia total de inteligencia
  2. No se puede saber en ningún momento la posición u orientación dentro del laberinto
  3. El robot no sabría en qué momento llega a la meta
  4. No se puede estimar el tiempo en que tardará de resolver el laberinto.

Por las razones anteriores el laberinto no es una elección adecuada.

(UNAM, 2013)

1. **Algoritmo de seguidor de paredes:**

Este algoritmo se basa en la regla de que siguiendo siempre por la derecha o por la izquierda se encontrará invariablemente la salida de un laberinto. El algoritmo se aplica de la siguiente manera:

**Paso 1:** Checa la presencia de la pared izquierda.

**Paso 2:** Si la pared izquierda está presente entonces se hace la bandera flag = 1, si no está presente, entonces se hace que el valor de la bandera sea 0.

**Paso 3:** Si flag = 1 entonces se activa el paso 4, caso contrario flag = 0 hace un giro de 90 grados.

**Paso 4:** Comprobar la presencia de una pared frontal.

**Paso 5:** Si la pared frontal está presente se hace que la bandera flag2 = 1, sino flag2 = 0.

**Paso 6:** Si la bandera flag2 = 0, el robot se moverá hacia adelante en línea recta, si la bandera tiene el valor complementario entonces se mueve hacia adelante.

Ventajas de este algoritmo:

* Pueden utilizarse solo algunos sensores y estos pueden ser tan simples como los detectores mecánicos de contacto.
* Se puede modificar el sistema de control electrónico lo que lo hace menos propensos a fallas.

(UNAM, 2013)

1. **Algoritmo de Tremaux:**

El algoritmo de Tremaux es un [algoritmo](https://es.wikipedia.org/wiki/Algoritmo) inventado por el ingeniero francés [Charles Tremaux](https://es.wikipedia.org/w/index.php?title=Charles_Tr%C3%A9maux&action=edit&redlink=1), para hallar la salida de un laberinto.

Lo primero que hay que hacer es marcar el camino que se va siguiendo, y seguir estos pasos:

* No siga el mismo camino dos 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.

Si se siguen estos pasos, aunque es posible que lleve horas, este sistema le llevará hasta la salida del laberinto, y si no la hay, de nuevo hasta la entrada.

1. **Algoritmo Pledge:**

Cuando no todas las paredes están conectadas (supongamos que la pared que escogimos es parte de un rectángulo dentro del laberinto), existe una modificación conocida como el algoritmo de Pledge, el cual funciona de la siguiente manera: escogemos una dirección arbitraria para avanzar, y cuando llegamos a una pared, utilizamos el algoritmo de la mano derecha. En el momento que la suma angular de las vueltas que hemos dado se vuelva cero, soltamos la pared y continuamos con lo dirección que habíamos escogido. Este algoritmo funciona sólo para laberintos en segunda dimensión. (Guillen, 2009)

1. **Algoritmo Vuelta Atrás:**

En su forma básica, la idea de backtracking se asemeja a un recorrido en profundidad dentro de un grafo dirigido. El grafo en cuestión suele ser un árbol, o por lo menos no contiene ciclos. Sea cual sea su estructura, existe sólo implícitamente. El objetivo del recorrido es encontrar soluciones para algún problema. Esto se consigue construyendo soluciones parciales a medida que progresa el recorrido; estas soluciones parciales limitan las regiones en las que puede encontrar una solución completa. El recorrido tiene éxito si, procediendo de esta forma, se puede definir por completo una solución. En este caso el algoritmo puede, o bien detenerse (si lo único que se necesita es una solución del problema) o bien seguir buscando soluciones alternativas (si deseamos examinarlas todas). Por otra parte, el recorrido no tiene éxito si en alguna etapa la solución parcial construida hasta el momento no se puede completar. En tal caso, el recorrido vuelve atrás exactamente igual que en un recorrido en profundidad, eliminando sobre la marcha los elementos que se hubieran añadido en cada fase. Cuando vuelve a un nodo que tiene uno o más vecinos sin explorar, prosigue el recorrido de una solución. (Diaz, 2013)

**Elección del algoritmo**

**¿Cuál algoritmo escogimos? ¿Por qué razón lo escogimos?**

El algoritmo que se decidió implementar en el proyecto, entre el resto de los mismos, es el de Tremaux, ya que este brinda una manera bastante fácil y ordenada de ir probando todas las posibles rutas que se pueden encontrar para salir de un laberinto. Con respecto al tiempo, puede que demore el robot (en este caso) en encontrar la salida, pero al tener posibilidades finitas de hacerlo, es seguro que la encontrará.

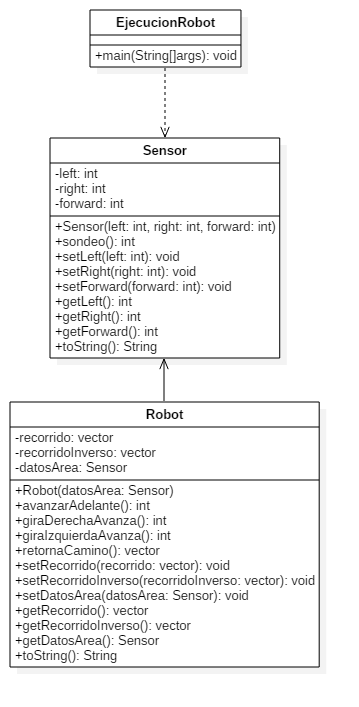
Por otro lado, no se escogió el de la mano derecha o seguidor de paredes según se conozca, ya que hay ocasiones en los que los laberintos (como en el caso de los circulares, a veces) poseen como “bloques” o “secciones” del mismo, que están desconectados del resto, y puede que en esa zona, se encuentre la salida. Y como algoritmo se trata de ir siempre verificando que haya pared a un costado (ya sea derecha o izquierda, según se escoja), en el caso anterior el robot no podría realizar el “salto” hacia esa zona separada del resto del laberinto. Por otra parte, tampoco se escogió el de Pledge, ya que implementaba el algoritmo de la mano derecha, con adición de que en este el robot siempre buscará una dirección (norte, sur, este, oeste), por lo que acá el salto entre bloques de laberinto no es ningún problema. Pero se vuelve a lo mismo, no es eficiente, o al menos no es lo que buscamos, ya que este no evalúa si ya ha pasado por un punto o no, como el de Tremaux.

Así mismo, por obvias razones que se mencionaron en la sección previa a esta, tampoco se escogió el algoritmo de azar, ya que lo que se busca es que el robot encuentre la salida del laberinto, y si no posee un sentido de orientación este jamás llegará, o al menos no lo sabrá, y seguirá corriendo el algoritmo, tratando de encontrar una salida. De igual forma con el de backtracking, ya que este funciona con una solución previamente definida, y si en dado caso algo en esta no sale según lo planteado, se vuelve a atrás y no es eficiente.

**Estructura de datos a utilizar:**

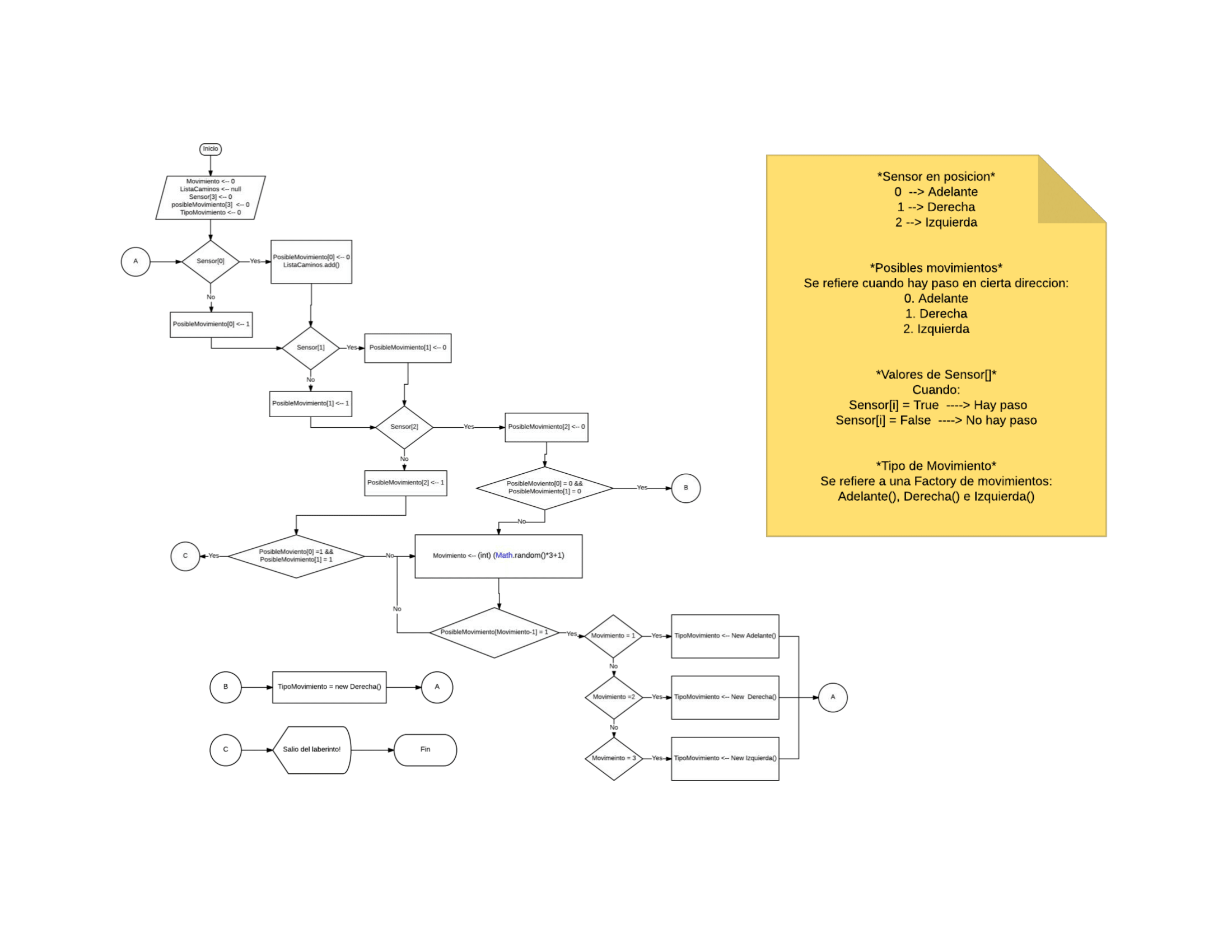
Se utilizará el tipo de dato vector para almacenar el recorrido del robot a lo largo del laberinto, porque se observó que su funcionamiento era ideal para el algoritmo escogido (Tremaux), porque se puede añadir elementos y eliminarlos a conveniencia, además de que proporciona un rápido acceso a sus datos en cualquier posición.

**Diagrama de clases:**

****

**Diagrama de flujo:**

(tomar los elementos “lista” como un vector)

****

# Bibliografía

Diaz, A. (1 de Febrero de 2013). *Academia.edu*. Obtenido de Academia.edu: http://www.academia.edu/4607412/Algoritmo\_de\_Backtracking\_Recursivo\_y\_no\_Recursivo\_para\_la\_Resoluci%C3%B3n\_de\_un\_Laberinto\_y\_su\_Aplicaci%C3%B3n\_en\_SDL

Guillen, P. (4 de Septiembre de 2009). *Pier.Guillen*. Obtenido de Pier.Guillen: http://pier.guillen.com.mx/algorithms/09-busqueda/09.4-camino\_laberintos.htm

UNAM. (13 de Abril de 2013). *ptolomeo.* Obtenido de ptolomeo: http://www.ptolomeo.unam.mx:8080/jspui/bitstream/132.248.52.100/307/11/A11.pdf