Universidad Del Valle de Guatemala
Algoritmos y estructura de datos
Catedrático: Douglas Barrios

Proyecto No. 1: Laberinto

Jorge Azmitia 15202
Jennifer Barillas 15307
Carlos Calderón 15219

¿Que algoritmos existen para salir de un laberinto ?

Algoritmo de Tremaux

Inventado por el ingeniero Charles Tremaux. el cual sugiere la siguiente serie de pasos:

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

Algoritmo random

Este es un método trivial, el cual simplemente depende de seguir el camino y al momento de tener la opción de girar hacia dos diferentes lugares, toma una decisión aleatoria, eventualmente se logra salir del laberinto, sin embargo puede llegar a ser extremadamente lento

Algoritmo de "seguir la pared"

También conocido como la regla de la mano izquierda o derecha. este algoritmo consiste en algo tan sencillo como que tanto la entrada como la salida del laberinto están conectadas. por lo tanto, si la persona que está dentro del laberinto, recorre este manteniendo la mano pegada a una de las dos paredes iniciales, sea cual sea. y sigue todo el laberinto así, eventualmente encontrará la salida, y de no existir ninguna salida regresará al punto de partida.

Algoritmo de pledge

Nombrado así por John Pledge. este algoritmo consiste en tomar de forma arbitraria una dirección por la cual ir. y que siempre gire hacia el mismo lado. en este algoritmo se tiene que ir tomando en cuenta los ángulos o giros que se dan al encontrar un obstáculo y girar en la dirección elegida. con esto se sabe si se dio una vuelta a una parte del laberinto, y si se regresa al giro anterior teniendo en cuenta que ya dio N vueltas y esta viendo hacia la dirección inicial, entonces ignora el giro y sigue hacia delante hasta encontrar el siguiente y eventualmente la salida. esto puede ser útil si en algún momento se encuentra uno con un laberinto segmentado. el cual podría solucionarse con el algoritmo anterior a menos que se le coloque como inicio en alguna parte central del laberinto y que esta no esté pegada a los bordes del laberinto.

Algoritmo Recursivo(con codigo en java)

Al algoritmo se le dan unas coordenadas (x,y) que son los valores iniciales, y si estos no están pegados a una pared del laberinto, el algoritmo automáticamente conseguirá unos adyacentes que lo estén. y se asegura de que estos valores no se hayan utilizado previamente. si los valores x,y al final son los que se designaron como el final. Va a guardar todos los previos como el camino que es la solución del laberinto.

El código:

```
int[][] maze = new int[width][height]; // The maze
boolean[][] wasHere = new boolean[width][height];
boolean[][] correctPath = new boolean[width][height]; // The solution to the maze
int startX, startY; // Starting X and Y values of maze
int endX, endY;     // Ending X and Y values of maze

public void solveMaze() {
    maze = generateMaze(); // Create Maze (1 = path, 2 = wall)
    for (int row = 0; row < maze.length; row++)
        // Sets boolean Arrays to default values
        for (int col = 0; col < maze[row].length; col++){
            wasHere[row][col] = false;
            correctPath[row][col] = false;
        }
    boolean b = recursiveSolve(startX, startY);
    // Will leave you with a boolean array (correctPath)
    // with the path indicated by true values.
    // If b is false, there is no solution to the maze
}
public boolean recursiveSolve(int x, int y) {
    if (x == endX && y == endY) return true; // If you reached the end
    if (maze[x][y] == 2 || wasHere[x][y]) return false;
    // If you are on a wall or already were here
    wasHere[x][y] = true;
    if (x != 0) // Checks if not on left edge
        if (recursiveSolve(x-1, y)) { // Recalls method one to the left
            correctPath[x][y] = true; // Sets that path value to true;
            return true;
        }
    if (x != width - 1) // Checks if not on right edge
        if (recursiveSolve(x+1, y)) { // Recalls method one to the right
            correctPath[x][y] = true;
            return true;
        }
```

```
    if (y != 0)  // Checks if not on top edge
        if (recursiveSolve(x, y-1)) { // Recalls method one up
            correctPath[x][y] = true;
            return true;
        }
    if (y != height- 1) // Checks if not on bottom edge
        if (recursiveSolve(x, y+1)) { // Recalls method one down
            correctPath[x][y] = true;
            return true;
        }
    return false;
}
```

Algoritmo backtracker

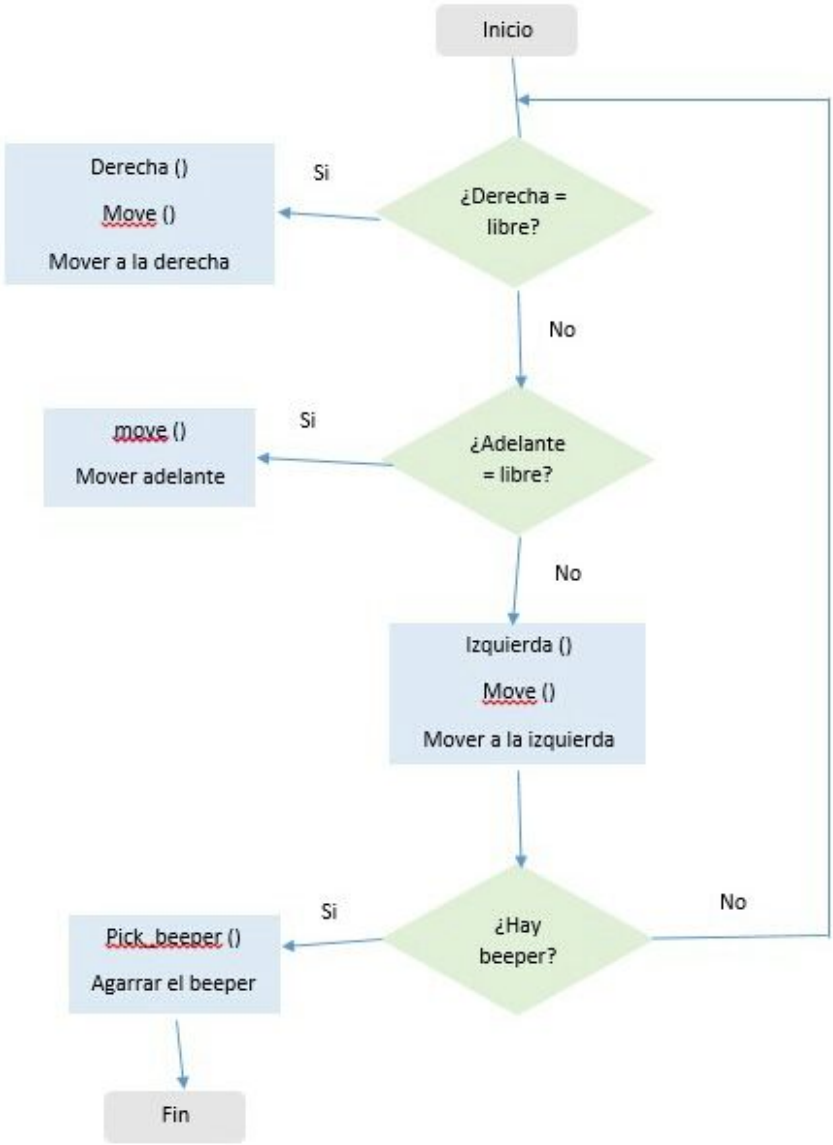
Este algoritmo se centra en el personaje dentro del laberinto, por lo general es rápido. Lo que hace este algoritmo es ver si ya se encontró con una pared o en un área ya conocida, regresa fracaso, asimismo si encuentra una salida retorna victoria. Sino, trata de forma recursiva de moverse en cuatro direcciones, (N,S,O,E). Cabe resaltar que este algoritmo, utiliza una pila con un espacio de memoria igual al del laberinto. Este dibuja una línea cuando prueba una dirección y lo borra cuando regresa fracaso.

Algoritmo a utilizar

Luego de estudiar las posibilidades, se vio que el algoritmo de Tremaux y el recursivo Backtracker tienen un garantía de solución bastante alta. Así también, se observó que tanto el algoritmo de Pledge como el de Seguidor de pared son aunque relativamente más simples que los ya mencionados, y por lo general son rápidos, aunque su garantía de solución es un poco más baja que los ya mencionados.

Por el momento, los algoritmos que nos parecieron más adecuados fueron el recursivo Backtracker, y el de Seguir a la pared. Sin embargo, por ahora se enfatizará un poco más en el de seguir a la pared, esto debido a que es más simple y aún se desconoce cómo va a funcionar el robot ante los algoritmos. Así que por ahora, se enfocará en este.

Diagrama de flujo:



Bibliografía

- https://www.cs.bu.edu/teaching/alg/maze/
- Turtle Geometry:computer as a medium for exploring mathematics
- MIT AI, Uses of technology to enhance education
- Edouard lucas: recreations Mathematiques V1, 1882