Universidad Del Valle de Guatemala
Algoritmos y Estructura de Datos
Sección 20

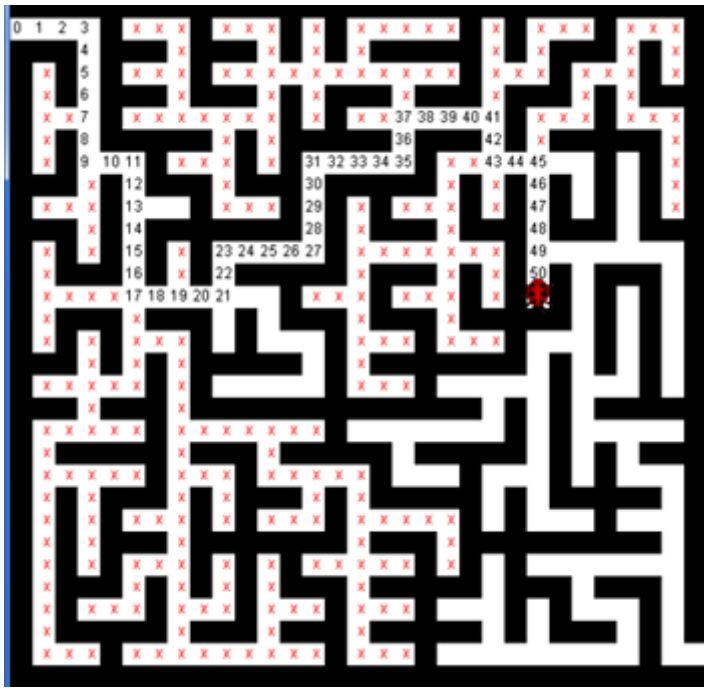
Juan Pablo Zea 15401
José Martínez 15163
David Sanchinelli 15549
Diego Hurtarte 15022

Proyecto 1

Simulación

Esta parte del proyecto consiste en realizar el algoritmo seleccionado (backtracking) utilizando código java. En nuestro caso, se decidió utilizar matrices para poder representar el laberinto, teniendo un carácter como una pared y otro como espacio vacío.

El algoritmo consiste en avanzar a través del laberinto hasta llegar a una división de caminos, este escogerá uno al azar y lo recorrerá hasta encontrar otra división, en la cual elegirá otra vez al azar, o un camino sin salida, por lo cual se regresara hasta la intersección anterior. Únicamente se podrán recorrer el mismo camino dos veces, uno cuando se entra por primera vez y, si es necesario, otra para regresar a la intersección anterior.



Arnold, Jarka. 2010

- *Descripción de clases*
    - Proyecto1

    Este es el programa principal en el cual se ejecutara el programa.

- Robot

Esta clase simula al robot que se utilizara al final del proyecto. Esta moverá al robot y verificara si el espacio que esta adelante o a cualquiera de sus dos lados esta vacío para poder proceder con el siguiente movimiento.

Atributos

- Posx: este atributo contiene la coordenada en x de la matriz, para poder evaluar si se puede realizar un movimiento.
- Posy: este atributo contiene la coordenada en y de la matriz, junto con posx se puede obtener un punto específico de la matriz, la cual es un laberinto.
- Dirección: este atributo contiene el movimiento que se quiere realizar en el laberinto, 1 si es hacia arriba, 2 si es a la izquierda, 3 si es hacia abajo y 4 si es hacia la derecha.

Métodos

- Analizar: este método analiza la posición en la que se encuentra en el laberinto.
- moverNorte: mueve el robot hacia arriba un espacio del laberinto si existe el espacio.
- moverOeste: mueve el robot hacia la izquierda un espacio del laberinto si el movimiento es permitido.
- moverEste: mueve el robot hacia la derecha si existe un espacio vacío en el laberinto.
- moverEste: mueve el robot hacia la izquierda un espacio, si no hay una pared.
- moverSur: mueve el robot un espacio hacia abajo si es permitido por el laberinto.
- LibreAdelante: este método verifica que el espacio de adelante del robot este libre, este es la representación del sensor delantero del robot que se utilizara.
- libreDerecha: este método verifica el lado derecho del robot para ver si en este hay una pared, este método simula el sensor derecho del robot.
- libreIzquierda: este método mira si del lado izquierdo del robot esta vacío o existe algún muro, este método representa el sensor izquierdo del robot.

- Subcamino:

Esta clase guarda los caminos que se recorren durante el laberinto, y tiene registro de lo que se realiza.

Atributos

- Subcamino: es un atributo de tipo vector que la información del recorrido del laberinto realizado, se utiliza vector ya que no se sabe que tan largo será.
- ii: es un contador que nos indica la posición del vector en la que se encuentra

Métodos

- setSubcamino: este método permite guardar información en el vector subcamino.
- getSubCamino: este método permite obtener el contenido que se encuentra en el vector subcamino.
- agregarMovimiento: este método permite la creación de un nuevo espacio en el vector.
- eliminarMovimiento: este método permite la eliminación del ultimo contenido del vector.

- CaminoP

Este método permite guardar los caminos previamente recorridos, para poder evitar volverlos a recorrer.

Atributos

- caminoP: es un atributo de tipo vector que sirve para guardar los subcaminos.
- I: es un contador que sirve para poder saber la posición en el vector caminoP.

Metodos

- setCamino: es un método que sirve para poder ingresar información dentro de una posición del vector
- getCamino: es un método el cual sirve para poder obtener la información del vector.
- agregarSubcamino: este método permite agregar un dato al vector, se ingresara un dato de tipo subcamino, descrito anteriormente.
- eliminarSubcamino: este método permite eliminar un elemento del vector.

Ref. Bibliográficas

Arnold, Jarka. Et. al. (2010). Finding the way through a labyrinth: backtracking as a solving strategy. http://www.java-

online.ch/gamegrid/gamegridEnglish/index.php?inhalt_mitte=backtracking/labyrinth.inc.php [10/08/2016]