Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2022.

RESCATE Y EXTRACCIÓN - CHAPIN WARRIORS

202003585 – Diego Andrés Huite Alvarez

Resumen

El proyecto *rescate y* extracción es un software elaborado para la empresa *chapín Warriors.* Esta empresa se encarga de patrullar ciudades en conflicto y ejecutar misiones para el rescate de civiles y misiones para la extracción de recursos dentro de la ciudad. El software se encarga de mostrarle al usuario los robots que se tienen disponibles para realizar la misión y de mostrarle las ciudades al usuario. Para que así el usuario sea capaz de visualizar el camino que tomará el robot para completar su misión. Esto nos acerca cada vez más a la automatización de tareas y permite que los seres humanos no sufran riesgos al adentrarse en estas ciudades en conflicto.
La empresa cuenta con robots capaces de combatir en las zonas de conflicto y extraer recursos dispersos por la ciudad, y también, cuenta con robots de rescate para unidades civiles, estos, no tienen la capacidad de combatir ante unidades hostiles.

Palabras clave

Estructura de datos; Forma de organizar datos en un computador para usarlos de manera eficiente

Matriz: Estructura que permite almacenar información en filas y columnas.

Abstract

The *rescue and extraction* project is software developed for the company *Chapín Warriors*. This company is in charge of patrolling cities in conflict and executing missions to rescue civilians and missions to extract resources within the city. The software is in charge of showing the user the robots that are available to carry out the mission and showing the cities to the user. So that the user is able to visualize the path that the robot will take to complete its mission. This brings us ever closer to the automation of tasks and allows human beings to be safe when entering these cities in conflict.
The company has robots capable of fighting in conflict zones and extracting resources scattered throughout the city, and it also has rescue robots for civilian units, which do not have the ability to fight against hostile units.

Keywords

Data Structure: A way of organizing data in a computer so that it can be used efficiently.

Matrix: Structure that allows information to be stored in rows and columns.

Introducción

El humano ha tenido conflictos con otros humanos desde hace tiempo, y es algo que probablemente siga así. Hasta el punto de lastimar a otros causando estragos en las ciudades donde estos viven. Se propone que sean robots los que ejecuten misiones de rescate de civiles y extracción de recursos en estas ciudades tan hostiles. El software analiza la ciudad a través de un dron. Y calcular el camino que deben de seguir los robots para llevar a cabo una misión. Estos robots pueden salvar muchas vidas de gente en estas ciudades sin arriesgar vidas al adentrarse en estas ciudades peligrosas.

En las ciudades existen 6 tipos de celdas, la celda verde es un punto de entrada seguro para el robot, celda roja es una unidad militar, celda azul es una unidad civil, celda blanca es camino libre, celda negra es intransitable, y celda gris es un recurso.

1. Identificación del problema

En una ciudad de MxN dimensiones pueden existir distintos caminos para un solo destino. Hay muchas formas de recorrer una ciudad y llegar a nuestro lugar destino. Ester es el principal problema al que nos enfrentamos al intentar plantear una solución para hallar caminos en donde el robot pueda extraer o salvar un recurso o una unidad civil.

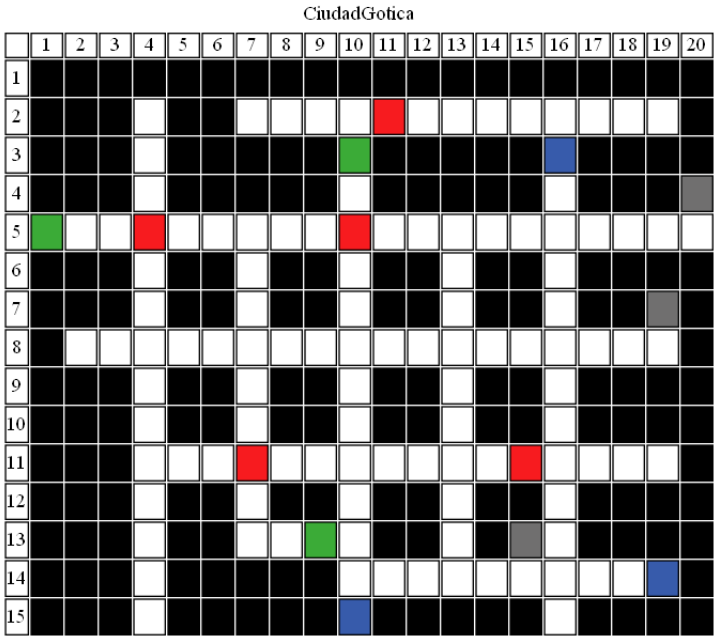

Figura 1. Ejemplo de ciudad.

Fuente: elaboración propia

Suponiendo que el recurso que se desea extraer se encuentra en la fila 4 y columna 20. Y desde el punto de entrada en la fila 13 y columna 9 hay muchísimos caminos que se pueden tomar, para comenzar a subir podemos hacerlo a través de todas las columnas y teniendo en cuenta que podemos atravesar las celdas rojas solo si el robot es capaz de hacerles frente. ¿Cómo sabemos que ruta tomar?, ¿Qué columna escogemos?, ¿Si quiera el camino que estamos tomando nos lleva al destino? Estas son algunas de las interrogantes que se plantearon al inicio de la búsqueda de un algoritmo que permitiera solucionar este problema.

2. Solución al problema

Para solucionar este problema se hizo uso de un algoritmo en teoría de grafos conocido como “Depth first search” o por sus siglas “dfs”. Es un algoritmo recursivo que permite recorrer cada celda y cada posibilidad de hacia dónde ir, a continuación, una

pequeña orientación al funcionamiento de este algoritmo:

Algoritmo dfs

Pasos:

1. El robot se posiciona en una celda de entrada
2. El robot considera las coordenadas de la celda de destino, existen 8 posibilidades. Estas son: (celda de destino arriba y a la izquierda, celda de destino justo debajo de donde se encuentra,etc.)
3. Dependiendo del paso anterior, le daremos cierta prioridad a los movimientos, por ejemplo, si la celda destino está justo arriba de nosotros, el robot le dará máxima prioridad a dar pasos hacia arriba y así con cada uno de los casos.
4. Antes de dar el paso el robot se asegura de que la celda a la que desea moverse no es una celda que le impida pasar y también se asegura de que no sea una celda antes recorrida.
5. Da el paso. Y se asegura que no esté en la celda de destino a la que deseamos llegar.
6. Esta función al ser recursiva recorre exhaustivamente cada celda, pero cuando ya no puede avanzar más hacia cierta dirección, la última celda visitada retornará un valor de falso, refiriéndose a que no hay más por recorrer acá, e indicándole a las demás celdas que ese camino ya fue explorado sin encontrar el destino en ese camino

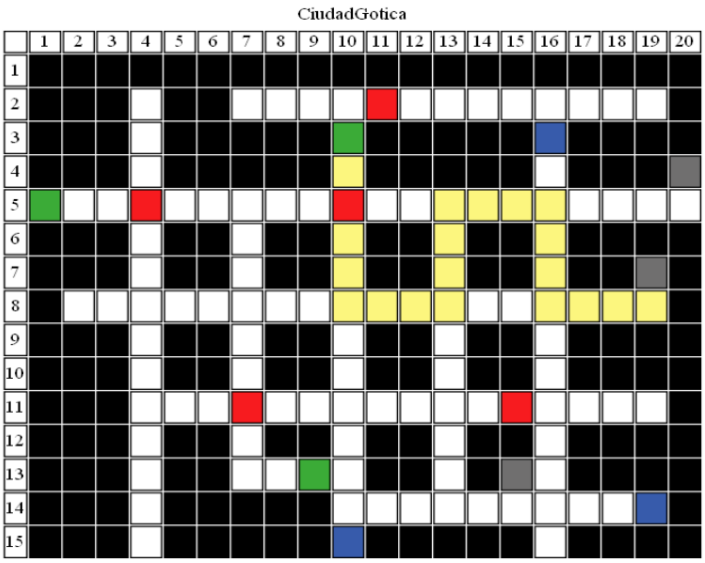

Figura 2. Ejemplo de ruta.

Fuente: elaboración propia

Siguiendo los pasos del algoritmo previamente mencionado se encuentra el camino hacia el recurso deseado, la ruta descubierta puede parecer extraña pero este algoritmo no encuentra el mejor camino.

3. Estructuración del software.

Clases usadas para el funcionamiento del software:

1. Ciudad
2. NodoCiudad
3. ListaCiudad
4. Robot
5. NodoRobot
6. ListaRobot
7. Celda
8. NodoInterno
9. Matriz
10. NodoEncabezado
11. ListaEncabezado

Las clases de la 1 a la 3 se encargan de guardar todas las ciudades en una lista doblemente enlazada, guardando datos como nombre, filas, columnas, el mapa de la matriz, etc.

Las clases de la 4 a la 6 se encargan de guardar toda la información sobre los robots cargados en el archivo .xml

Las clases de la 7 a la 11 son las que se encargan de la estructuración de una matriz con nodos apuntando hacia arriba, abajo, derecha e izquierda.

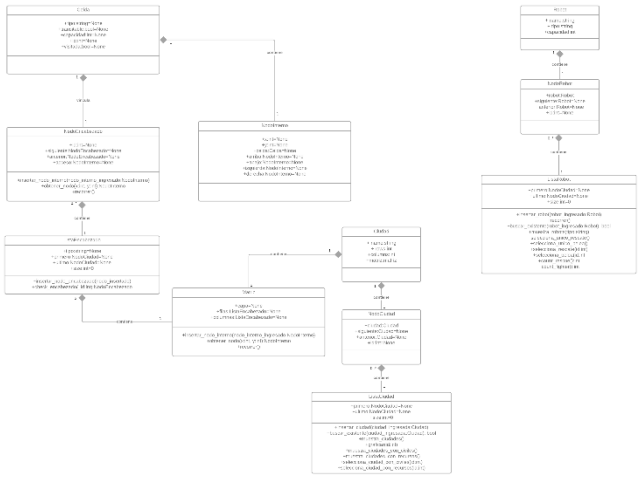

Figura 3. Diagrama de clases del software.

Fuente: elaboración propia

Conclusiones

Las estructuras de datos permiten un mejor manejo de ciertos algoritmos, en este caso la matriz dispersa ayudó a tener nodos internos apuntando a las 4 direcciones cardinales facilitando así la navegación del robot. Cosa que hubiera sido imposible con listas, tuplas, etc.

Los algoritmos de búsqueda de nodos en grafos fueron la clave para resolver este problema y encontrar un camino hacia el destino del robot.

Referencias bibliográficas

Búsqueda en profundidad. (2021, 27 de junio). *Wikipedia, La enciclopedia libre*. Fecha de consulta: 22:22, marzo 29, 2022 desde https://es.wikipedia.org/w/index.php?title=B%C3%BAsqueda_en_profundidad&oldid=136615639.

Matriz dispersa. (2019, 23 de octubre). *Wikipedia, La enciclopedia libre*. Fecha de consulta: 22:23, marzo 29, 2022 desde https://es.wikipedia.org/w/index.php?title=Matriz_dispersa&oldid=120684678.