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 2021.

PROYECTO 1. ALGORITMO PARA LA AGENCIA GUATEMALTECA DE EXPLORACIÓN ESPACIAL

202003381 – Luisa María Ortíz Romero

Resumen

Se desarrolla un algoritmo para la Agencia Guatemalteca de Investigación Espacial que permite que el robot r2e2 pueda encontrar el camino que consuma menos combustible moviéndose ortogonalmente en un terreno cuya información es obtenida mediante un archivo XML recibido gracias a la comunicación con el satélite Quetzal01.

El programa presentado se ejecuta en memoria dinámica y funciona en base al algoritmo de Dijkstra, encontrando un camino que parte de la coordenada inicial y se detiene en la coordenada final, y que considera un movimiento ortogonal, de acuerdo con las capacidades motrices de r2e2.

El algoritmo es capaz de encontrar el camino óptimo de cualquier terreno recibido, convirtiendo cada coordenada en un nodo de una matriz ortogonal, siempre y cuando el número de filas y columnas estén entre 1 y 100.

La solución propuesta garantiza el gasto mínimo de combustible, lo cual permite la exploración de un mayor número de terrenos.

Palabras clave
nodo, lista, grafo, ortogonal, TDA

Abstract

An algorithm is developed for the Guatemalan Space Research Agency that allows the r2e2 robot to find the road that consumes less fuel by moving orthogonally on a terrain whose information is obtained through an XML file received thanks to communication with the Quetzal01 satellite.

The presented app is executed in dynamic memory and works based on the Dijkstra algorithm, finding a road that starts from the initial coordinate and stops at the final coordinate, and that considers an orthogonal movement, according to the r2e2's motor capabilities.

The algorithm can find the optimal road of any received terrain, converting each coordinate into a node of an orthogonal matrix, as long as the number of rows and columns are between 1 and 100.

The proposed solution guarantees the minimum cost of fuel, which allows the exploration of a greater number of fields.

Keywords
Node, list, orthogonal, ADT

Introducción

El método más aceptado para encontrar rutas óptimas, en este caso, en base al consumo de combustible es el algoritmo de Dijkstra, parte de la teoría de grafos. Para lograr el funcionamiento óptimo del programa se utilizan TDA (Tipos de dato abstracto) que permiten la asignación de memoria dinámica, para no afectar el rendimiento del robot r2r2.

Para cumplir con el algoritmo se utilizan los TDA: Lista enlazada simple y Lista enlazada doble, según sea el caso. Los terrenos son guardados en una lista enlazada simple y las coordenadas de estos se almacenan en una lista enlazada doble, esta última con el propósito de recorrerla en ambas direcciones (hacia el inicio y hacia el fin) sin problemas.

El desarrollo de los TDA es logrado a través de la abstracción del programa utilizando Programación Orientada a Objetos en el lenguaje Python.

A. Algoritmo de Dijkstra

El Algoritmo de Dijkstra permite encontrar el camino más corto entre dos vértices de un grafo. Para describir el algoritmo se parte de la definición de un grafo, como la representación gráfica de un conjunto de nodos o vértices unidos por enlaces llamados aristas o arcos.

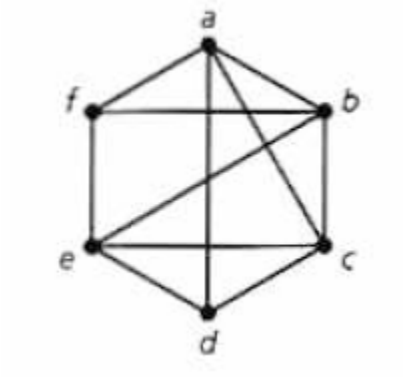

Figura 1. Representación de un grafo
Fuente: Matemáticas discreta y combinatoria. Ralph R. Grimaldi. (1977). P.551

Para la solución del problema, cada coordenada del mapa obtenido se convierte en un nodo y la arista corresponde a la cantidad de combustible necesaria para pasar por él.

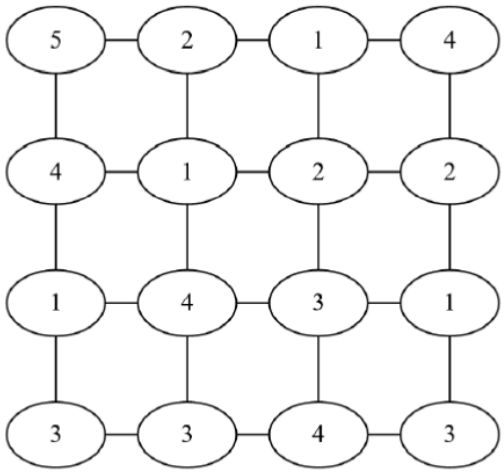

Figura 2. Representación de los nodos de un terreno

Fuente: Enunciado de Proyecto. Escuela de Ciencias y Sistemas

La solución es encontrada mediante el proceso de etiquetado, en el que se analizan los nodos a los que

el robot es capaz de moverse en ese instante (arriba, abajo, derecha o izquierda) y se etiquetan tomando el nodo del que proviene y la cantidad de combustible que se acumula al llegar hasta él, para después elegir el nodo que acumule menos combustible y se considera un nodo terminal (ya analizado).

A lo largo del análisis el algoritmo compara todas las etiquetas existentes y no terminales, aunque no sean vecinos próximos al nodo actual, sin embargo, eso no es inconveniente, pues al llegar a la coordenada final, utiliza la etiqueta anterior para regresar posiciones hasta llegar a la coordenada inicial, encontrando la ruta óptima.

El etiquetado antes mencionado se realiza de la siguiente manera:

$$[A, a]_{(i)} \qquad (1)$$

Donde:

A= Nodo predecesor

a= Acumulado de combustible

i= Número de iteraciones realizadas

B. Tipos de Dato Abstracto (TDA)

Muchos de los lenguajes de programación de la actualidad implementan estructuras de datos como listas, tuplas y diccionarios, sin embargo, la mayoría de estos son ejecutados utilizando memoria estática, es decir, se le asigna un espacio en memoria fijo, lo que provocaría desperdicio o falta de memoria.

Para procurar el funcionamiento óptimo del robot r2e2 se propone una solución que se ejecute en memoria dinámica, es decir, crezca su espacio mientras el programa así lo requiera. La estructura dinámica es lograda gracias a la abstracción de los elementos más importantes de los terrenos.

B.1. Listas enlazadas

Este tipo de estructura es de forma lineal y permite almacenar elementos comúnmente llamados nodos, en donde cada uno de ellos puede almacenar distintos datos y un apuntador hacia otros nodoas.

Son estructuras dinámicas que pueden almacenar datos que cambian con frecuencia. Las listas enlazadas se propagan y se retraen para que sean más sencillas al añadir o eliminar información y permiten almacenar información en diferentes posiciones de memoria

B.1.1 Listas Enlazadas Simples

Es un conjunto de nodos con una sola dirección y forman una estructura de datos lineal. Cada nodo es un objeto que almacena un elemento (datos) y una dirección a otro nodo.

La dirección que guarda un nodo de otro nodo se llama puntero y el salto que los une, se llama salto de puntero. El primer nodo de una lista es la cabecera y el último se denomina final. El nodo final apunta a null, haciendo referencia a un nodo nulo, es decir, inexistente.

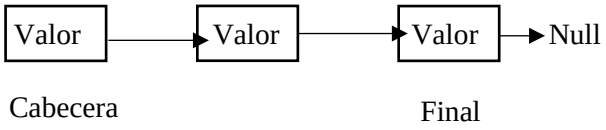

Figura 3. Representación de una Lista enlazada simple

Fuente: Elaboración propia

B.1.2 Listas Enlazadas Dobles

Cuando los nodos tienen un doble enlace, es decir, un enlace al siguiente (dirección sucesora) y otro enlace al nodo anterior (dirección predecesora), entonces se conocen como lista doblemente enlazada.

La lista doblemente enlazada aventaja a la lista simplemente enlazada en su recorrido en ambos sentidos, de izquierda a derecha como de derecha a izquierda, los nodos primero y último apuntan a null, que simboliza el final de la lista. El puntero anterior permitirá el acceso hacia el elemento anterior de la lista, mientras que el puntero siguiente permitirá el acceso hacia el próximo elemento. Su desventaja con relación a las listas simplemente enlazadas es que ocupan más memoria por nodo.

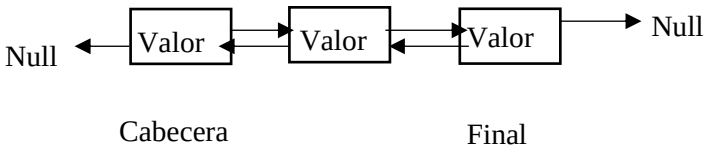

Figura 4. Representación de una Lista enlazada doble

Fuente: Elaboración propia

Conclusiones

La Programación Orientada a objetos permite abstraer los elementos más importantes de los terrenos recibidos por la comunicación con el satélite Quetzal01 y utilizar la información a lo largo del programa en numeroso tipo de métodos y operaciones que garantizan la viabilidad del algoritmo.

La utilización de estructuras dinámicas como Tipos de Dato Abstracto permite que el robot r2e2 ejecute el programa con la memoria necesaria y no afecten a su rendimiento, además de permitir las separación de los datos en nodos que guardan la información necesaria, creando una Lista enlazada simple para almacenar los terrenos y asociando a cada nodo de esta otra lista, esta del tipo enlazada doble. De esta manera se cuenta con una estructura con una lista en una sola dirección que contiene información de los terrenos y una lista con apuntadores en dos direcciones que permiten recorrer las coordenadas hacia adelante y hacia atrás hasta encontrar la ruta que consuma menos combustible.

El algoritmo de Dijkstra demostró ser apto para resolver el problema, encontrando la ruta que consuma menos combustible. La abstracción de los datos en nodos facilita la tarea de el etiquetado y cambio de nodos. Sin embargo, el programa parece presentar ambigüedades cuando se abstrae un terreno con 100 filas o más, de la misma manera con las columnas, esto asociado a la gran cantidad de iteraciones necesarias para comparar todos los nodos, por lo que se recomienda reducir el tamaño de los terrenos a explorar.

Referencias bibliográficas

Álvarez Morales, A. M. (2015). Implementación de un TAD que gestione una lista enlazada de nodos de tipo Árbol AVL y aplicación práctica.

Cristina, H. L. E. ED Lineales.

Martínez, L. M., Colina, E. R., & Ramírez, C. M. (2013). Toma de Decisiones Basadas en el Algoritmo de DIJKSTRA. *Redes de Ingeniería*, *4*(2), 35-42.

Anexos

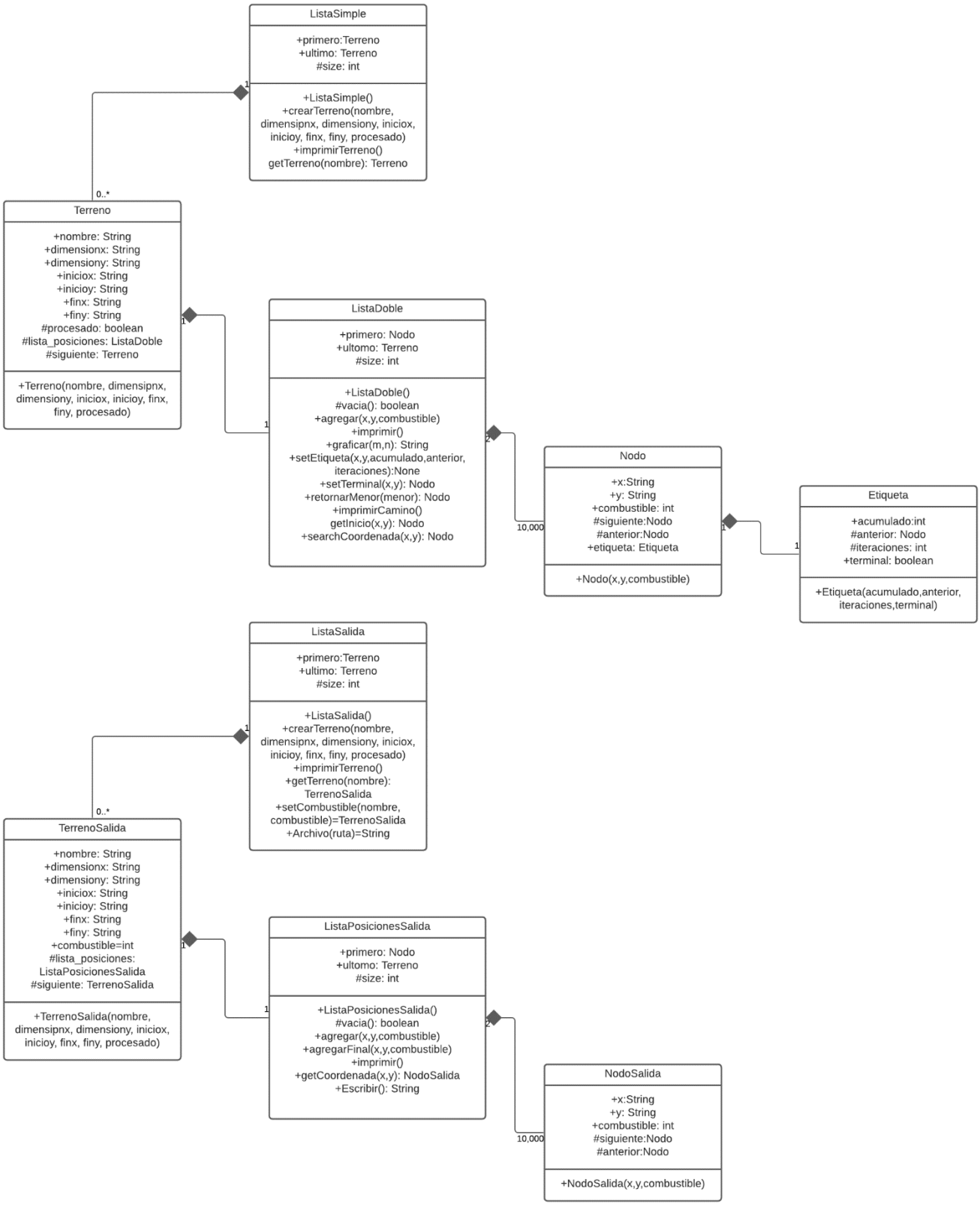

Figura 5. Diagrama de clases que modela la solución del software

Fuente: Elaboración propia