Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2021.

PROYECTO 1

202004765 – Javier Alejandro Gutierrez de León

Resumen

Se realizó un programa, utilizando diferentes estructuras de datos, principalmente matrices ortogonales, el cual tiene la capacidad de recibir información acerca de un terreno de un planeta en un archivo XML, siendo analizado con cElementTree, y al leer esta información un robot tiene que decidir por el mejor camino desde un nodo de inicio hasta un nodo final, consumiendo la menor cantidad de gasolina, ya que esta es limitada, por lo que se utilizó el algoritmo "A*" para tomar la mejor decisión, teniendo en consideración la distancia y el menor consumo de gasolina.

Por último, se genera un reporte también en XML de la decisión que ha tomado indicando únicamente las coordenadas del camino elegido y la gasolina que se ha consumido, además de generar una imagen utilizando Graphviz, también de la decisión que ha tomado, pero pintando el camino escogido sobre la matriz que se le ha proporcionado como entrada.

Palabras clave

Dijkstra: operaciones para determinar la longitud del camino más corto entre dos vértices
Nodo: es un punto de intersección, conexión o unión de varios elementos
Algoritmo: instrucciones no ambiguas, ordenadas y finitas utilizadas para solucionar un problema.

Abstract

Was made a program, using different data structures, mainly orthogonal matrices, which has the ability to receive information about a planet's terrain in an XML file, being analyzed with cElementTree, and when reading this information a robot has to decide along the best path from an initial node to an end node, consuming the least amount of gasoline, since this is limited, using "A *" algorithm to make the best decision, taking into account the distance and the lowest gasoline consumption.

Finally, a report is generated in XML of the decision that has been made, indicating only the coordinates of the chosen path and the gasoline that has been consumed, in addition to generating an image using Graphviz, also of the decision that has been made but painting the path chosen over the matrix provided as input.

Keywords

Dijkstra: operations to determine the length of the shortest path between two vertices
Node: it is a point of intersection, connection or union of several elements
Algorithm: unambiguous, ordered, and finite instructions used to solve a problem.

Introducción

En este ensayo se describe a grandes rasgos las funcionalidades del proyecto, indicando los tipos de estructura de datos, algoritmos y librerías utilizadas para brindar una relación entre estos y las funcionalidades que cumplen dentro del programa.

La importancia de este proyecto está en que cada día el concepto de viajes a otros planetas es más común, el utilizar conceptos como los que utilizan las aplicaciones como Waze para encontrar rutas óptimas dentro del tráfico de una ciudad, son importantes para crear interés hacia la exploración espacial.

Desarrollo del tema

Para el desarrollo de este proyecto fue necesario utilizar el paradigma orientado a objetos, ya que este nos ayuda en la creación de "n" terrenos que se deben analizar durante la ejecución del programa, así como la implementación distintos algoritmos, estructuras de datos y librerías para crear e interpretar archivos, siendo estos:

Algoritmo A*: Es un algoritmo de búsqueda creado por Nils Nilsson, con el fin de incrementar el algoritmo Dijkstra. Este algoritmo se utilizó para lograr encontrar la mejor ruta por la cual el robot debe pasar para llegar al nodo destino. Como agregado para aumentar la precisión del algoritmo se evaluó también un estado futuro en el caso se tomará la decisión que se está evaluando para evitar caer en nodos aparentemente mejores.

$$F(n) = g(n) + h(n) + F(n+1) \quad (\mathbf{1})$$

Donde:

- F(n): costo total de utilizar el nodo.
- g(n): Valor absoluto de distancia entre el nodo actual y el nodo final
- h(n): valor de gasolina que representa recorrer el nodo
- F(n+1): costo total del siguiente nodo elegido en el caso se utilice el que se está evaluando.

"El algoritmo encuentra siempre el camino de menor costo entre un nodo origen y uno objetivo siempre y cuando exista." (Aymar S., 2019)

TDA (Tipo de dato abstracto): Es una colección de datos con operaciones infinitas que forman un modelo matemático. Entre las estructuras más conocidas y utilizadas para el proyecto son las siguientes:

- **Lista simplemente enlazada:** Estructura de datos en la cual cada dato (nodo) apunta al siguiente, con el fin de referenciarse entre sí y al guardarse en memoria poder llamar a cada uno de estos datos conociendo la posición del primero. Esta estructura de datos se utilizó para pasar los datos de un método a otro.

- **Lista doblemente enlazada:** Estructura de datos en la cual cada dato (nodo) apunta al siguiente y el siguiente apunta al anterior, con el fin de referenciarse entre sí y al guardarse en memoria poder llamar a cada uno de estos datos conociendo la posición de cualquiera. Esta estructura se utilizó de base para formular la forma en al cual se tendrían que ir enlazando los

nodos ya existentes con los que se fueran agregando en la matriz ortogonal que se elaboró.

- **Matriz ortogonal:** Estructura de datos en la que cada dato (nodo) apunta hacia arriba, abajo derecha e izquierda con el fin de poder recorrer este arreglo por fila y columna. Utilizando los parámetros de coordenadas de los datos de entrada ordenando de esta forma las filas, formando así la matriz ortogonal utilizada tanto para guardar los datos de entrada como los datos de salida con la ruta a utilizar.

cElementTree: API que parsea y crea datos XML por medio de métodos que facilitan la extracción y escritura de datos. Utilizada en el proyecto para poder leer los archivos de entrada, así como escribir el archivo de salida.

Graphviz: Programa de visualización gráfica de código abierto con el cual se pueden crear gráficos para presentar información estructural. Utilizado en el proyecto para generar una imagen en la cual se visualiza el mapa completo y la ruta que se obtuvo por el algoritmo.

Conclusiones

Utilizar el lenguaje Python fue de gran ayuda, ya que al ser un lenguaje mas simple se lograron realizar funciones complejas en pocas líneas de código, las cuales hubieran sido muchas mas si se hubiese utilizado algún otro lenguaje de programación.

La utilización de la memoria dinámica con los TDA al principio parecía algo complicada, pero al entender bien estos se lograron implementar funciones que para aspectos prácticos de búsqueda y toma de decisiones con el algoritmo implementado fue más práctico que utilizar.

La manipulación de archivos XML por medio de cElementTree fue muy sencillo ya que con las funciones que este posee fue fácil extraer la información dentro del archivo, ahorrando la interpretación de línea por línea y la eliminación de caracteres que no fueran relevantes que se debería realizar si no utilizáramos esta librería.

La utilización de Graphviz fue complicado ya que esta herramienta tiene varios usos, por lo que la confusión por parte paginas no oficiales dificultó su uso, pero al leer la documentación se logró implementar de forma rápida al programa.

Referencias bibliográficas

- Aymar, S., Barahona, G., Gomez, M., & Martinez, C. (2019). INTELIGENCIA ARTIFICIAL (BÚSQUEDA A ESTRELLA). UNIVERSIDAD MAYOR DE SAN SIMÓN FACULTAD DE CIENCIAS Y TECNOLOGÍA. http://www.cs.umss.edu.bo/doc/material/mat_gral_139/Busqueda%20A%20estrella-2.pdfMuniz

- [Estructura de datos] -Lista doblemente enlazada, use Python para crear una lista simple doblemente enlazada (explicación detallada) - programador clic. (s. f.). Programmerclick. Recuperado 16 de agosto de 2021, de https://programmerclick.com/article/2000768743/

- F. T., & Severance, C. R. (2015). Python para informaticos: Explorando la informacion. Createspace Independent Publishing Platform.

- Graphviz. (s. f.). Graphviz. Recuperado 15 de agosto de 2021, de https://graphviz.org/

- xml.etree.ElementTree — La API XML de ElementTree — documentación de Python - 3.9.6. (s. f.). docs.Python. Recuperado 16 de agosto de 2021, de https://docs.python.org/es/3/library/xml.etree.elementtree.html

Anexos