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: RUTA ÓPTIMA

202000166 – Gerson Rubén Quiroa del Cid

Resumen

El proyecto presente es un programa capaz de encontrar una ruta con el menor gasto de combustible para un robot de exploración, llamado r2e2.

El usuario solamente tiene que ingresar la ruta del archivo con extensión .xml y luego se empezará a ejecutar los terrenos y a guardarse en memoria.

Luego el usuario tiene la opción de procesar los terrenos, que es donde la magia ocurre, la salida será el camino indicado con menor combustible y también el gasto total que dicho camino consumirá.

Cuando haya sido procesado, el usuario puede generar un archivo xml donde se encontrará con algunos datos de archivo que ingresó y con datos extras como: el consumo de combustible y las coordenadas por donde r2e2 puede pasar para consumir el menor combustible posible.

Palabras clave
Ruta, algoritmo, xml, lista enlazada, grafos.

Abstract

The present project is a program capable of finding a route with the lowest fuel consumption for an exploration robot, called r2e2.

The user only has to enter the path of the file with an .xml extension and then the terrain will begin to be executed and saved in memory.

Then the user has the option of processing the terrain, which is where the magic happens, the exit will be the indicated path with less fuel and also the total expense that said path will consume.

When it has been processed, the user can generate an xml file where they will find some data from the file that they entered and with extra data such as: fuel consumption and the coordinates through which r2e2 can pass to consume the least fuel possible.

Keywords
Path, algorithm, xml, linked list, graphs

Introducción

El presente proyecto tiene como finalidad encontrar la ruta con menor gasto de combustible para un robot de exploración, llamado r2e2. Con el lenguaje de programación Python, muy versátil y de fácil uso, se programará un algoritmo capaz de detectar el mejor camino posible.

Básicamente el programa pide al usuario que ingrese la ruta con los terrenos donde r2e2 estará explorando y el programa en cuestión de segundos proporcionará una ruta que el robot debería de seguir.

Luego, se tiene la opción de que el usuario pueda graficar los terrenos para tener una visualización mejor de cada terreno.

Desarrollo del proyecto

El análisis del presente proyecto se empezó primero que nada investigando acerca de las herramientas que se utilizarían en la misma como pueden ser las listas enlazadas, nodos, archivos xml, etc.

Después, se comenzó a elaborar el diagrama de clases de nuestro proyecto, como una visión general de todo el programa, creando las clases que usaremos en lo largo del proyecto. Dicho diagrama se puede visualizar completo en la sección de Apéndices.

Pasaremos a describir brevemente las clases que se utilizaron:

Las clases Nodo y LinkedList:

Estas clases se utilizaron para crear una estructura de dato muy conocida como lo son las listas enlazadas, en resumidas cuentas, se creo una lista dinámica donde se puede agregar, eliminar y buscar cualquier dato que haya sido ingresada en dicha lista.

En este caso se ingresaron los terrenos que fueron analizados, así como también las posiciones de cada terreno, que varían dependiendo del terreno.

Las clases Terreno y Posiciones:

En estas clases se almacena toda la información que se encuentra en el archivo de entrada xml, en la clase Terreno se almacenan datos como la posición inicial, final, la dimensión, las posiciones.

La clase posición se usó para guardar todas las posiciones de las matrices, como se mencionó anteriormente.

El módulo Archivo:

Este módulo es el encargado de procesar el archivo de entrada y pasarlos a la lista enlazada, con un algoritmo que posteriormente se explicará.

También, en este módulo, se escribe el archivo de salida, con los nuevos datos y se crea en una ruta específica que el usuario ingrese. La extensión será igual que la entrada, es decir, xml.

Módulo Algoritmo:

Probablemente este módulo sea el más importante del programa, ya que es el algoritmo que genera la ruta que menos consumo de gasolina emite el robot. Esta clase consiste en varias funciones, que en su conjunto hace que el algoritmo funcione de manera muy eficaz.

Posteriormente se describirá dicho algoritmo para la mejor comprensión de estos algoritmos indispensables para el programa.

Módulo Grafica:

Este es el último módulo del programa y es el encargado de crear un grafo del terreno que se seleccione. En la carpeta donde se encuentre el programa, ahí mismo se guardar la gráfica en formato pdf y gv.

Luego de echar un breve vistazo a las clases y módulos que se utilizó en el programa, pasaremos a explicar cómo se analizó el proyecto.

Primero se investigó sobre los diferentes algoritmos de ordenamiento, sobre teoría de grafos, etc. Todo lo que pudiera sumar al proyecto.

Luego de analizar cada información y algoritmo, se procedió a escoger el algoritmo a implementar en código. El algoritmo elegido en esta ocasión fue la teoría de Dijkstra.

Este algoritmo en forma resumida compara las aristas de cada vértice y el que tenga menor recorrido, pasa a dicho vértice. De forma cíclica, se hace el mismo procedimiento hasta llegar a la posición final.

Algoritmo analizarArchivo:

Este algoritmo se encargará de pasar los datos del archivo de entrada xml a una lista enlazada creada por nosotros.

Primero se leerá todo el archivo con ayuda de la herramienta Minidom. Primero se procesará la etiqueta raíz, que en este caso será <terrenos>, a partir de acá, se almacenará de una lista temporal todos los terrenos por medio de la etiqueta <terreno>.

Luego con ayuda de un for, se recorrerá cada Terreno y por cada etiqueta se guardará en un objeto de tipo Terreno como por ejemplo <dimensión>, <posicióninicial>, etc.

Cuando se haya recorrido el terreno y se haya guardado cada atributo en los objetos Terrenos se agrega a la lista enlazada creada anteriormente, y es allí donde se guardará cada terreno por medio de la función “append()”.

Algoritmo Dijkstra:

Bueno, pasando a código este algoritmo, se empezó buscando la posición inicial donde el robot iniciará su recorrido. Una vez se haya encontrado la posición inicial se empezará a sumar y a restar una posición en (x) y (y). Esto para acceder una posición en los cuatro cuadrantes Norte, Oeste, Sur y Este (NOSE).

Se usará condiciones para saber si existen posiciones ya sea arriba, abajo, a la derecha o a la izquierda. Por ejemplo, si se inicia en la posición (1,1) no se podrá ir a la izquierda, ni hacia arriba, ya que no hay posiciones menores a éstos.

Por medio de recursividad, este proceso se repetirá comparando en los cuatro cuadrantes cual es la posición de menor consumo de gasolina. Todas las posiciones donde el robot pueda seguir se guardará en una lista y se comparará con las demás posiciones y las futuras.

De esta manera, si el algoritmo encuentra una posición de menor consumo, redireccionará la ruta y se irá por la que menos combustible genere.

Si la posición siguiente es igual a la posición final, significa que ha llegado al destino y, por lo tanto, se terminará el algoritmo. Mientras se va ejecutando el algoritmo, irá imprimiendo mensajes para que el usuario esté al tanto de cómo va el proceso del terreno.

Cuando termine el algoritmo se imprimirá que se ha encontrado una ruta, seguido de una representación gráfica del terreno en forma de matriz, donde estará conformada por ceros y unos, los ceros serán las posiciones donde el robot no pasará, y los unos, las

posiciones donde el robot debería de pasar para consumir la menor cantidad de gasolina.

Pasando un poco al flujo del programa, es de la siguiente manera:

Cuando se inicia el programa se observará el menú de inicio, donde lo primero que se deberá de hacer antes de utilizar cualquier otra opción será cargar el archivo de entrada con extensión xml.

Cabe resaltar que si la extensión es de otro tipo diferente al .xml el programa rechazará el archivo, así que se recomienda cargar solo archivos con extensión xml.

Una vez cargado el archivo, este se procesará y se cargará a memoria los datos de cada terreno. Cuando el proceso termine se imprimirá un mensaje donde se verá que fue exitoso el procesado del archivo.

Las demás opciones serán habilitadas cuando el archivo de entrada haya sido cargado. Si se ingresa a la segunda opción "Procesar terreno" le pedirá al usuario el nombre de algún terreno de los que se hayan cargado anteriormente.

Si no se encuentra el nombre del terreno, retornará un mensaje diciendo que no se ha encontrado el terreno, de forma contraria empezará a analizar el terreno.

En el transcurso del proceso de analizado del terreno se irán imprimiendo mensajes de cómo está el estado del terreno y cuando se haya terminado el proceso de analizado se podrá visualizar el terreno con el camino que se ha encontrado, siendo el camino representado por una serie de unos.

También se podrá visualizar la cantidad de combustible que se gastará utilizando el camino encontrado. Después de este proceso el programa regresará al menú principal.

La opción tres "Escribir archivo de salida" es donde se rescribirá el archivo que se analizó en la opción dos. Se resalta que solamente los terrenos analizados serán escritos, sino sufre ninguna modificación algún terreno, no aparecerá en el nuevo archivo.

El usuario elegirá la ruta donde el archivo será guardado. Cuando termine la escritura del archivo se imprimirá un mensaje de éxito.

Luego pasamos a la opción cuatro "Mostrar datos del estudiante. Aquí se podrá observar los datos del estudiante que creó el programa. Datos como: nombre, carné, clase, facultad y semestre.

Por último, tenemos la opción de "Generar gráfica", cuando ingresemos en esta opción se desplegará una lista de los terrenos que han sido cargados; luego, el usuario tiene que ingresar en número del terreno del que quiere graficar.

Cuando se indique qué grafica se requiere, la función gráficaG empezará a ejecutarse y a crearse el grafo. Cuando se haya hecho dicho grafo se imprimirá un mensaje de éxito.

La gráfica se podrá encontrar en la misma carpeta donde se encuentra el programa con extensión .png.

Por último, la opción seis, que es la finalización del programa, si presiona el usuario esta opción se terminará de inmediato el programa.

Esto es, en general, el programa que se creó para identificar una ruta con el menor combustible posible. Y con esto se termina la explicación del proyecto. Gracias.

Conclusiones

Se aprendió a usar de forma correcta y a crear las diferentes estructuras de datos, en este caso, creando la clase Nodo y ListaEnlazada, se analizó y comprendió dichas clases.

Se determinó un algoritmo para encontrar una ruta con el menor combustible, utilizando el algoritmo de Dijkstra y se implementó en código Python.

Se aprendió a procesar archivos tipo xml, a guardar la información del archivo en objetos y listas enlazadas. Así como a escribir archivos del mismo tipo xml, con su estructura característica.

Se comprendió el uso de la herramienta Graphviz para crear grafos gráficamente y así tener una visualización mejor del problema y del algoritmo.

Referencias bibliográficas

Auxiliares de IPC2, (2021). *Proyecto 1*. Disponible en: https://bit.ly/3z6LE6d

Anónimo, (2021). *Teoría de Grafos*. Wikipedia. Disponible en: https://bit.ly/3D8tDH6

Anónimo, (2021). Algoritmo de Dijkstra. Wikipedia. Disponible en: https://bit.ly/3D7rdbF

Anónimo, (sin fecha). Estructura Lista Enlazada. PDF. Disponible en: https://bit.ly/3sJyt9a

Anónimo, (sin fecha). Estructuras de datos: listas enlazadas, pilas y colas. Calcifer.org. Disponible en: https://bit.ly/3j6CscB

Apéndices

Gerson Rubén Quiroa del Cid
202000166

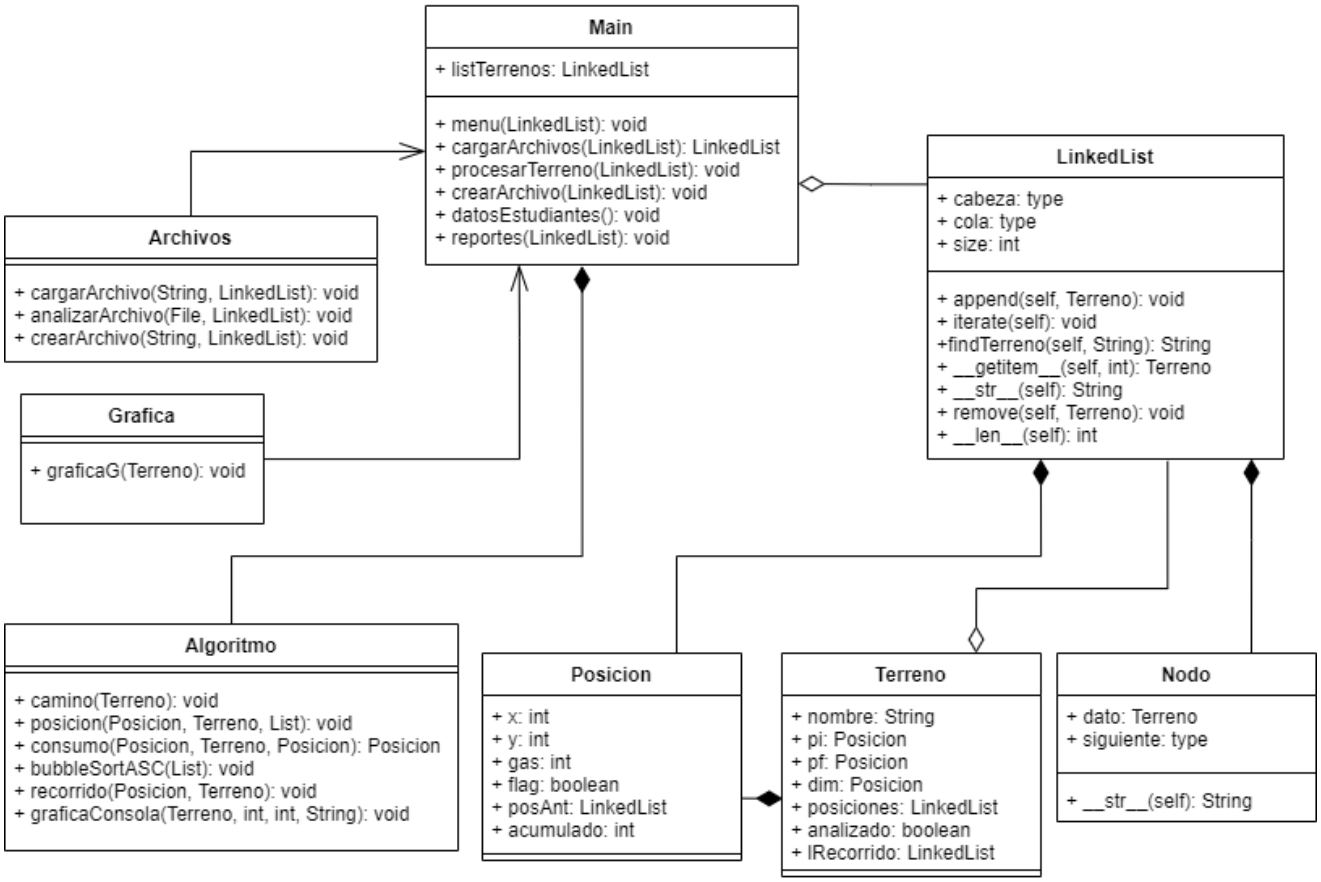

Figura 1. Diagrama de clases proyecto 1

Fuente: elaboración propia.