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: Analizador de Caminos

Erick Daniel Ajche Hernandez 201701043

Resumen:

El proyecto ayudará a r2e2, y a los futuros robots de exploración, sus empresas en un ahorro increíble de recursos, siendo AGIE con ayuda de sus satélites la primera en implementarlo, este podrá obtener sus reportes para el máximo control de lo que ocurrirá con el robot, los datos obtenidos son exactos ya que matemáticamente están respaldados. Todo el problema se basó principalmente en el algoritmo, y otra parte en el ingreso de datos, ya que cada terreno se maneja por sus distintas áreas. Con la condición de que r2e2 solo puede desplazarse solo en 4 direcciones. El programa fue desarrollado a tal punto que la ejecución y todos los métodos son muy rápidos, este necesita un archivo de entrada que tenga como máximo 100 áreas por dimensión, cada área con sus respectivas coordenadas y el gasto de combustible que este tendría. El programa los ordenada y encontrara la mejor opción.

Palabras Clave:

Coordenada:
Sistema de valores, referencias, etc., que aclaran la situación de algo o alguien y facilitan su análisis.

Estructura Dinámica:
Es aquella en la que el tamaño ocupado en memoria puede modificarse durante la ejecución del programa.

Matriz:
Es un arreglo bidimensional de números.

Puntadores:
Es una variable que contiene una dirección de memoria, la cual corresponderá a un dato o a una variable que contiene el dato.

XML:
Es una especificación de W3C como lenguaje de marcado de propósito general.

Summary:

The project will help r2e2, and future exploration robots, their companies in an incredible saving of resources, being AGIE with the help of their satellites · the first to implement it. This will be able to get your reports for maximum control of what will happen with the robot, the data obtained are exact since mathematically backed up. The whole problem was based mainly on the algorithm, and another part on it.
data entry, since each land is managed by its different areas. With the quer2e2 condition can only scroll only in 4 addresses. The program was developed to the point that execution and all · methods are very fast, it needs an input file that has max 100 areas per dimension, each area with its respective ones coordinates and the cost of fuel that this would have. The program · sorted and find the best choice .

Keywords:

Coordinate:
System of values, references, etc., that clarify the situation of something or someone and facilitate its analysis.

Structure Dynamics:
It is the one in which memory occupied size can be modified during execution of the program.

Matrix:
It is a two dimensional arrangement of numbers.

Pointers:
It is a variable that contains a memory address, which · will correspond to one data or one. variable that contains the data.

XML:
It a W3C specification as language of general purpose markup.

Introducción:

Tenemos muchos problemas en las misiones espaciales, además de que son muy costosas, no podemos arriesgar vidas sin saber el fin de la misión, hacemos un gasto basto de recursos no renovables, principalmente el combustible, por lo cual se implementó la idea de tratar de extender y ahorrar la mayor cantidad de combustible posible, como hicimos esto. Pues se desarrolló un método complejo mediante algoritmos los cuales obtendrán el camino más conveniente en un terreno, ya que, al tener recursos limitados al visitar otras lunas o planetas, detiene el desarrollo de la misión. Tenemos la opción de mostrarlo gráficamente los terrenos a recorrer, un XML si lo necesitamos escritos para mantener un control de los hechos.

Desarrollo del Tema:

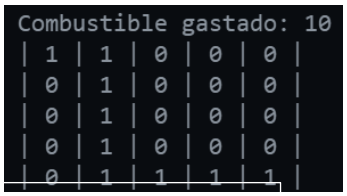
Entrada de archivo:

Desde el planteamiento del problemas, nuestras ideas fueron, el ingreso de los datos, como seria, nos reclinamos por un archivo XML, ya que la identificación de los datos seria fácil, y es una estructura de texto fácil de entender para personas que no están muy familiarizadas con la informática.
Dividimos un archivo XML en diferentes secciones.
Se pedirá que ingrese la dirección del archivo XML y así trabajar los terrenos.
Para empezar a identificar los terrenos el archivo debe de empezar con <terrenos>, desde ese momento se empiezan a tomar en cuenta los terrenos, para identificar y separar cada terreno del otro <terreno nombre =”name”>. Para identificar los puntos de inicio y final, usamos <posicioninicio> y <posicionfinal> respectivamente, dividimos las coordenada de cada una en <x> y <y>. Para cada área, este divide por <posición x=”X” y=”Y”>”C” donde X y Y son sus coordenadas, y C es el combustible que este gastaría.
Al ingresar un archivo con la estructura tal como se indica, cada terreno ingresara a una lista simple, así se guardarán los terrenos,
Para las distintas áreas, las tratamos como nodos, las áreas las ingresamos en una matriz ortogonal, ya que este tipo de estructura es perfecta para su manejo, al tener apuntadores hacia arriba, abajo, derecha e izquierda. Pudiendo tomar las direcciones en las que solo puede desplazarse el robot.

Proceso de terreno:

Siendo esta la parte mas complicada del al desarrollar el software, a principio, nos ubicamos en las áreas iniciales y finales, se intento implementar un algoritmo recursivo el cual, hacia cada posible viaje, pero este no era eficiente, ya que los viajes dependiendo el tamaño aumentaban exponencialmente, así calculando una cantidades extremadamente grande de caminos distintos. Por lo cual cambiamos implementamos en la programación el algoritmo de grafos de Dijkstra. El cual fue mas eficiente, ya que los recorridos que hace van reduciendo y descartando distintos caminos, con el fin de que las ejecuciones sean significativamente menos.
Con el fin que este método nos retorne la lista de áreas por la que el robot deberá pasar. Así retornara en la consola un tipo de grafico donde mostrar la camino, con 0 todos los nodos y 1 los nodos del mejor camino y mostrara la suma total de todo el combustible que gastara en el recorrido.

para procesar el terreno se necesita ingresar el nombre exacto del terreno.



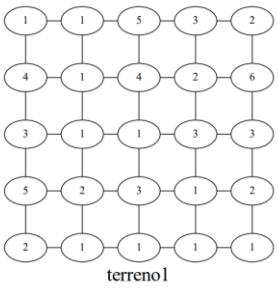
1. Grafica de consola

Escribir archivo de salida:

Tal como en el archivo de entrada, el de salida es un archivo XML donde mostrar lo mismo a diferencia que las áreas que mostrar son solo las del mejor camino, además de su gasto de combustible, para obtener el archivo de salida se necesita ingresar el nombre exacto del terreno.
El programa creara el archivo XML en la carpeta donde está el programa.

Generar gráfica:

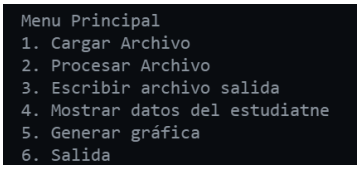
Se mostrar un gráfico con la ayuda de la herramienta llamada graphviz, el terreno, donde mostrara como seria el terreno gráficamente, los nodos y como se conectan, estos mostraran el consumo de combustible que tendrían cada área. Para dar comprensión de como seria el terreno. Para este ser graficado se desplegará la lista de terreno ingresados, solo ingresar el id que le pertenece al terreno, y lo graficará, al generar la gráfica, esta se abrirá automáticamente.



2. Grafica de Graphviz

Interfaz:

El programa se ejecuta solo en consola, aunque tiene acceso a archivos externos, ya que, aunque ya es funcional, hay mucho aun que se le puede implementar, la navegación es sencilla, están los créditos de desarrollador. Al iniciar el programa se muestra el menú principal, este donde se mostrarán todas las acciones que puede hacer el software. Si no se ha ingresado ningún archivo con terrenos, no se podrá ingresar a Procesar terreno, Graficar terreno, ni archivo de salida.
Para ingresar el archivo de entrada necesita la ubicación de este y su nombre, para procesar el archivo su nombre, para el archivo de salida su nombre, para la grafica seleccionar de los que están ingresado. Si el archivo no se ha procesado, este no podrá generar su archivo de salida.
Por último, la opción de salir, que cerrara el programa.



3. Menú principal

Código:

Se implemento a programación orientada a objetos, en el código Python, en su versión 3.9.6.
Para la lista y matriz, creamos TDA´s dinámicos, la lista contiene los terrenos, y la matriz contiene las áreas. Para el ingreso de áreas, de un terreno, al ser ingresada la nueva área en la matriz, esta conecta los apuntadores, y los mueve, para que encaje bien, siempre dependiendo de sus coordenadas. Este también podemos recorrer los nodos área, y obtener

información para la generación de reporte o procesar el archivo.

Al procesar el archivo, con la implementación de Dijkstra, abarcamos los caminos, y encontramos el mas corto, hasta cierto punto hasta llegar al punto que final que se nos indica, este método va recolectando los nodos por el cual iría el camino hasta que este y guardando el combustible para devolver una lista de nodos, y sus respectivos gastos de combustible.

Para los reportes ya solo importamos datos de nuestros objetos, y así obtenemos todo lo que necesitamos para obtener cualquier reporte

Para la gráfica usamos la librería graphviz de Python, escribimos el archivo, y con ayuda de la librería, solo renderizamos y creamos la gráfica.

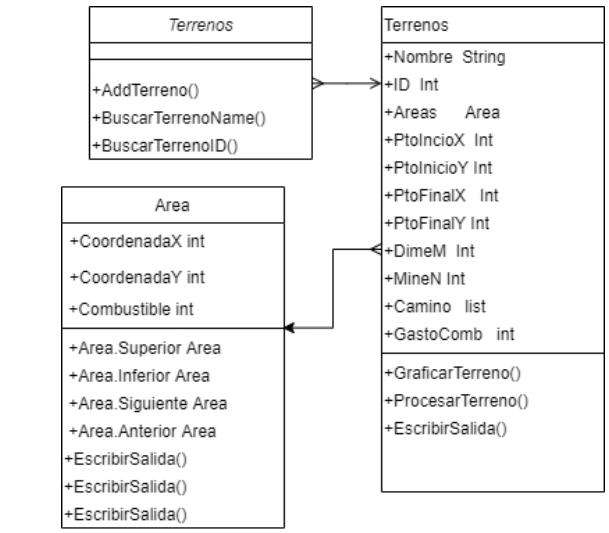
```
class Nodo: ...
class NodoGround: ...
class ListGround: ...
class Area : ...
class Ground: ...
```

4. Clases

El programa se trató de reducir en su mayoría de errores, para que nunca se detenga, verificamos la existencia de la mayoría de los objetos, atributos, nodo, apuntadores, para que nunca de erros, verificamos la existencia de los mismo, si fueron procesados o no.

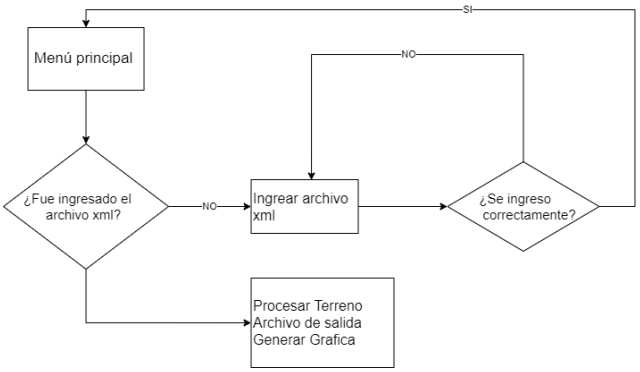
Todos los archivos externos que se crearán en el tiempo de ejecución serán generados en la carpeta donde el programa está ubicado.

Esta es la manera mas realista de ver los objetos creados.



5. Diagrama de Clases

El flujo del programa



6. Diagrama de Flujo

Conclusión:

Para solucionar el problema planteado, hay que pensar en todo lo que necesitamos para saber que se necesita, así obtenemos una mejor manipulación de datos, y no cambiar el código constantemente o generar más código del necesario, es una de las ventajas de la programación orientada a objetos.

Los TDA´s fueron la clave para el ingreso de datos, sin una matriz el acceso a los datos hubiese sido muy complicado.

Por otro lado, su creación al ser la primera vez es más complicada, pero el entender como debe de funcionar, al ser tan lógico es más fácil de entender e implementarlo.

Hay mucha importancia dentro de la implementación de argumentos y métodos matemáticos, nos podemos ahorrar mucho código al aplicarlos. Y evitar si es posible los métodos que tengan ejecuciones exponenciales, ya que pueden volverse muy tardados.

Gráficamente es más fácil entenderle a cualquier problemas, mas en los TDA´s una vez visualizado los problemas su desarrollo y solución será más fácil de implementar.

Referencias Bibliográficas:

https://www.it swarm es.com/es/python/como crear una matriz ortonormal aleatoria en python numpy/827266049/

https://www.ecured.cu/Algoritmo_de_Dijkstra

Estructura de datos en C++ Joyanes