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 – IPC2

202000194 – Alvaro Emmanuel Socop Pérez

Resumen

El presente proyecto consiste en la creación de un algoritmo de programación que resuelva el problema de la Agencia Guatemalteca de Investigación espacial para explorar por medio de su robot "r2e2" nuevos terrenos y encontrar el camino de los terrenos recibidos del satélite "Quetzal01" más óptimo.

Se hace el uso del lenguaje de programación Python con el paradigma de programación orientada a objetos, modular y funcional, memoria dinámica, manejo de archivos XML, estructuras condicionales, secuenciales y cíclicas para solucionar el problema.

Con la implementación del programa el robot "r2e2" ahora podrá optimizar el uso de su combustible y disminuir el tiempo de llegada desde una posición a otra. En la ejecución del programa se cargan los datos de los terrenos para luego procesar y mostrar el camino que se considera optimo, posteriormente se muestran los datos en un archivo XML demostrando gráficamente comportamiento de las matrices de los terrenos.

Palabras clave
Algoritmo, optimización, Estructuras secuenciales, memoria dinámica.

Abstract

This project consists of the creation of a programming algorithm to solve the problem of the Guatemalan Space Research Agency to explore new terrains by means of its robot "r2e2" and to find the most optimal path of the terrains received from the satellite "Quetzal01".

The Python programming language is used with the object-oriented, modular and functional programming paradigm, dynamic memory, XML file management, conditional, sequential and cyclic structures to solve the problem.

By implementing the program, the robot "r2e2" will now be able to optimize the use of its fuel and decrease the arrival time from one position to another. In the execution of the program the data of the terrains are loaded and then processed to display the optimal way, then the data are displayed in an XML file graphically demonstrating the behavior of the matrices of the terrains.

Keywords
Algorithm, optimization, structures, dynamic memory.

Introducción

El proyecto tiene como enfoque la implementación de estructuras de datos TDA´s utilizando listas enlazadas y doblemente enlazadas, para tomar una serie de datos correspondiente a coordenadas de un terreno enviado por el satélite "Quetzal01" y el valor de combustible que el robot r2e2 consumirá desde un punto inicial hacia un punto final, y luego por medio del algoritmo de Dijkstra enviar cada uno de los nodos enlazados e ir encontrando las coordenadas del terreno donde debe recorrer el robot, esto para optimizar el uso de gasolina.

El programa es capaz de graficar cada uno de los terrenos que se han leído por medio del archivo XML, graficando en una imagen, para mostrar las gráficas se utiliza la herramienta Graphviz.

El algoritmo de Dijkstra implementado para obtener los nodos de menor consumo de gasolina determinando los vértices que se encuentran en nuestra matriz, implementando condicionales cíclicas.

Programación en Python

Python es un lenguaje de scripting que no depende de plataforma y es orientado a objetos, realiza cualquier tipo de programa, aplicaciones para Windows hasta servidores de red o, páginas web. Es un lenguaje interpretado, por lo cual no es necesario compilar el código fuente para su ejecución.

Una de las ventajas que se presentan es la rapidez de desarrollo, y su pequeña desventaja o inconveniente es que ofrece una menor velocidad.

Se considera es un lenguaje de programación que admite el uso de varios paradigmas de programación (modelos de desarrollo), por lo que no exige a los programadores un estilo único para programar.

Estructuras de programación secuenciales

La estructura secuencial es aquella en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso.

La asignación consiste, en el paso de valores o resultados a una zona de la memoria. Dicha zona será reconocida con el nombre de la variable que recibe el valor. La asignación se puede clasificar de la siguiente forma:

a. Simples: Consiste en pasar un valor constante a una variable.
b. Contador: Consiste en usarla como un verificador del número de veces que se realiza un proceso.
c. Acumulador: Consiste en usarla como un sumador en un proceso.
d. De trabajo: Donde puede recibir el resultado de una operación matemática que involucre muchas variables.

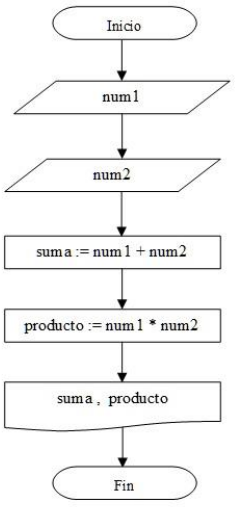


Figura 1. Diagrama de flujo como estructura secuencial.

Fuente: elaboración propia, o citar al autor, año y página.

Estructuras cíclicas

Estas estructuras también llamadas bucles, se utilizan para realizar o ejecutar operaciones o acciones repetitivas, las cuales tienen un fin o limite

Las estructuras cíclicas se utilizan para ejecutar fragmentos de código un número limitado de veces. Existen tres estructuras cíclicas generales, las cuales son:

a. Desde Hasta (For)
b. Hacer Mientras (While)

Las estructuras tienen el mismo objetivo, ejecutar un fragmento de código un número limitado de veces, su principal diferencia se encuentra en la forma en la cual limitan el número de ocasiones que se ejecutará el código.

Estructuras condicionales

Las estructuras condicionales existen en todos los lenguajes de programación y nos dejan realizar algo según una condición.

La estructura if permite que un programa ejecute unas instrucciones cuando se cumplan una condición.

La estructura de control if else permite que un programa ejecute unas instrucciones cuando se cumple una condición y otras instrucciones cuando no se cumple esa condición.

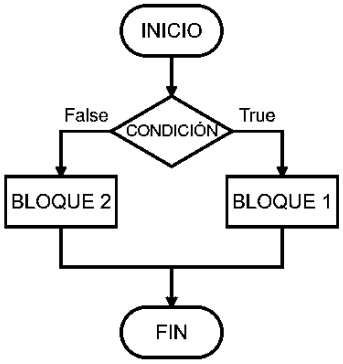


Figura 2. Diagrama de flujo sentencia if-else.

Fuente: elaboración propia.

Graphviz

Graphviz es una herramienta de software para el diseño de diagramas de cualquier tipo, definido en el lenguaje descriptivo DOT.

DOT

Dot es una herramienta de comandos que genera imágenes en capas de grafo dirigido con varios colores y diseños.

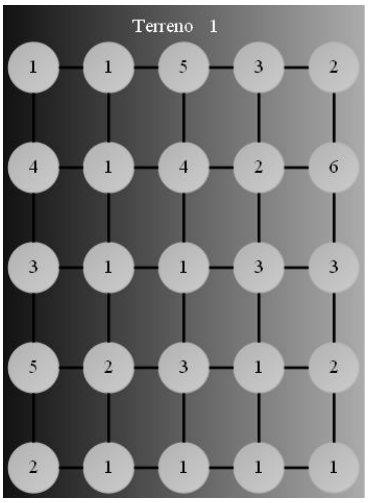


Figura 3. Gráfico en Graphviz de un terreno.

Fuente: elaboración propia.

Archivos XML

XML consiste en un lenguaje de marcado que tiene la finalidad de definir una sintaxis para la codificación de documentos, que tanto los usuarios como las propias máquinas en sí puedan ser capaces de leer.

Lo hace mediante la utilización de una serie de etiquetas que definen la estructura que posee el documento en cuestión, además de cómo debe ser transportado y almacenado.

Se parece mucho al lenguaje de marcado de hipertexto (HTML), usado esencialmente para la codificación de las páginas web, y que utiliza un conjunto de símbolos de marcado predefinidos que

describen el formato que posee el contenido de una página web.

Tipo de dato abstracto

Un tipo de dato abstracto es un modelo matemático compuesto por una colección de operaciones definidas sobre un conjunto de datos para el modelo.

Los lenguajes de programación orientados a objetos (POO) tienen la característica de que no son lenguajes lineales, sino que se forman de diversas funciones, las cuales son llamadas en el orden en que el programa mismo las pide o el usuario determina.

Un elemento fundamental de las Estructuras de Datos denominado Abstracción de Datos y que es parte importante de estos Lenguajes y de la manera en que funciona la mayoría del software comercial de nuestros días.

La abstracción de datos consiste en ocultar las características de un objeto y obviarlas, de manera que solamente utilizamos el nombre del objeto en nuestro programa.

En el método principal solamente se llama a la función Área y el programa hace el resto.

Es frecuente que el Encapsulamiento sea usado como un sinónimo del Ocultación de información, aunque algunos creen que no es así [1] (BlogdeCompu,2010)

Listas

Las listas son una colección de elementos homogéneos (del mismo tipo: TipoElemento) con una relación LINEAL establecida entre ellos. Pueden estar ordenadas o no con respecto a algún valor de los elementos y se puede acceder a cualquier elemento de la lista.

Operaciones

a. Crear()
b. Imprimir()
c. Insertar()
d. Eliminar()
e. Buscar()

Implementación

Representación secuencial: el orden físico coincide con el orden lógico de la lista. Por ejemplo para insertar o eliminar elementos podría exigirse que se desplazaran los elementos de modo que no existieran huecos.

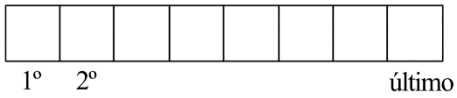


Figura 3. Implementación de una lista.

Fuente: Diseño de algoritmos, 2015, web

Variables de tipo puntero:

permite dimensionar en tiempo de ejecución (cuando se puede conocer las necesidades de memoria). La implementación de las operaciones son similares a las ya vista sobre la lista enlazada de punteros.

Figura 4. Punteros en una lista enlazada.

Fuente: Diseño de algoritmos, 2015, web.

Variables estáticas: se utilizan arrays de registros con dos campos: elemento de la lista y campo enlace (de tipo índice del array) que determina el orden lógico de los elementos, indicando la posición del siguiente elemento de la lista (simulan un puntero).

Por tanto, en realidad coexisten dos listas enlazadas dentro del array de registros, una de memoria ocupada y otra de memoria libre.

Memoria Dinámica

Estas estructuras se caracterizan porque no sólo pueden variar su contenido durante la ejecución del programa sino que además pueden variar su tamaño. Debido a esto, las variables que son de este tipo se denominan estructuras de datos dinámicas

Las estructuras dinámicas lineales son: las listas simplemente enlazadas, las listas circulares, las listas doblemente enlazadas, las pilas y las colas.

En una asignación dinámica de memoria se hace referencia al hecho de crear variables o reservar espacio en memoria para estas variables en tiempo de ejecución del programa, así como liberar el espacio reservado para dichas variables anónimas, cuando ya no son necesarias, también durante el tiempo de ejecución.

Tabla I.

Tipos de estructuras dinámicas en tipos de datos abstractos.

Lista	elementos
Simplemente enlazada	Mismo tipo
Circular	El ultimo nodo apunta primero
Doblemente Enlazada	Nodo con acceso directo
Pilas	Mismo tipo con un elemento cima
Colas	Elemento de final es el inicial

Fuente: elaboración propia.

La tabla muestra las listas y elementos que existen en las estructuras dinámicas.

Método de optimización de Rutas

Algoritmo de Dijkstra

El algoritmo de Dijkstra nos va a permitir calcular la ruta más corta entre un nodo y todos los demás nodos en el grafo.

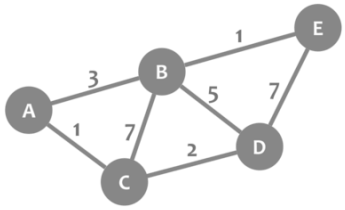
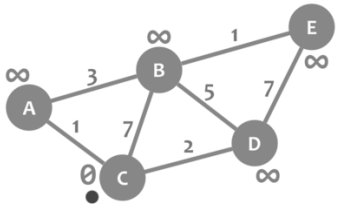


Figura 5. Grafo demostración de Dijkstra.

Fuente: Diseño de algoritmos, 2015, web.

Durante la ejecución del algoritmo, se va marcando cada nodo con su distancia mínima a un nodo C por ejemplo el nodo que hemos elegido. Para el nodo C, esta distancia es 0. Para el resto de nodos, como todavía no conocemos esa distancia mínima, se le coloca una etiqueta de infinito:

Figura 6. Grafo demostración de Dijkstra.



Fuente: Diseño de algoritmos, 2015, web.

Posteriormente, se revisa a los vecinos de nuestro nodo actual (A, B y D) sin importar el orden. Podemos empezar con B. Se le suma la distancia mínima del nodo actual que en nuestro ejemplo es 0 con el peso de la arista que esta conectada al nodo actual con B para nuestro caso es 7, obteniendo:
0 + 7 = 7.
Se compara el valor con la mínima distancia de B que es infinito, el valor más pequeño es el que queda

como la distancia mínima de B , y como 7 es menor que infinito entonces queda algo como esto:

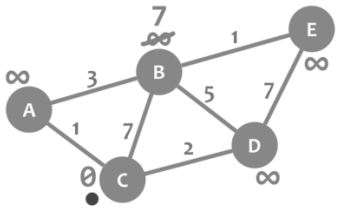


Figura 7. Grafo demostración de Dijkstra..

Fuente: Diseño de algoritmos, 2015, web.

Luego se mira al vecino A. Se le suma 0 (que es la distancia mínima de C, nuestro nodo actual) con 1 (que es el peso de la arista que conecta nuestro nodo actual con A) para obtener 1. Se compara ese 1 con la mínima distancia de A (infinito) y se deja el menor valor, algo asi:

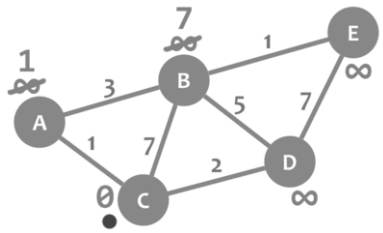


Figura 8. Grafo demostración de Dijkstra.

Fuente: Diseño de algoritmos, 2015, web.

Y volvemos a repetir el proceso con el siguiente nodo por ejemplo el D

Conclusiones

Se ha reconocido que existen varios métodos para escoger el camino más optimo utilizando programación orientada a objetos y el uso de memoria dinámica para ahorrar recursos en nuestro sistema.

El uso de estructura dinámica facilita el manejo de datos a gran escala y ahorra memoria en el tiempo de ejecución.

En problemas reales pueden presentarse problemas donde necesitemos almacenar cierto conjunto de datos y es buena práctica utilizar memoria dinámica para ir guardando cada uno de estos datos.

Existen varios métodos de optimización de rutas o caminos, y el mas optimo resulta ser el de Dijkstra y fácil de implementar relacionándolo a los nodos que se tienen en una matriz como en el problema que se ha resuelto.

Python es un lenguaje de programación fácil de entender y programar con variedad de librerías de las cuales nos ha servido para la resolución del problema y construcción de clases necesarios para el funcionamiento del algoritmo.

Graphviz es una herramienta que ayudó a ver de una forma gráfica como esta construido el terreno que nos ha enviado el satélite “Quetzal01” siendo un método para entender lo que esta haciendo el programa de una forma dinámica.

Referencias bibliográficas

C. Clase, (2013, 20 octubre). *C Con Clase | Estructuras de datos (cap4)*. Estructura de Datos (TDA). http://conclase.net/c/edd/cap4?cap=004

C. S. Severance, (2009). *Python para informaticos.* Explorando la informacion, version 2.7.2, Inc.

C. Larman, (2003). *Introducción al análisis y diseño orientado a objetos*, Hall Prentice.

J. A. Luis, (2008). *Fundamentos de Programación.* McGraw-Hill, Inc.

R. Hernandez, (2001). *Estructuras de datos y Algoritmos*, Hall Prentice.

Diagrama de Clases

Figura 9. Diagrama de Clases.

Fuente: Elaboración propia.

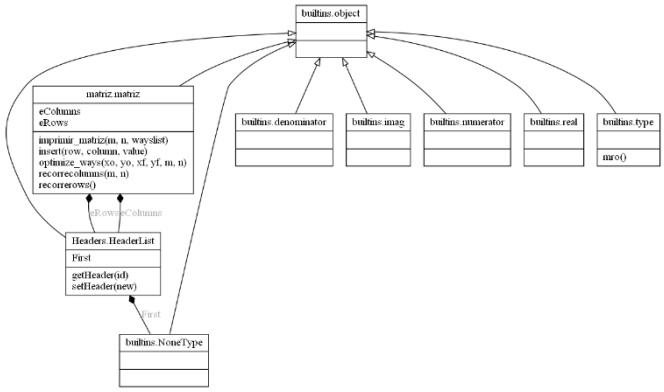


Figura 9. Diagrama de Clases (Matriz).

Fuente: Elaboración propia.

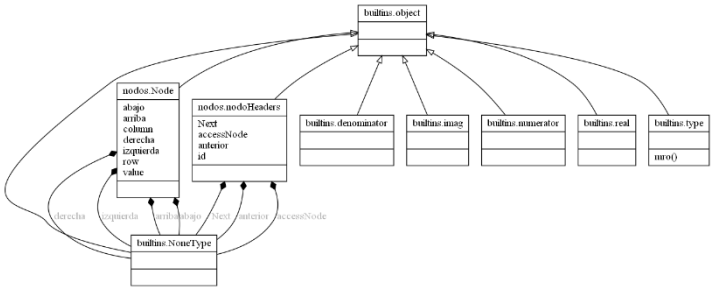


Figura 9. Diagrama de Clases.

Fuente: Elaboración propia

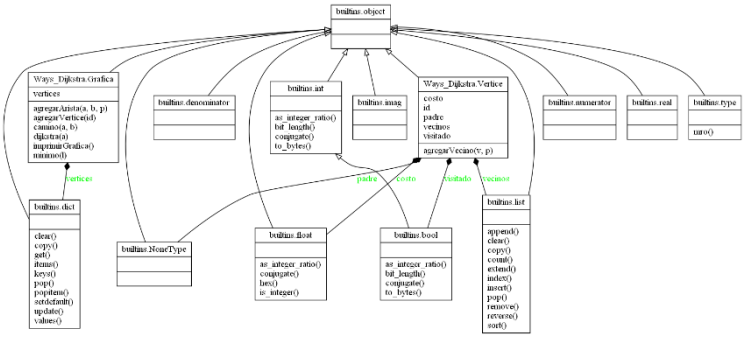


Figura 10. Diagrama de Clases (Dijkstra).

Fuente: Elaboración propia

Algunas clases más importantes

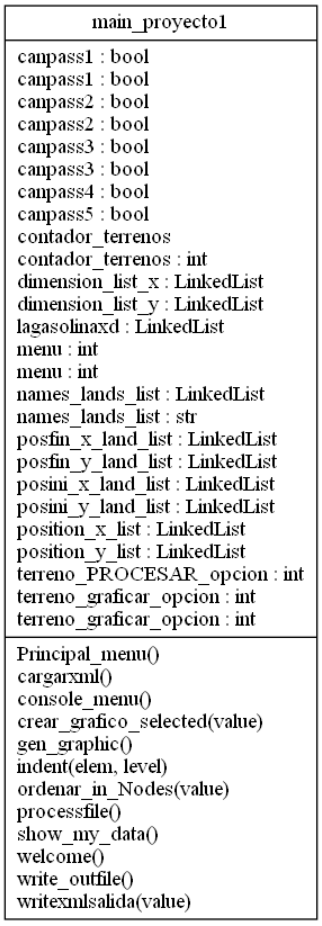


Figura 11. Diagrama de la clase Main

Fuente: Elaboración propia

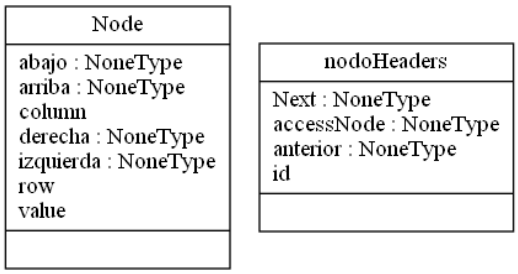


Figura 11. Diagrama de la clase Nodos

Fuente: Elaboración propia

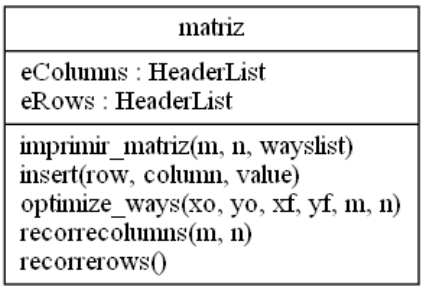


Figura 11. Diagrama de la clase Matriz

Fuente: Elaboración propia

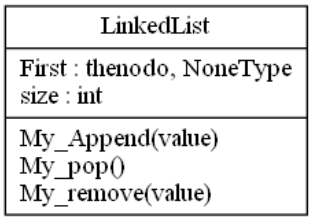


Figura 11. Diagrama de la clase Lista Enlazada

Fuente: Elaboración propia

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

DIAGRAMA DE CLASES

Diagrama General del programa

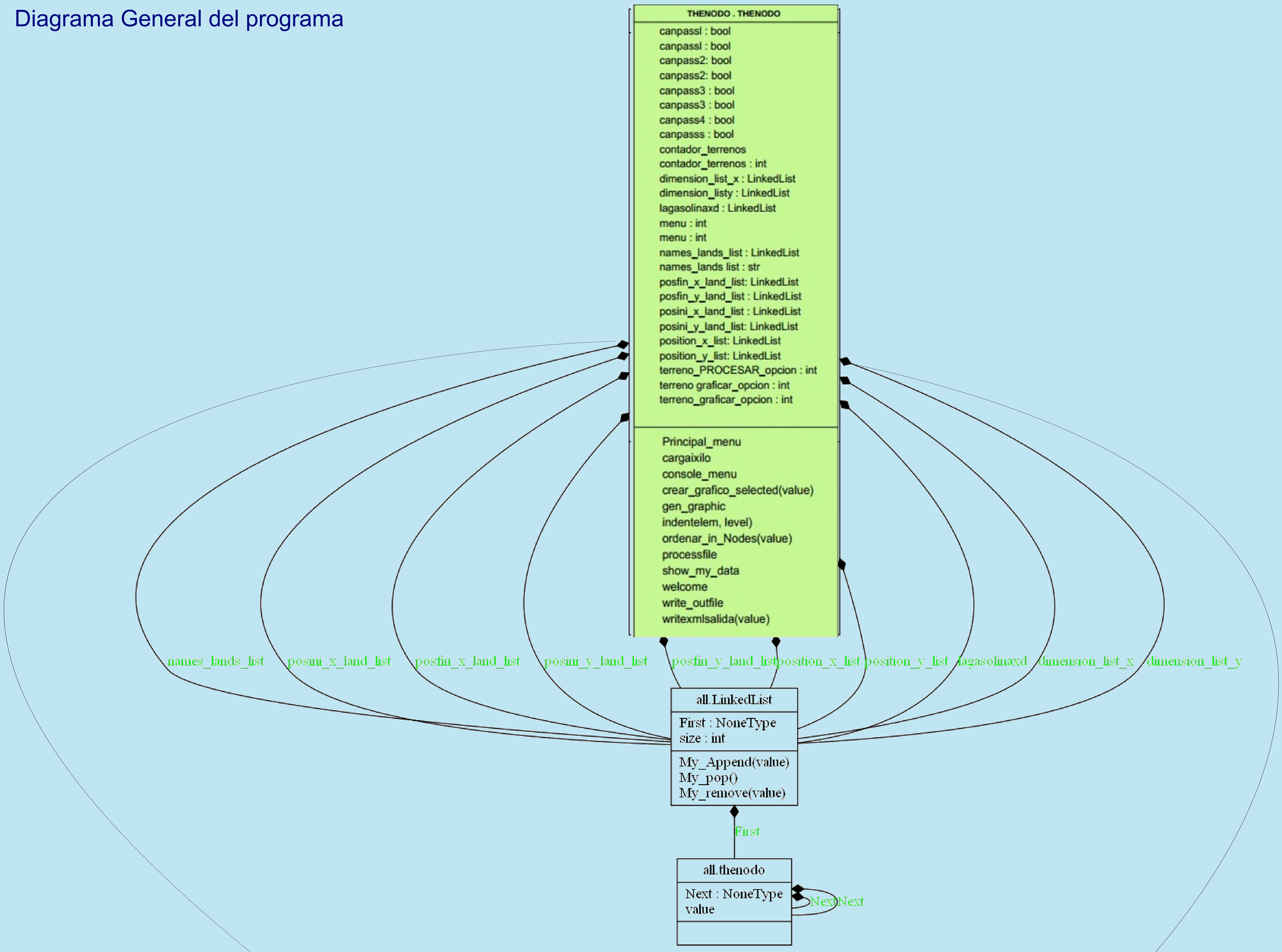


Diagrama de los Nodos

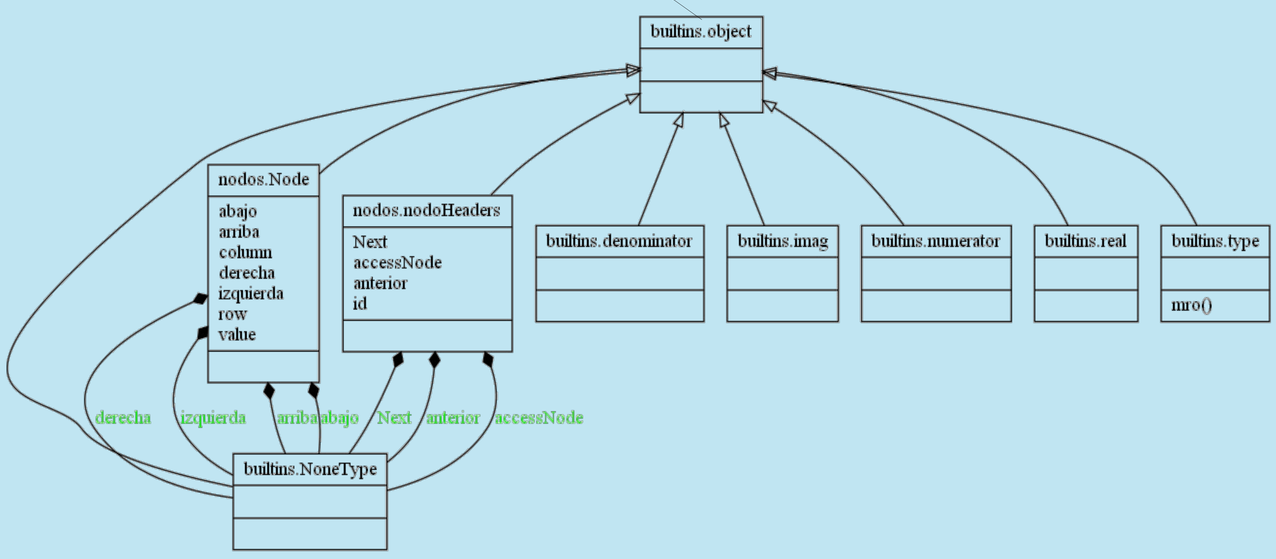


Diagrama de mi matriz

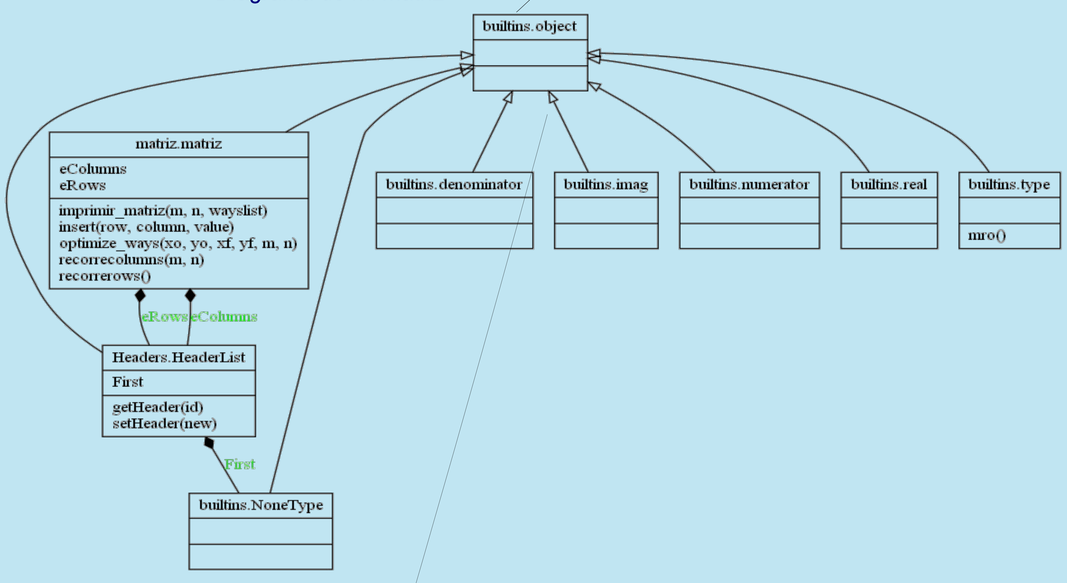


Diagrama del Algoritmo de Dijkstra

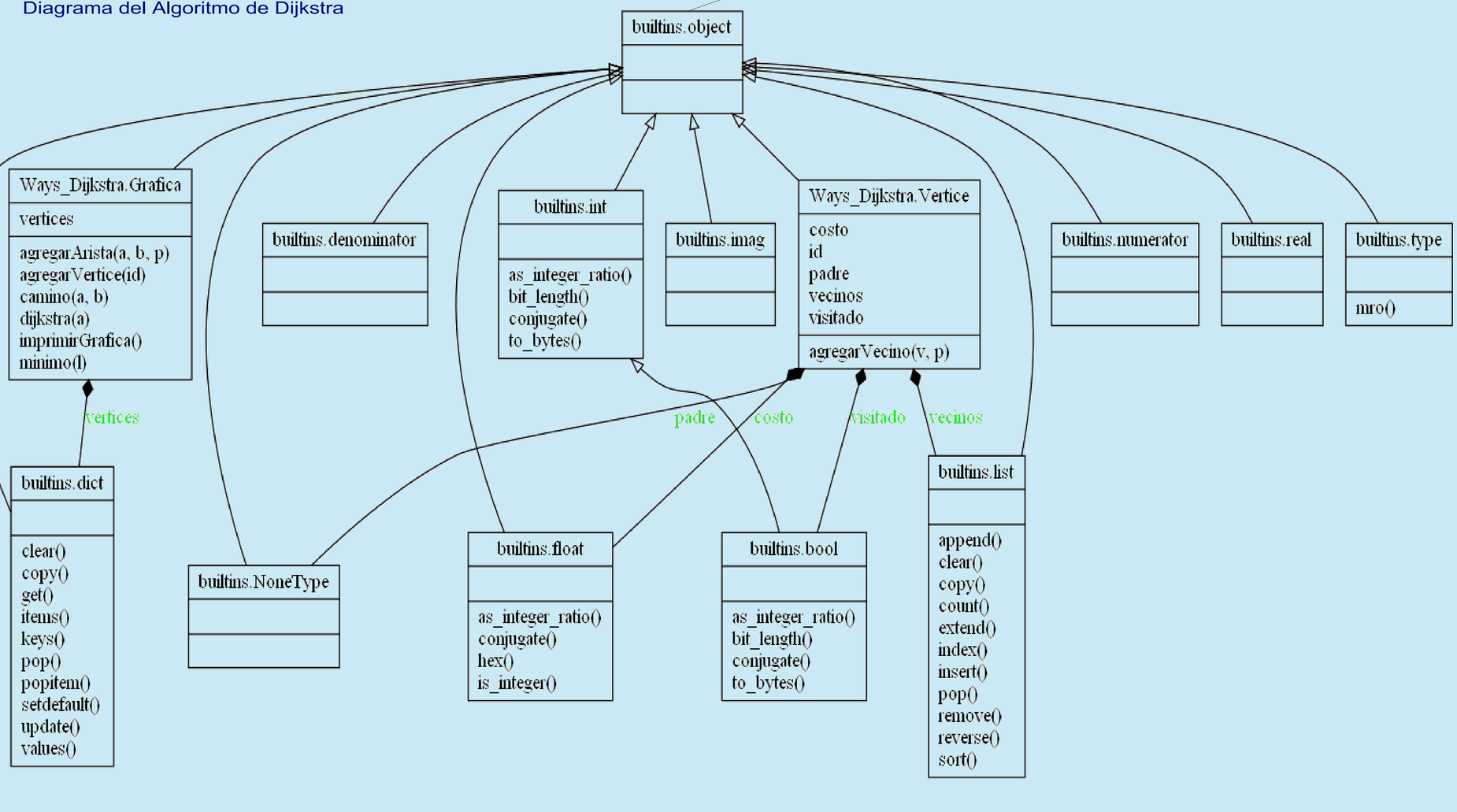


Diagrama General del Programa