Algoritmos y Programacion II (75.41)
Cátedra Wachenchauzer

Trabajo Práctico 3

Corrector:

Martín Buchwald

Alumnos:

Lauría, María Dulce (94772)

Acevedo, Facundo (94311)

Análisis:

Nuestro “GPS” trabaja con dos grafos, uno para las calles y otro para los nodos. Posee una estructura muy modular y de código ordenado. Para las rutas optimas se implemento un algoritmo basándose en el de Dijkstra, de manera que se obtienen las rutas optimas desde un vértice hacia todos los demás.

Para visualizar de manera mas legible las rutas se genera un archivo kml el cual puede ser cargado en cualquier GIS que soporte este tipo de archivos.

Especificación:

Sistemas operativos:

Aquellos en los que este portado python

Ejecución:

Correr *./gps_pizeria.sh* en la consola. Para máxima compatibilidad se debe poseer python 2.7, ya que con esta versión fue probado.

Datos de entrada:

A lo largo de la ejecución se pedirá que se ingrese texto, avisando el formato esperado. Este sera validado y modificado para que cumpla con el estándar usado en la aplicación.

Las entradas obligatorias son:

-Ruta del mapa: ubicación en la computadora del archivo del mapa

-Intersección de la pizeria: ubicación física de la pizeria.

Luego variara según la opción a utilizar.

Datos de salida:

Luego de encontrar y procesar la ruta optima, la aplicación generara parcialmente un archivo kml, el cual estará completo al cerrar el programa, este archivo cuenta con marcas especificas en la ubicación de la pizeria, y la casa de los clientes, como así en cualquiera de los trayectos que se hayan procesado.

Diseño:

El código del programa se divide en siete archivos *archivos.py constantes.py grafo.py interaccion.py main.py sexykml.py texto.py*

archivos.py:

Contiene las funciones encargadas de validar el archivo .csv, ya sea verificando que el archivo sea un fichero ordinario (osea no un dispositivo, ni una carpeta), verificando que no este vació y comprobando que se tienen los permisos necesarios para leerlo. Luego genera dos grafos y un diccionario , carga los datos en los respectivos grafos, y obtiene la información de los nodos guardándola en el diccionario. Finalizada la tarea, cierra el archivo.

constantes.py:

Contiene las constantes globales, que se utilizan en todo el programa, ya sean mensajes, como también el valor referencia de infinito.

grafo.py:

Contiene clases y funciones referidas a la utilización del grafo, maneras de imprimir una ruta por pantalla, como así también el grafo mismo. Varias de las funciones no son usadas en el programa, pero sirven en el caso de querer ver agilmente los datos en pantalla, sin tener que generar un archivo kml.

Contiene también una función para validar que las calles ingresadas pertenezcan al mapa y se intersequen. Una función para obtener la ruta optima basanadose en el algoritmo de Dijkstra. Una función para procesar la ruta y otra para adecuarla a la entrada de las funciones de generación del kml.

Por ultimo posee una función que ejecuta de manera ordenada el resto de las funciones, necesarias para obtener la ruta y agregarla al archivo kml.

interaccion.py:

Contiene funciones que interactuan con el usuario, incluyendo al menú, estas funciones imprimen mensajes, piden datos y también validan entradas.

main.py:

Llama a las funciones de manera ordenada para que el programa funcione, importa a todos los módulos de la carpeta src, también inicializa el archivo kml, luego pide que se le ingrese la ruta al mapa, valida que este exista y genera los grafos, a continuación ingresa al bucle principal de la aplicación donde se mostrara el menú y según la opción elegida se procederá a generar las rutas.

sexykml.py:

Contiene la clase SexyKML, que es la encargada de generar el archivo kml, al inicializarla debe recibir el titulo que tendrá el mapa, también se le puede pasar la ubicación donde se quiere que se guarde el archivo con las rutas, pero no es obligatorio, ya que por defecto genera el archivo "rutas.kml". Esta clase tiene las funciones para agregar un marcador, agregar una ruta, agregar un comentario, y finalizar, esta ultima debe ser llamada obligatoriamente ya que es la encargada de cerrar la estructura y escribir el archivo.

texto.py:

Contiene las funciones de tratamiento de entradas, modificandolas para que cumplan con el estándar utilizado por esta aplicación. También posee una función para validar y devolver una intersección.

Pruebas:

Se pide la ubicación del mapa:

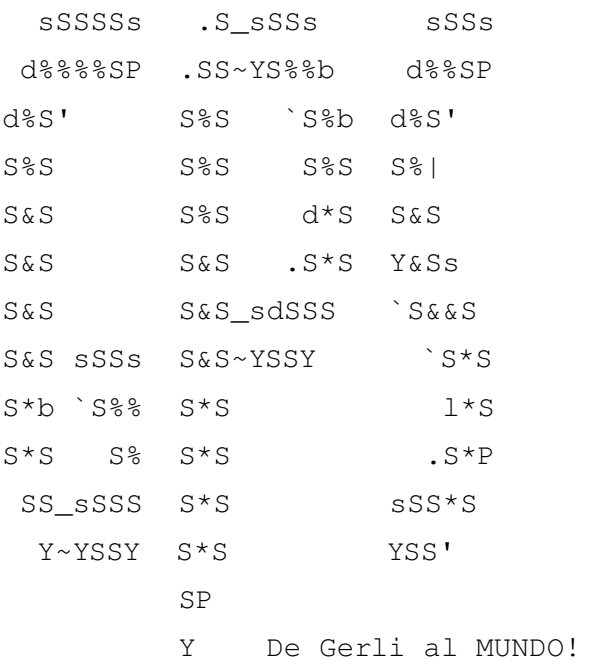
```
[facu@wtf tp3-algo2]$ ./gps_pizeria.sh
Ingrese la ubicación del mapa:
:>
```

Se pide la intersección. de la pizeria:

```
Ingrese la intersección. con el siguiente formato:
        calle1,calle2

INGRESE LA DIRECCION DE LA PIZERIA:
:>
```

Luego de ingresar la intersección. se mostrara el menú:




```
1)Generar ruta
2)Ruta e/puntos cualesquiera -Consigna Opcional-
3)Salir

:>
```

Seleccionada la opción 1, pedirá l intersección. de la calle del cliente:

```
Ingrese la intersección. con el siguiente formato:
        calle1,calle2

INGRESE LA DIRECCION DEL CLIENTE:
:>
```

Luego de ingresar la intersección. nos avisara:

```
Ida...
Generando rutas, puede tardar...
Generando trazas...
Vuelta...
Generando rutas, puede tardar...
Generando trazas...
Generado correctamente, el archivo .kml
```

y volverá a mostrar el menú.

Selecciono la opción 2, pedirá la intersección. del vértice A:

```
Ingrese la intersección. con el siguiente formato:
        calle1,calle2

INGRESE INTERSECCION DEL VERTICE_A:
```

Luego la del vértice B:

```
Ingrese la intersección. con el siguiente formato:
        calle1,calle2

INGRESE INTERSECCION DEL VERTICE_B:
:>
```

Luego genera el kml y nos avisa:

```
Ida...
Generando rutas, puede tardar...
Generando trazas...
Vuelta...
Generando rutas, puede tardar...
Generando trazas...
Generado correctamente, el archivo .kml
```

Elijo la opción 3, para finalizar el programa:

```
Finalizando KML...
Cerrando archivo ...
```

Posibles errores en el ingreso de datos:

Mapa:

```
Ingrese la ubicación del mapa:
:>/dev/null
El archivo no existe, o no es del tipo esperado
Ingrese la ubicación del mapa:
:>mapa_vacio.csv
El archivo esta vació

Ingrese la ubicación del mapa:
:>mapa_sin_permisos.csv
Ocurrio un error al intentar abrir el archivo <type
'exceptions.IOError'>
```

Calles:

```
:> general nazar y caxaraville
Formato no valido

- Si la dirección de la pizeria es la dirección del cliente:
INGRESE LA DIRECCION DE LA PIZERIA:
:> general nazar,caxaraville

[.. MENU..]
INGRESE LA DIRECCION DEL CLIENTE:
:> general nazar,caxaraville
El lugar de partida y el de destino, no deben ser el mismo!
```