Algoritmos

Archivo “Linkedlist”

Clase **Node**
#Nodo
Método constructor

Sus parametros son: (nombre, distancia, ancho de banda, usuarios conectados, trafico y tipo de medio)

Sus caracteristicas:

Nombre que recibe nombre del parametro
Distancia que recibe distancia del parametro
Ancho de Banda que recibe ancho de banda del parametro
Usuarios conectados que recibe usuarios conectados del parametro
Trafico que recibe trafico del parametro
Tipo de medio que recibe tipo de medio del parametro
Arista que recibe la Lista enlazada
Siguiente que recibe Nada

Clase **LinkedList**
#Lista enlazada
Método constructor

No recibe parametros

Sus caracteristicas:

Un Primero que recibe Nada
Una Lista tipo2 que recibe un arreglo

Método add
#Agregar
Recibe como parametro (Nombre,Distancia,Ancho de Banda,Usuarios conectados, Trafico y Tipo de medio)

Si no hay un primero
    El primero se crea un Nodo

Caso contrario
    El actual se le asigna el Primero
    Mientras haya un siguiente del actual
        El actual avanza al siguiente
    El siguiente del actual se crea un Nodo

Método searhInLL
#Buscar en la lista.
Recibe como parametro (nombre, estado = 0)
#Busqueda de un elemento, state =0 para devolver el nodo, state = 1 para regresar un boolean

El actual se le asigna el Primero

Si el estado es igual a 0
Si el nombre del primero es igual al nombre del parametro
Retornar el primero
Caso contrario
Mientras haya un siguiente del actual
El actual avanza al siguiente
Si el actual es diferente de Nada
Si el nombre del actual es igual al nombre del parametro
Retornar el actual

Caso contrario si el estado es igual a 1
Si el nombre del actual es igual al nombre del parametro
Retornar verdadero
Caso contrario
Mientras haya un siguiente del actual
El actual avanza al siguiente
Si el nombre del actual es igual al nombre del parametro
Retornar falso

Método length
#Obtener el tamano de la lista.

El actual se le asigna el Primero
SI el actual es diferente de Nada
Tamaño se le asigna 1
Mientras haya un siguiente
El actual avanza al siguiente
Tamaño incrementa en 1
Retornar tamaño
Casi contrario
Retornar 0

Método atPosition
#Obtener un item dado un indice.

Recibe como parametro (index)

Tamaño se le asigna el método length
Actual se le asigna el primero
SI el index es mayor igual que el tamaño
Imprimir (“ Busqueda fuera de rango ”)
Retornar -1
Caso contrario
SI el index es igual a 0
Retornar el actual
Caso contrario
Contador se le asigna 0

Mientras haya un siguiente
El actual avanza al siguiente
Contador incrementa en 1
Si el contador es igual al index
Retornar el actual

Método _printToNormal
#Imprime los elementos de la lista

Recibe como parametro (estado)

El actual se le asigna el primero
SI el actual es diferente de Nada
Mientras haya un siguiente del actual
Si el estado es igual a 1
Imprime (El actual)
Caso contrario
Imprime (Nombre del actual)
Si el estado es igual a 1
Imprime (El actual)
Caso contrario
Imprime (Nombre del actual)
Caso contrario
Retornar nada

Archivo “BuildGraf”
#Clase para los vertices.

Clase Vertex

Método constructor

Sus parametros son (Nombre)

Sus caracteristicas son:

Nombre que recibe nombre del parametro
Vertices que recibe un diccionario
Peso que recibe Nada

Método addEdge
#Utilizada para agregar una arista a un vertices, recibiendo sus caracteristicas.

Recibe como parametro (nodo, nombre, distancia, ancho de banda, usuarios conectados, trafico y tipo de medio)

Dentro del arreglo de vertices recibe el nombre del nodo que se le asigna el método getWeight

Método getWeight
#Retortnar el peso calculado.

Sus parametros son (distancia, ancho de banda, usuarios conectados, trafico y tipo de medio)

El subtotal para la distancia se le asigna el método decreaseInReliability
El subtotal para la banda ancha se le asigna el método calculateForBandwidth

Fiabilidad total se le asigna el subtotal para la distancia + el subtotal para la banda ancha

Si Fiabilidad total es menor a 0
Fiabilidad total se le asigna 0
Si Fiabilidad total es mayor a 1
Fiabilidad total se le asigna 1

Fiabilidad total se le asigna un formato para escribir dos números después del decimal del valor de Fiabilidad total
2.7800000001 --> 2.78

Se le asigna la Fiabilidad total al Peso

Retornar punto flotante del Peso

Método decreaseInReliability

Recibe como parametro (Distancia y tipo de medio)

Fiabilidad se le asigna nada
Partición se le asigna nada

SI tipo de medio es igual a ‘CAT 5’
Fiabilidad = 0.98
Disminución de confiabilidad = 0.02
Partición = 50

SI tipo de medio es igual a ‘CAT 6’
Fiabilidad = 0.98
Disminución de confiabilidad = 0.01
Partición = 50

SI tipo de medio es igual a ‘Fibra-Optica’
Fiabilidad = 0.90
Disminución de confiabilidad = 0.05
Partición = 100

SI tipo de medio es igual a ‘Wifi’
Fiabilidad = 0.7
Disminución de confiabilidad = 0.06
Partición = 6

SI tipo de medio es igual a ‘Coaxial’
Fiabilidad = 1
Disminución de confiabilidad = 0.04
Partición = 100

SI tipo de medio es igual a ‘Par-Trenzado’
Fiabilidad = 1
Disminución de confiabilidad = 0.01
Partición = 100

Sub total por distancia = (distancia/partición*disminución de confiabilidad)

#Sub total por distancia se le asigna el valor entero de la distancia entre la partición eso multiplicado por la disminución

Sub total por distancia = Fiabilidad - sub total por distancia

#Sub total por distancia se le asigna la fiabilidad menos el sub total por distancia actual.

Método calculateForBandwidth

Recibe como parametros (Ancho de banda, usuarios conectados, trafico)

Ancho de banda se le asigna el valor entero del ancho de banda
Cantidad de usuarios se le asigna el valor entero de la cantidad de usuarios
Trafico se le asigna el valor entero del trafico
Sub total por banda ancha se le asigna trafico - (ancho de banda) / cantidad de usuarios #mbps

Porcentaje se le asigna el subtotal por banda ancha / banda ancha * 100

Fiabilidad se le asigna 0

Si el porcentaje es mayor o igual a 1 y el porcentaje es menor a 25
Fiabilidad se le asigna 0.05
Si el porcentaje es mayor o igual a 25 y el porcentaje es menor a 50
Fiabilidad se le asigna 0.10
Si el porcentaje es mayor o igual a 50 y el porcentaje es menor a 75
Fiabilidad se le asigna 0.15
Si el porcentaje es mayor o igual a 75 y el porcentaje es menor a 100
Fiabilidad se le asigna 0.20
Si porcentaje es menor a 1
Fiabilidad se le asigna 0

Retornar fiabilidad

Clase Graph
#Clase para gestion de los vertices en el grafo.

Método constructor

No tiene parametros

Sus caracteristicas son:

Vertices se le asigna un diccionario

Método addVertex
#Agregar vertice.

Recibe como parametro vertex

Vertices se le asigna un arreglo concatenando el nombre del vertice asignando la arista del vertice

Método printGraph
#imrpimir el grafo.

```
Graf se le asigna vertices
Para k,v en los items de graf
    Para aris,w en items de v
        Imprimir (“ Vertices: k       Arista: aris - peso: w ”)
```

Método searchEdgeWeight
#Busca el peso entre un Vertice y Otro.

Recibe como parametro NombreVertex1, NombreVertex2)

```
Para k,v en los items de vertices
    Para aris, w en los items de v
        SI aris es igual al nombreVertex2
        Retornar w
```

Class BuildPaths
#Clase para encontrar las rutas de un grafo.
Método Constructor

No recibe parametros

Sus caracteristicas son:

Pila se le asigna un arreglo
Rutas se le asigna un arreglo

Método findPaths
#Encuentra y guarda TODAS las rutas entre un vertice origen, a un destino.

Sus parametros son (Inicio, destino, dict)

Agrega el vertice inicio.
Mientras la cola tenga valores.
Extrae el último indice de la lista
Extrae el ultimo valor del elemento
Llama a la función que 'resta' los elementos de las listas dadas, devolviendo otra lista.
Stop si el valor de la lista 'resta' es el destino deseado
Se agrega a la variable de rutas
En caso que no sea el valor destino, se sigue agregando rutas, volviendo al while

#'Resta' los valores de las listas dadas.
#ejm: ['A','B','C'] - ['A','C'] = ['B']

Método subtractList

Recibe como parametros (ListaA, ListaB)

listTemporal se le asigna un arreglo

Para i en la listaA
Si i esta en la posición 0 de la listaB
Pasar

Caso contrario
ListTemporal adjunta i en la posición 0

Retornar listTemporal

Método getPaths
#Retornar rutas
Retornar paths

Archivo “QuickSort”

Método constructor

No recibe parametros

No tiene caracteristicas

Método sort

Recibe como parametro (arreglo = [ ])

#Recibe el arreglo a ordenar

Retorna el llamado a la siguiente función (quicksort)

Método quickSort

Recibe como parametro (arreglo, minimo, max)

SI el minimo es mejor al max
#Ordenamiento por seleccion de pivote.
Pivote se le asigna el método partition

Se llama a si mismo el método quicksort enviando (arreglo, minimo y pivote -1)
#Particion 1(izquierda pivote)

Se llama a si mismo el método quicksort enviando (arreglo, pivote+1, elevado)
#Particion 2(derecha pivote)

Método partition

Recibe como parametros (arreglo, minimo, max)

i se le asigina (mimino - 1)

Pivote se le asigna arreglo en la posición del valor de max

Para j en el rango de minimo hasta max

SI el arreglo en la posición j es menor igual a pivote
i se le asigna i+1
Arreglo en la posición i, arreglo en la posición j se intercambian valores entre ellos.
Arreglo en la posición i+1, arreglo en la posición max se intercambian valores entre ellos.
Retorna (i+1)

Archibo “eventMouse”

Clase hoverButton(QPushButton)

Método constructor

Recibe como parametros (progenitor=nada)

Sus caracteristicas son:

QPushButton constructor

cssLeaveEvent
#Cambios visuales en el botón al soltar el punto.

cssEnterEvent
#Cambios visuales en el botón al pasar el puntero.

Establecer seguimiento del ratón

Fuente
#Tipo de fuente

Posición X en enteros
Posición Y en enteros
#Posición del mouse

Método enterEvent
#Funcion/evento al pasar el mouse por el boton.
Recibe como parametro un evento

Posición X recibe la posición del mouse en coor X
Posición Y recibe la posición del mouse en coor Y
setStyleSheet(cssEnterEvent)
#Establece un estilo
animacionCursos
#Todo lo relacionado a la animación y efectos del cursos
setFixedSize
#Cambia el tamaño
Fuente.setPointSize
#Fuente y tamaño
setFont
#Donde se establece (cambia)

Método leaveEvent

#Regresa el valor dado predeterminado.

Archivo “PrincipalWindow”

Clase Pwindow(QMainWindow)

Método constructor

No recibe parametros

Sus caracteristicas son:

Super()
#devuelve un objeto temporal de la superclase que luego le permite llamar a los métodos de esa superclase.

Title = “Ventana Principal”
#Titulo de la ventana
setGeometry(0,0,450,550)
#Geometria de la ventana [left,top,width,height]

#Creación de objetos y variables
“””
boxsText()
Buttons()
layoutPWindow()
objectGraph se le asigna Nada

“””

Paint = **Código CSS**
setStyleSheet(Paint)
#Estableciendo el estilo de los Widgets

centerWindow()
#Ventada centrada

Método boxsText
#Define y establece las cajas de texto para la ruta origen y destino.
No recibe parametros

boxOfCharacteristics se le asigna la clase QtextEdit
boxOfCharacteristics se le establece una geometria (10,60,430,380)
#Estos datos definen (x,y,TamañoX,TamañoY)
BoxOfOrigin se le asigna la clase QLineEdit
#Se le asigna un texto titular “Nodo origen” y un tamaño de 212,40 coor 10,450
boxOfDestiny se le asigna la clase QLineEdit
#Se le asigna un texto titular “Nodo Destino” y un tamaño de 212,40 coor 10,480

LabelTextMessege se le asigna la clase Qlabel
LabelTextMessege Su estado inicial es apagado
Su geometria es 10,480,400,40

Font se le asigna QFont
#Negrita activada y subrayado activado

LabelTitle se le asigna Qlabel
Se establece una geometria (0,10,460,30)
#Color blanco, texto definido: “PACKET TRACER”, alineado al centro, Fuente de Font

Método buttons

#Creacion de los botones de la ventana.

btnFileUpload
#Cargar Archivos
btnCreateMap
#Crear Mapa
btnCreateTable
#Crear Tabla
Minimize
#Minimizar la ventana
Close
#Cerrar la ventana

Método layoutPWindow

#Ordenamiento por cajas de los widgets

#----------FUNCIONES GENERALES----------

Método enableCreateMap
#Funcion que se activa al presionar el boton de crear mapa.

G = nx.DiGraph()
#Almacena nodos y bordes con datos opcionales o atributos
fig = plt.figure()
#Crear un objeto figura.
enableLabel(False)
#Etiqueta oculta
LLOfVertex = self.extractValuesToTextFormat()
#Extrae todos los vertices,aristas y caracteristicas del contenido de la caja de texto guardandolos en una lista enlazada el cual se asigna su valor a la variable LLOfVertex.

Si LLOfVertex es diferente de “”
objectGraph = self.convertLLInDict(LLOfVertex)
#Convierte y almacena la lista enlazada en diccionario (JSON)
Guardar peso de rutas se le asigna un arreglo

Si objectGraph es diferente a vacio
#Si hay un diccionario.

#Recorre el diccionario creado del texto, extrayendo sus llaves,valores,peso para dibujar. Agrega 2 pesos en las aristas, ejm: A-B y de B-A

Pos almacena el diseño (radio) del circulo
Labels almacena el retorno de dos tuplas en forma de (u,v)

Para llave,w en los items de Labels
#Devuelve el diccionario que contiene todo el grafo.
Concatenar se le asigna un string de la llave en posición 0 y 1
Concatenar replaza su forma a (“ ”, “ ”)
Guardar peso de las rutas adjunta lo concatenado

Si buscar rutas guardadas de la llave en posición 0 y , Guardar Peso de rutas es igual a Verdadero

Peso se le asigna el peso de la arista
Labels en la posición de las llaves almacena “(Peso)► Salto de linea◄(w)”

nx.draw_circular
#Establece y determina el dibujado utilizando 'nx'.
nx.draw_networkx_edge_labels
#Establece el color de letra, fuente, tamaño

fig.set_facecolor('white')
#Establece un fondo blanco
plt.savefig('graph.png',facecolor=fig.get_facecolor())
#Guarda la imagen dado una ruta.

openMapWindow
#Abre otra ventana para mostrar el mapa creado.

Caso contrario
enableLabel(True,"No hay un archivo cargado para dibujar el mapa.")
Caso contrario
enableLabel(True,"No hay contenido escrito.")

Método enableButtonUpload
#Carga el archivo de texto, y lo excribe en la caja de EditText

Usando la clase QFileDialog.getOpenFileName abrimos un explorador de archivos

El archivo seleccionado se almacena en “ruta del archivo”

Si se ha elegido un archivo, caso contrario muestra un mensaje.

Guarda el contenido extraido del archivo cargado.

Traspasa el contenido cargado a la caja de texto.

Método enableButtonCreateTable

LLOfVertex
#Extrae todos los vertices,aristas y caracteristicas del contenido de la caja de texto guardandolos en una lista enlazada el cual se asigna su valor a la variable LLOfVertex.

Si LLOfVertex es diferente de " "

objectGraph
#Convierte y almacena la lista enlazada en diccionario (JSON)
Guardar peso de rutas se le asigna un arreglo

NuevoJson se le asigna un diccionario { }
Construir Rutas hereda de la clase construir rutas
Rutas se le asigna nada
Sort hereda QuickSort
#Algoritmo de ordenamiemto, utilizado para ordenar ascendentemente los pesos de las rutas.
enableLabel(False)
#Etiqueta oculta

Si objectGraph es diferente de nada
#Si hay un diccionario, == None cuando no se ha cargado un archivo.

jsonGraph almacena los vertices del diccionario

Para k,v enlos items de jsonGraph
#Recorre el diccionario.
NuevoJson se le asigna un arreglo en la posición de k
Para aris,w en los items de v
Temporal se le asigna un arreglo
Temporal adjunta aris
Temporal adjunta w
NuevoJson adjunta el Temporal

Origin
#Extrae el texto de la caja origen.
Destino
#Extrae el texto de la caja destino.

EstadoO, EstadoD
#Almacenan el estado de la existencia del origen y el destino.

#----------Control del input del origen y destino----------

Si EstadoO, EstadoD son verdaderos
#En caso que los valores de las cajas, existan en el diccionario.
buildPaths.findPaths(origin,destination,newJson)
#Busca todas las rutas desde el punto origen, al destino.
buildPaths.getPaths()
#Devuelve y guarda las rutas encontradas.

self.labelTextMessage.setVisible(False)
#No mostrar mensaje

```
        Titulo = ("%s\n%s%s%s\n%s\n" % ('='*107,'\t'*4,'T A B L A   D E   R U T A
        S\n','\t\t\t---La ruta mas optima, es la de mayor peso(confiabilidad)--','='*107))
        labelPath = ("\t\t\t\tRuta desde %s hasta %s\n"% (origin,destination))
        titleSubTable = ("%s" % ('\tPeso\t|\tRuta\n')) dates = ""


#Este ciclo recorrera todos las rutas, para guardarlas en una LL, para asi mas
adelante poder ordenar las rutas desde el menor peso, al mayor peso...
#...por el cual el primer o primeros elementos de la lista en tabla, sera la ruta mas
corta.

        Para i en rutas
        #Ruta de vertices

        Peso almacena la busqueda de peso
        Agregar a LLOfPathsAndWeigth
        #name:ruta,distance:peso

        Arreglo Peso adjunta Peso

    Ordenar arreglo Peso
    Ruta temporal se le asigna un arreglo

    Para j en arreglo peso
        Estado se le asigna verdadero
        Para k en el rango de la longitud de LLOfPathsAndWeigth
            Nodo actual se le asigna la posición de LLOfPathsAndWeigth en k

            Si j es igual al nodo actual yel nombre del nodo actual no esta en ruta
            temporal y estado es verdadero
            #En el atributo distance esta guardado el peso.
                Ruta Tempora adjunta nombre del nodo actual
                Temporal se le asigna ("%s\n\t%s\t|\t%s\n" %
                ('-'*190,j,",".join(currentNode.name)))
                Datos se concatena con temporal
                Estado ahora pasa a ser falso
        Contenido   se le asigna titulo, etiqueta de la ruta,titulo de la sub tabla y
        datos
        #Contiene todo el contenido a mostrar en la ventana de la tabla.

    Abrir ventana de rutas

    Caso contrario SI EstadoO y EstadoD Verdadero, falso
        enableLabel(True,'El valor destino no existe')
    Caso contrario SI EstadoO y EstadoD falso, Verdadero
    enableLabel(True,'El valor destino no existe')
    Caso contrario SI EstadoO y EstadoD falso, falso
    enableLabel(True,'El valor origen y destino no existen')

    Caso contrario
    enableLabel(True,"No hay un archivo cargado para obtener la tabla.")
Caso contrario
    enableLabel(True,"No hay contenido escrito.")
```

#---------- FUNCIONES LLAMADAS POR LAS GENERALES ----------

Método searchWeight
#Funcion que extrae el peso de las aristas, reciendo las rutas y el objeto grafo.

Recibe como parametros (rutas, jsongraph)

ArregloPeso se le asigna un arreglo
Peso se le asigna 0

Para i en el rango de la longitud de las rutas
    Index se le asigna i+1
    Si index es menor que la longitud de las rutas
    Adjuntar al arregloPeso el peso de la arista del jsongraph

Para j en arregloPeso
    Peso se le asigna peso + j
Retornar peso

Método searchPathsSaved
#Esta funcion es utilizada a la hora de dibujar los pesos entre las aristas, para poder mostrar ambos pesos, ejm: peso de A - B y peso de B - A => (5,4)

    Parametros recibidos (ruta1, ruta2, arreglo)

    Item se le asigna un string concatenado de ruta2 y ruta1
    Item se le asigna remplazar a formato " ", " "
    Para i en arreglo
        Si i es igual a item
        Retornar Verdadero

Método extractContentToBoxOfCharacteristics
#Esta funcion extrae todo el contenido del archivo seleccionado.

    Recibe como paramatro (ruta del archivo)

    Lista temp se le asigna un arreglo
    Se abre el archivo en modo lectura
    Se leen todas las lineas
    Size almacena el tamaño de lineas
    Para i en el rango de la longitud de lineas
        Adjuntar en lista temp lineas en el rango de i

        Si i es igual al size-1
            Si linea en el rango de i es diferente a un salto de linea
                Adjuntar a la lista temp un salto de linea.
    Cerrar el archivo
    Retornar la lista temp

Método checkVertexExists
#Revisa si los vertices existen en el diccionario dado.

Recibe como parametro (origen, destino, json)

Arreglo temporal se le asigna un arreglo
Para k en los indices del json
Adjuntar en Arreglo temporal el indice (k)

Para i en arreglo temporal
SI origen en arreglo temporal y destino no esta en arreglo temporal

Retornar Verdadero, Falso

Si origen no esta en arreglo temporal y destino esta en arreglo temporal

Retornar Falso, Verdadero

Si origen no esta en arreglo temporal y destino no esta en arreglo temporal

Retornar Falso, Falso

Si origen y destino estan en arreglo temporal

Retornar Verdadero, Verdadero

Método extractValuesToTextFormat

#Extrae todos los vertices,aristas y caracteristicas del contenido de la caja de texto guardandolos en una lista enlazada.

No recibe parametros

textOfBoxOfCharacteristics
#Almacena el texto plano
Si textOfBoxOfCharacteristics es diferente de “”
Hacer una separación en cada salto de linea
vertexLL
#Lista enlazada de vertices

Para i en el rango de la longitud de textOfBoxOfCharacteristics
#Recorrera el contenido de la caja, para obtener los vetices.
Linea actual
Si el vertice no tiene tabulado
Buscar item se le asigna tabulado linea actual
Para j en el rango de la longitud de textOfBoxOfCharacteristics
Item actual secundario se le asigna textOfBoxOfCharacteristics en la posición j
Si item actual secundario es igual a buscar item
Arista con el mismo nombre del vertice actual, para extraer sus caracteristicas.
Agregar a la lista enlazada
Parar
Para k en rango de la longitud de textOfBoxOfCharacteristics
#Recorrera el contenido de la caja, para obtener las aristas.

Método convertLLInDict

#Convierte los datos extraidos del texto que fueron almacenados en una LL, a un diccionario, para luego crear el mapa.

Recibe como parametros (LLOfVertex)

graph de la clase Graph
LLOfEdges se le asigna una lista enlazada

Para i en rango de la longitud de LLOfVertex
#Recorre los vertices de la lista enlazada.
    El vertice actual
    Vertex se le asigna el nombre del vertice actual
    LLOfEdges.add
    #Almacena los vertices

Para j en el rango de la longirud de LLOfVertex
#Recorre los vertices de la lista enlazada.

    El vertice actual en la posición de J
    Vertex almacena la busqueda de las aristas en la posición de j
    Para k en el rango de la longitud de los vertices de las aristas
        Arista almacena la posición de la arista del vertice en la posición de k
        #Nodo arista
        EL objeto arista almacena el nombre de la busqueda de la lista enlazada de aristas

        Vertex agrega la arista
        #Nodo arista(clase vertex)
    Graph agrega el vertice
Retornar el grafo

Método extractValueToString
#Extrae el valor de una caracteristica(str). ej: Distancia:100 => 100

    Recibe como parametro (string, a buscar = ":")
    Indice a buscar se le asigna string buscado
    Nuevo caracter se le asigna string[indice a buscar +1 hasta longitud de string]
    Retorna nuevo caracter

Método enableLabel
#Activa un texto en la pantalla, mandando como parametro el texto a mostrar.

    Recibe como parametros (Estado, texto = '')

Método centerWindow
    Toma la geometria de la pantalla
    Mueve la ventana multiplicando el ancho por 1.5, alto se deja igual.
#Centrado de la ventana.

Método openMapWindow
#Abre la ventana que muetra el mapa.
MapWindow (clase interna)
Mostrar MapWindow

Método openTableWindows

Recibe como parametros (contenido)

TableWindow (Clase interna)
Mostrar TableWindow

Método minimizar
showMinimized (Minimizar)
Método Closewindow
sys.exit() (Terminar ejecución)

Clase MapWindow(QDialog)

Método constructor

Recibe como parametro (progenitor)

Titulo = "Grafica del grafo"
#En la etiqueta label se pega la imagen del grafo y se muestra en la ventana y la ventana se ajusta al tamaño de la imagen

Clase TableWindow(QWidget)

Método constructor

Recibe como parametro (contenidor de Rutas)

Titulo = "Tabla de rutas"
Almacenamos el contenido de las rutas
La geometria de la ventana es (0,0,750,400) #Ancho, Alto
Center #Centrado
Content se le asigna QTextEdit
La geometria de la caja es (0,0,750,400)
El texto de Content es el contenido de las rutas

Método center

Toma la geometria de la pantalla
Mueve la ventana dividiendo entre dos el alto y ancho de la misma
#Centrado de la ventana.