📒

Parcial 2 documentación

Integrantes:

🦸 `Franco Pignanelli`

🏌️ ***Augusto Olivera***

Métodos utilizados

1. Carga de datos

Con este método obtenemos los datos del archivo línea por línea, dentro nos vamos a encontrar 2 IF y un ELSE.

1. Dentro del primer IF vamos a leer la línea correspondiente a los datos de la cantidad de esquinas, la posición del colectivo y la posición de la escuela.
   Utilizamos dos variables auxiliares para manejar el subíndice de de la posición del colectivo y la escuela.
   También hay un contador llamado ✨*contador*✨ que se utiliza para controlar si se ingresa en el IF actual o en los próximos.
2. En el segundo IF se leen la cantidad de calles.
3. El ELSE se trabajan todas las demás líneas del archivo de entrada agregando los datos de las aristas (con quienes se conecta y su coste)

```
def cargar_datos(self):

        archivo = open("entrada.in")
        contador = 0

        for linea in archivo:

            if (contador == 0): #primera linea (esquinas-pos_colectivo-pos_escuela)
                datos = linea.split(" ")

                #Validamos los datos para que se mantengan detro de un rango aceptado
                datos[0] = self.validar( int(datos[0]), 1, 80000)

                for dato in range(0, int(datos[0]) ) :
                    self.vertices.append( str( dato + 1) )

                #Auxiliar para corregir el subindice,
                aux1 = int( datos[1] ) -1
                aux2 = int( datos[2] ) -1
                self.colectivo = self.vertices[aux1]
                self.colegio   = self.vertices[aux2]
                contador += 1
                continue

            if (contador == 1): #Segunda linea (numero_calles)
                total_calles = linea.split(" ")
                total_calles[0] = self.validar( int(total_calles[0]), 1, 250000)
                contador += 1
                continue

            else: #todas las demás lineas (esq_salida; esq_llegada; peso)
                calle = linea.split(" ")
```

```
                calle[2] = self.validar( int(calle[2]), 1, 50) ## validamos los decámetros
                self.aristas.append( (calle[0], calle[1], int( calle[2] ) ) )
                pass
```

⚠️ Se utiliza el método `validar` para asegurarnos que los datos de entrada se mantenga dentro de un valor mínimo y máximo, en caso de que esté por fuera, los ubica en los extremo mayor o menor según corresponda.

2. Creación del grafo

En este método primer se cargan los vértices, las aristas.

En una segunda parte guardamos el recorrido del camino mínimo de el bus hacia la escuela y luego guardamos el coste de dicho viaje.

Por último tenemos las funciones propias de la librería que fabrican al grafo. 🤭

```
def fabricar_grafo( self ):
        self.Grafo.add_nodes_from( self.vertices )
        self.Grafo.add_weighted_edges_from( self.aristas )

        self.dijkstra = nx.dijkstra_path( self.Grafo, self.colectivo, self.colegio )
        self.distancia = nx.dijkstra_path_length( self.Grafo, self.colectivo, self.colegio ) #* se guarda el costo de dikjstra

        nx.draw( self.Grafo, pos=nx.circular_layout( self.Grafo ), node_color='r', edge_color='g', with_labels=True )
        plt.show()
```

3. Cambios de calle

- Inicializamos la variable numerito_de_calle en 1
- Tenemos un primer FOR que itera por cada arista del total de aristas y lo que haces es:
  - En calle_girada guardamos el valor de Invertir la dirección de la arista.
  - Tenemos un IF que compara cada una de las aristas por la que pasa el resultado de aplicar dijkstra contra el valor de calle_girada, en caso de que se cumpla significa que la tiene que girar a la calle entonces guardamos su número en el vector de calles por girar.
  - Aumenta el contador numerito_de_calle para en la próxima iteración trabajar con la siguiente calle.

```
def buscar_cambios_de_calle( self ):

        numerito_de_calle = 1
        for arista in self.aristas:

            calle_girada = (arista[1], arista[0])

            for i in range (1, len(self.dijkstra)):
                if ( ( calle_girada[0] == self.dijkstra[i-1] ) and ( calle_girada[1] == self.dijkstra[i]) ):
                    self.vector_de_calles_cambiables.append(numerito_de_calle)

            numerito_de_calle += 1
```

Casos de prueba

Caso de prueba 1

Caso de pruebas con los datos de la consigna.

▼ Entrada.in:

```
8 2 7
13
2 4 4
2 1 5
2 3 2
3 1 3
4 1 4
7 5 3
1 5 3
1 6 7
8 4 2
6 8 1
8 7 6
6 7 8
5 3 2
```

▼ Salida.out

```
7
6 13
```

Caso de prueba 2

▼ Entrada.in

8 2 7
12
3 4 4
1 2 5
2 3 2
3 4 3
8 4 4
1 3 3
2 6 3
2 7 7
7 5 2
5 9 1
7 8 6
5 6 8

▼ Salida.out

7

La salida es solo 7 porque no hay ninguna calle por invertir debido a que el camino mínimo se recorre como está.

Verificamos que esto se cumple en el siguiente caso de prueba

Caso de prueba 3

Aumentamos el peso de unas de las aristas (mirar número subrayado) de la calle tomada en el caso de prueba anterior para forzar a que cambie el camino y verificar que la salida sea distinta

▼ Entrada.in

```
8 2 7
12
3 4 4
1 2 5
2 3 2
3 4 3
8 4 4
1 3 3
2 6 3
2 7 70
7 5 2
5 9 1
7 8 6
5 6 8
```

▼ Salida.out

```
13
9 12
```

Efectivamente la salida cambió, ya que se tuvo que buscar un camino distinto y se tuvieron que girar algunas calles.