1IEE14 - Laboratorio 11

Instrucciones para el laboratorio:

- Materiales permitidos: Wiki del curso, apuntes de clase, consultar foros, tutoriales o documentación de python online.
- Está prohibido el uso de cualquier modelo de lenguaje como ChatGPT o alguno similar. A cualquier alumno que se le detecte que ha consultado un modelo de lenguaje se le pondrá nota 0(cero) en el laboratorio.
- Usted debe subir a Paideia 1 solo archivo comprimido (.zip o .rar) con el nombre L11_CODIGOPUCP.zip o L11_CODIGOPUCP.rar. Este archivo comprimido debe tener archivos de python(extensión .py) para cada pregunta. No se aceptarán soluciones en Jupyter notebook.
- Se espera que el nombre de los archivos sea: pregunta1.py, pregunta2.py, etc. A no ser que la pregunta especifique un nombre para su(s) archivo(s).
- El horario máximo permitido para subir el archivo es a las 10:00:00 pm. Pasada esa hora, habrá una penalidad de 2 puntos por cada minuto extra que se demore en entregar su archivo.

Pregunta 1 **(9 puntos)**

Se solicitará dos archivos: “P1-cliente.py” y “P1-servidor.py”.

a. (2 puntos) Cree un primer thread en el archivo servidor.py el cual, luego de establecer conexión con un cliente, leerá el archivo PartesDeElectrónica.csv

La cabecera de este archivo es: [ID, Nombre, Número de parte, Cantidades, Peso, Costo unitario] Este archivo contiene los datos de varios componentes electrónicos que se quieren comprar en un proveedor de China.

b. (2 puntos) El segundo Thread del servidor, en intervalos de 1 segundo, le enviará al cliente la cadena de caracteres de una fila (datos de un componente electrónico), así consecutivamente hasta completar de leer el archivo.

c. (1 punto) En el archivo P1-cliente.py, cree un thread que se encargue de recibir los datos del servidor.

d. (2 puntos) Cree un segundo thread en el cliente que se encargue de procesar los datos de cada fila recibida. Debe calcular lo siguiente:

- Cálculo del costo total (CT = Precio unitario * cantidades)
- Clasificación del componente por costo total:

  (Menor a 75): Costo bajo

  (75.0 – 149.9): Costo regular

  (150 - 199.9): Costo alto

  (200 a más): Costo elevado

- Conteo de componentes que tengan costo elevado, conteo de componentes con **peso mayor a 2000g** y con costo elevado, y conteo de componentes con costo bajo (conteos actualizados por cada componente que se procese).

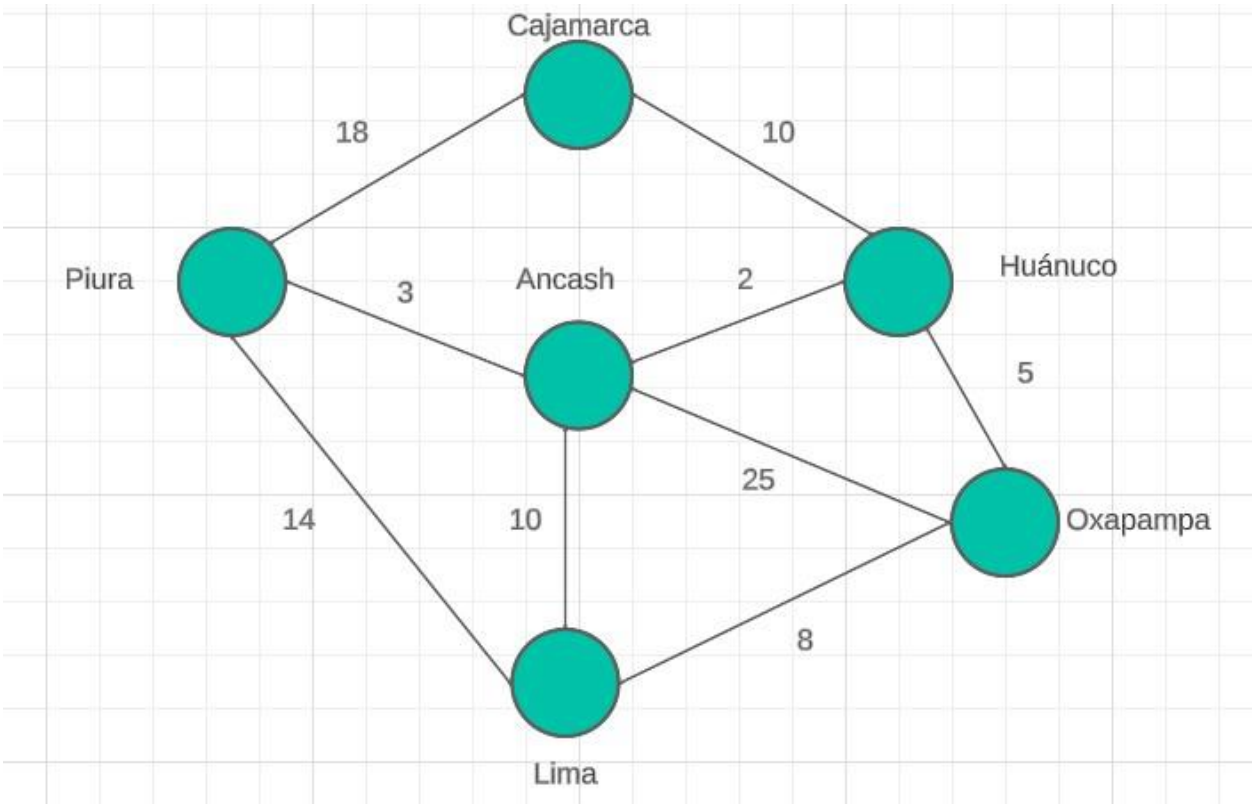
e. **(2 puntos)** Asimismo, el segundo thread imprimirá en terminal los resultados de cada fila de la siguiente forma:

```
--------------Nombre: XXXX-----------------
Costo total: XXX
Clasificación por costo: XXXX
Número de componentes con costo elevado: XXX
Número de componentes con costo elevado y con peso mayor a 2000g: XXX
Número de componentes con costo bajo: XXX
--------------Nombre: XXXX-----------------
Costo total: XXX
Clasificación por costo: XXXX
Número de componentes con costo elevado: XXX
Número de componentes con costo elevado y con peso mayor a 2000g: XXX
Número de componentes con costo bajo: XXX
--------------Nombre: XXXX-----------------
Costo total: XXX
Clasificación por costo: XXXX
Número de componentes con costo elevado: XXX
Número de componentes con costo elevado y con peso mayor a 2000g: XXX
Número de componentes con costo bajo: XXX
```

Nota: solo debe imprimir en terminal si y solo si se tiene un nuevo dato a imprimir

Pregunta 2 **(6 puntos)**

Se tiene el siguiente gráfico, donde cada círculo representa una ciudad y las líneas representan las carreteras que unen cada ciudad. Los números en cada línea representan la cantidad de horas que tomaría viajar por tierra entre cada ciudad:



Se desea averiguar cuál es el camino más corto para ir desde Lima (punto de partida) hasta Cajamarca (destino final).

(5 puntos) Calcule el destino más corto usando el método de fuerza bruta: Ejecute una corrrutina por cada combinación posible y que cada corrtuina calcule el tiempo del recorrido. Descargue la plantilla *lab11_pregunta2_plantilla.py* la cual contiene la lista de todas las combinaciones válidas posibles. Su programa debe comparar los tiempos de viaje y debe imprimir la ruta más corta.

Indicaciones de cómo se espera que implemente el programa:

- Debe usar corrutinas.
- Cada corrutina debe recibir como argumento de entrada la ruta que va a evaluar. El argumento de entrada debe ser una cadena de texto. Por ejemplo:

  - Si quiere que la corrutina evalúe la ruta Lima-Piura-Cajamarca, el argumento de entrada debe ser “LPC”.

  - Si quiere que la corrutina evalúe la ruta Lima-Oxapampa-Huánuco-Cajamarca, el argumento de entrada debe ser “LOHC”.

  - Dentro de cada corrutina, para simular la demora por cada tramo, debe usar la función asyncio.sleep(). Por ejemplo: Si su corrutina recibió como parámetro de entrada la cadena de texto “LPC”, su corrutina debe poder separar cada tramo, es decir, darse cuenta que tiene que evaluar 2 rutas: Lima-Piura, PiuraCajamarca. Como son 2 rutas que tiene que evaluar, ejecutará 2 veces el asyncio:

  ```
  asyncio.sleep(14)  # Esto es para simular la demora de Lima a Piura
  asyncio.sleep(18)  #  Esto es para simular la demora de Piura a Cajamarca
  ```

  Si lo desea, puede convertirlo a milisegundos para que no espere mucho tiempo.

- Las corrutinas deben retornar la cantidad de tiempo que demora recorrer su ruta. Por ejemplo, la que recibió como argumento de entrada “LPC” debe retornar 32, la que recibió como argumento “LAHC” debe retornar 22.
- Cuando todas las corrutinas hayan terminado de ejecutarse, su programa debe procesar los resultados e imprimir la siguiente frase:

  La ruta más corta es ABCD con una duración de X horas.

(1 puntos) Si se implementa el mismo programa, pero utilizando threads, ¿qué programa sería más eficiente? Jusitifique.