**1IEE14 - Laboratorio 10**

**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 usar Github CoPilot. A cualquier alumno que se le detecte que ha consultado un modelo de lenguaje se le pondrá nota 0(cero) en el laboratorio.
- No está permitido el uso de la librería Pandas ni ninguna otra librería para leer y/o escribir en los archivos csv.

- Usted debe subir a Paideia 1 solo archivo comprimido (.zip o .rar) con el nombre L10_CODIGOPUCP.zip o L10_CODIGOPUCP.rar. Este archivo comprimido debe tener archivos de python para cada pregunta. Note que se le está pidiendo archivos de python para cada pregunta, no se aceptarán soluciones en jupyter notebook.

- 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** (5 puntos)

Se desea descargar un conjunto de 29 imágenes, que se pueden encontrar en los siguientes enlaces:

| |
|---|
| https://raw.githubusercontent.com/SebastianMerino/Threading/main/images/01.png |
| https://raw.githubusercontent.com/SebastianMerino/Threading/main/images/02.png |
| ... |
| https://raw.githubusercontent.com/SebastianMerino/Threading/main/images/29.png |

a) (1 punto) Descargue los archivos secuencialmente y mida el tiempo de ejecución. Puede utilizar la función urlopen(), la cual funciona de manera similar a la función open() de Python:

```
from urllib.request import urlopen
url = ''

with urlopen(url) as page:
    image_data = page.read() # data como objeto binario
```

b) (2 puntos) Descargue los archivos utilizando hilos. Genere un hilo por cada imagen a descargar. Mida el tiempo de ejecución y comente sus resultados

c) (2 puntos) Descargue los archivos utilizando solo 3 hilos. Mida el tiempo de ejecución. ¿Obtuvo mejores o peores resultados que en b)? ¿Por qué?

**Nota**: Para la medición de los tiempos de ejecución en la pregunta 1, se debe realizar la prueba 5 veces y obtener el tiempo mediano.

**Pregunta 2** (5 puntos)

En el archivo *lab10_pregunta2_plantilla.py* se ha definido la función get_ntp_time() la cual imprime la hora en un país determinado. Su argumento de entrada es cualquiera de las cadenas definidas en la lista servidores_ntp[].Lo que hace internamente la función es obtener la hora de esas URL según el país solicitado. Por ejemplo:

Ejemplo1: si Ud. llama a la función de esta manera: get_ntp_time("1.es.pool.ntp.org"); la función imprimirá la hora actual en España.

Ejemplo2: si Ud. llama a la función de esta manera: get_ntp_time("0.uk.pool.ntp.org"); la función imprimirá la hora actual en Reino Unido.

Indicaciones sobre la función get_ntp_time():

- Aparte de imprimir la fecha-hora, también retorna el timestamp en formato datetime.datetime.

- Los países disponibles en la plantilla son aquellos donde se encuentran las principales bolsas de valores del mundo donde uno puede comprar o vender acciones, lo cual será importante para este ejercicio. No tiene que modificar_get_ntp_time() por dentro, tampoco es necesario que entienda cómo es que funciona, solo úsela para lo siguiente:

a) (2 puntos) Escriba una función que itere sobre todas las cadenas en la lista servidores_ntp[] para determinar cuál es el país cuya bolsa de valores está más próxima a abrir con respecto a nosotros(en Perú). Asuma que todas las bolsas de valores abren a las 08:00am en sus respectivos países.

Por ejemplo: Supongamos que en actualmente en Perú fueran las 12:00, y en los otros países las horas son:

- UK: 17:00
- España: 18:00
- Estados Unidos: 12:00
- Hong Kong: 01:00
- Japón: 02:00

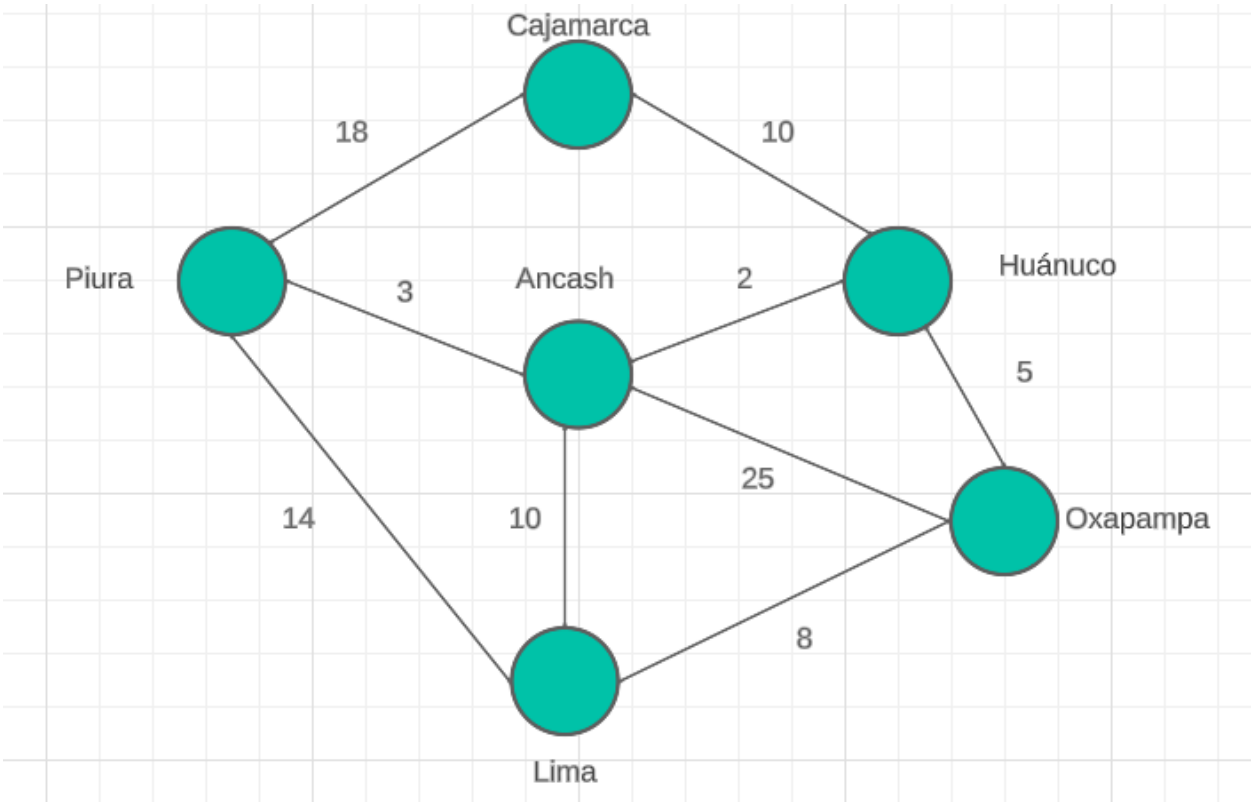
Entonces la respuesta correcta es que Japón sería el más próximo a abrir, porque es el que está más cercado a las 08:00am (pues solo le faltan 6 horas para abrir). Su programa al final debe imprimir el país que ha calculado tener la hora más cercana. Imprima el tiempo de ejecución de esta función.

**Nota**: Esos servidores son gratuitos y usan un tipo de protocolo que no responde el 100% de las veces, así que puede que a veces alguno de ellos no responda. Si Ud. ve que alguna URL se está demorando más de 10 segundos en imprimir su hora, cancele la ejecución de su programa(ctrl+C) y vuélvalo a ejecutar. Si el problema persiste, puede quitar esa URL de la lista servidores_ntp[].

b) (3 puntos) Haga lo mismo que en la parte a) pero en vez de iterar sobre la lista, use threads para cada uno de los elementos en la lista servidores_ntp[]. Una vez terminado los threads, su programa debe imprimir el país que ha calculado tener la hora más cercana. Imprima el tiempo de ejecución de esta función. Indique cuál función fue más rápida, si la implementada en la parte a) o en la parte b) en base a sus tiempos de ejecución.

**Pregunta 3** (5 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).

Calcule el destino más corto usando el método de fuerza bruta: Ejecute una corrrutina por cada combinación possible y que cada corrtuina calcule el tiempo del recorrido. Descargue la plantilla *lab10_pregunta3_plantilla.py* la cual contiene la lista de todas las combinaciones válidas possibles. 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:

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

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

  a) 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, Piura-Cajamarca. 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.*