Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2115 – Programación como Herramienta para la Ingeniería (II/2021)

Taller 1b

Objetivos

- Utilizar estructuras de datos básicas.

Entrega

- **Lenguaje a utilizar:** Python 3.6 o superior
- **Lugar:** repositorio privado en GitHub. Recuerde incluir todo en una carpeta de nombre **T1b**.
- **Entrega: lunes 30 de agosto a las 16:50 hrs.**
- **Formato de entrega:** archivos python notebook (**T1b.ipynb**) y python simple (**T1b.py**) con la solución a lo solicitado en el enunciado. Ambos archivos deben estar ubicados en la carpeta **T1b**. No se debe subir ningún otro archivo a la carpeta. En el archivo python notebook, utilice múltiples celdas de texto y código para facilitar la revisión de su programa. El archivo python simple debe ser exportado a partir del python notebook.
- **NO SE ADMITEN ENTREGAS POR CORREO U OTROS MEDIOS NI ENTREGAS FUERA DE PLAZO**
- **Entregas con errores de sintaxis y/o que generen excepciones serán calificadas con nota 1.0.**

Introducción

Con el fin de ejercitar el uso de estructuras de datos, en este taller deberá desarrollar 2 ejercicios. El primero será guiado a través de instrucciones en el enunciado, mientras que en el segundo el desarrollo será por su cuenta.

Ejercicio 1: caminos en grafos no dirigidos

Utilizando OOP y estructuras de datos básicas, modele un verificador de caminos en grafos no dirigidos de dos maneras distintas: i) utilizando listas y ii) utilizando diccionarios. Los nodos serán identificados con *strings* de largo 1, compuestos por letras mayúsculas. Para realizar la modelación, considere los siguientes instrucciones:

- Cree una clase para cada tipo de verificador (`Verificador_Lista` y `Verificador_Diccionario`). Las clases deben implementar al menos 2 métodos: `__init__(arcos)` y `existe_camino(camino)`. Puede implementar más métodos si lo prefiere o requiere.
- Los arcos de los grafos serán entregados utilizando una lista de tuplas (una tupla por arco), donde cada tupla contendrá los 2 nodos que une el arco. Puede asumir que los arcos estarán bien formados y sin errores, pero no que siguen un orden específico.
- Los nodos de los caminos serán entregados como una lista de *strings*, donde cada *string* indica un nodo del camino. El orden de la lista indica el orden del camino. Puede asumir que cada uno de los nodos en el camino existe en el grafo, pero no puede asumir que el camino existe en el grafo o que el camino no tiene nodos o arcos repetidos.
- Para cada una de las clases, almacene los arcos en la estructura de datos correspondiente, es decir, lista o diccionario. Recuerde que el grafo es **no dirigido**.
- Para verificar la existencia de los caminos, itere por cada arco considerado en este y compruebe si está almacenado en la estructura de datos utilizada. El método debe retornar (no imprimir en pantalla) una tupla, indicando si el camino existe (`True` o `False`) y cuántas verificaciones de arcos debió realizar en la estructura almacenadora. Por ejemplo, si un camino no existe y el método iteró sobre 45 arcos en total para realizar la verificación, el retorno debe ser `(False, 45)`. **IMPORTANTE:** Para la clase basada en una lista, no está permitido utilizar métodos que realicen una búsqueda automática en esta (por ejemplo `index` o `in`), para verificar si existe un arco.

- Un ejemplo de ejecución es el siguiente:

  Código

```
arcos = [("A","B"), ("A","E"),("A","C"), ("B","D"),("B","E"),
         ("C","F"),("C","G"), ("D","E"), ("E","F")]
camino = ["A","C","F","E","A","B"]
verificador = Verificador_Lista(arcos)
res = verificador.existe_camino(camino)
print(res)
verificador = Verificador_Diccionario(arcos)
res = verificador.existe_camino(camino)
print(res)
```

  Salida

```
(True, 21)
(True, 10)
```

  Es importante mencionar que tanto el 21 como el 10 de la salida son resultados posibles, pero no necesariamente únicos.

- **OPCIONAL**: modifique la clase basada en diccionarios para que además retorne la cantidad de veces que el camino para por el arco más visitado, sin utilizar una estructura de datos adicional.

Ejercicio 2: corrección de claves secretas corruptas

Considere la siguiente situación: ud. recibe una clave secreta en forma de número natural, que debe ser utilizada para desactivar una bomba. Lamentablemente, por problemas en el canal de comunicación, la clave recibida tiene algunos errores. Después de realizar análisis estadísticos, se ha identificado que el valor recibido tiene dígitos duplicados. Escriba un programa que los elimine, manteniendo sólo la primera aparición de cada dígito y entregue como respuesta el número resultante. Un ejemplo de ejecución del algoritmo es el siguiente:

Código

```
def eliminar_duplicados(n):
    #solucion al problema


n = 1242241
n_ = eliminar_duplicados(n)
print(n_)
```

Salida

```
124
```

Objetivo parcial de participación

Para obtener el puntaje máximo asociado a participación durante la clase, debe implementar completamente la clase `Verificador_Diccionario` del Ejercicio 1.

Política de Integridad Académica

Los alumnos de la Escuela de Ingeniería deben mantener un comportamiento acorde al Código de Honor de la Universidad:

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un procedimiento sumario. Ejemplos de actos deshonestos son la copia, el uso de material o equipos no permitidos en las evaluaciones, el plagio, o la falsificación de identidad, entre otros. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica en relación a copia y plagio: Todo trabajo presentado por un alumno (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), sin apoyo en material de terceros. Si un alumno (grupo) copia un trabajo, se le calificará con nota 1.0 en dicha evaluación y dependiendo de la gravedad de sus acciones podrá tener un 1.0 en todo ese ítem de evaluaciones o un 1.1 en el curso. Además, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir un procedimiento sumario. Por "copia" o "plagio" se entiende incluir en el trabajo presentado como propio, partes desarrolladas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.