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

IIC2115 – Programación como Herramienta para la Ingeniería (II/2019)

Laboratorio 3 - Técnicas y algoritmos

Objetivos

- Aplicar los contenidos de técnicas y algoritmos, a través de la resolución de problemas específicos.

Entrega

- **Lenguaje a utilizar:** Python 3.6
- **Lugar:** repositorio privado en GitHub. Recuerde incluir todo en una carpeta de nombre **L03**.
- **Entrega: domingo 6 de octubre a las 23:59 hrs.**
- **Formato de entrega:** archivo python notebook con nombre **solucionL03** (**solucionL03.ipynb**) y archivo python con el mismo nombre (**solucionL03.py**) con la solución de este enunciado. Los archivos deben estar ubicados en la carpeta **L03**. No se debe subir ningún otro archivo a la carpeta. Utilice múltiples celdas de texto y código para facilitar la revisión de su tarea. Los archivos **.ipynb** y **.py** deben contener la misma solución.
- **Descuentos:** se descontará 0.5 puntos por cada hora de atraso y fracción en la entrega final. Tareas que no cumplan el formato de entrega tendrán un descuento de 0.5 ptos.
- **Tareas con errores de sintaxis y/o que generen excepciones serán calificadas con nota 1.0.**
- Si su laboratorio es entregado fuera de plazo, tiene un hasta el **lunes 7 de octubre a las 23:59 hrs** para responder el formulario de **entregas fuera de plazo** disponible en el Syllabus.
- Las discusiones en las *issues* del Syllabus en GitHub son parte de este enunciado.

Resolución de problemas

En este laboratorio deberán resolver una serie de problemas cortos sobre técnicas y algoritmos. Cada problema es independiente de los demás y se espera que sean resueltos utilizando los conocimientos de los capítulos 2 y 3 del curso. Para obtener el máximo de puntaje por problema, cuando se indique, estos deben ser solucionados usando los tópicos indicados en cada uno de ellos. De lo contrario, sólo obtendrán una parte pequeña del puntaje.

Para entregar sus soluciones, debe definir una función llamada `problema_x()` (con minúsculas), donde `x` corresponde al número del problema. Además, cada función debe recibir como input y retornar su respuesta de la forma indicada en cada problema. Por ejemplo, Si el problema 1 requiere, un entero, un texto y una lista (en ese orden) y se le pide retornar un entero, debe crear la siguiente función:

```
def problema_1(entero, texto, lista):
    #Lógica y resolución
    return numero
```

funciones prohibidas y limitadas

Para resolver los problemas, se prohíbe y/o limita el uso de algunas funciones de Python.

- Métodos de ordenamiento. Funciones como `sort()` no pueden ser utilizadas. En su defecto pueden programar alguna lógica de ordenamiento. Es importante que tengan conocimiento de la complejidad de lo que van a utilizar.
- Métodos de borde o valor. Funciones como `max()`, `min()` o `index()` debe controlarse su uso. Recuerden que uso corresponde con una iteración por todo el arreglo. Esto puede elevar considerablemente la complejidad de su algoritmo.

Problemas

1 **Tirar la cuerda - dificultad básica** (1 pts)

Un grupo de niños ha decidido jugar al clásico juego de tirar la cuerda. Este juego consiste en que los niños se separan en dos grupos, se paran junto a una cuerda y jalan de ella en direcciones opuestas. El grupo que logre tirar al otro hacia su lado, es el ganador. Cada niño posee un índice de fuerza que corresponde a un número entero. Dado esto, se tiene una lista que representa a los niños alineados

junto a la cuerda. Uno de los niños puede ser juez si los grupos que separa (a la izquierda y a la derecha) poseen la misma suma de fuerzas. Para evaluar a los niños que se encuentran en los extremos, entonces los demás deben sumar 0. El objetivo es saber quiénes pueden ser jueces dada una lista de niños. Para identificar a los niños, utilice la posición de estos en la lista.

Entrada:

```
Lista de enteros: [0,-4,6,-4,-3,3,2,0]
```

Salida:

```
Lista de enteros: [0,3,7]
```

(a) Resuelva el problema utilizando completamente fuerza bruta, es decir, analice todos los casos para entregar su respuesta y no use ninguna lógica que le ahorre tiempo de resolución. Llame a esta función `problema_1a`. Responda en la celda siguiente: ¿De qué orden es la solución construida?

(b) Resuelva el mismo problema pero ahora debe reducir el orden de solución. Es decir, trate de optimizar el tiempo de solución de la parte (a). Llame a esta función `problema_1b`. Responda en la celda siguiente: ¿De qué orden es la solución construida?

(c) Utilice la librería entregada junto con el enunciado (uso explicado más adelante). La librería cuenta con el método `graficar`, úselo para construir dos gráficos *tamaño de input vs tiempo de ejecución* para las soluciones de las parte (a) y (b). Comente lo que observa.

2 **Los Terremotos diciocheros - dificultad media** (1 pts)

Para estas fiestas patrias, usted ha decidido ir a recorrer las fondas de su comuna. Este año, los locales ofrecen una gran variedad de terremotos. Con el fin de cuidar a la población de posibles borracheras, el gobierno ha obligado a todos los locales a publicar el nivel de alcohol presente en este trago típico. Usted ha decidido tomar tres terremotos diferentes (no repetirse ninguno). Además, por una restricción impuesta en casa, usted debe estar lo menos ebrio posible. El nivel de ebriedad se determina como el producto del nivel de alcohol de cada terremoto consumido. Su objetivo sera determinar qué combinación de tres terremotos obtiene el mínimo nivel de borrachera. Usted recibe una lista con el nivel de alcohol de todos los terremotos disponibles. Nota: Recuerde no utilizar fuerza bruta, o no podrá resolver de inputs muy grandes.

Entrada:

```
Lista de enteros: [4,-1,3,5,9]
```

Salida:

```
tupla de enteros: [-1,5,9]
```

3 **Doblelectura secreta (Recursión) - dificultad media** (1 pts)

En un antiguo imperio se inventó un antiguo sistema para codificar información importante. El sistema oculta palabras en única secuencia de letras. Las palabras que se manejan en este imperio siempre se leen igual, tanto de izquierda a derecha, como de derecha a izquierda. Por ejemplo "ORO" o "ANILINA". Su objetivo es, dada una secuencia de letras, determinar el número mínimo de cortes necesarios para que solo se formen palabras admitidas en este imperio.

Entrada:

```
Texto: 'PAPAPEPADED'
```

Salida:

```
entero: 2
```

4 **Vuelta a Clases (backtracking) - dificultad alta** (1,5 pts)

Se acabaron las fiestas patrias y te encuentras atrapado en una isla solo con un bote y unos remos. Dado que no te quieres perder la próxima clase de tu ramo favorito, Programación Como Herramienta para la Ingeniería, te quieres idear la forma de llegar lo más rápido posible a Santiago. En un principio no sabes exactamente cómo hacerlo, pero luego recuerdas haber soñado con el Dios Francisco. En tu sueño te ofreció un trato: si cumples ciertas reglas dentro de su archipiélago, él personalmente te llevará a la universidad. Sus palabras fueron:

- Te daré el nombre de la isla en que estás y el de la isla final a la que debes llegar.
- Te daré los nombres de todas las islas de este archipiélago.
- Todos los nombres de mis islas son distintos pero del mismo largo.

- Nunca te diré que mi isla inicial será igual a la isla final.
- Sólo permitiré que te muevas entre islas que tengan **una** letra de diferencia en su nombre.
- Sólo aceptaré llevarte a tu ciudad si es que llegas desde la isla inicial a la isla final parando en la menor cantidad de islas posibles.

A continuación se te dará un ejemplo de cómo debe funcionar tu programa. Recibirás el nombre de la isla inicial, el de la isla final y una lista con el resto de las islas del archipiélago. Deberás retornar una lista de listas en donde cada lista contenida tenga los nombres de las islas que se recorrieron para llegar de la isla inicial a la isla final (solo debes incluir los resultados con menor cantidad de islas). En caso de que no exista forma de llegar desde el inicio a la isla final se debe retornar una lista **vacía**. Es importante que sepas que la isla final debe estar contenida en la lista de islas entregada para poder llegar a ella, no así la isla inicial. Puedes asumir que los nombres entregados contendrán solo letras del alfabeto y que estarán en minúscula.

Entrada:

```
Lista de islas:
    ["mhu, "phu, "mai", "phe", "zai", "kae", "pai", "rho", "phi"]
Isla inicial: "mau"
Isla Final: "phi"
```

Salida:

```
Listas de listas:
    [
        ["mau", "mhu", "phu", "phi"]
        ["mau", "mai", "pai", "phi"]
    ]
```

5 **¿Quién llega más lejos? - dificultad alta** (1,5 pts)

Estás con tus amigos en un lugar lleno de bloques de concreto de distintas alturas y uno de ellos te desafía a competir por quién es el que puede hacer la secuencia más larga saltando entre un bloque y otro. Para hacerlo más complicado te dice que sólo se puede saltar hacia el sur y hacia el este, otras

direcciones estarán prohibidas en la competición. Además te dice que no se pueden utilizar las manos para saltar entre bloques por lo tanto tu sabes que **a lo más** podrías saltar a un bloque adyacente que tenga una altura hasta un metro superior o hasta un metro inferior desde el bloque en que estas parado.

Se te entrega como ejemplo una matriz en donde cada número representa la altura en metros del bloque correspondiente. El output es una lista de listas con las secuencias máximas[1] entre dos bloques (puede ser una o más). La secuencia está representada por un conjunto con las alturas de los bloques que se recorrieron.

Entrada:

```
Lista de listas:
[[8, 6, 3, 4, 2]
 [4, 5, 2, 5, 5]
 [2, 6, 7, 8, 9]
 [4, 5, 6, 9, 10]
 [4, 3, 3, 8, 7]]
```

Salida:

```
Lista de listas:
[[6, 5, 6, 7, 8, 9, 8, 7], [4, 5, 6, 7, 8, 9, 8, 7]]
```

Librería l03

Dado que en este laboratorio se le pide graficar, junto con el enunciado se ha suministrado de una librería que le ayudará en este objetivo. Para que esta funcione correctamente, debe instalar *matplotlib* en su entorno.

```
pip/pip3 install matplotlib
```

[1] Se define como secuencia máxima aquella que contiene más bloques

Luego, debe posicionar el archivo **l03.py** en la misma carpeta que su archivo de trabajo. En este último, debe escribir la siguiente línea de código.

```
import l03
```

La librería cuenta con el método `graficar` que recibe una lista de valores de la variable independiente (x), una lista de valores de la variable independiente (y), un "título", "nombre eje x" y "nombre eje y" (estos tres últimos como texto). Las listas x e y deben tener la misma cantidad de elementos. A continuación se muestra un ejemplo para graficar los puntos (1,5), (2,4), (3,7) y (4,9).

```
import l03
x = [1,2,3,4]
y = [5,4,7,9]
l03.graficar(x,y,'Ejemplo','valor 1','valor 2')
```

De esto resulta

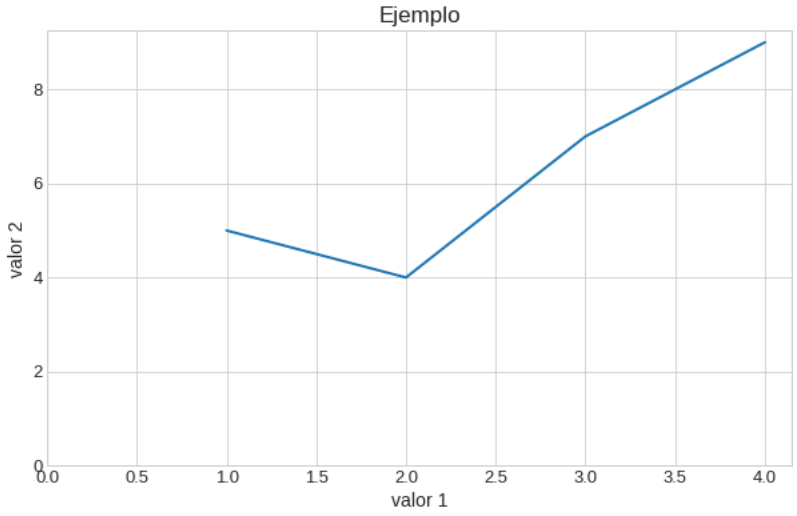

Figure 1: Ejemplo de gráfico

Corrección

Para la corrección de este laboratorio, se revisarán dos aspectos, cada uno con igual valor en la nota (50%). El primero serán los contenidos y mecanismos utilizados para resolver cada uno de los problemas propuestos. De este modo, es importante que comente correctamente su código y que sea más sencilla la

corrección. Recuerde que debe utilizar las estructuras de datos adecuadas a cada problema. Además, se evaluará el orden de su trabajo.

El segundo aspecto será la correctitud de los resultados y el tiempo de ejecución de las soluciones entregadas. Por cada problema se probarán distintos valores de entrada, y para cada uno de estos, el tiempo de respuesta no deberá ser superior a 1 segundo. Por lo tanto, para cada caso probado, si el resultado entregado por el algoritmo no es correcto, o si el tiempo de resolución supera 1 segundo, la respuesta será considerada como incorrecta.

El 22 de septiembre se entregarán referencias para el tamaño del problema y el tiempo de ejecución esperado.

Política de Integridad Académica

"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.