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

IIC2115 – Programación como Herramienta para la Ingeniería (II/2023)

Ejercicio Formativo 2 Capítulo 1

Aspectos generales

- **Objetivos:** Utilizar estructuras de datos para solucionar problemas de manera eficiente.
- **Lugar de entrega:** domingo 20 de agosto a las 23:59 hrs. en repositorio privado.
- **Formato de entrega:** archivo Python Notebook (**C1E2.ipynb**) con el avance logrado para el ejercicio. El archivo debe estar ubicado en la carpeta **C1**. Utilice múltiples celdas de texto y código para facilitar el trabajo del cuerpo docente.

Introducción

Con el fin de ejercitar el uso de estructuras de datos, en este ejercicio deberá resolver 3 problemas, donde deberá utilizar varias de estas para hacerlo manera eficiente. Qué estructura utilizar en cada situación será parte esencial del proceso de solución del ejercicio. Dado esto, para los primeros 2 problemas se indica una recomendación de la estructura a utilizar, mientras que para el último esta debe ser elegida por uds. Además de esto, será importante utilizar POO para modelar el problema.

Problemas

1. **Camino máximo (Listas):** considere dos listas de números naturales, que se encuentran ordenados de menor a mayor. Encuentre el camino de suma máxima compuesta por elementos de ambas listas, partiendo por el primer elemento de cualquiera de las dos, y sólo pudiendo cambiarse de lista cuando se utilice un número común entre ambas. Un ejemplo de ejecución del algoritmo es el siguiente:

   Código

```
def camino_maximo(lista1, lista2):
    #solucion al problema

lista1 =  [3,6,7,8,10,12,15,18,100]
lista2 =  [1,2,3,5,7,9,10,11,15,16,18,25,50]
s = camino_maximo(lista1, lista2)
print(s)
```

Salida

```
[1,2,3,6,7,9,10,12,15,16,18,100]
```

2. **Anagramas (Diccionarios):** dos strings son anagramas si al reordenar los caracteres de uno, es posible formar el otro. Teniendo esto en consideración, escribe un programa que dados dos strings, responda si estos son o no anagramas. Un ejemplo de ejecución del algoritmo es el siguiente:

Código

```
def son_anagramas(string1, string2):
    #solucion al problema

string1 = 'solucion'
string2 = 'oclusion'
s = son_anagramas(string1, string2)
print(s)
```

Salida

```
True
```

3. **Mansión embrujada:** por algún extraño motivo, ud. se encuentra atrapado en una mansión embrujada y lógicamente debe encontrar la salida. Lamentablemente, la única manera de escapar es viajar a través de las habitaciones de la mansión, buscando alguna que tenga una puerta de salida. Lamentablemente, la mansión tiene una cantidad de habitaciones que en la práctica es infinita y además, cada una de ellas tiene entre 1 y 5 puertas (contando la de entrada), que llevan a otras habitaciones de la mansión.

   En este escenario desolador, ud. afortunadamente descubre un libro en el suelo, donde se indica que cada habitación solo puede alcanzarse a través de una secuencia única de habitaciones. Además, este libro contiene un mapa parcial, que indica para solo algunas habitaciones, qué puerta se debe elegir.

   Además de lo anterior, su solución deberá cumplir las siguientes condiciones:

   - La mansión con sus habitaciones debe ser construida de manera automática, asignando aleatoriamente tanto la cantidad de puertas en cada habitación, como la ubicación de las salidas.
   - Cada habitación debe tener un identificador único, que no debe guardar relación con el lugar de la habitación en la mansión, ni con las habitaciones que la anteceden o siguen.
   - La cantidad de habitaciones debe ser exponencialmente mayor que la de salidas.
   - El mapa parcial solo podrá contener información para una cantidad de habitaciones que no debe exceder el logaritmo de la cantidad total de habitaciones.
   - Dado todo lo anterior, su objetivo es construir un programa que encuentre una salida de la mansión lo antes posible. No es necesario que esta salida sea la más cercana a la entrada.