FUNDAMENTOS DE COMPUTACIÓN

Grado en ingeniería en tecnologías industriales

Universidad de Cantabria

Examen extraordinario(laboratorio)

2 de septiembre del 2020

Nombre y apellidos:

---

1. [50 %] En computación un conjunto es una estructura de datos que representa los conjuntos finitos, que ya conocemos en matemáticas y los cuáles son una colección de diferentes valores sin ningún orden alguno ni valores duplicados. Una lista no es un conjunto porque es posible que un elemento aparezca repetidas veces. Podemos simular el tipo de datos `conjunto` con una simple lista, pero deberíamos tener presente muchos aspectos. Por ejemplo, precaución de no insertar un elemento si ya está presente, además las listas tienen orden. Los diccionarios no están ordenados, pero sus elementos están asociados a las claves. Para no complicar nuestros programas, es mejor definir un nuevo tipo de datos que, internamente, realice las comprobaciones pertinentes: la clase **Conjunto**.

```
class Conjunto(object):
    def __init__(self, lista=[]):
        self.elementos =lista
```

   1. [10 %] Definir el método `__eq__` para que devuelva el booleano True si los dos objetos son iguales o False, en caso contrario
   2. [5 %] Definir el método `__str__` para mostrar, apareciendo por pantalla con sus elementos separados por comas y encerrados entre llaves.
   3. [5 %] Implementar un método `inserta` que con dos argumentos: `self`(como siempre) y el elemento que queremos añadir, pero antes debemos comprobar que no está presente.
   4. [5 %] Implementar un método `borrar` que con dos argumentos: `self` y el elemento que queremos eliminar, obviamente solo habrá que borrar el elemento de la lista si pertenece.
   5. [5 %] Implementar el método `cardinal` para que devuelva el número de elementos del objeto conjunto.
   6. [20 %] Definir los métodos `union` e `interseccion` con dos argumentos, `self` y `otro` y devuelva el objeto unión e intersección.

Ejemplo de ejecución del programa

```
>>> A = Conjunto([2,1,3])
>>> B = Conjunto([1,3,2,1])
>>> C= Conjunto([7,3,6,1,3])
>>> A == B
True
>>> print(B)
{1,3,2}
>>> A.inserta(5)
>>> A.inserta(2)
>>>print(A)
{2,1,3,5}
>>> print(B.borrar(3))
{1,2}
>>> print(A.union(C))
{2,1,3,5,7,6}
>>> print(A.interseccion(C))
{1,3}
>>> C.cardinal()
4
```

2. [50 %] Elaborar un programa que lea una matriz de un archivo llamado *matriz.txt*, por ejemplo:

matriz.txt

```
1 0 0 2 -3 2
0 1 0 7 -5 1
3 0 0 1 -2 0
0 6 0 3 0 1
0 0 0 0 0 0
```

y realice las siguientes tareas:

1. [20 %] Comprobar si la matriz tiene alguna(s) fila(s) y/o columna(s) de ceros, indicando en su caso cual(es) son.
2. [15 %] Intercambiar los valores de dos filas (columnas) y mostrar la matriz resultante por pantalla. El usuario dirá cuáles son las filas (columnas) involucradas.
3. [15 %] Sea $M = (m_{i,k})$ una matriz de n filas, se pide cambiar los valores de una columna k usando una columna j, tal que cada elemento de la columna k sea igual a ese elemento mas dos veces el elemento correspondiente de la columna j, es decir:

$$m_{i,k} = m_{i,k} + 2m_{i,j}, \quad i = 1, \ldots, n$$

Implementar una función para mostrar la matriz. Esta función será invocada en el programa cada vez que se necesite. Implementar una función para comprobar si una fila (o columna) está llena de ceros en la matriz. La función recibe una fila (columna) y devuelve un booleano, según el caso.

El programa debe funcionar para cualquier tamaño (filas y columnas) y valores enteros de la matriz en el archivo de entrada "matriz.txt"

Ejemplo de ejecución del programa

```
1 0 0 2 -3 2
0 1 0 7 -5 1
3 0 0 1 -2 0
0 6 0 3 0 1
0 0 0 0 0 0
la fila 5 está llena de ceros
la columna 3 está llena de ceros
Intercambiar dos filas
Dame filas a intercambiar 1,2
0 1 0 7 -5 1
1 0 0 2 -3 2
3 0 0 1 -2 0
0 6 0 3 0 1
0 0 0 0 0 0
Intercambiar dos columnas
Dame columnas a intercambiar 1,2
1 0 0 7 -5 1
0 1 0 2 -3 2
0 3 0 1 -2 0
6 0 0 3 0 1
0 0 0 0 0 0
Dame columnas en juego 3,4
1 0 14 7 -5 1
0 1 4 2 -3 2
0 3 2 1 -2 0
6 0 6 3 0 1
0 0 0 0 0 0
```