```python
# -*- coding: utf-8 -*-
"""
Created on Sat Oct 16 12:52:06 2021

@author: itzam
"""
import numpy as np
class tableau:

    def __init__(self,A,b,c):
        self.n = np.shape(A)[1]  #Número de variables
        #El indexado comienza en cero así que las variables serán
        #x_0, x_1,..., x_{n-1}

        self.p = np.shape(A)[0]  #Número de restricciones
        self.A = np.zeros((self.p,self.n+self.p))  #Añadimos variables de holgura
        #x_n,...,x_{n+p-1}

        self.A[:,:self.n]=A
        self.A[:,self.n:]=np.eye(self.p)
        self.b = b
        self.c = np.zeros((self.p+self.n,1))
        self.c[:self.n]=c
        self.c[self.n:]=0



        #Definición de la base
        #Al inicio la base esta conformada por las variables de holgura
        #Es un vector de dimensión p donde la k-ésima entrada contiene
        #el número de la k-ésima variable en la base
        #Ejemplo: si self.base = (1,3,5), las variables en la base son
        #x_1, x_3, x_5
        self.base=np.arange(self.n,self.n+self.p)

        #Definimos las no básicas
        #Serán de las n+p variables, las que no estén en la base
        self.nobase = np.delete(np.arange(0,self.n+self.p),self.base)

        #Por último definimos una solucion provisional
        self.solucion = np.zeros(self.n+self.p)


        self.ahorros = np.zeros(self.n+self.p)

    def Simplex(self):
        itera = 0

        while(True):

            print(itera)
            print(self.base)

            A_b = self.A[:,self.base]
            A_n = self.A[:,self.nobase]
            c_b=self.c[self.base]
            lambda_simplex = np.linalg.solve(A_b.T,c_b)
```

```
ahorros_nb = lambda_simplex.T@A_n-self.c[self.nobase].T

if((ahorros_nb<=0).all()):
    #En este caso hemos terminado y ya encontramos la solucion
    z_0 = lambda_simplex.T@self.b

    self.solucion[self.base]=np.linalg.inv(A_b)@self.b
    return (self.solucion,z_0,0,itera)


#En este punto, existe al menos un ahorro positivo
self.ahorros[self.nobase]=ahorros_nb
self.ahorros[self.base]=0


#Obtenemos el índice de entrada
#Esta línea nos da el índice de la primer variable con ahorro
#positivo (Bland)
#Por ejemplo probar:
#a = array([ 0, -1,  1,  0,  5])
#np.where(a==5)[0][0]
e = np.where(self.ahorros>0)[0][0]

H_e = np.linalg.inv(A_b)@self.A[:,e]
if((H_e<=0).all()):
    #El problema es no acotado

    return (self.solucion,0,1,itera)


#Ahora obtengamos la variable de salida
#s_tilde es el número de variable en la base que saldra
#entonces s = base[s_tilde] tiene el índice de la variable que sale
h = np.linalg.inv(A_b)@self.b
#print(h)

#Cuidado en esta división,
#Donde las entradas de la columna H_e son
#negativas, h hago que tome el valor -infinito
#y cuando -inf/H_e=inf mayor que cualquier numero


h_temp = np.where(H_e<=0,-np.inf,h.T)
s_tilde = np.argmin(h_temp/H_e)

#s = self.base[s_tilde]
#Sale x_s y entra x_e
self.base[s_tilde]=e

#la nueva base difiere en la anterior por una variable
#ahora no quitaremos s, pero sí e
self.nobase = np.delete(np.arange(0,self.n+self.p),self.base)

itera+=1
```

```
A = np.array([[ 1,  1,  2],
       [ 1,  1, -1],
       [-1,  1,  1]])

b = np.array([[9],
       [2],
       [4]])

c = np.array([[ 1],
       [ 1],
       [-4]])

tabla1 = tableau(A,b,c)
tabla1.Simplex()
```