47-PROGR-divpor7

November 4, 2017

Definición de la función
Comenzamos definiendo la función F que vamos a iterar:

```
In [1]: def F(n):
            if n==0:
                return 0
            else:
                L = (n).digits(base=10)
                n0 = L[0]
                return (((n-n0)//10)-2*n0)

In [2]: F(49)

Out[2]: -14

In [3]: F(-14)

Out[3]: 7
```

Una definición alternativa podría ser:

```
In [4]: def F1(n):
            if n==0:
                return 0
            else:
                y = n%10
                x = n//10
                return x-2*y
```

El problema con esta segunda definición de la función viene de su comportamiento cuando n es negativo:

```
In [5]: F1(-14)

Out[5]: -14
```

Vemos que F y $F1$ no son la misma función, al menos para los negativos. £Son la misma para n positivo?

```
In [6]: all([F(n)==F1(n) for n in srange(1000)])

Out[6]: True
```

No es difícil ver cómo aparece ese comportamiento diferente:

```
In [7]: L = (-14).digits(base=10)
        print L
        x = (-14-L[0])//10
        print x
        print x-2*L[0]

[-4, -1]
-1
7


In [8]: print (-14)//10
        print (-14)%10
        print ((-14)//10)-2*((-14)%10)

-2
6
-14
```

Vemos que el problema es que $-14//10$ es -2 y queremos que sea -1. Al calcular la órbita de la función $F1$ si llegamos a -14 se queda ya para siempre en -14 y si no se incluye $F(n) = n$ como condición de parada del bucle *while* se produce un bucle infinito. Además, como estamos añadiendo los sucesivos valores de $F(n)$ a la lista en la que acumulamos la órbita, esa lista crece sin límite y satura la memoria RAM, es decir, no sólo el proceso no para sino que cuelga la máquina.

Órbita

£Cuál debe ser la condición de parada para un while? Estudiamos algunos trozos de órbitas, tomando un número de iteraciones N suficientemente grande , pero no demasiado, es decir, tanteamos cuál puede ser un buen valor dependiendo de los valores *ini* que usamos:

```
In [9]: def orbita(ini,N,f):
            L = [ini]
            for _ in srange(N):
                ini = f(ini)
                L.append(ini)
            return L

In [10]: for item in [orbita(J,20,F) for J in srange(1,20)]:
             print(item)

[1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4]
[2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8]
[3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12]
```

```
[4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16]
[5, -10, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1]
[6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3]
[7, -14, 7, -14, 7, -14, 7, -14, 7, -14, 7, -14, 7, -14, 7, -14, 7, -14, 7, -14, 7]
[8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11]
[9, -18, 15, -9, 18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15]
[10, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2]
[11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2]
[12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6]
[13, -5, 10, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11]
[14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14]
[15, -9, 18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15, -9, 18]
[16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1]
[17, -13, 5, -10, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16]
[18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15, -9, 18, -15, 9]
[19, -17, 13, -5, 10, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8]
```

Parece claro que al iterar F se obtienen enteros pequeños que parecen entrar en ciclos. Hagamos otra prueba con enteros iniciales mayores:

```
In [11]: for item in [orbita(J*1000+randint(-1000,1000),20,F) for J in srange(1,20)]:
             print(item)
```

```
[1841, 182, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14]
[1056, 93, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3]
[3868, 370, 37, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16]
[3880, 388, 22, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1]
[4917, 477, 33, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12]
[5595, 549, 36, -9, 18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15]
[6312, 627, 48, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6]
[7418, 725, 62, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1]
[9795, 969, 78, -9, 18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15]
[9168, 900, 90, 9, -18, 15, -9, 18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15, -9, 18, -15]
[10526, 1040, 104, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1]
[11509, 1132, 109, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4]
[13837, 1369, 118, -5, 10, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8]
[14220, 1422, 138, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12]
[14113, 1405, 130, 13, -5, 10, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4]
[15131, 1511, 149, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2]
[17445, 1734, 165, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3]
[17580, 1758, 159, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12]
[19440, 1944, 186, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3]
```

Parece que en cuanto una iteración tiene un único dígito entra en un ciclo. Lo comprobamos:

```
In [12]: for item in [orbita(J,20,F) for J in srange(-9,10)]:
             print(item)
```

```
[-9, 18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15, -9, 18, -15]
[-8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11]
[-7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7]
[-6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3]
[-5, 10, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1]
[-4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16]
[-3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12]
[-2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8]
[-1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4]
[2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8]
[3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12]
[4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16, -11, 1, -2, 4, -8, 16]
[5, -10, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1]
[6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3, -6, 12, -3, 6, -12, 3]
[7, -14, 7, -14, 7, -14, 7, -14, 7, -14, 7, -14, 7, -14, 7, -14, 7, -14, 7, -14, 7]
[8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11, -1, 2, -4, 8, -16, 11]
[9, -18, 15, -9, 18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15, -9, 18, -15, 9, -18, 15]
```

£Cuántos de estos ciclos hay?

Parece que una condición de parada razonable puede ser "parar cuando se cae en el intervalo $[-9, 9]$". Si no fuera correcta se entraría en bucles infinitos debidos al while.

```
In [13]: def orbita1(ini,f):
             L = []
             while not (-10<ini<10):
                 L.append(ini)
                 #print L
                 ini = f(ini)
             L.append(ini)
             return L[-1]
```

En esta función no nos interesa toda la órbita completa, sino únicamente el ciclo final. £Es razonable (eficiente) programarla así?

Comprobamos primero que la condición de parada del bucle *while* es correcta, es decir, que no se entra en bucles infinitos:

```
In [14]: len([orbita1(n,F) for n in xsrange(100)])

Out[14]: 100

In [15]: len([orbita1(n,F) for n in xsrange(1000)])

Out[15]: 1000

In [16]: len([orbita1(n,F) for n in xsrange(10000)])
```

```
Out[16]: 10000

In [17]: time len([orbita1(n,F) for n in xsrange(1000000)])

CPU times: user 14.5 s, sys: 252 ms, total: 14.8 s
Wall time: 14.6 s


Out[17]: 1000000

In [18]: print [(n,orbita1(n,F)) for n in xsrange(100)]

[(0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 1), (11,
```

Observamos que sólo los múltiplos de 7 todos terminan pasando por $-7, 0$ o 7. Comprobamos si esto es cierto:

```
In [19]: def orbita2(ini,f):
             while not((ini == -7)or(ini == 0)or(ini==7)):
                ini = f(ini)
             return ini

In [20]: OO = [orbita2(n,F) for n in xsrange(10,10000) if n%7 == 0] ##Definimos una variable ig
```

Como no ha entrado en un bucle infinito, vemos que para todos los múltiplos de 7 en el intervalo $[10, 9999]$ la órbita termina en $-7, 0$ o 7. £Cómo vemos que ningún primo con 7 tiene esa misma propiedad?

```
In [21]: orbita(7,10,F)

Out[21]: [7, -14, 7, -14, 7, -14, 7, -14, 7, -14, 7]

In [22]: orbita(-7,10,F)

Out[22]: [-7, 14, -7, 14, -7, 14, -7, 14, -7, 14, -7]

In [23]: LL = [orbita1(n,F) for n in xsrange(10,10000) if n%7 != 0]

In [24]: 7 in LL;-7 in LL;0 in LL

Out[24]: False
```

Esto comprueba que los NO múltiplos de 7 tienen órbitas que intersecan el intervalo $[-9, 9]$ en enteros diferentes de $-7, 0, 7$.

De hecho, el primer apartado del ejercicio, tal como se enuncia en las notas, pedía que se probara que "N es múltiplo de 7 si y sólo si $F(N)$ también lo es".

Si se ha resuelto este primer apartado, el si y sólo si nos garantiza que un primo con 7 no puede tener en su órbita al -7, ni al 0, ni al 7. Parece claro entonces, que el criterio de divisibilidad por 7 que estamos buscando debe ser:

"Un entero n es divisible entre 7 si y sólo si la órbita de n, mediante la iteración de F, interseca al conjunto $\{-7, 0, 7\}$."

Supuesto que se ha demostrado el primer apartado, para terminar la demostración habrá que ver que todas las órbitas intersecan al intervalo $[-9, 9]$, afirmación que hemos comprobado experimentalmente usando la función $orbita1(n)$.

Los dos ejercicios se pueden ver resueltos en el archivo "multiplos7.pdf" en la carpeta "PDFs/PROGR/".

```
In [25]: print [orbita1(n,F) for n in srange(-20,201)]
```

```
[-2, 5, -9, -5, -1, 9, 7, 5, 3, 1, -1, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6
```

```
In [ ]:
```