51-COMPL-trucos

December 3, 2017

Potencias

```
In [1]: def potencia(a,k):
            if k==0:
                return 1
            elif k %2 == 0:
                b = potencia(a,k/2)
                return (b*b)
            else:
                b = potencia(a,(k-1)/2)
                return (a*b*b)

In [2]: %time p=potencia(77,2^30)

CPU times: user 44.8 s, sys: 2.1 s, total: 46.9 s
Wall time: 46.9 s


In [3]: def potencia_mod(a,k,m):
            if k==0:
                return 1
            elif k%2 == 0:
                b = potencia_mod(a,k/2,m)
                return (b*b)%m
            else:
                b = potencia_mod(a,(k-1)/2,m)
                return (a*b*b)%m

In [4]: %time pm = potencia_mod(7777^1234,2^157,10991^987654+1)

CPU times: user 1min 1s, sys: 1.28 s, total: 1min 3s
Wall time: 1min 3s
```

Listas binarias

```
In [7]: def polinomios(k):
            L = []
            Lsal = []
```

```
            for m in srange(2^k):
                L.append(m.digits(base=2,padto=k))
            for L1 in L:
                Lsal.append(sum([L1[int]*x^int for int in srange(k)]))
            return Lsal
```

```
In [8]: print polinomios(3)
```

```
[0, 1, x, x + 1, x^2, x^2 + 1, x^2 + x, x^2 + x + 1]
```

```
In [9]: def polinomios(k,j):
            L = []
            Lsal = []
            for m in srange(j^k):
                L.append(m.digits(base=j,padto=k))
            for L1 in L:
                Lsal.append(sum([L1[int]*x^int for int in srange(k)]))
            return Lsal
```

```
In [10]: print polinomios(3,3)
```

```
[0, 1, 2, x, x + 1, x + 2, 2*x, 2*x + 1, 2*x + 2, x^2, x^2 + 1, x^2 + 2, x^2 + x, x^2 + x + 1,
```

Archivos binarios

```
In [11]: %time L=[randint(0,1) for _ in srange(1024*1024)] #Generamos una lista de 1048576 bits
```

```
CPU times: user 3.48 s, sys: 88 ms, total: 3.57 s
Wall time: 3.52 s
```

```
In [12]: from string import *

         def archivo(L):
             C = ''
             while L != []:
                 C += (chr(int(join(map(str,L[:8]),sep=''),base=2)))
                 L = L[8:]
             outfile = open("prueba-bits","wb") #Abrimos el archivo para escribir en el
             outfile.write(C)                   #Escribimos la cadena C al archivo
             outfile.close()                    #Cerramos el archivo
```

Comentarios:

El archivo en el que se va a escribir tiene nombre prueba-bits y está en el directorio desde el que hemos arrancado Sage.

La línea importante es [C += (chr(int(join(map(str,L[:8]),sep='),base=2))).] Analizemos su contenido:

$L[:8]$ corta los primeros ocho bits de L.

$map(str, L[:8]), sep = '')$ convierte la lista de ceros o unos en una lista igual pero con los números transformados en caracteres (en lugar de 0 pone $'0'$).

$join(map(str, L[:8]), sep = '')$ transforma la lista de caracteres individuales en una cadena de caracteres.

$int(join(map(str, L[:8]), sep = ''), base = 2)$ produce el entero decimal que corresponde a la cadena de ceros o unos.

Finalmente, $chr(< entero >)$ convierte un entero en el caracter que ocupa en ASCII ese número de orden.

```
In [13]: %time archivo(L)

CPU times: user 3min 9s, sys: 304 ms, total: 3min 10s
Wall time: 3min 9s
```

```
In [14]: def leer_archivo():
             L = []
             infile = open("prueba-bits","rb") #Abrimos el archivo /tmp/prueba-bits para lectu
             LL = infile.readlines()                #Leemos cada linea como uno de los element
             infile.close()
             for C in LL:
                 while C != '':
                     L1 = ZZ(ord(C[0])).digits(base=2,padto=8)
                     L1.reverse()
                     L += L1
                     C = C[1:]
             return L
```

Comentarios:

$ord(C[0])$ nos devuelve el n'umero entero que corresponde al primer caracter de la cadena C.

$ZZ(ord(C[0]))$ lo convierte en un entero de SAGE.

$ZZ(ord(C[0])).digits(base = 2, padto = 8)$ nos devuelve la lista de sus dígitos binarios, completada con ceros hasta que haya 8 ceros o unos.

La lista del apartado anterior hay que invertirla porque los dígitos están escritos de izquierda a derecha ($[unidades, decenas, ...]$).

```
In [15]: L1 = leer_archivo()

In [16]: len(L);len(L1);L == L1 # Las dos listas coinciden

Out[16]: True

In [ ]:
```