87-CRIPT-mhellman

March 5, 2018

Ejercicios 2 y 3

```
In [1]: def is_superincreasing(L):
            if len(L) == 1 and L[0] <= 0:
                return False
            elif len(L) == 1:
                return True
            elif L[-1:][0] <= sum(L[:-1]):
                    return False
            is_superincreasing(L[:-1])
            return True
```

```
In [2]: is_superincreasing([1,2,2])
```

```
Out[2]: False
```

```
In [3]: is_superincreasing([1,2,5])
```

```
Out[3]: True
```

```
In [4]: def superincreasing(n,N):
            L = [randint(1,N)]
            for cont in srange(n-1):
                L.append(randint(sum(L)+1,sum(L)+N))
            return L
```

```
In [5]: is_superincreasing(superincreasing(5,3))
```

```
Out[5]: True
```

```
In [6]: all([is_superincreasing(superincreasing(7,12)) for int in srange(100)])
```

```
Out[6]: True
```

Ejercicio 5

```
In [7]: def listas (K):
            L = []
            for k in srange(2^K):
                L.append(k.digits(base=2,padto=K))
            return L
```

```
In [8]: def resolver_fb(L,A):
            K = len(L)
            LL = listas(K)
            for lista in LL:
                if sum([L[i]*lista[i] for i in srange(K)])==A:
                    return lista
                else:
                    continue

In [9]: L = superincreasing(10,12)
        print L
        %time resolver_fb(L,sum(L)-L[5]-L[3])

[11, 23, 40, 85, 170, 339, 671, 1345, 2693, 5381]
CPU times: user 68 ms, sys: 8 ms, total: 76 ms
Wall time: 64.2 ms


Out[9]: [1, 1, 1, 0, 1, 0, 1, 1, 1, 1]

In [10]: L = superincreasing(15,12)
         %time resolver_fb(L,sum(L)-L[5]-L[3])

CPU times: user 1.94 s, sys: 128 ms, total: 2.07 s
Wall time: 1.91 s


Out[10]: [1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]

In [11]: L = superincreasing(20,12)
         # time resolver_fb(L,sum(L)-L[5]-L[3])
```

Ejercicio 6

```
In [12]: def resolver_r(L,A):
             SOL = []
             if len(L) == 0:
                 return SOL
             elif A > sum(L):
                 return []
             else:
                 if A >= L[-1]:
                     L1 = resolver_r(L[:-1],A-L[-1])
                     SOL= L1+[1]
                 else:
                     L1 = resolver_r(L[:-1],A)
                     SOL=L1+[0]
             return SOL
```

```
In [13]: L = superincreasing(7,12)
         print L
         %time resolver_r(L,sum(L)-L[5]-L[3])

[11, 12, 27, 62, 121, 240, 481]
CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 51 ţs


Out[13]: [1, 1, 1, 0, 1, 0, 1]

In [14]: L = superincreasing(7,12)
         print L
         %time resolver_r(L,sum(L)-L[5]-L[3])

[6, 14, 23, 47, 91, 182, 375]
CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 50.1 ţs


Out[14]: [1, 1, 1, 0, 1, 0, 1]

In [15]: L = superincreasing(15,12)
         %time resolver_r(L,sum(L)-L[5]-L[3])

CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 84.2 ţs


Out[15]: [1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

Merkle-Hellman

```
In [16]: def generar_clave_privada(N):
             L = superincreasing(N,10^3)
             m = randint(2*L[-1],2*L[-1]+10^4)
             w = next_prime(m^3)
             return L,m,w

In [17]: def generar_clave_publica(t):
             L = []
             for item in t[0]:
                 L.append(t[2]*item%t[1])
             return L

In [18]: def generar_claves(N):
             Cpr = generar_clave_privada(N)
             return Cpr,generar_clave_publica(Cpr)

In [19]: CL = generar_claves(10)
```

```
In [20]: alfb = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

In [21]: L_alfb = list(alfb)

In [22]: texto = "Through the use of abstraction and logical reasoning, mathematics developed

In [23]: def ord2(c):
             return L_alfb.index(c)

In [24]: def chr2(n):
             return L_alfb[n]

In [25]: print map(ord2,[x for x in alfb])

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]


In [26]: (7).digits(base=2,padto=5)

Out[26]: [1, 1, 1, 0, 0]

In [27]: def cadena(L):
             C = ''
             L.reverse()
             for item in L:
                 C = C+str(item)
             return C

In [28]: def bin_2(n):
             return cadena(ZZ(n).digits(base=2,padto=5))

In [29]: bin_2(7)

Out[29]: '00111'

In [30]: print map(chr2,map(ord2,[x for x in alfb]))

['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S'


In [31]: from string import *
         def limpiar(texto,alfb):
             L = map(ord,[x.capitalize() for x in list(texto)])
             L1 = [item for item in L if item in map(ord,[x for x in alfb])]
             C1 = join(map(chr,L1),sep = "")
             return C1

In [32]: texto2 = limpiar(texto,alfb);texto2

Out[32]: 'THROUGHTHEUSEOFABSTRACTIONANDLOGICALREASONINGMATHEMATICSDEVELOPEDFROMCOUNTINGCALCULA
```

```
In [33]: texto_cod0 = map(ord2,list(texto2));print texto_cod0

[19, 7, 17, 14, 20, 6, 7, 19, 7, 4, 20, 18, 4, 14, 5, 0, 1, 18, 19, 17, 0, 2, 19, 8, 14, 13, 0

In [34]: texto_cod1 = map(bin_2,texto_cod0);print texto_cod1

['10011', '00111', '10001', '01110', '10100', '00110', '00111', '10011', '00111', '00100', '10

In [35]: texto_cod3 = join(texto_cod1,sep='');texto_cod3

Out[35]: '1001100111100010111010100001100011110011001110010010100100100010001110001010000000001

In [36]: CL

Out[36]: (([701, 1473, 3154, 6210, 12291, 24437, 48611, 97432, 194638, 389398],
           785128,
           483973293986417189),
          [25937,
           54501,
           116698,
           229770,
           454767,
           119041,
           228351,
           464472,
           135454,
           275422])

In [37]: def encriptar(C,Cpu):
             EN = []
             r = len(C)%10
             C1 = C + '0'*(10-r)
             while len(C1) > 0:
                 C2 = C1[:10]
                 sum = 0
                 for i in srange(10):
                     sum += ZZ(C2[i])*Cpu[i]
                 EN.append(sum)
                 C1 = C1[10:]
             return EN

In [38]: CL[1]

Out[38]: [25937, 54501, 116698, 229770, 454767, 119041, 228351, 464472, 135454, 275422]

In [39]: texto_encr = encriptar(texto_cod3,CL[1]);print texto_encr

[1585822, 1308981, 742561, 1331152, 1265707, 397130, 944975, 571465, 709262, 1104937, 135454,
```

```
In [40]: CL[0]

Out[40]: ([701, 1473, 3154, 6210, 12291, 24437, 48611, 97432, 194638, 389398],
          785128,
          483973293986417189)

In [41]: def cadena(L):
             C = ''
             for item in L:
                 C = C+str(item)
             return C

In [42]: cadena([1, 0, 0, 1, 1, 1, 1, 1, 1, 0])

Out[42]: '1001111110'

In [43]: def desencriptar(texto_encr,Cpr):
             R = Integers(Cpr[1])
             w1 = R(1/Cpr[2])
             #print w1
             C = ''
             for item in texto_encr:
                 C += join(cadena(resolver_r(Cpr[0],R(w1*item))),sep='')
             return C

In [44]: texto_cod3 in desencriptar(texto_encr,CL[0])

Out[44]: True

In [45]: desencriptar(texto_encr,CL[0])[:-5]==texto_cod3

Out[45]: True

In [46]: ZZ('10011',base=2)

Out[46]: 19

In [47]: def descod(C):
             C1 = ''
             while C != '':
                 C1 = C1+chr2(ZZ(C[:5],base=2))
                 C = C[5:]
             return C1

In [48]: descod(desencriptar(texto_encr,CL[0])[:-5])

Out[48]: 'THROUGHTHEUSEOFABSTRACTIONANDLOGICALREASONINGMATHEMATICSDEVELOPEDFROMCOUNTINGCALCULA
```