nombre.apellido-labot-2015-ex4-SOL

April 23, 2018

CALIFICACIÓN:

Por favor, empieza cambiando *nombre.apellido* en el nombre de esta hoja por los tuyos, tal como aparecen en tu dirección de correo eletrónico de la UAM, y deja el resto del nombre como está.

Debes intentar programar tus funciones de forma que no saturen la RAM del ordenador con valores razonablemente grandes de sus parámetros. Si con los valores que se indican en los enunciados no te funciona puedes reducirlos.

Recuerda que debes entregar esta hoja de SAGE, con tus respuestas, en la carpeta $ENTREGA\_examen\_nombre.apellido - labot - ex4$ que está en el escritorio de la cuenta que utilizas para el examen. Para guardar la hoja, una vez terminado el examen, utiliza el menú "FILE>Save worksheet to a File...."

Ejercicio 1

(3 puntos) Calcula, experimentalmente, la probabilidad de que al lanzar dos dados se obtenga una suma de puntos de al menos 10 y además, lanzado un tercer dado, la suma total de los puntos alcance al menos 15 puntos. No es difícil resolver este ejercicio de manera exacta, y puedes hacerlo (es opcional) como comprobación escribiendo la respuesta en el reverso de la hoja con la contraseña.

```
In [1]: def prob(N):
            cont = 0
            for muda in xsrange(N):
                x = randint(1,6)
                y = randint(1,6)
                if x+y >= 10:
                    z = randint(1,6)
                    if x+y+z >= 15:
                        cont += 1
            return (cont/N).n()

In [2]: prob(10^5)

Out[2]: 0.0741300000000000
```

Ejercicio 2

(6 puntos) En algunos sistemas criptográficos, en particular en RSA, utilizamos como clave pública el producto de dos primos muy grandes. Decimos que un entero es semiprimo si su factorización consiste en dos primos, diferentes o iguales (es decir, son enteros de la forma $n = p.q$ o bien $n = p^2$ con p y q primos).

Define una función $semip1(R)$ que cuente el número de semiprimos en el intervalo $[1, R]$. Podemos decir que la densidad de los semiprimos en el intervalo $[1, R]$ es $semip1(R)/R$, que también podemos interpretar como la probabilidad de que un entero elegido al azar en el intervalo $[1, R]$ sea semiprimo.

Calcula la densidad y el tiempo de cálculo de la función anterior con $R = 10^4, 10^5, 10^6, 10^7$. En este último caso el cálculo puede durar del orden de 200 segundos, y si tarda mucho más puedes reducir el valor de R , por ejemplo a la mitad . £Cómo depende el tiempo de R?

Los tiempos obtenidos en el apartado anterior nos parecen demasiado altos y decidimos probar a calcular la densidad eligiendo al azar un número N de enteros en el intervalo $[1, R]$ y contando cuántos son semiprimos. Para los valores de R del apartado anterior, calcula tiempos y densidades mediante una función $semip2(R, N)$ que debes definir, y estudia cómo conseguir una buena aproximación a la densidad sin hacer N demasiado grande. Es claro que si N está próximo a R las dos funciones hacen casi lo mismo y tardarán tiempos similares, y lo que queremos es, por ejemplo, calcular dos cifras decimales correctas de la densidad con el "mínimo" N, lo que va a hacer que el tiempo sea también "mínimo". No se trata de calcular con toda precisión un N mínimo (no tendría mucho sentido porque cada vez que ejecutamos la función obtenemos resultados diferentes) sino de obtener una idea correcta del orden de magnitud del N mínimo, en función de R, con el que conseguimos la precisión deseada.

```
In [3]: def semip1(R):
            cont = 0
            for int in xsrange(1,R):
                L = list(factor(int))
                if (len(L)==2 and L[0][1]==1 and L[1][1]==1) or (len(L)==1 and L[0][1]==2):
                    cont += 1
            return cont

In [4]: time semip1(10^4)

Out[4]: 2625
        Time: CPU 0.19 s, Wall: 0.31 s

In [5]: time semip1(10^5)

Out[5]: 23378
        Time: CPU 1.94 s, Wall: 1.95 s

In [6]: time semip1(10^6)

Out[6]: 210035
        Time: CPU 19.84 s, Wall: 19.84 s

In [7]: time semip1(10^7)

Out[7]: 1904324
        Time: CPU 203.76 s, Wall: 203.83 s
```

Se observa que al multiplicar por 10 el trabajo se multiplica por 10 el tiempo. Esto parece querer decir que con estos valores de R, no muy altos, la factorización de enteros mucho mçás grandes no tarda más y el tiempo se multiplica por diez porque hay que tratar 10 veces más enteros. La relación entre el tiempo y R parece ser lineal en este rango de valores ($R \leq 10^7$).