CC3101 — Matemáticas Discretas

Tarea 1

Nombre: Gabriel Iturra Bocaz
Fecha de entrega: 1 de mayo de 2017

1. En el reino de Nlogonia se quiere eliminar todos los billetes exepto los de 3 y de 5 NlogDolares. Un grupo de mercaderes está nervioso de que exisitan valores que no se pueden formar solo con esos billetes. Decimos que un valor n se puede formar con esos billetes si existen a, b positivos o cero tales que $n = 3a + 5b$

   a) Para evitar este problema, el rey quiere prohibir vender productos a precios que no se puedan formar con esos billetes. Encuentre cual es el valor n más grande que no se puede formar con estos billetes para ayudar al rey a pasar su ley.

   b) Los mercaderes siguen intranquilos. Exijen una demostración rigurosa de que ese es el n más pequeño. Pruebe usando inducción fuerte que todo número mayor que n puede formarse con esos billetes.

   c) Una vez convencidos de que funciona, a los mercaderes de Nlogonia les encantó el nuevo sistema y se entretienen durante horas haciendo secuencias con sus nuevos billetes. Encuentre un método basado en la inducción fuerte para calcular cuantas secuencias distintas se pueden hacer que valgan exactamente d Nlogdólares. (Estas secuencias son sensibles al orden, por ejemplo "$3, 5$" es distinta de "$5, 3$")

   d) Escriba un programa, basado en la inducción fuerte, que reciba un número d y entregue la cantidad de secuencias de billetes distintos que valen exactamente d o entregue 0 en caso de que no haya ninguna secuencia posible que valga en total d.

   Respuestas:

   a) Como cada valor n se puede formar, si existen a, b constantes que pertenecen a $\{0\} \cup \mathbb{N}$, tal que $n = 3a + 5b$, considerando esto, existen algunos números (pertencientes al conjuntos de los naturales, que no es posible formar con esta suma), para verlo de mejor formar consideraremos algunos casos importantes (aunque no lo dice explicitamente, asumimos que $n \in (\{0\} \cup \mathbb{N})$):

   - Cuando $n = 0$ es posible formar es número, cuando tanto a y b, con iguales 0, es decir, $n = 3 \cdot 0 + 5 \cdot 0 = 0$.
   - Cuando $n = 1$, dicho número no es posible formarlo mediante ninguna combinación de valores para a y b, en la formula del enunciado.
   - Cuando $n = 2$, sucede lo mismo para el caso anterior.
   - Cuando $n = 3$, es posible formarlo cuando $a = 1$ y $b = 0$, es decir, $n = 3 \cdot 1 + 5 \cdot 0 = 3$.
   - Cuando $n = 4$, no es posible formarlo mediante ninguna combinación de valores para a y b.
   - Cuando $n = 5$, sucede lo mismo para el caso cuando $n = 3$, pero con los roles invertidos para a y b, es decir, $n = 3 \cdot 0 + 5 \cdot 1 = 5$.
   - Cuando $n = 6$, se puede formar cuando $a = 2$ y $b = 0$, es decir, $n = 3 \cdot 2 + 5 \cdot 0 = 6$.
   - Cuando $n = 7$ dicho número no es posible formarlo por ninguna combinación de valores para a y b.
   - Cuando $n = 8$, se forma cuando a y b valen 1, es decir, $n = 3 \cdot 1 + 5 \cdot 1 = 8$.

- Luego, el resto de los casos para $n > 7$ es posible formarlo para cualquier combinación de valores de a y b. Con esto concluimos que el n más grande que que no se puede formar con billetes especificados por el rey, cuando $n = 7$.

b) Para tranquilizar a los mercaderes hemos propuesto una demostración más rigurosa, basandonos en el método de inducción fuerte sobre todos n mayores a 8 en los naturales. Para ello, tomanos la propiedad que cualquier $n \geq 8$, es posible formarlo usando solo billetes de 3 y 5 NlogDolares ponderados por las constantes a y b. Para comenzar la demostración consideraremos 3 casos bases, cuando n toma los valores 8, 9, 10.

- Casos base:
  1) Sí $a = 1$, y $b = 1$, tenemos que $3 \cdot 1 + 5 \cdot 1 = 8$.
  2) Sí $a = 3$, y $b = 0$, tenemos que $3 \cdot 3 + 5 \cdot 0 = 9$.
  3) Sí $a = 0$, y $b = 2$, tenemos que $3 \cdot 0 + 5 \cdot 2 = 10$.
- Hipótesis Inductiva:
  Cualquier valor i, donde i se encuentra entre $8 \leqslant i \leqslant n$, y $n \geq 10$, puede ser expresado como una combinación de billetes $i = 3 \cdot a + 5 \cdot b$ con a y b como constantes enteras no negativas.
- Caso inductivo:
  Mostraremos $n+1$ tambíen puede ser escrito como una combinación tal que, $n+1 = 3 \cdot a + 5 \cdot b$, asumiendo que esto se cumple para los casos desde 8 hasta n.
  Considerando esto, podemos asumir (por H.I.), que la propiedad se cumple para $n - 2$ tal que:

$$n - 2 = 3 \cdot a + 5 \cdot b$$
$$n - 2 + 3 = 3 \cdot a + 5 \cdot b + 3$$
$$n + 1 = 3 \cdot a + 5 \cdot b + 3$$
$$n + 1 = 3 \cdot a + 3 + 5 \cdot b$$
$$n + 1 = 3 \cdot (a + 1) + 5 \cdot b$$

  Donde $a + 1$ y b, siguen siendo constantes enteras no negativas, y asumiendo la propiedad para $n - 2$, y agregando 3 NlogDolares podemos probar la propiedad para el caso $n + 1$.

c) Siguiendo con el juego de los mercaderes, hemos propuesto un método basado en inducción fuerte para calcular todas las posible secuencias de valores para d NlogDolares, antes de explicarlo, definiremos s_n como la cantidad de secuencias que existe para un determinado d para sumas de 3 y 5, donde d es un número que pertenece $\{0\} \cup \mathbb{N}$.
La demostración consta de 6 casos bases, y una definición inductiva donde:

$$s_n = \begin{cases} 0 & \text{si } n = 0 \\ 0 & \text{si } n = 1 \\ 0 & \text{si } n = 2 \\ 1 & \text{si } n = 3 \\ 0 & \text{si } n = 4 \\ 1 & \text{si } n = 5 \\ s_{n-3} + s_{n-5} & \text{si } n > 5 \end{cases}$$

Comenzamos con la demostración:

1) Casos base:
   - Sí d = 0, no tenemos combinación de sumas de 3 y 5, por lo que no hay secuencias, por lo tanto $s_0 = 0$.

- Sí d = 1, no tenemos combinación de sumas de 3 y 5, por lo que no hay secuencias, por lo tanto $s_1 = 0$.
- Sí d = 2, no tenemos combinación de sumas de 3 y 5, por lo que no hay secuencias, por lo tanto $s_2 = 0$.
- Sí d = 3, tenemos que es posible armar una secuencias de un 3 y cero 5, por lo tanto $s_3 = 1$.
- Sí d = 4, no tenemos combinación de sumas de 3 y 5, por lo que no hay secuencias, por lo tanto $s_4 = 0$.
- Sí d = 5, tenemos que es posible armar una secuencias de un 5 y cero 3, por lo tanto $s_5 = 1$.

2) Hipótesis inductiva, como la definición es recursiva vamos a asumir que para todo k, donde k está entre $0 \leq k \leq n$, que la definición $s_{k-3} + s_{k-5}$ es cierta, para todo valor d.

3) Caso inductivo, vamos a mostrar que s_{n+1} también cumple con la definición inductiva, cuando $n + 1 \geq 5$, usando la defínción recursiva, para un $n + 1 \geq 5$ tenemos que:

$$s_{n+1} = s_{n+1-3} + s_{n+1-5} = s_{n-2} + s_{n-4}$$

$$s_{n+1} = s_{n-2} + s_{n-4}$$

Por H.I. sabemos que s_{n-2} y s_{n-4} que cumple con la definición recurviva, ya que los $k = n-2$ y $k = n-4$ se encuentra en el intervarlo $0 \leq k \leq n$, por lo que podriamos seguir aplicando la definición hasta llegar a los casos bases esto es:

$$s_{k+1} = s_{k-5} + s_{k-7} + s_{k-1} + s_{k-9}$$

$$s_{k+1} = s_{k-8} + s_{k-10} + s_{k-10} + s_{k-13} + s_{k-4} + s_{k-6} + s_{k-12} + s_{k-4}$$

...

$$s_{k+1} = s_0 + s_1 + s_2 + s_3 \ ...$$

Como sabemos que la definición recursiva se cumple para todo k perteneciente a $0 \leq k \leq n$, y por lo hemos mostrado por inducción s_{n+1} que satisface la propopiedad.

d) Aquí esta código de la implementación:

```
def secuencias(d):
    if (d == 0):
        return 0
    if (d == 1):
        return 0
    if (d == 2):
        return 0
    if (d == 3):
        return 1
    if (d == 4):
        return 0
    if (d == 5):
        return 1
    return secuencias(d - 3) + secuencias(d - 5)
```

2. Consideremos el famoso problema de las torres de Hanoi. Se trata de una serie de discos, todos distintos, puestos uno encima del otro, que van desde el más pequeño arriba hasta el más grande abajo. Tenemos 3 posiciones donde se pueden dejar los discos (A, B y C) y debemos mantener la condición de nunca poner un disco más grande sobre uno más pequeño.

a) En este problema vamos a agregar una nueva restricción, solamente podemos mover discos desde A hacia B, de B hacia C o de C hacia A.
Ningun otro movimiento está permitido. Con esta nueva restricción llamaremos Q_n a la mínima cantidad de movimientos para mover una torre de n pisos desde A hasta B y llamaremos R_n a la cantidad de movimientos que se necesitan para retornar la misma torre desde B hasta A. Muestre que se cumple lo siguiente:

$$Q_n = \begin{cases} 0 & \text{si } n = 0\,(1) \\ 2R_n + 1 & \text{si } n > 0 \end{cases}$$

$$R_n = \begin{cases} 0 & \text{si } n = 0\,(2) \\ Q_n + Q_{n-1} + 1 & \text{si } n > 0 \end{cases}$$

b) Escriba un programa que para una torre de n pisos de instrucciones al usuario de como mover los discos para pasar de la posición A a la posición B. (Las instrucciones son de la forma $'A \rightarrow 'B$ que significa tomar el disco más pequeño de A y ponerlo arriba de B, por supuesto todos los movimientos deben ser legales usando las restricciones de la parte a).

Respuestas:

a) Para mostrar que ambas propiedades se cumplen, lo mostraremos de forma separada, tomando una definición inductiva en alguna de ellas, y tomando la otra como hipótesis. Por lo tanto:
Supongamos que la relación se cumple para (2), mostraremos que de forma inductiva se cumple la relación para (1).
Si Q_n es la mínima cantidad de movimientos para mover una torre de n pisos de A hasta B.
Tomamos el caso base para (1), con $n = 0$: esto quiere decir que no tenemos discos que mover de la pila A hasta la B, por lo que se cumple el caso base.
Como hipótesis inductiva tenemos que $Q_n = 2R_{n-1} + 1$ para algún $n > 0$.
Ahora mostraremos el caso inductivo para Q_{n+1}, tomando la definición de R_n (La cantidad de movimiento que se necesitan para retornar desde la torra B hasta A, o más específicamente la cantidad mínima de movimientos para mover n discos desde una pila hasta la pila anterior, siguiendo el orden del enunciado, de A a B, B a C, y C a A). Esto quiere decir, que podemos tomar R_n movimientos para mover n discos desde la pila A pasando por B hasta la pila C, luego, tomamos el disco n + 1 y lo movemos de la pila A hasta la pila B, en la que gastamos un movimiento, finalmente necesitamos R_n movimientos para mover los primeros n discos que se encuentra en la pila C, que habíamos movido anteriormente, para dejarlos en B, pasando de C a A hasta B, es decir:

$$Q_{n+1} = R_n + 1 + R_n \; movimientos$$

Que es igual a, $Q_{n+1} = 2R_n + 1$ por que la relación se cumple para el caso inductivo.
Usaremos un razonamiento parecido para mostrar que se cumple la relación (2), tomaremos como hipótesis (1).
Para el caso base, con n = 0:
Sí tenemos 0 discos no es necesario realizar movimientos, por que $R_0 = 0$, por lo que se cumple el caso base.
Para caso inductivo la cantidad movimientos R_{n+1}, se puede calcular moviendo n discos de la pila B hasta la pila A (pasando de B a C y luego A), con R_n movimientos, luego tomamos el disco n + 1 de la pila B hasta la pila C, con lo que gastamos un movimiento, ahora tomamos los n discos de la pila A y los movemos a la pila B en Q_n movimientos, y movemos el discos n + 1, de la pila C a la pila A, con lo que gastamos un movimiento, finalmente, es necesario mover los n discos de la pila B hasta pila A, en R_n movimientos, es decir:

$$R_{n+1} = R_n + 1 + Q_n + 1 + R_n \; movimientos$$

$$R_{n+1} = 2R_n + 1 + Q_n + 1 \; movimientos, \; por \; (1)$$

$$R_{n+1} = Q_{n+1} + Q_n + 1 \; movimientos$$

De lo que se concluye que se cumplen ambas relaciones.

b) Como queremos generar un problema que cumpla las restricciones anteriores basta programas la formulas de recurrencia anteriores, para Q_n la programaremos de la misma forma que sale en la pregunta, pero para R_n, usaremos la forma encontrada en el desarrollo de la pregunta, esta es:

$$R_{n+1} = R_n + 1 + Q_n + 1 + R_n$$

```
def Qn(n, A= "A", B ="B", C="C"):
  if (n == 1):
      print "Mueva el disco " + str(n) + " '"+A+"--->"+B+"'"
  else:
      Rn(n-1, A, C, B)
      print "Mueva el disco " + str(n) + " '"+A+"--->"+B+"'"
      Rn(n-1, C, B, A);
```

```
def Rn(n, A= "A", C="C", B="B"):
  if (n == 1):
      print "Mueva el disco "+str(n)+" '" +A+"--->"+B+"'"
      print "Mueva el disco " +str(n)+" '" +B+"--->"+C+"'"
  else:
      Rn(n-1, A, B, C)
      print "Mueva el disco "+str(n)+" '"+A+"--->"+B+"'"
      Qn(n-1, C, A, B)
      print "Mueva el disco "+str(n)+" '"+B+"--->"+C+"'"
      Rn(n-1, A, C, B)
```

3. Un problema común para los algoritmos aleatorios es seleccionar entre una lista de n elementos seleccionar aleatoriamente alguno de ellos. Si la lista es pequeña y conocemos n entonces nos bastaría con ir a random.org para que nos diga que elemento seleccionar. Pero si la lista es muy grande y no sabemos el tamaño de la lista entonces el problema deja de ser trivial. Es por ello que muchas veces se usan algoritmos como el que se propone a continuación:

   - Guarde el primer elemento.
   - Cuando llegue al i-esimo elemento ($i \geq 2$), con probabilidad 1/i guarde el nuevo elemento y descarte el antiguo. En caso contrario, simplemente quédese con el antiguo elemento.

     Por ejemplo, si la lista tiene 2 elementos, partirías guardando el primer elemento, y en el segundo paso cambiaríamos el elemento guardado con probabilidad 1/2.

   a) Pruebe que cada elemento de la lista es elegido con probabilidad 1/n

   b) Programe el algoritmo para que al recibir una lista separada por comas entregué un elemento de la lista con probabilidad n. Ejemplo:

      - Input: 1,3,5,2,1,9
      - Output:5

   Respuestas:

a) Vamos a probar que cada elemento de la lista es elegido con probabilidad 1/n, para esto supongamos que n (desconocido) es la largo de lista.

- Caso base, sí n = 1, esto significa que un solo elemento puede ser elegido, y por lo tanto la probabilidad de que este elemento sea elegido es 1.
- Hipótesis inductiva, supogamos el algoritmo funciona con n elementos, y cada elemento de la lista es elegido con probabilidad 1/n.
- Caso inductivo, supogamos ahora que agregamos un nuevo elemento al final de la lista, y la probabilidad de que se elegido $\frac{1}{n+1}$, por HI sabemos que la probabilidad de elegir un elemento entre los primeros n elementos de la lista es 1/n, y también sabemos que la probabilidad de que el último agregado elemento no sea elegido es $\frac{n}{n+1}$, luego para calcular la probabilidad de que mantengamos el elemento que tenemos cuando inspeccionemos el último elemento agregado es:

$$\frac{1}{n} \cdot \frac{n}{n+1} = \frac{1}{n+1}$$

Con demostramos que luego del último paso del algoritmo cada elemento es elegido con la misma probabilidad.

b) Esta pregunta se me hizo un tanto complicada, porque no sabía cuando parar el loop cuando se llegaba al final de la lista (ya sea que hubiese optado por un método recursivo o iterativo), consultando con los auxiliares tanto en persona como por el foro dijeron que una alternativa podia ser esta.

```
import random

def aleatorio(list):
    i = 0.0
    c = 0
    num = list[0]
    i = 1.0
    while True:
        try:
            rand = random.random()
            prob = float((1.0 / (i+1.0)))

            if prob < rand:
                i += 1.0
                c += 1
            else:
                num = list[c]
                i += 1.0
                c += 1
        except IndexError:
            break
    print '"'+str(num)+'"'
```