Tarea II

Para entregar todos menos pregunta 8 el lunes 17 de septiembre en mi buzón de correo.

1. Resuelva la siguiente recursión:

   $$T(n) = 2T(n/4) + \sqrt{n}.$$

   Verifica tu resultado con lo que dice el Teorema Master para este caso.

2. Considera la siguiente recursión:

   $$T(n) = 2T(n/2) + nlog(n) \quad T(1) = 2$$

   Resuélvala con la técnica general que vimos para ecuaciones lineales de diferencias. Puedes suponer que $n = 2^m$.
   Verifica que es un ejemplo donde el Teorema Master no aplica.

3. Tomamos el contador binario de la clase. Además de la operación `increment` para aumentar con una unidad el contador, se tiene ahora también la operación `reset` para ponerlo en zero. Considera la siguiente implementación que hace uso de la variabla auxiliar `max(A)` que contiene la posición más a la izquierda donde hay un 1 (y -1 si el contador está en 0). Demuestra con analisis amortizado que el costo de n operaciones es $O(n)$.

```
INCREMENT(A)
i ← 0
while i < length[A] and A[i] = 1
    do A[i] ← 0
       i ← i + 1
if i < length[A]
  then A[i] ← 1
       ▷ Additions to book's INCREMENT start here
       if i > max[A]
         then max[A] ← i
  else max[A] ← −1

RESET(A)
for i ← 0 to max[A]
    do A[i] ← 0
max[A] ← −1
```

4. Hay n personas cada una sentada en una silla. Definimos a_n como el número de maneras de (re)acomodar estas n personas tal que nadie se siente en su silla original.

   (a) Argumenta que:

   $$a_n = (n-1)(a_{n-1} + a_{n-2}).$$

   (hint: supongamos que persona 1 se siente en silla i. Distingue entonces el caso donde persona i se siente en silla 1 del caso donde persona i se siente en otra silla).

   (b) Define la función generatrice exponential de $\{a_n\}$ como

   $$G(x) = \sum_{n \geq 0} a_n \frac{x^n}{n!}$$

   Verifica que $G()$ satisface $(1-x)G'(x) = xG(x)$.

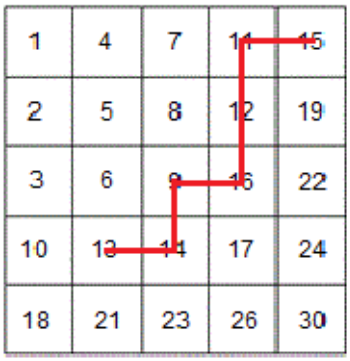
   (c) Verifica que $G(x) = \exp(-x)/(1-x)$. Determina el valor de a_n.

5. Supongamos que tenemos A, un arreglo $2^n \times 2^n$ ordenado, es decir:

   $$A[i,j] \leq A[i,j+1] \quad A[i,j] \leq A[i+1,j], \forall i,j$$

   Queremos ver si aparece el valor x en A.

   Existen varios algoritmos $O(n)$. Uno está basado en empezar en la celda de la esquina superior derecha y dependiendo de la comparación de la celda con el elemento que se busca, uno va un paso hacia abajo o un paso hacia la izquierda (ver la figura para la búsqueda de 13)



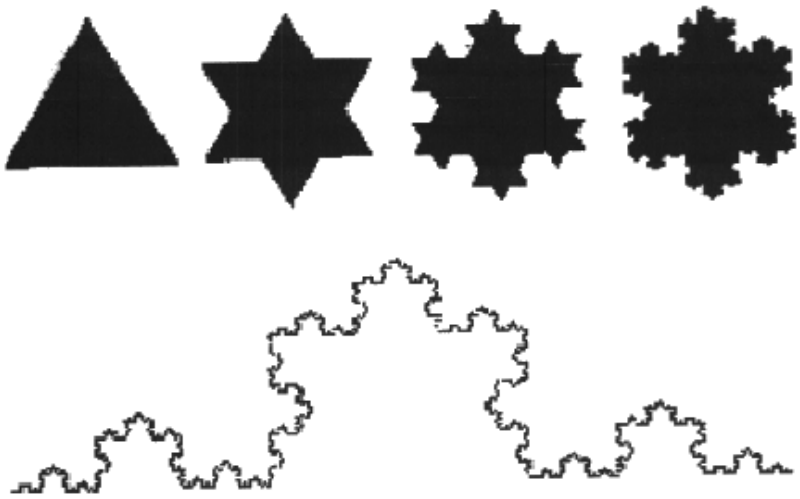
   Escribe un algoritmo en Python que implementa esta idea. Verifica que es $O(n)$.

6. Dada una muestra $\{x_i\}_1^n$ de $\mathcal{U}([0,1])$. Se colocan los datos en canastas como vimos en el algoritmo de Bucketsort pero sin ordenar cada canasta.

   Calcula el número promedio de comparaciones que se va a tener que hacer para localizar un elemento si sabes que este elemento es uno de la muestra, elegido al azar de la muestra (o sea todos tienen la misma probabilidad).

   Calcula el número promedio de comparaciones que se va a tener que hacer para (tratar de) localizar un elemento si sabes que este elemento NO pertenece a la muestra y fue elegida de una distribucion con densidad f sobre $[0,1]$.

7. Una curva (isla) de von Koch es un fractal que se define por el siguiente proceso iterativo:



   Tiene la particularidad que el perímetro converge a infinito pero el área está acotada. Leer `https://en.wikipedia.org/wiki/Koch_snowflake`

   Escribe un programa recursivo que dibuja una curva de von Koch en Python.

8. (para pensar) La idea es usar lo anterior para construir un objeto de decoración con la cortadora laser de CIMAT. Puedes usar toda tu fantasia. Basta hacer una búsqueda en Google con `laser cut fractal`.

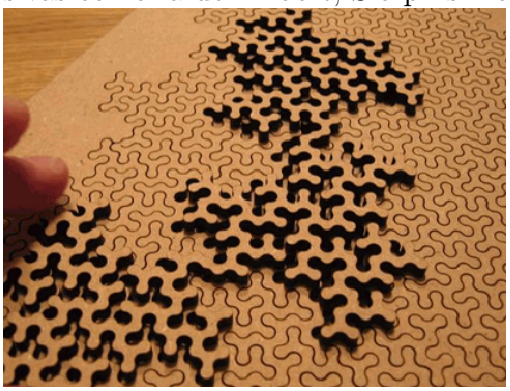
Lo anterior requiere exportar tu diseño a formato `dxf`.
Para eso, hay que instalar la libreria `dxfwrite`, por ejemplo a través de `conda install -c travis dxfwrite=1.2.0`

Cambiar cada llamada a `plot` por una llamada a `dxf.line`. Por ejemplo:

```
from dxfwrite import DXFEngine as dxf
drawing = dxf.drawing('test.dxf')
drawing.add(dxf.line((0, 0), (10, 0), color=7))
drawing.save()
```

Por definición una unidad corresponde a 1mm. Puedes ver el resultado con el software `FreeCAD` (usar la opción `importar`).

En lugar de un fractal de von Koch puedes usar también curvas recursivas como la de Hilbert, Sierpinski etc.



9. (no entregar) Considera las secuencias $\{u_i\}$ y $\{v_i\}$. Si $u_0 = v_0 = 1$ y

$$u_n = u_{n-1} + 2v_{n-1} \text{ y } v_n = u_{n-1} + v_{n-1}$$

Encuentra una expresión explícita para u_n y v_n.

10. (no entregar) Define $a_n = \sum_{i=0}^{n} i^2$. Encuentra una recursión lineal para a_n y resuélvela para obtener una expresión explícita para a_n.