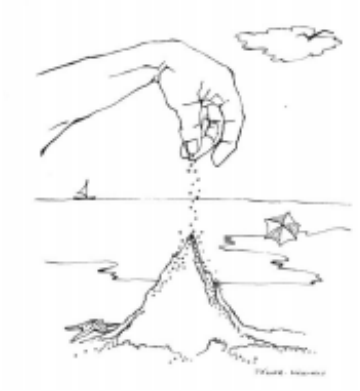
Tarea 1

Pilas de Arena Abelianas

Prof: Nelson Baloian, Patricio Poblete
Auxiliares: Rodrigo Llull, Gabrial Norambuena, Javier Oliva, Matías Ramírez
Ayudantes: Salvador Alveal, Daniel Báez, Nicolás Canales, Anastassia Carter, Gabriel Chandía, Cristián Llull, Valentina Ramos, Ignacia Robert

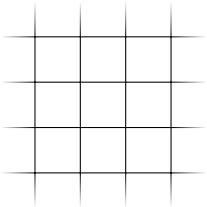
El problema

El objetivo de esta tarea es estudiar un problema inspirado en un fenómeno físico, a través de un modelo matemático, el cual si bien es sencillo, produce resultados que presentan una estructura muy interesante.



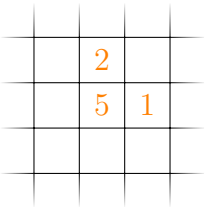
La idea es que si uno va formando una pila de arena, llega un momento en que se produce un pequeño derrumbe, y la arena de esa pila se derrama hacia los lugares vecinos. Estos a su vez se pueden derrumbar, y el proceso continúa hasta que finalmente se estabiliza.

Para estudiar este proceso, usaremos un modelo ultra simplificado. Supondremos que la arena se deposita sobre una superficie plana, la cual está dividida en pequeñas celdas cuadradas, las cuales forman un tablero como se muestra en la figura siguiente:

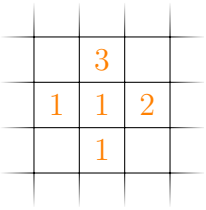


El modelo supone que si se apilan demasiados granos de arena en una celda, se produce un derrumbe. En particular, la regla es que si en una celda hay 4 o más granos, se le quitan 4 granos, que se reparten equitativamente hacia las celdas vecinas en los cuatro puntos cardinales.

Para simular este proceso, supondremos que cada celda almacena un número entero, que es la cantidad de granos almacenados en su interior. Aplicando la regla antes descrita (y suponiendo que las celdas que aparecen vacías tienen 0 granos), desde la configuración



se pasaría a



Cuando hay más de una casilla con exceso de granos de arena, la regla se puede aplicar a ellas en cualquier orden y el resultado final es el mismo. Esta propiedad es la que hace que estas pilas de arena se denominen *abelianas*.

La idea va a ser partir desde una configuración inicial, y luego aplicar esta regla en todos los casilleros que se pueda, hasta que no quede ninguno que tenga 4 o más granos de arena. Esa configuración final la vamos a visualizar asignando un color distinto a cada número de granos.

En particular, nos va a interesar estudiar lo que ocurre cuando la configuración inicial tiene todas las celdas vacías, excepto la del centro, en la cual hay N granos de arena (donde N es un parámetro del problema).

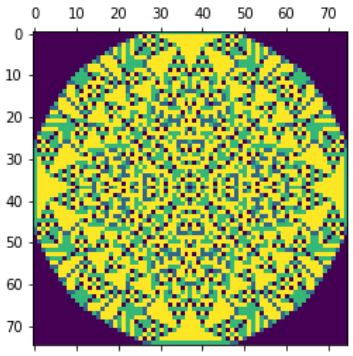
Note que en teoría el tablero es infinito, en el sentido que siempre hay espacio para colocar los granos de arena que se van distribuyendo. Para su simulación, usted debe calcular un tamaño de tablero suficientemente grande como para estar seguro que ningún grano de arena se salga hacia afuera de los bordes (calcule la máxima área que se puede cubrir con N granos de arena, y después calcule cuán grande debe ser el tablero para poder contener esa área).

La Tarea

Parte 1

Usted debe escribir un programa en python, en un archivo llamado PilaArena.py, que le pida al usuario ingresar el valor del número N y luego simule el proceso anteriormente descrito hasta que se estabilice. El programa debe contar e imprimir en la salida estándar el número total de veces que se aplicó la regla que distribuye 4 granos de arena hacia los vecinos. Además, debe visualizar en la pantalla el tablero resultante, usando los métodos que se describen más adelante.

Por ejemplo, al simular con N = 10000, la figura que resulta es:



Parte 2

Observe que cuando en una celda hay un número grande de granos de arena, es muy ineficiente ir quitándole de 4 en 4, y sería mejor quitar de una sola vez lo más que se pueda. Podemos mejorar nuestro programa si cambiamos la regla de distribución, y decimos que si en una celda hay un número de granos de arena mayor o igual a 4, le quitamos de una sola vez el mayor múltiplo de 4 posible, y todos esos granos los repartimos equitativamente entre los vecinos de los cuatro puntos cardinales. Modifique su programa de acuerdo a esta nueva regla.

Parte 3

Ejecute ambos programas con valores crecientes de N, hasta el mayor número que pueda dentro de un tiempo de ejecución razonable. Escriba un informe en donde describa muy brevemente el problema, señale cómo dimensionó el tamaño del tablero, ilustre con la imagen correspondiente a N = 128 y la correspondiente al mayor N que haya logrado ejecutar, y compare a través de una tabla y un gráfico el número de aplicaciones de la regla que hace el programa de la Parte 1 y el de la Parte 2, para los distintos valores de N que usted haya calculado.

Discuta si valió la pena la optimización y discuta también (pero no implemente) otras posibles optimizaciones que se le ocurran. En su informe debe describir claramente qué cambio le hizo al programa de la Parte 1 para transformarlo para la Parte 2.

Debe entregar un archivo .ipynb con su código ejecutable con comentarios explicando su funcionamiento y resultados de experimentos.
Debe incluir

- Introducción
- Código explicado
- Conclusión

Si lo desea puede hacer la entrega como un link de google colab en lugar de entregar el archivo .ipynb, para esto, no debe modificarlo después de la fecha de entrega, sino se considerará como atraso.

Visualización

Para la visualización se recomienda utilizar arreglos de numpy y la libreria matplotlib. Aquí un ejemplo

```
import matplotlib.pyplot as plt
import numpy as np

#Crea una matriz de numpy llena de ceros de dimensiones 5 x 4
mat = np.zeros((5, 4))

#Asigna un uno a la posicion (0, 1)
mat[0][1] = 1

#Imprime
plt.matshow(mat)
plt.show()
```

Reglas

- La tarea debe hacerse en python3
- La tarea es individual
- Consulte sus dudas a través del foro del curso
- No se aceptan atrasos