Actividad 4 - Incendio forestal

Pensamiento Computacional

Segundo Cuatrimestre 2024

Pasos para resolver los ejercicios

1. **Entender qué hay que hacer**
   a) **Leer en detalle** el texto de la descripción de la función e interpretar qué se espera (e.g., qué parámetros tiene, qué hace con la entrada y cómo genera la salida).
   b) Escribir en el cuaderno distintos ejemplos de **entradas** y qué **salidas** se esperan para cada una.
2. **Armar el esqueleto de la función**
   a) Escribir exactamente como se pidió el **encabezado de la función**, incluyendo los parámetros.
   b) Definir la **variable de la salida** y ponerle un valor inicial válido (por ej., 0 para un contador)
   c) **Agregar el** `return` donde se devuelve la variable anterior.
   d) Probar **ejecutar la función** con diferentes valores, aunque todavía no hace lo que se pide.
3. **Implementar la función**
   a) **Escribir de a poco** el contenido de la función.
   b) **Usar otras funciones** para resolver el problema, ya sea que estén de antes (y las reutilizan), o que crean otras nuevas (porque nos conviene partir el problema en otros más pequeños y manejables).
4. **Probar la función**
   a) **Probar la función solamente por la consola**, usando distintos valores para los parámetros, y ver que funciona bien. No hacer las pruebas desde el editor de texto de Spyder ni usar prints.
   b) **Ver qué pasa con casos especiales**, como cuando le dan una lista vacía u otro caso donde se usa una entrada válida, pero que pone a prueba los límites de lo que se hizo.
5. **Preguntar al equipo docente** por cualquier duda o consulta.

Incendio forestal

El objetivo de esta actividad es modelar la dinámica de un bosque utilizando el modelo de incendio forestal propuesto por Back et al. en 1990 y Drossel y Schwabl en 1992.

Representación y dinámica

Representaremos un bosque como una grilla de n celdas en una dimensión (numeradas del 0 al $n-1$). Cada celda puede estar en tres estados posibles:

0 representa una posición vacía,

1 representa un árbol, y

-1 representa un árbol quemado.

Inicialmente todas las celdas del bosque están vacías. En la dinámica se suceden cuatro etapas:

1. **Etapa de brotes**: cada celda vacía tiene probabilidad p de que le brote un árbol.
2. **Etapa de caída de rayos**: en cada celda puede caer un rayo con probabilidad f. Si un rayo cae sobre un árbol, éste se quema (por lo que el bosque resultante tendrá lugares vacíos, árboles vivos y árboles quemados).
3. **Etapa de propagación**: se propaga el incendio todo lo posible. Cada árbol quemado propaga el fuego a los árboles vecinos vivos inmediatos (el de la derecha y el de la izquierda). Nota: la propagación termina cuando no queda ningún árbol quemado que pueda propagar el fuego.
4. **Etapa de limpieza**: se tiran abajo los árboles quemados y esas celdas pasan a estar vacías nuevamente.

Ejercicios

La implementación de las cuatro etapas, cada una con su regla, cierra un ciclo anual, luego de lo que se cuenta la cantidad de árboles que sobrevivieron ese año. Vamos a desarrollar algunas funciones útiles para el modelo.

Importante: Para modelar a nuestros bosques vamos a usar arrays de **NumPy**, y para simular con azar vamos a usar `random`. Antes de empezar a resolver los ejercicios, agregue el siguiente `import` al principio de su archivo *.py*:

```
import random
import numpy as np
```

1. Implemente la función `generar_bosque(n)`, que recibe como parámetro el número de celdas y devuelve un bosque vacío de ese tamaño. Representamos el estado del bosque mediante una array de NumPy de n elementos, donde cada elemento representa una celda del bosque. Una posición del array con valor 0 representa una celda vacía del bosque.

   Ayuda: Usar la función `np.repeat` de NumPy.

   Para probar en consola:

   Crear un bosque de 10 celdas, imprimirlo y verificar que se muestra un array de 10 ceros:

   ```
   bosque = generar_bosque(10)
   bosque # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   ```

2. Construya la función `cuantos(bosque, tipo_celda)` que devuelve la cantidad de celdas que hay en el bosque de la categoría `tipo_celda`. Por ejemplo `cuantos(bosque, 1)` devuelve la cantidad de árboles vivos, `cuantos(bosque, -1)` devuelve la cantidad de árboles quemados y `cuantos(bosque, 0)`, la cantidad de celdas vacías.

   Para probar en consola:

   Generar un bosque de 10 celdas, con 1's y -1's en algunas posiciones, y verificar que la función cuente bien los tres tipos de celda:

   ```
   bosque = np.array([0, 1, 0, -1, 0, 1, 0, -1, -1, 0])
   cuantos(bosque, 1) # 2
   cuantos(bosque, -1) # 3
   cuantos(bosque, 0) # 5
   ```

3. Construir una función `suceso_aleatorio(p)` que toma como parámetro p, un valor entre 0 y 100 que indica qué tan probable es que ocurra un suceso. La función debe generar un número al azar, evaluar si occure o no el suceso aleatorio (comparando el número generado al azar con p), y devolver `True` o `False`.

   Por ejemplo, `suceso_aleatorio(40)` debería devolver `True` aproximadamente el 40 % de las veces que es invocada, permitiendo así modelar un suceso aleatorio con una probabilidad de 40 % de ocurrir.

   Para probar en consola:

   Para distintos valores de p, llamar varias veces a la función:

   ```
   suceso_aleatorio(0) # Cada vez que se la llama debe devolver False

   suceso_aleatorio(100) # Cada vez que se la llama debe devolver True

   suceso_aleatorio(50) # Alrededor de la mitad de las veces True y mitad False
   ```

4. Implemente la función `brotes(bosque, p)` que a partir de un bosque (es decir, una array que representa un bosque) y de un valor p entre 0 y 100, en cada celda vacía genera un árbol nuevo con probabilidad `p`. Debe devolver el bosque modificado.

   Ayuda: Usar la función implementada en el item anterior en cada celda vacía del bosque.

Para probar en consola:

Para distintos valores de p, llamar varias veces a la función con bosques vacíos de tamaño 10:

```
bosque = generar_bosque(10) # Nuevo bosque
bosque = brotes(bosque, 0)
bosque # Tiene que devolver igual que el original

bosque = generar_bosque(10)
bosque[1] = 1 # bosque con un solo arbol, en 2da celda
bosque = brotes(bosque, 0) # No se agrega ningun otro arbol
bosque # Tiene que devolver bosque con un solo arbol, en 2da celda

bosque = generar_bosque(10) # Se genera un nuevo bosque
bosque = brotes(bosque, 100) # Se agregan arboles en todas las posiciones
bosque # Tiene que devolver un bosque lleno de arboles

bosque = generar_bosque(10) # Bosque nuevo
bosque = brotes(bosque, 50) # Genera arboles como tirando una moneda
bosque # Habran brotado alrededor de la mitad de las posiciones
```

¿Qué pasa con el bosque si llamamos a `bosque = brotes(bosque, 50)` muchas veces?

5. Implemente la función `rayos(bosque, f)` que realiza la caída de rayos en un bosque con probabilidad f. Suponga que en cada celda hay una probabilidad f de que caiga un rayo, si en la celda había un árbol y cayó un rayo, entonces el árbol se quema (la celda pasa del estado 1 al estado -1), si la celda estaba vacía y cayó un rayo, no pasa nada (sigue en estado 0). La función debe devolver el bosque modificado.

Para probar en consola:

Probar correr la función con diferentes bosques (por ej., un bosque lleno de árboles y un bosque limpio que tenga la mitad de las celdas ocupadas por árboles). ¿A qué fracción de los árboles le cayó un rayo?

6. Implemente la función `propagacion(bosque)` que realiza la fase de propagación de incendios en un bosque y devuelve el bosque modificado. Luego de aplicar esta función en el bosque **no queda ningun árbol vivo que sea vecino de un árbol quemado**. La función debe devolver el bosque modificado.

Para probar en consola:

Llamar a la función usando estos bosques:

- `bosque_1 = np.array([1, 1, 1, -1, 0, 1, 0, -1, 1, 0])`
- `bosque_2 = np.array([-1, 1, 1, -1, 1, 1, 0, 0, -1, 1])`
- `bosque_3 = np.array([-1, 1, 1, 0, 1, 1, -1])`
- `bosque_4 = np.array([1, 1, -1, 0, -1, 1, 1])`
- `bosque_5 = np.array([1, 1, 1, -1, 1, 1, 1])`

¿Se obtienen los resultados esperados?

7. Implemente la función `limpieza(bosque)` que reemplaza en el bosque los árboles quemados por celdas vacías y lo devuelve modificado.

Para probar en consola:

Llamar a la función usando `bosque = np.array([-1, -1, 0, -1, 0, 1, 0, -1, -1, -1])`

¿Se obtiene el resultado esperado?

Para resolver como tarea

Estos ejercicios no esperamos que los resuelvan en el tiempo de clase, sino que queremos que los vean luego. Sin embargo, vamos a discutirlos en el pizarrón para ver que estamos de acuerdo en qué es lo que hay que hacer.

8. Defina una función llamada `dinamica(n, a, p, f)` que simula la dinámica de un bosque a lo largo de los años, la cual debe tener como parámetros a `n`, el tamaño del bosque, `a`, la cantidad de años a considerar, `p` y `f` que representan las probabilidades de brotes y de rayos. La función debe registrar la cantidad de árboles que sobreviven cada año y calcular el promedio al cabo de los `a` años. Es decir, se debe hacer lo siguiente:

   a) Generar un bosque vacío de `n` celdas.

   b) En cada año implementar la dinámica anual del bosque:

      1) **Brotes:** hacer brotar árboles con probabilidad `p` en el bosque.
      2) **Rayos:** simular que caen rayos con probabilidad `f` sobre el bosque que resulta de i) (si la celda donde cayó un rayo estaba ocupada por un árbol, el árbol se quema), obteniendo un bosque con algunos árboles quemados.
      3) **Propagación incendio:** simular la propagación del incendio a partir del bosque que resulta de ii), obteniendo un bosque quemado en su máximo alcance.
      4) **Limpieza:** los árboles quemados pasan a estar vacíos (las celdas en estado -1 vuelven al estado 0), obteniendo un bosque limpio nuevamente.

   c) Contar y registrar la cantidad de árboles que sobreviven en el bosque obtenido y luego volver a i) con el bosque actual, hasta completar la cantidad especificada de años.

   d) Calcular y devolver el promedio de los árboles sobrevivientes entre todos los años.

9. *a*) Defina la función `arboles_sobrevivientes(f, a, n)` que devuelve una lista con el promedio de árboles sobrevivientes para diferentes valores de `p`, desde 0 % a 100 %, al cabo de ciertos `a` años, dada una probabilidad `f` de caída de rayos, para un tamaño de bosque `n`. Se deberá usar la función `dinamica` hecha anteriormente.
   Nota: La lista a devolver debe tener 101 elementos, uno por cada valor de `p`, desde el 0 hasta el 100 (inclusive).

   b) Defina la función `p_optimo(f, a, n)` que devuelve el valor de óptimo de `p`, dada una probabilidad de rayos `f`, para `a` años de simulación, y con un bosque de `n` posiciones.

   c) Usando `arboles_sobrevivientes`, generar un gráfico de los árboles sobrevivientes (eje x) para los distintos valores de `p` (eje y), dado un `f` de 2, `a` de 500 y `n` de 100.
   Nota: Recordar que para el gráfico se debe importar el paquete pyplot usando `import matplotlib.pyplot as plt`, y seguir el ejemplo de las diapos de esta clase.

   d) Usando `p_optimo` obtener el `p` que genera la mayor cantidad de sobrevivientes, con los mismos parámetros.

Extensiones (optativo)

Las siguientes son extensiones de lo que realizamos anteriormente. No es obligatorio que las hagan, pero, si ya terminaron lo anterior que había que hacer y vieron que funciona bien, estos ejercicios les pueden servir para acercarse a un modelado más complejo y cercano a la realidad.

10. **Dinámica evolutiva.** En esta versión del modelo se permite que haya una heterogeneidad en el bosque: no todas las celdas tienen la misma probabilidad de hacer brotar árboles. Cada celda usa su propio valor de p para hacer brotar árboles y en cada paso de tiempo t, después de la propagación del incendio y antes de la época de limpieza, cada celda en la que había un árbol en la época de brotes modifica su valor de p:

    - si el árbol sobrevivió entonces p de la celda aumenta en 5 %.
    - si el árbol no sobrevivió entonces p de la celda disminuye en 5 %.

Las celdas donde no había árbol mantienen su valor de p. En un bosque de $n = 100$ celdas, inicialice un valor de p para cada celda; por ejemplo, $p = 50\,\%$ para cada celda, o un número al azar entre $0\,\%$ y $100\,\%$ para cada celda. Guarde la información en un array `pes`. Simule la dinámica de propagación de incendios evolutiva con una probabilidad de caída de rayos fija de $f = 10\,\%$. Compare los resultados obtenidos con la dinámica simple. Por ejemplo, evalúe la distribución final de p para la grilla de celdas. Grafique la cantidad de árboles que sobrevive en cada año en función del tiempo. ¿Cuál es la producción promedio en cada caso?

11. **Otras dinámicas.** Suponga que cada árbol vivo representa a una persona sana y cada árbol quemado a una persona enferma. La propagación del fuego es ahora la propagación de la enfermedad y agregue la siguiente regla: después de la propagación, elegimos al azar dos lugares i y j y si en uno de los lugares había una persona sana y en el otro una enferma, la persona sana se contagia, luego se vuelve propagar la enfermedad. Simule la propagación de la enfermedad y evalúe el valor óptimo de probabilidad p que hace que se maximice la cantidad de personas sanas, con un valor de f fijo en $f = 2\,\%$ (probabilidad de enfermarse).