🧬Resident Evil UDEM: Simulación de propagación de infección en una matriz con árbol de contagio

Contexto general

En esta práctica implementarás una simulación de propagación de una infección dentro de una **matriz NxN**, donde **N personas** (p1, p2, …, pN) se mueven aleatoriamente en todas las direcciones. Al inicio de la simulación, **una persona aleatoria** será seleccionada como el **paciente cero** (infectada).

Durante cada iteración o ronda, todas las personas se desplazan aleatoriamente en una de las 8 direcciones posibles (norte, sur, este, oeste y diagonales). Cuando una persona infectada **se cruza** o **comparte posición** con una persona sana, esta última **reduce su nivel de defensa** en 1. Cuando el nivel de defensa llega a **0**, la persona se infecta de inmediato.

El sistema debe mantener y actualizar un **árbol de propagación de infección**, que registre quién contagió a quién. La simulación termina cuando el usuario lo decida o cuando **todas las personas estén infectadas**.

Reglas y dinámica de la simulación

1. **Inicialización**
    - El usuario define el tamaño de la matriz `N` y el número de personas `n`.
    - Se ubican aleatoriamente las `n` personas en la matriz (una por celda al inicio).
    - Una persona es seleccionada aleatoriamente como **paciente cero** (infectada).
    - Todas las demás comienzan **sanas** con un **nivel de defensa inicial** de 3 (puede parametrizarse).
2. **Movimiento**
    - En cada ronda, todas las personas se mueven a una celda adyacente aleatoria (en cualquiera de las 8 direcciones posibles).
    - Si una persona intenta salir de los límites, puede:
        - a) rebotar (quedarse en la misma celda), o

- b) desplazarse de forma circular (modo toroide).
  (El grupo debe documentar cuál opción utiliza).

3. **Cruces e infección**

   - Si una persona sana comparte celda con una o más personas infectadas:
     - Pierde **1 punto de defensa** por cada cruce.
     - Si su defensa llega a **0**, se **infecta de inmediato**.
     - En ese instante se actualiza el **árbol de propagación**, agregando la arista `(infectador → nuevo infectado)`.

4. **Curación**

   - El usuario puede ejecutar la acción `curar(x, y)` indicando las coordenadas de una persona.
   - Si la persona está infectada:
     - Pasa a estar sana nuevamente.
     - Se **elimina del árbol** de propagación.
     - Los **descendientes** de esa persona en el árbol pasan a ser **hijos directos del infectador original** de la persona curada.

5. **Agregar nuevas personas**

   - El usuario puede agregar nuevas personas (`pN+1`, `pN+2`, …) en coordenadas específicas de la matriz.
   - Estas nuevas personas comienzan sanas con un nivel de defensa inicial de 3.

6. **Defensa especial**

   - Cada **tres rondas** completadas, todas las personas sanas **aumentan su defensa en 1 punto**.
   - Cada vez que una persona sana se cruza con una infectada, su defensa **disminuye en 1**.
   - Cuando el nivel de defensa llega a **0**, la persona se **infecta automáticamente**.

7. **Visualización**

   - La simulación debe mostrar después de cada ronda:

     - **La matriz**:
       - Personas **sanas en verde** (🟩 o texto verde en consola).
       - Personas **infectadas en rojo** (🟥 o texto rojo en consola).
       - Las celdas deben mostrar el identificador (p1, p2, etc.).
       - Un listado de las personas sanas con nombre: nivel de defensa.

        - **El árbol de propagación de infección**, actualizado y legible (por ejemplo, como árbol ASCII o lista de adyacencia).

8. **Finalización**

    - La simulación termina cuando:
        - El usuario lo indique, o
        - Todas las personas estén infectadas.

Requisitos técnicos

1. **Obligatorio:** uso de **type hints** en todas las funciones y clases.
2. **Programación orientada a objetos:** se deben definir clases para representar cada artefacto de la solución.
3. **Visualización:** puede implementarse en consola o con una interfaz gráfica sencilla.
   Código limpio: debe haber separación clara entre la lógica de simulación, las estructuras de datos y la visualización.
4. **Reproducibilidad:** se debe permitir usar una semilla (`random.seed`) para repetir experimentos idénticos.

Entregables

1. **Repositorio en Github con Código fuente completo** con type hints y sin comentarios. Se revisará el aporte individual de cada miembro del equipo.

2. **README.md** que contenga:

    - Descripción del proyecto
    - Cómo ejecutar la simulación
    - Estructura de clases
    - Supuestos asumidos

3. **Capturas de ejecución** donde se evidencie:
    - Propagación progresiva
    - Curación y reparenting
    - Incremento y decremento de defensa
    - Visualización del árbol por ronda

Trabajo en equipo

- Grupos de **1 o 2 personas**.

- Cada integrante debe participar en el desarrollo de la solución vía repositorio y debe hacer sustentación para demostrar su conocimiento del código.

Rúbrica de evaluación

Criterio	Descripción	Peso
Sustentación (60%)		
Adición / modificación práctica	Cambio en la implementación presentada	20%
Explicación solución	Explica la implementación presentada y la adición / modificación práctica	40%
Implementación (40%)		
Correctitud funcional	Cumple con las reglas de movimiento, infección y curación.	15%
Implementación del árbol	El árbol se actualiza correctamente y se visualiza en cada iteración.	10%
Uso de type hints y limpieza del código	Tipado correcto, buena organización y documentación.	10%
Evidencias y pruebas	Presenta pruebas o ejemplos documentados.	5%

Entregas:
Grupo 62: Octubre 31
Grupo 64: Octubre 30
Grupo 61: Noviembre 6