UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA CIVIL EN BIOINFORMÁTICA

Programación Avanzada
Proyecto - Unidad III

Objetivo

El objetivo es poder trabajar con los conceptos asociados al modelado de problemas mediante el paradigma de orientación a objetos considerando los saberes conceptuales y aplicarlos al contexto adecuado. Al término de este trabajo, se espera que el estudiante sea capaz de:

- Desarrollar una solución computacional que aplique el paradigma de programación orientada a objetos acorde a los objetivos y requerimientos de un problema específico.
- Aplicar el uso de una API de interfaz gráfica en el desarrollo de una solución computacional.

Tarea

Desarrollar una simulación basada en objetos que modele el comportamiento dinámico de una colonia bacteriana, considerando su crecimiento, competencia por recursos, mutaciones aleatorias y condiciones ambientales. Se espera que la simulación refleje fenómenos realistas como expansión poblacional, competencia intraespecífica y aparición de resistencia a condiciones adversas.

Contexto Biológico

Las bacterias viven en ambientes donde deben crecer, dividirse y competir por nutrientes limitados. Esta simulación busca modelar una colonia bacteriana en una placa de Petri, donde cada bacteria puede:

- **Consumir nutrientes:** Las bacterias absorben nutrientes desde la celda en la que se encuentran para incrementar su energía.
- **Dividirse si alcanza un umbral de energía:** Una vez que una bacteria acumula suficiente energía, puede reproducirse mediante mitosis.
- **Morir si no logra suficientes recursos:** Si una bacteria no obtiene la energía mínima necesaria, entra en estado de muerte celular.
- **Mutar y adquirir resistencia a condiciones ambientales:** Durante la reproducción, algunas bacterias pueden presentar mutaciones que les otorguen ventajas, como resistencia a antibióticos.

El entorno se representa como una grilla bidimensional, donde las bacterias pueden interactuar con sus vecinas. Esto permite visualizar fenómenos como la expansión de clones, formación de biofilms y zonas resistentes.

Clases Requeridas

Bacteria

- **id**: identificador único.
- **raza**: identificador genético o especie.
- **energia**: nivel de energía actual.
- **resistente**: booleano que indica si es resistente.
- **estado**: activa o muerta.
- **Métodos**: `alimentar()`, `dividirse()`, `mutar()`, `morir()`.

Ambiente

- **grilla**: matriz que representa el entorno.
- **nutrientes**: cantidad de nutrientes por celda.
- **factor_ambiental**: puede representar antibióticos u otras presiones.
- **Métodos**: `actualizar_nutrientes()`, `difundir_nutrientes()`, `aplicar_ambiente()`.

Colonia

- **bacterias**: lista de objetos `Bacteria`.
- **ambiente**: instancia de la clase `Ambiente`.
- **Métodos**: `paso()`, `reporte_estado()`, `exportar_csv()`.

Simulador

Controla los pasos de simulación e inicializa los valores.

- **Métodos**: `run(pasos)`, `graficar_crecimiento()`, `graficar_resistencia()`.

Main

Inicializa la simulación con parámetros definidos.

Eventos a Simular

- **Reproducción por mitosis:** Las bacterias duplican su información genética y se dividen en dos células hijas cuando alcanzan un umbral de energía suficiente. Esta acción depende de la disponibilidad de nutrientes y del espacio libre en la grilla.
- **Difusión y consumo de nutrientes:** Los nutrientes están distribuidos en la grilla y son consumidos por las bacterias cercanas. La difusión representa el movimiento natural de los nutrientes hacia zonas menos concentradas, permitiendo que otras bacterias puedan también alimentarse.
- **Muerte por inanición o antibiótico:** Una bacteria puede morir si su nivel de energía cae por debajo de un límite crítico, lo que ocurre si no logra alimentarse. También puede morir si entra en contacto con una zona que contiene un antibiótico, a menos que posea resistencia adquirida.

- **Mutaciones aleatorias:** Algunas bacterias, al dividirse, pueden presentar mutaciones genéticas. Estas mutaciones pueden ser neutras, perjudiciales o beneficiosas. Un ejemplo de mutación beneficiosa es la adquisición de resistencia a un antibiótico presente en el ambiente.
- **Eventos probabilísticos:** Muchos procesos del sistema están regidos por la probabilidad: la absorción de nutrientes, la aparición de mutaciones, o incluso la posibilidad de supervivencia frente al antibiótico. Esto permite simular un entorno más realista y estocástico.

Ejemplo de Evento Probabilístico: Supervivencia al antibiótico

Supongamos que una bacteria entra en una celda de la grilla que contiene un antibiótico. No todas las bacterias mueren inmediatamente al contacto; se introduce una probabilidad para simular el efecto estocástico del ambiente:

```python
import random

def aplicar_antibiotico(bacteria):
    if not bacteria.resistente:
        # 15% de posibilidades de sobrevivir sin resistencia
        probabilidad_supervivencia = 0.15
        if random.random() > probabilidad_supervivencia:
            bacteria.estado = "muerta"
            print(f"{bacteria.id} ha muerto por antibiótico.")
        else:
            print(f"{bacteria.id} sobrevivió al antibiótico.")
```

Listing 1: Ejemplo de evento estocástico

En este ejemplo:

- Si la bacteria **no es resistente**, se le asigna una **probabilidad del 15 % de sobrevivir**.
- Se utiliza `random.random()` para generar un número entre 0 y 1.
- Si el número aleatorio es mayor a 0.15, la bacteria muere.
- Si es menor o igual a 0.15, sobrevive aleatoriamente.

Otros ejemplos de eventos probabilísticos

- **Mutación al dividirse:** Cada vez que una bacteria se divide, existe una pequeña probabilidad (por ejemplo, 5 %) de que su descendiente mute.
- **Consumo de nutrientes variable:** En una celda con nutrientes, una bacteria puede consumir entre 15 y 25 unidades con una distribución uniforme.
- **Herencia de resistencia:** Si una bacteria resistente se divide, su hija tiene un 95 % de probabilidad de heredar la resistencia.

Salidas Esperadas

- Archivo CSV con estado de bacterias por paso.
- Gráficas de crecimiento y resistencia.
- Interfaz gráfica.
- Uso obligatorio de control de versiones (20 commits mínimo).

Ejemplo Narrado de Simulación

Paso 1: 20 bacterias activas colonizan aleatoriamente la placa. Todas comienzan con energía = 50. No hay divisiones ni muertes.
Paso 2: 18 bacterias alcanzan energía > 60 y se dividen. 2 mueren por falta de nutrientes.
Paso 3: 3 bacterias mueren al ingresar a zona con antibiótico. Una muta y se vuelve resistente.
Paso 4: 5 nuevas mutaciones; 3 efectivas. 20 divisiones. Empieza la escasez de nutrientes.
Paso 5: 6 muertes por inanición. La zona central es dominada por bacterias resistentes.

Visualización de la Grilla Bacteriana

Esto podría representar el paso número dos del ejemplo:

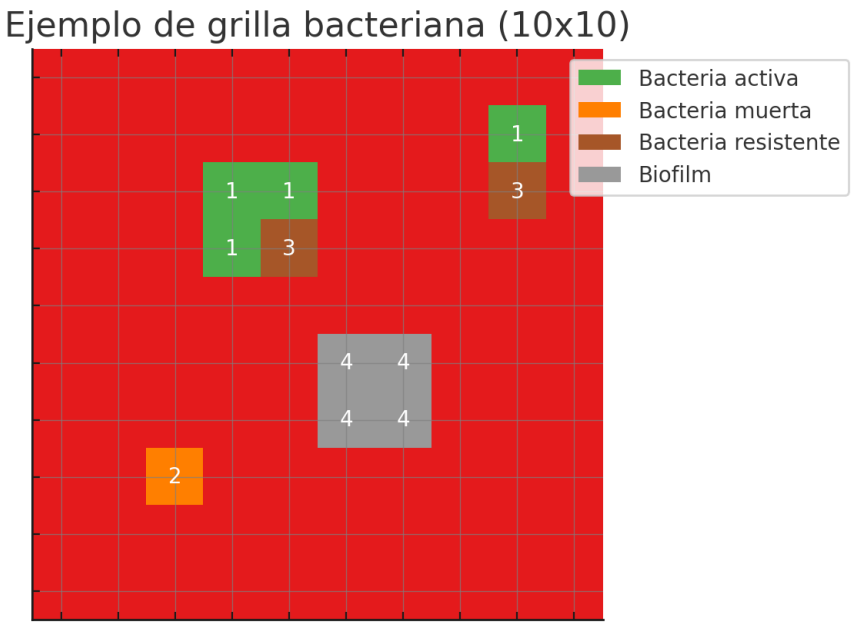

Figura 1: Ejemplo de grilla bacteriana.

Explicación de la Grilla

Esta grilla representa una placa de Petri de 10x10, donde cada celda puede contener o no una bacteria. Las bacterias pueden estar activas, muertas, ser resistentes o formar parte de un biofilm.

- **1 (Verde)**: Bacteria activa
- **2 (Naranja)**: Bacteria muerta
- **3 (Café)**: Bacteria resistente
- **4 (Gris)**: Zona de biofilm

Esta representación permite visualizar cómo se expande una colonia, cómo interactúan entre sí, y cómo ciertas zonas pueden dominarse por bacterias resistentes o morir por falta de nutrientes. Se recomienda generar visualizaciones periódicas para evaluar la evolución de la simulación.

```python
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.patches import Patch

# Crear una grilla 10x10
grilla = np.zeros((10, 10))

# Definimos posiciones de bacterias activas (1), muertas (2), resistentes (3), biofilm (4)
grilla[2, 3] = 1
grilla[2, 4] = 1
grilla[3, 3] = 1
grilla[3, 4] = 3  # resistente
grilla[5, 5] = 4  # biofilm
grilla[5, 6] = 4
grilla[6, 5] = 4
grilla[6, 6] = 4
grilla[7, 2] = 2  # muerta
grilla[1, 8] = 1
grilla[2, 8] = 3

# Crear un mapa de colores con 5 categorías (0 = vacío)
cmap = plt.cm.get_cmap('Set1', 5)

fig, ax = plt.subplots(figsize=(6, 6))
cax = ax.matshow(grilla, cmap=cmap)

# Agrega leyenda personalizada
legend_elements = [
    Patch(facecolor=cmap(1/5), label='Bacteria activa'),
    Patch(facecolor=cmap(2/5), label='Bacteria muerta'),
    Patch(facecolor=cmap(3/5), label='Bacteria resistente'),
    Patch(facecolor=cmap(4/5), label='Biofilm'),
]

ax.legend(handles=legend_elements, loc='upper right', bbox_to_anchor=(1.45, 1))

# Configuración de la grilla
ax.set_xticks(np.arange(0, 10, 1))
ax.set_yticks(np.arange(0, 10, 1))
ax.set_xticklabels([])
ax.set_yticklabels([])
ax.grid(color='gray', linestyle='-', linewidth=0.5)

# Mostrar valores en cada celda
for i in range(10):
    for j in range(10):
        val = grilla[i, j]
        if val > 0:
            ax.text(j, i, int(val), va='center', ha='center', color='white
```

```
    ')

plt.title("Ejemplo de grilla bacteriana (10x10)")
plt.tight_layout()
plt.show()
```

Listing 2: Ejemplo de código para generar el gráfico la grilla paso 2

Bibliotecas del ejemplo

- **matplotlib.pyplot**: Biblioteca para la creación de gráficos 2D. Se utiliza para dibujar la grilla, establecer etiquetas, mostrar el gráfico y agregar una leyenda visual.
- **numpy**: Biblioteca para cálculos numéricos y manipulación eficiente de arreglos. Se emplea para crear y modificar la matriz que representa el entorno simulado.
- **matplotlib.patches.Patch**: Componente usado para definir elementos visuales en la leyenda, permitiendo asignar colores específicos a los distintos estados de las bacterias.

Condiciones y entrega

- Miércoles 2 de julio: entrega final del proyecto.
- La entrega del proyecto está condicionada a la realización de una defensa oral individual obligatoria sobre el problema y el código entregado.
- Miércoles 18 de junio y miércoles 25 de junio: presentación de avances parciales.
- Se evaluarán rigurosamente los conceptos relacionados con la programación orientada a objetos (POO).
- Debe implementar el concepto de API para ajustar una interfaz gráfica.
- Puede integrar las bibliotecas que estime conveniente.
- Se espera que la solución propuesta refleje creatividad, precisión técnica e innovación en el diseño e implementación. La propuesta debe demostrar que se pensó en dar una solución que sea capaz de integrar adecuadamente los conceptos del paradigma orientado a objetos con el contexto biológico del problema. Desde el punto de vista lógico, se valorarán aquellas soluciones que permitan interpretar y explicar el comportamiento del sistema más allá de clasificar respuestas como correctas o incorrectas; es decir, se privilegiarán enfoques que entreguen análisis y justificación desde el resultado observado.