Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 — Estructuras de Datos y Algoritmos
2019- 1

Tarea 0

Fecha de entrega código: Domingo 31 de Marzo
Fecha de entrega informe: Martes 2 de Abril

Objetivos

- Familiarizar al alumno con el lenguaje de programación C
- Implementar y analizar un algoritmo básico.

Introducción

Los investigadores del centro de estudios biológicos INCUBACT desean simular el la interacción entre un grupo de 3 bacterias que estudian arduamente, para lo cual te han contactado. Deberás escribir un programa en C que a partir de un estado inicial de la comunidad bacteriana puedan predecir la configuración de las siguientes generaciones. Para mostrar estos resultados te han provisto de una interfaz gráfica con la que te tendrás que comunicar para mostrar la evolución del ecosistema.

Mundo

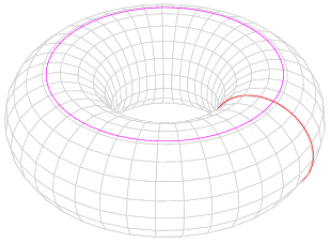


Figura 1: Toro

El ambiente en que se habitan las bacterias corresponde a un toro (un ejemplo de esta superficie se muestra en la Figura 1), el cual visualmente se representará a través de una cuadrícula de dimensiones M x N de tal forma que la siguiente columna luego de la última es la primera columna y la siguiente fila luego de la última fila es la primera fila.

Cada celda de la cuadrícula habita uno de los tres tipos de bacterias, apodados según su comportamiento:

- **Bacteria "Piedra"**: Derrota a las bacterias tipo Tijera. Pierde ante bacterias tipo Papel.
- **Bacteria "Papel"**: Derrota a las bacterias tipo Piedra. Pierde ante bacterias tipo Tijera.
- **Bacteria "Tijera"**: Derrota a las bacterias tipo Papel. Pierde ante bacterias tipo Piedra.

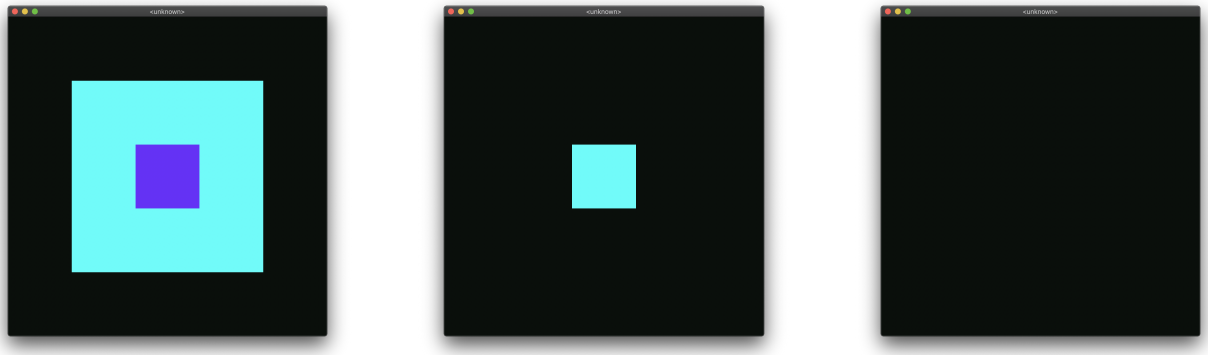
Cuando se hable de los "vecinos" de una celda de la cuadrícula se refiere a las 8 celdas con las que esta comparte vértices o aristas. [1]

Generaciones

La simulación funciona en base a generaciones, donde a partir de la generación actual se calcula el estado de la colonia en la siguiente generación. La regla es una sola:

Si en la generación i una bacteria se encuentra rodeada por 3 o más bacterias que la derrotan, en la generación $i+1$ la bacteria de esta celda pasa a ser del tipo que la derrotó

A continuación, un ejemplo:



Generación i Generación $i+1$ Generación $i+2$

Figura 2: bacterias y su evolución

En la generación i vemos el estado de la colonia: al centro una bacteria tipo piedra rodeada de bacterias tipo papel, y estas rodeadas a su vez de bacteria tipo tijera. La célula piedra al medio del tablero tiene 8 vecinos papel, por lo que en la siguiente generación será papel. Así mismo, todas las celdas papel tienen 3 o más vecinos del tipo tijera, por lo que para la siguiente generación pasan a ser del tipo tijera. En la generación $i+1$ vemos que al centro hay una bacteria de tipo papel, rodeada completamente por bacterias de tipo tijera, por lo que para la siguiente generación esta también se transformaría en tijera.

[1] También conocida como la "Vecindad de Moore"

Interfaz Gráfica

Como se mencionó anteriormente, los investigadores ya cuentan con una Interfaz Gráfica implementada para poder visualizar la evolución del ecosistema de bacterias. Los detalles de los métodos disponibles se dan a continuación:

- `watcher_open(height, width)`: Abre una ventana con una grilla que contiene `height` filas y `width` columnas.
- `watcher_set_cell(row, col, state)`: Modifica para la siguiente generación el estado de la celda en la fila `row` y columna `col` según el `int state` donde `state` puede ser 0 = Piedra, 1 = Papel o 2 = Tijera.
- `watcher_refresh()`: Actualiza la imagen de la interfaz. Hace efectivas las modificaciones hechas a la grilla desde la última llamada a esta función. Llama a esta función luego de actualizar el estado de cada celda para la siguiente generación.
- `watcher_close()`: Cierra la ventana.

Ejecución

Tu programa debe ejecutarse de la siguiente forma:

```
./simulate <test.txt> <número de iteraciones>
```

Donde `<test.txt>` corresponde al archivo con el tablero inicial a resolver t `<número de iteraciones>` corresponde a la cantidad de generaciones que se quieren simular.

Input

Los archivos de prueba contienen el siguiente formato:

- La primera línea contiene los números `M` y `N`, separados por un espacio, que indican la cantidad de filas y columnas que tendrá la grilla, respectivamente.
- Cada una de las siguientes `M` líneas corresponderán a `N` números; 0, 1 y 2 separados por espacios que denotan los estados de las celdas, donde 0 es una bacteria tipo piedra, 1 es una bacteria tipo papel y 2 es una bacteria tipo tijera.

Output

El único *output* de tu programa debe ser la visualización del mundo en la ventana, por lo cual debes asegurarte que tanto los resultados intermedios como el resultado final se muestren correctamente. Debes utilizar la función `watcher_refresh()` solo una vez por generación.

Informe

Deberás entregar un informe donde hagas un análisis teórico y empírico de la complejidad de la simulación y como esta varía con la cantidad de celdas de la grilla. Utiliza gráficos para contrastar la teoría con la practica y así apoyar tu análisis.

Evaluación

Tu código será evaluado con tests aleatorios distintos a los entregados como ejemplo (ver Anexo). Intenta que tu solución sea eficiente: tu programa tendrá como tiempo límite tantos segundos como generaciones tengas que calcular. Si se demora más será cortado y tendrás 0 puntos en ese test.

La nota de tu tarea se descompone como se detalla a continuación:

- 80% A la nota de tu código:
  - 60% que el output de tu programa sea correcto
  - 10% a que `valgrind` diga que tu programa no tiene errores de memoria
  - 10% a que `valgrind` diga que tu programa no tiene *memory leaks*
- 20% A la nota de tu informe

Entrega

- **Código:** GIT - Repositorio asignado (asegúrate de seguir la estructura inicial de éste).
  - En la carpeta *Programa* debe encontrarse el código.
  - Se recogerá el estado en la rama *master*.
  - Se espera que el código compile con `make` dentro de la carpeta *Programa* y genere un ejecutable de nombre `simulate` en esa misma carpeta.
- **Informe:** SIDING - En la encuesta correspondiente, en formato PDF
- **Hora Límite:** 1 minuto pasadas las 23:59 del día de la entrega.
- No se permitirán entregas atrasadas.

Anexo: Generador

Además de los tests que vienen en tu repositorio, puedes usar el generador para probar tu programa con tests aleatorios o para crear tus tests de distintos tamaños para el análisis empírico del informe.

Este recibe 4 parámetros:

1. La ruta al archivo donde se guardará el test generado
2. La cantidad de filas que deberá tener la cuadrícula
3. La cantidad de columnas que deberá tener la cuadrícula
4. La semilla aleatoria, para poder replicar resultados

Al momento de corregir generaremos tests nuevos con una semilla distinta, los que tendrán los mismos tamaños que los tests que se encuentran actualmente en sus repositorios.