Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2133 - Estructuras de Datos y Algoritmos

Informe Tarea 0

1 de abril de 2019

1º semestre 2019 - Profesor Yadran Eterovic

Paul Heinsohn Manetti - 1562305J

Implementación

Por temas de legibilidad y simplicidad de código decidí abordar la simulación mediante mi propia estructura de datos. Esta consiste en una variable llamada 'table' del tipo 'Node***' que funciona como una matriz de nodos, cuyas filas son del tipo 'Node**' y columnas del tipo 'Node*'. Gracias a lo anterior, pude acceder a estos. Cada nodo apunta a dos variables: 'state' y 'neighbours'. La primera es un número entero (int), mientras que la segunda es un puntero de punteros del mismo tipo (Node**). La idea de 'state' es almacenar el tipo de bacteria presente dicho nodo, mientras que 'neighbours' para guardar los nodos vecinos.

Esta modelación me permite realizar una iteración en base a las filas 'i' y otra en base a las columnas 'j' para acceder a los nodos mediante la sentencia 'table[i][j]', que a su vez facilita la asignación y el acceso a los vecinos para realizar los cambios pertinentes.

Complejidad

En cuanto al algoritmo ejecutado en cada generación, tenemos que en su mejor caso se realizan $3 \cdot M \cdot N$ comparaciones, ya que basta que cada nodo sea comparado con otros tres que le ganen para cambiarlo de estado. En cuanto a los otros casos, es posible apreciar que realizan la misma cantidad de comparaciones debido a que la probabilidad de que un nodo le gane a otro es de $1/3$, y como se requieren tres para cambiarlo de estado, se tiene que en el caso promedio tres de cada ocho nodos van a ganarle al actual, y como $8/3 < 3$, necesariamente se van a realizar las ocho comparaciones al igual que en el peor caso, ergo, ambos realizan $8 \cdot M \cdot N$ comparaciones.

Es por lo anterior que el algoritmo tiene complejidad O(M·N) en cualquier caso, lo que inicialmente uno tiende a pensar que no está tan mal, sin embargo, a medida de que iba probando los tests de mayor tamaño, más se demoraba la ejecución de cada generación. Entrando en detalle, es posible apreciar que el primer (test_00), cuya matriz es de 5x5, tarda en llegar a la generación de un solo tipo de bacteria (3 en adelante) en menos de medio segundo, mientras que al ejecutar los últimos dos testes (test_06 y test_07) demoraba más de un segundo por generación.

Tabla 1: Complejidad en casos

Caso	Comparaciones	Complejidad
Peor	$8 \cdot M \cdot N$	$O(M \cdot N)$
Promedio	$8 \cdot M \cdot N$	$O(M \cdot N)$
Mejor	$3 \cdot M \cdot N$	$O(M \cdot N)$

Lo anterior puede explicarse tanto por el algoritmo como por las estructuras empleadas. Seguramente, mi algoritmo puede ser modificado para que la cantidad de comparaciones disminuya, pero creo que lo que más afecta el funcionamiento de éste es el uso de estructuras, ya que mi variable 'table' podría ser solamente del tipo 'int**', la cual mediante el correcto uso de punteros evita la múltiple 'instanciación' de nodos, por lo que disminuirían los accesos a la memoria haciendo más fluido el transcurso de las generaciones. Finalmente, me gustaría destacar que la Implementación de 'int' en vez de 'uint_8' también influyó en el desempeño del algoritmo, ya que se reservó más memoria y se terminó leyendo innecesariamente bloques que no aportaban información relevante.