Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2022.

PROYECTO #1: ANALIZADOR DE COMPORTAMIENTO DE CELULAS

202100953 – Damaris Julizza Muralles Véliz

Resumen

El tema que se busca exponer al realizar este ensayo y proyecto es el manejo de TDA en estructuras de datos, el cual está estrechamente relacionado con la programación orientada a objetos y el manejo de datos.

A través de la resolución al problema planteado para el desarrollo del proyecto, se puede observar que el uso de TDA en la programación puede ser eficiente para manejar una gran cantidad de datos y por lo tanto se vuelve una buena opción para ser implementada en el desarrollo de software que necesite manejar mucha información.

Palabras clave
TDA, POO, Estructuras de datos, Graficas.

Abstract

The topic that is sought to be exposed when carrying out this essay and project is the management of ADT in data structures, which is closely related to object-oriented programming and data management.

Through the resolution of the problem posed for the development of the project, it can be seen that the use of ADT in programming can be efficient to handle a large amount of data and therefore it becomes a good option to be implemented in development. of software that needs to handle a lot of information.

Keywords
ADT, OOP, Data estructures, Graphs.

Introducción

Un Tipo de dato abstracto (TDA), es un conjunto de datos u objetos al cual se le asocian operaciones y un mismo TDA puede ser implementado en distintas estructuras de datos proveedor la misma funcionalidad. (CC30A Algoritmos y Estructuras de Datos: Tipos de datos abstractos, s. f.)

Este ensayo tiene el propósito explicar el desarrollo de la solución al problema propuesto en donde se implementa TDA, estructura de datos y programación orientada a objetos (POO).

Se propuso el desarrollo de un software que permita observar el comportamiento de células contagiadas de acuerdo con los patrones propuestos y determinar mediante esto si la enfermedad será leve, grave o mortal.

Se utilizará estructuras de datos para guardar los patrones proporcionados y estos serán presentados en forma gráfica representando una rejilla cuadrada que servirá como medio de simulación del comportamiento de las células contagiadas y sanas, así como la forma de infección y expansión de estas para finalmente determinar el tipo de enfermedad provocada.

Desarrollo del proyecto

En el proyecto se realizó por medio de POO, en donde se implementaron distintas clases y métodos para interactuar y modificar los datos necesarios en el programa.

En el problema planteado se nos hace referencia a que unos investigadores han encontrado un patrón relacionado con el comportamiento de enfermedades mediante el comportamiento de las células contagiadas al momento de infectar a otras y su expansión. Por lo que se pide que el programa sea capaz de cargar al sistema los datos desde un archivo XML que tendrá la información necesaria de los pacientes a diagnosticar y posteriormente poder manejar esos datos de acuerdo a lo necesario para poder proporcionar una gráfica donde se visualice el comportamiento de las células contagiadas y luego generar un archivo XML con la información correspondiente al diagnóstico de la enfermedad, sea esta leve, grave o mortal y el periodo en donde se encontró la coincidencia de los patrones.

Tomando en cuenta lo anterior, en la solución del problema planteado se hizo uso de estructuras, específicamente se implementaron listas doblemente enlazadas, que es creada a partir de nodos los cuales tienen dos enlaces asociados que permite moverse hacia adelante y hacia atrás en la lista para manejar la información necesaria. Como en toda estructura de datos esta estructura implementada puede contener distintos métodos que permitan el manejo de los datos contenidos en ellos, como lo es el insertar información, eliminar, mostrar, etc. (Estructura de Datos : Lista Enlazada Doble, s. f.).

Figura 1. Representación de listas doblemente enlazadas.

Fuente: elaboración propia.

Para desarrollo del programa se modelo un diagrama de clases de manera que se pudiera tener una idea de

la forma en que se estructuraría el programa realizado y las relaciones que tendría.

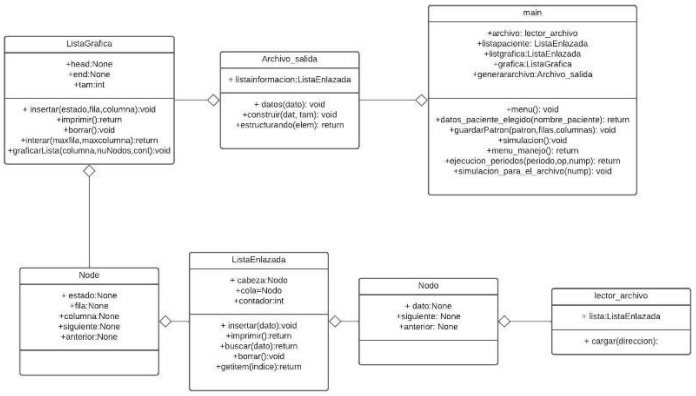

Figura 2. Diagrama de clases.

Fuente: elaboración propia.

En el diagrama de clases que modela la solución del problema podemos observar, como se mencionaba al inicio, que se utilizan distintas clases con distintos atributos y métodos. Las clases que se por ponen son las siguientes:

a. Lector_archivo
b. Nodo
c. ListaEnlazada
d. Node
e. ListaGrafica
f. Archivo_salida
g. Main

Clase Lector_archivo

Esta clase tiene como propósito leer el archivo que sea proporcionado y cargar la información contenida a una lista enlazada que permitirá manejar toda la información mientras el programa se encuentre en ejecución.

Debido a su propósito se hizo uso de la librería minidom de Python la cual proporciona métodos más amigables al momento de leer archivos.

Esta clase esta diseñada para requerir un atributo el cual será la dirección en donde se ubica el archivo a leer y procesar, este dato será pedido por medio de consola desde la clase main quien se encargará de llamar a este método y proporcionarle el dato necesario.

Clase Nodo

Esta clase tiene una relación de composición con la clase ListaEnlazada y tendrá como atributos a dato, siguiente y anterior, los últimos dos son los encargados de manejar los datos permitiendo ver los datos que contienen los nodos anteriores y posteriores.

Clase ListaEnlazada

Aquí podemos observar la lógica que hace posible la creación de la lista doblemente enlazada, esta clase contiene atributos denominados cabeza y cola los cuales serán de tipo nodo, aparte de ellos también tendremos como atributo a la variable contador que será de tipo int el cual manejara la información sobre el tamaño de la lista.

Los métodos que se implementaron en esta clase son el de insertar, que permite agregar un nuevo nodo con información, el de imprimir, que permite recorrer la lista y mostrar la información contenida en todos los nodos, el de borrar, que permite borrar toda la

información en la lista, el método __getitem__, que permite que la estructura tenga una función parecida a las listas nativas de Python, es decir, que podemos obtener la información de los nodos por medio de un índice y el método de buscar, este método está enfocado de manera específica a buscar un dato/patrón dado y comparar con los datos que ya se encuentran en la lista para posteriormente retornar una respuesta.

Clase Node

Esta clase cumple la misma función que la clase nodo y sus atributos son casi similares solo que en lugar del atributo dato, esta clase posee tres atributos bajo el nombre de estado, fila y columna los cuales contendrán la información relacionada a si las células están contagiadas o no y en que fila y columna están posicionadas.

Clase ListaGrafica

Al igual que la clase ListaEnlazada esta estructura es una de tipo lista doblemente enlazada y contiene métodos similares a la clase antes mencionada, sin embargo, esta posee otros dos métodos importantes que están específicamente diseñados para el manejo de los patrones generados en representación de las células en las rejillas para generar nuevos patrones y representarlos gráficamente.

Uno de estos métodos es el método graficarLista el cual necesita los atributos de columna, nuNodos y cont los cuales ayudaran a manejar la informacion de la manera adecuada para poder ser graficados, aquí se hace el uso de la librería graphviz en específico su método diagraph para modelar una gráfica en simulación de una tabla de m*m, este dato m que representa el tamaño se proporciona desde el archivo xml cargado al inicio.

Es necesario tomar en cuenta que este método no genera nuestra grafica directamente, si no que antes construimos un archivo .dot que contiene las relaciones entre los nodos, el tipo de figura utilizada, colores, etc. y convertiremos este archivo a ,svg para que pueda ser abierto desde Edge, haciendo que cada grafica pueda visualizarse inmediatamente una vez que fue creada.

El otro método es interar, este permite navegar entre los datos guardados previamente en todos los nodos de la estructura y hacer un análisis de el nodo, que representa a la célula, estará contagiado o no en el siguiente periodo dependiendo de cuantas células contagiadas estén a su alrededor.

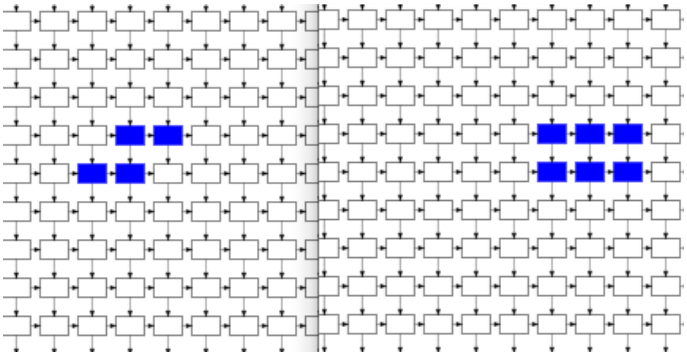

Figura 3. Grafica de periodos.

Fuente: elaboración propia.

La lógica utilizada en este método está basada en crear dos variables de tipo nodo uno actual y uno temporal las cuales se van a recorrer de forma anidada y mediante condiciones se hará comparación para identificar si las células alrededor del nodo analizado

cumplen con las condiciones para que la célula se contagie o siga contagiada en el siguiente periodo.

La variable actual llevara el control del nodo que necesita ser analizado y el temporal se encargara de recorrer todos los nodos desde el principio hasta el fin cada vez que analicemos un nuevo nodo de la variable actual. Las condicionales también estarán dentro de nuestro segundo ciclo para temporal y se debe tomar en cuenta que existen un total de 9 condiciones a tomar en cuenta debido a los puntos críticos que se deben evaluar debido a la rejilla.

La rejilla prácticamente es como una tabla cuadrada entonces tendremos como puntos críticos al evaluar un nodo que este en alguna esquina, cuando se encuentre en la primera fila o columna, cuando se encuentre en la última columna, cuando se encuentre en la última fila o bien cuando no se encuentre en ninguna de estas zonas criticas y sea posible evaluar tanto los datos que se encuentre en la fila anterior como en la posterior y diagonales al nodo actual.

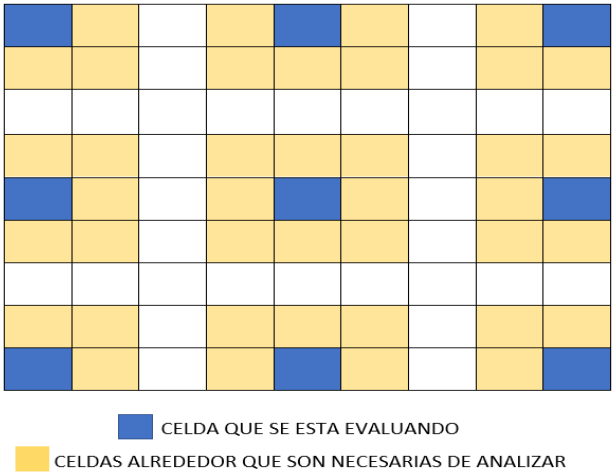

Figura 4. Representación de posibles nodos a analizar.

Fuente: elaboración propia.

Clase Archivo_salida

Esta clase tiene el propósito de construir un archivo xml con los resultados después de interar los patrones las n cantidad de veces especificadas en el archivo de entrada y por lo tanto a parte de los datos esenciales del paciente como nombre y edad, necesita contener el tamaño de rejilla número de interacciones, si la enfermedad es leve, grave o mortal que dependerá si los patrones de rejilla se repiten y en que periodos se presenta esta coincidencia en los patrones. Toda la información anteriormente descrita es proporcionada por una lista enlazada que contiene esta información.

Para construir y crear el archivo se utilizó ElementTree y se realizo un “Prettyprint” con ayuda de minidom para darle una estructuración más agradable a la vista.

Clase main

En esta clase se programa toda la lógica detrás del programa en general y prácticamente cumple la función de una clase controladora que maneja las llamadas de las otras clases y el manejo de información que se debe de pasar a estas.

Contiene varios métodos de tipo menú que nos permiten entrar a distintas opciones de ejecución en el programa, de ejemplo el leer un archivo, ejecutando la clase de lector_archivo.

Se hace uso de la librería sys, re y time, sys para detener el programa, re para encontrar simular los split que es un método para listas nativas, cumpliendo así el requisito de no involucrar ninguna lista nativa y

time para detener la ejecución de un método por un tiempo determinado haciendo uso del método sleep.

En el constructor de esta clase hacemos instancia a todas nuestras estructuras para tener un control sobre ellas desde esta clase.

Conclusiones

1. La implementación de TDA en las estructuras de datos hace que el manejo de información en el programa sea más eficiente.

2. Un mal manejo de la memoria dinámica al momento de implementar estructuras de dato puede provocar enciclamientos y un stack overflow.

Referencias bibliográficas

CC30A Algoritmos y Estructuras de Datos: Tipos de datos abstractos. (s. f.). Recuperado 6 de septiembre de 2022, de https://users.dcc.uchile.cl/%7Ebebustos/apuntes/cc30a/TDA/

colaboradores de Wikipedia. (2022, 2 septiembre). Programación orientada a objetos. Wikipedia, la enciclopedia libre. Recuperado 2 de septiembre de 2022, de https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos

Estructura de Datos : Lista Enlazada Doble. (s. f.). Recuperado 6 de septiembre de 2022, de https://www.fceia.unr.edu.ar/estruc/2005/listendo.htm

Apéndices

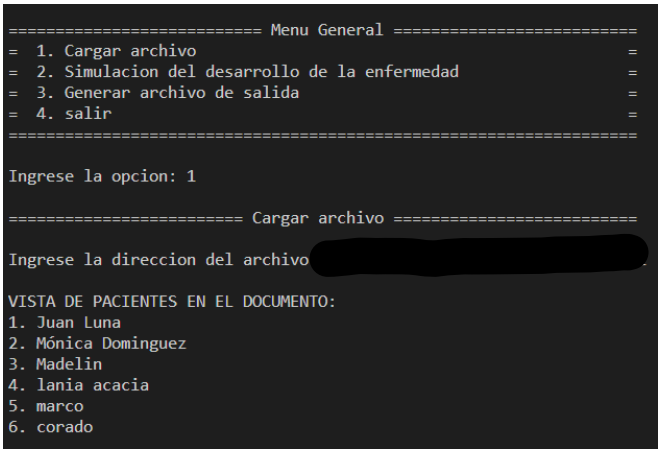

Figura 5. Menú general y carga de datos.

Fuente: elaboración propia.

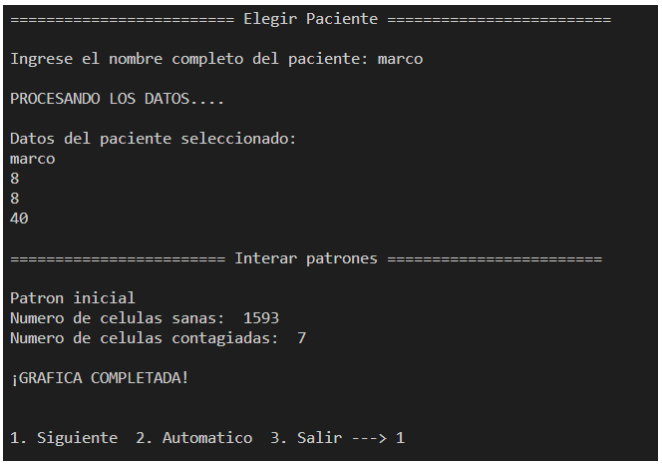

Figura 6. Elección de un paciente y vista de los periodos.

Fuente: elaboración propia.