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 2025

TAPLICACIÓN DE TDA, POO, XML Y GRAPHVIZ EN IPC2: ANÁLISIS DEL PROYECTO GUATERIEGOS 2.0 Y LA OPTIMIZACIÓN DE ESTAIONES BASE

202405828 – Danny Josué González Lémus

Resumen

Este ensayo analiza dos insumos complementarios del curso IPC2: (1) el proyecto "GuateRiegos 2.0", una implementación en Python que integra Programación Orientada a Objetos (POO), Tipos de Datos Abstractos (TDA) propios, lectura/escritura de XML, generación de reportes y visualización con Graphviz; y (2) el enunciado oficial que describe un problema NP■Hard de distribución de estaciones base mediante agrupamiento por patrones de matrices de frecuencias. Se contrasta el enfoque de simulación de riego con drones —que modela invernaderos, hileras, plantas y planes— con la metodología de reducción de estaciones base basada en patrones idénticos (Fp y Fr). A partir de esta comparación, se extraen principios de diseño transferibles: construcción de TDA sin estructuras nativas, separación de responsabilidades (parsing, dominio, simulación, visualización), y reportabilidad. El trabajo concluye con una propuesta para adaptar el andamiaje de GuateRiegos 2.0 al problema de estaciones, destacando beneficios técnicos y

Palabras clave

POO, TDA; XML, Graphviz; Optimización

Abstract

This essay examines two complementary IPC2 inputs: (1) the "GuateRiegos 2.0" project, a Python implementation that integrates Object■Oriented Programming (OOP), custom Abstract Data Types (ADTs), XML I/O, reporting
and Graphviz visualization; and (2) the official statement describing an NP■Hard problem for base■station distribution via frequency■matrix pattern clustering. We contrast the irrigation■by■drone simulation—modeling greenhouses, rows, plants and plans—with the station■reduction methodology based on identical patterns (Fp and Fr). From this comparison we derive transferrable design principles: building ADTs without native structures, clear separation of concerns (parsing, domain, simulation, visualization), and reportability. We conclude with a proposal to adapt the GuateRiegos 2.0 scaffolding to the stations problem, highlighting technical and educational benefits.

Keywords

OOP; ADT; XML; Graphviz; Optimization

Introducción

El presente documento ofrece un panorama de dos artefactos clave en IPC2. Por un lado, GuateRiegos 2.0 materializa POO y TDA propios en una solución web que parsea XML, simula planes de riego con drones y genera reportes y grafos de estado. Por otro, el enunciado académico plantea la optimización de estaciones base mediante el agrupamiento de patrones en matrices de frecuencias, integrando XML y Graphviz. El propósito es identificar convergencias de ingeniería —abstracciones, estructuras y visualización— que permitan transferir aprendizajes entre ambos contextos. La contribución principal es una guía de adaptación: cómo reutilizar el esqueleto de GuateRiegos 2.0 (capas TDA/domino/parsing/visualización) para resolver el problema NP■Hard de estaciones, manteniendo la restricción didáctica de no usar estructuras nativas y conservando la trazabilidad requerida por el curso.

Desarrollo del tema

a) Marco del problema y requisitos curriculares. El enunciado oficial exige una solución con POO, TDAs creados por el estudiante, entrada/salida en XML, graficación con Graphviz y menú con operaciones de carga, procesamiento, escritura, datos del estudiante y reporte gráfico. La esencia técnica es transformar las matrices de frecuencias F[n,s] y F[n,t] en patrones Fp y agrupar filas equivalentes para obtener matrices reducidas Fr, con la consiguiente disminución de estaciones base. La restricción de no utilizar listas, diccionarios o tuplas nativas orienta a diseñar listas enlazadas, colas y pilas propias, sobre las que se construyen los algoritmos.

b) Análisis de la implementación GuateRiegos 2.0. El proyecto organiza el código en capas: tda (Lista/Nodo), domain (Invernadero, Hilera, Dron, Planta, PlanRiego), core (Scheduler/Simulador), xmlio (Parser/Writer) y viz (Graphviz). La web en Flask orquesta la carga de XML, la simulación y la generación de reportes. Destaca el uso consistente de la ListaEnlazada en lugar de estructuras nativas, la separación de responsabilidades y un renderizador Graphviz que valida la presencia de dot en el PATH y produce grafos de estado de los TDA tras cada simulación.

c) Transferencia de aprendizajes al problema de estaciones. El andamiaje anterior se adapta naturalmente al problema de agrupamiento de estaciones base: (1) xmlio define lectores/escritores para las matrices F[n,s] y F[n,t]; (2) domain modela Estación, SensorS, SensorT y Campo; (3) tda aporta listas y colas para recorrer estaciones y acumular frecuencias; (4) core implementa el algoritmo de construcción de patrones (binarios o multi■conjunto) y la fusión de filas idénticas; y (5) viz muestra F, Fp y Fr con Graphviz, permitiendo validar visualmente la reducción. Este mapeo favorece la mantenibilidad y la evaluación por rúbrica.

d) Resultados esperados y mejora continua. La arquitectura orientada a TDA facilita pruebas por unidad y de sistema: lectura XML, generación de planes/estados y consistencia de grafos. Para estaciones base, las métricas clave son el número de estaciones reducidas, la corrección de frecuencias acumuladas y la legibilidad del grafo. Se recomiendan pruebas smoke automáticas, validación de

Es conveniente describir brevemente el contenido de una tabla, evitando los aspectos obvios.

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 2025

Criterio	Enunciado (Estaciones)	GuateRiegos 2.0	Observación
POO + TDA propios	Obligatorio	Implementado (Lista/Nodo)	Conforme
XML I/O	Entrada y salida	Parser/Writer XML	Conforme
Graphviz	F, Fp o Fr	Grafo de estado TDA	Transferible
Menú/Interfaz	Consola requerida	Interfaz web Flask	Ajustar a consola o justificar

Tabla I. Matriz de cumplimiento entre requisitos y la implementación disponible. Fuente: elaboración propia.

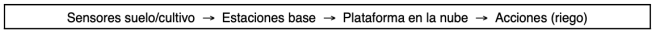

Figura 1. Flujo conceptual entre sensores, estaciones, nube y acciones. Fuente: elaboración propia.

Conclusiones

• Los TDA propios y la separación de capas demuestran ser reutilizables entre dominios distintos.
• El patrón XML → Dominio → Núcleo → Visualización acelera el desarrollo y la evaluación por rúbrica.
• La metodología de patrones (Fp/Fr) es compatible con el andamiaje de GuateRiegos 2.0 con cambios mínimos.
• Para cumplir estrictamente el enunciado de estaciones, se sugiere añadir un menú de consola y reportes específicos.
• La visualización con Graphviz incrementa la trazabilidad y la verificación de resultados.

Referencias bibliográficas

1. Universidad de San Carlos de Guatemala (2025). Enunciado del Proyecto 1 – IPC2 (matrices F, Fp y Fr, Graphviz, XML).
2. Proyecto GuateRiegos 2.0 (2025). Implementación en Python con POO, TDA, XML y Graphviz.
3. Graphviz (2025). Graph Visualization Software. Documentación técnica.
4. Flask (2025). Microframework de Python para aplicaciones web. Documentación oficial.