Universidad San Carlos de Guatemala
Centro Universitario de Occidente
División Ciencias de la Ingeniería
Ingeniería en Ciencias y Sistemas
Laboratorio de Estructura de Datos
Primer Semestre 2025

Proyecto: SISTEMA DE TRÁFICO

OBJETIVO GENERAL

Desarrollar un sistema de simulación de tráfico inteligente que gestione de manera eficiente la circulación de vehículos en una ciudad, priorizando vehículos especiales y aplicando algoritmos de análisis, búsqueda y ordenación a través de estructuras de datos implementadas desde cero.

OBJETIVOS ESPECÍFICOS

- Aplicar estructuras de datos lineales como listas, colas, pilas y árboles para representar el flujo vehicular y semáforos inteligentes.
- Utilizar funciones hash para el manejo de placas vehiculares y búsquedas eficientes.
- Implementar árboles AVL o B para calcular el nivel de dificultad de las rutas y coordinar intersecciones complejas.
- Comparar distintos algoritmos de ordenación aplicados al tráfico y reportes.
- Evaluar la eficiencia de los algoritmos aplicando notación BigO

DESCRIPCIÓN GENERAL

El sistema simulará un mapa urbano dividido en zonas y carriles con tráfico constante. Cada carril funciona como una cola en donde se encolan vehículos con diferentes niveles de prioridad. El sistema determina el orden de paso utilizando una cola de prioridad basada en tipo de vehículo, tiempo de espera y dirección.

Además, se almacenan intersecciones como nodos de un árbol AVL o árbol B, donde se define la complejidad o carga de cada ruta. También se implementa un sistema de historial que permite analizar el flujo vehicular, los tiempos de espera y los eventos importantes registrados en una pila de eventos.

Universidad San Carlos de Guatemala
Centro Universitario de Occidente
División Ciencias de la Ingeniería
Ingeniería en Ciencias y Sistemas
Laboratorio de Estructura de Datos
Primer Semestre 2025

ESPECIFICACIONES DEL SISTEMA

1. Estructura de Ciudad

- Representada con matrices ortogonales 2D o 3D de nodos conectados.
- Cada nodo representa una intersección, semáforo o cruce.
- Las rutas deben construirse con estructuras enlazadas, no arreglos planos.

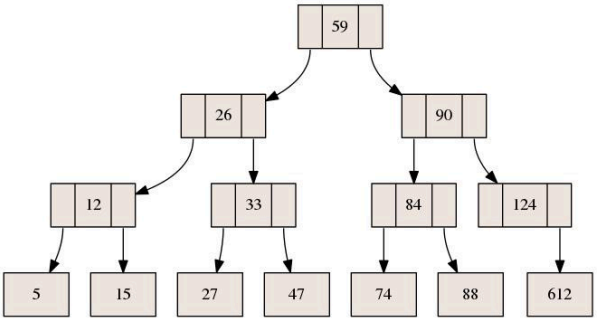
2. Vehículos y Prioridades

- Cada vehículo tendrá:
  - Tipo (Ambulancia, Policía, Transporte Público, Particular)
  - Placa única (hash)
  - Tiempo de espera
  - Nivel de urgencia
- Los vehículos se almacenan en una cola de prioridad manual (implementación tipo heap o lista ordenada).

3. Intersecciones y Árbol AVL/B

- Cada intersección se guarda como un nodo en un **árbol AVL** o **árbol B**.

- El valor del nodo representa el tráfico o complejidad (vehículos procesados, bloqueos, etc)
- Se permite búsqueda, inserción y reorganización de rutas en **O(log n)**
- Árboles reequilibran las rutas dinámicamente para encontrar la ruta más fluida.



4. Registro de Eventos en Pila

- Cada acción relevante se almacena en una pila:
  - Vehículo que cruzó
  - Prioridad atendida
  - Tiempo de paso
  - Cambios de semáforo o bloqueos
- Permite deshacer últimas acciones para análisis o simulación paso a paso.

5. Registro de Placas con Hash

- Uso de tabla hash implementada desde cero.
- Permite:
  - Búsqueda rápida de vehículos.
  - Evitar colisiones por placas duplicadas.
  - Consultas sobre vehículos que ya cruzaron o están en espera.

6. Ordenamiento y Clasificación

- Los reportes pueden ordenarse por:
  - Tiempo promedio de espera

  - Tipo de vehículo
  - Número de vehículos atendidos por carril
- Se deben implementar y comparar al menos 3 métodos de ordenamiento:
  - Inserción directa
  - QuickSort
  - Método de la sacudida

7. Simulación del tráfico

- El sistema aceptará vehículos de forma:
  - Manual (El sistema permitirá ingresar vehículos manualmente)
  - Por archivo (entrada controlada)
- El sistema decide qué vehículo pasa basándose en:
  - Nivel de prioridad
  - Tiempo de espera
  - Complejidad de la intersección (extraído del árbol)

8. Reportes de la Simulación

Al finalizar la simulación o en puntos de control preestablecidos, se generarán reportes que incluirán:

- Ranking de vehículos por nivel de prioridad y tiempo de cruce.
- Cantidad total de vehículos que cruzaron por cada carril.
- Tiempo promedio de espera por tipo de vehículo.
- Gráfico de intersecciones más congestionadas, basado en la información del árbol AVL/B. Este reporte incluirá:
  - Un recorrido del árbol impreso en consola (in-order, pre-order o post-order).
  - Un archivo gráfico (PNG o JPG) generado a partir del árbol.
- Lista de placas duplicadas o en conflicto, obtenidas desde la tabla hash.
- Registro de los últimos 20 eventos relevantes del sistema, almacenados en la pila.

9. Reportes de Estado en Tiempo Real

Durante la ejecución de la simulación, el sistema podrá generar reportes que muestren el estado actual del tráfico, tales como:

- Visualización en consola del contenido actual de cada cola, simulando el flujo vehicular por carril.
- Identificación del carril más congestionado o más fluido, calculado mediante el conteo de vehículos en espera.

10. Archivo de Entrada

El sistema leerá un archivo llamado trafico.csv con el siguiente formato por línea:

tipo_vehiculo,placa,interseccion_origen,interseccion_destino,prioridad,tiempo_espera

Ejemplo:

- AMBULANCIA,GUA-123A,A1,B3,5,3
- POLICIA,GUA-222B,A1,A2,4,5
- PARTICULAR,GUA-789C,B1,C1,2,8
- TRANSPORTE,GUA-333T,C3,B3,3,4
- AMBULANCIA,GUA-444A,C2,D1,5,2
- PARTICULAR,GUA-555P,A2,A4,1,9
- POLICIA,GUA-666P,B1,A1,4,6
- TRANSPORTE,GUA-777T,C1,D2,3,5
- PARTICULAR,GUA-888X,D1,C3,1,10

Cada campo se interpreta como:

- tipo_vehículo: AMBULANCIA, POLICIA, PARTICULAR, TRANSPORTE
- placa: identificador único (clave para hash)
- interseccion_origen/destino: nodos de la matriz de ciudad
- prioridad: número de 1 (baja) a 5 (alta)
- tiempo_espera: tiempo acumulado en cola

Importante

- Lenguaje a utilizar Java.
- El programa debe ejecutarse en consola.
- Las malas prácticas de desarrollo serán penalizadas.
- Las copias obtendrán nota de cero y se notificará a coordinación.
- Se requiere la creación de un archivo JAR ejecutable, para obtener el compilado del programa.
- Documentación Técnica:
    - Se debe incluir la complejidad de cada algoritmo utilizado en el proyecto.
    - La notación BigO debe justificarse adecuadamente para cada implementación.
- Estructuras de Datos Implementadas desde Cero:
    - Los árboles, listas enlazadas y tablas hash deben ser implementados por los estudiantes desde cero.
- Prohibición de Librerías de Estructuras de Datos:
    - No está permitido utilizar librerías o herramientas externas que proporcionen estructuras de datos como listas enlazadas, árboles, etc.
- Optimización y Análisis de Algoritmos:
    - Se deben comparar diferentes métodos de ordenación para determinar cuál es el más adecuado para cada caso.
    - Se debe analizar el impacto de las estructuras de datos utilizadas en la biblioteca.

Entrega

La fecha de entrega es el día <FECHA>. Los componentes se entregarán en un repositorio de Git a través Classroom, siendo estos:

- Código fuente.
- Documentación técnica.
- Manual de Usuario.

Nota

El repositorio debe ser privado, sin embargo debe de incluir los siguientes usuarios como colaboradores: **LuisCif-20, David15Barrera, daaaniel6** para la calificación correspondiente, estos deben ser agregados antes de la fecha y hora de entrega.