Proyecto Semáforo

Estructura de Datos

Martínez Rivera Luis Fernando

Juárez Morales Salvador

Vázquez Almanza Emanuel

Problema por modelar

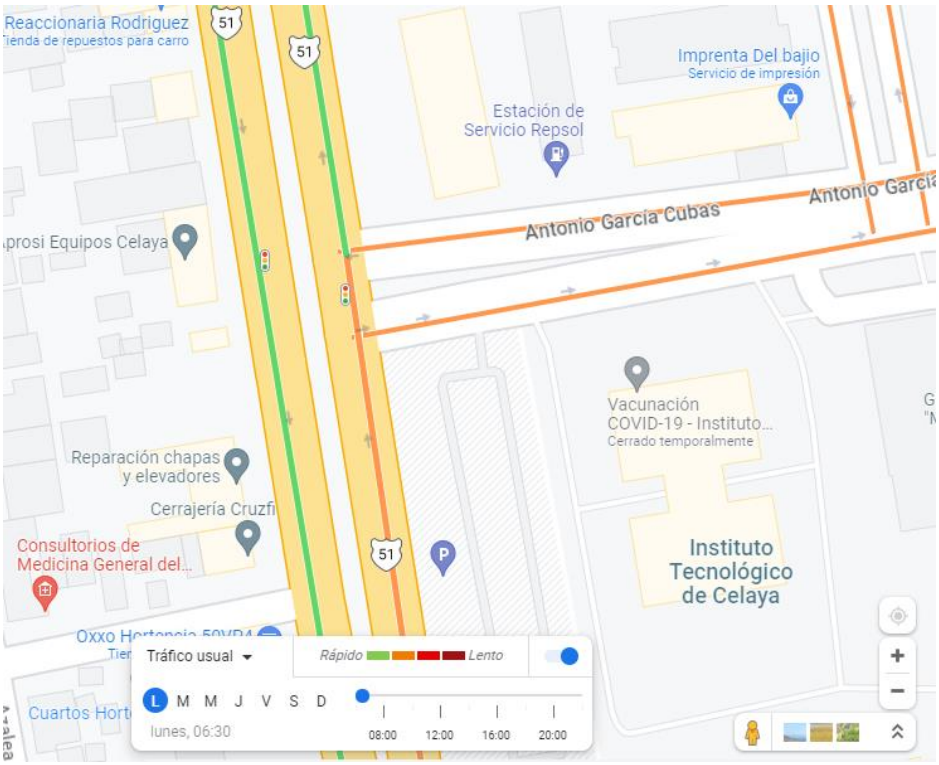
El problema por resolver es el embotellamiento de tráfico en la esquina de la Av. Tecnológico y la esquina de Antonio Garcia Cubas.

Este cruce posee una serie de elementos que afectan de manera negativa el flujo de tráfico en ciertas franjas horarias que se explicarán a continuación.

Factores originan el problema:

- Cruce peatonal en la esquina
- Semáforo con mala temporización
- Botones en el cruce peatonal que no controla el semáforo de carros (efecto placebo, no sirve de nada el botón)
- 3 carriles principales
- Ciclovía
- Parada de autobuses
- Puente peatonal a 100m (que no se usa) que congestiona aún más el tráfico
- Franjas horarias

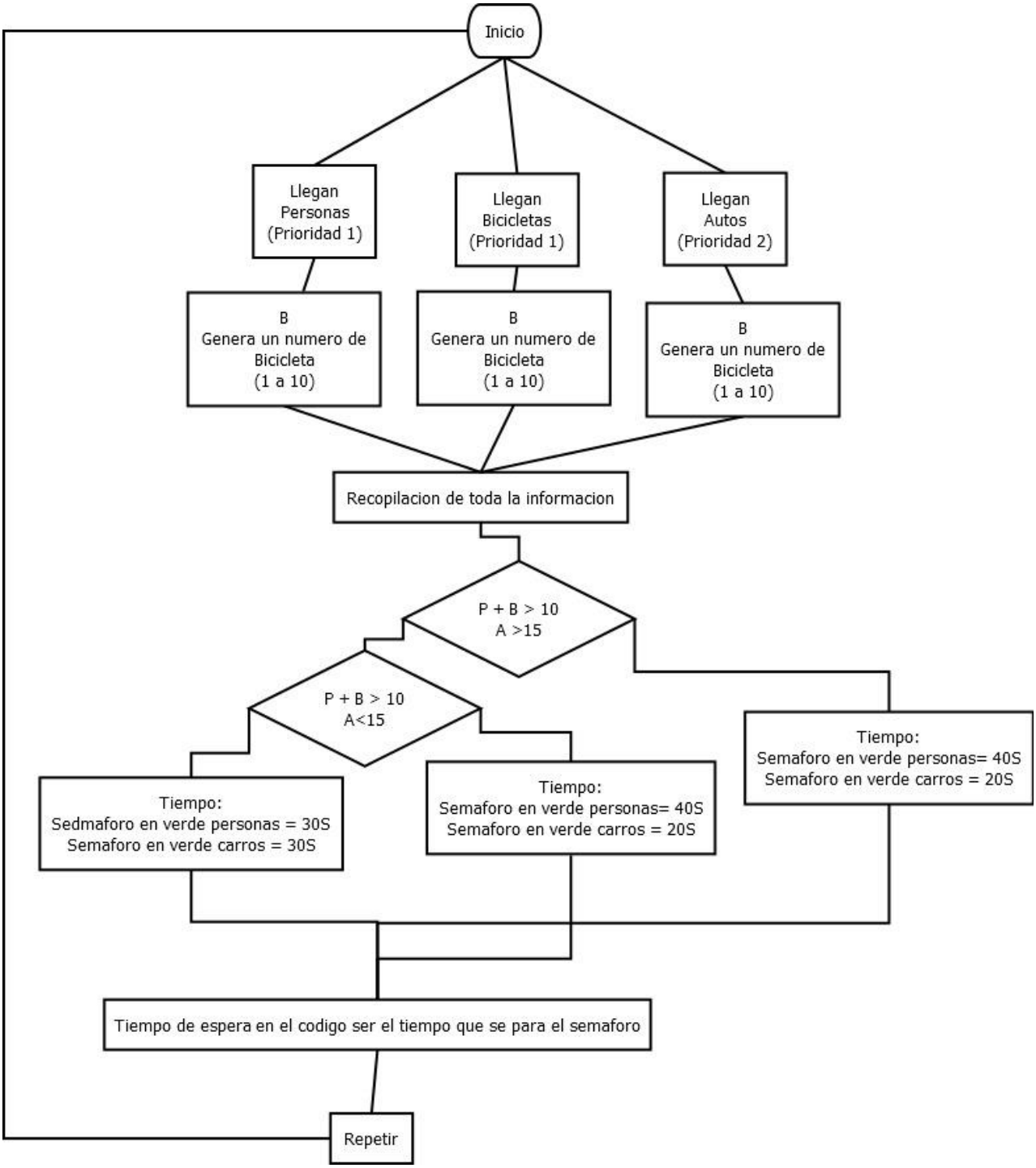
Para este fenómeno nos basamos en analizar como se comporta el tráfico gracias a los datos extraidos de google maps.



Como podemos ver, el tráfico usual está muy congestionado en general los días entre semana en una franja horaria de 6:30-7:00, 9:00-9:30, 14:30-15:00 y 18:45-19:30

Diagrama de flujo

Una vez teniendo claro nuestro problema, hemos propuesto este diagrama de flujo el cual va a estar basado nuestro modelo en Python para optimizar el flujo de tráfico.



Estas decisiones fueron previamente analizadas y planteadas de acuerdo con

condiciones reales del cruce: longitud desde el puente peatonal al semáforo, longitud de la parada hasta ciclistas y acumulación de personas.

Vista aérea con condiciones:

Repositorio del proyecto

Adicionalmente una de las tecnologías que utilizamos para este proyecto fue un sistema de control de versiones, elegimos GitHub por la flexibilidad que éste tiene para gestionar proyectos desde bajo a alto nivel.

https://github.com/TheVampi/TrafficModelTecNM

En el repositorio está alojado el proyecto, capturas de ejecución y video de ejecución.

Estructuras de datos a utilizar

En este modelo se utilizó colas estáticas para representar a nivel de programación la llegada de carros, peatones y ciclcistas.

Es decir, se van “encolando” n cantidad de personas, ciclistas y carros, se evalúan, y se toma la decisión de “desencolar” de acuerdo a las condiciones ya mencionadas arriba en el diagrama de flujo.

Tecnologías y librerías utilizadas

Python

Decidimos usar este lenguaje por su versatilidad y facilidad a la hora de programar, adicionalmente, de base ya incluye muchísimas librerías muy útiles para facilitar el trabajo a la hora de programar. **Usamos la última versión 3.12.0**

Referencias:

Status of Python versions. (n.d.). Python Developer’s Guide. https://devguide.python.org/versions/#versions

Visual Studio Code

El IDE que optamos fue éste, debido a toda la rama de herramientas y extensiones que se pueden acoplar, adicionalmente que posee de las mejores herramientas de highlighting de código que nos son de mucha utilidad para comprender mejor la sintaxis. **Usamos la versión 1.83**

Referencias:

Visual Studio Code September 2023. (2021, November 3). https://code.visualstudio.com/updates/v1_83

Tkinter

Utilizamos esta librería puesto que es una que ya está incluida por default en Python, que nos permite generar interfaces graficas de manera sencilla, en nuestro caso, solo rectángulos que se muevan de una posición a otra ejemplificando los carros, peatones y ciclistas. Se utilizaron funciones y conceptos a continuación:

Creación de canvas: En resumidas palabras, creamos un lienzo o ventana para hacer la representación gráfica del modelo, modificando ciertos atributos como sus dimensiones en pixeles, nombre, etc.

Creación de rectángulos: Se crearon rectángulos de diferentes colores con sus respectivas dimensiones medidas en pixeles.

Creación de animaciones: Para esto diseñamos métodos que simplemente desplazaran hacia una coordenada en específico los rectángulos, simulando así el movimiento de los carros, peatones y ciclistas.

Referencias:

Tkinter Canvas. (n.d.). https://www.tutorialspoint.com/python/tk_canvas.htm

Graphical user interfaces with TK. (s. f.). Python documentation. https://docs.python.org/3/library/tk.html

Tkinter Canvas. (s. f.). https://www.tutorialspoint.com/python/tk_canvas.htm

Código Fuente:

```
import random
import tkinter as tk
import csv
import time

carrosArray = []
personasArray = []
bicicletasArray=[]
indice_actual = 0

def escrituraArchivo():
    with open(r'C:\Users\luisi\Desktop\DataStructures-
TecNM\Python\lab08Semaforo\valores.csv', 'w+', newline='') as file:
        file.write('valor1' + "|" + 'valor2' + '\n')
        writer = csv.writer(file, delimiter='|', quotechar='"',
quoting=csv.QUOTE_NONNUMERIC)
        for i in range(20):
            carros = random.randint(1, 20)
            personas = random.randint(1, 10)
            bicicletas=random.randint(1,10)
            writer.writerow([carros, personas,bicicletas])

escrituraArchivo()

def leerArchivo():
```

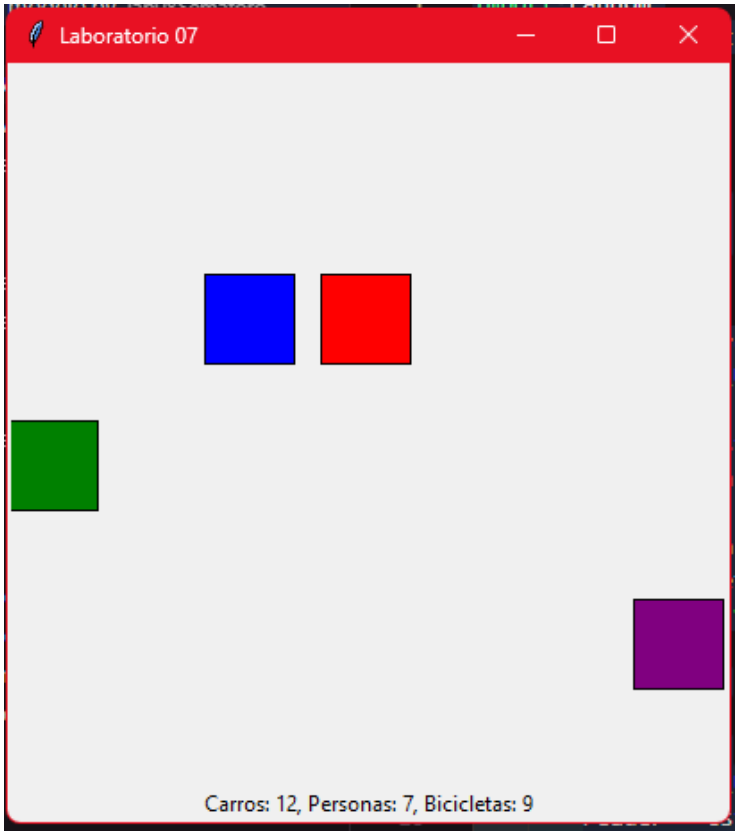
```
    with open(r'C:\Users\luisi\Desktop\DataStructures-
TecNM\Python\lab08Semaforo\valores.csv', 'r') as file:
        reader = csv.reader(file, delimiter='|')
        next(reader)  # Saltar la primera fila
        for row in reader:
            carros = int(row[0])
            personas = int(row[1])
            bicicletas= int(row[2])
            carrosArray.append(carros)
            personasArray.append(personas)
            bicicletasArray.append(bicicletas)

leerArchivo()

print("Valores de carrosArray:", carrosArray)
print("Valores de personasArray:", personasArray)
print("Valores de bicicletasArray:", bicicletasArray)

def leer_valores():
    global indice_actual
    if indice_actual < 20:
        valor1 = carrosArray[indice_actual]
        valor2 = personasArray[indice_actual]
        valor3= bicicletasArray[indice_actual]
        print('Valor1: ' + str(valor1))
        print('Valor2: ' + str(valor2))
        print('Valor3: ' + str(valor3) )
        label.config(text=f"Carros: {valor1}, Personas: {valor2},
Bicicletas: {valor3}")

        #Prioridad a personas
        if valor1<15 and valor2+valor3<10:
            mover_rectangulo_morado()
            mover_rectangulo_verde()
            print('pasan primero carros')
        else:
            if (valor1<15 and valor2+valor3>=10):
                mover_rectangulo_rojo()
                mover_rectangulo_azul()
                print('pasan primero personas')
            else:
                mover_rectangulo_morado()
                mover_rectangulo_verde()
                print('pasan primero carros')
```

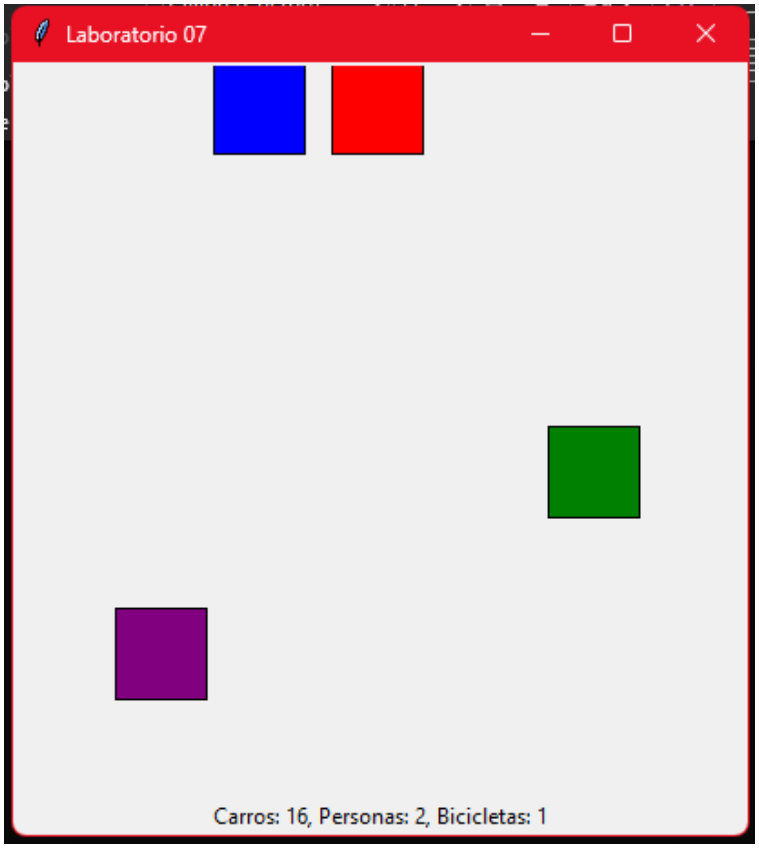
```
        indice_actual += 1

def mover_rectangulo_rojo():
    move_rectangulo_abajo()

def move_rectangulo_abajo():
    coords = canvas.coords(rectangulo)
    y1 = coords[1]  # Coordenada y de la parte superior
    y2 = coords[3]  # Coordenada y de la parte inferior
    if y2 < 400:
        canvas.move(rectangulo, 0, 2)
        root.after(10, move_rectangulo_abajo)
    else:
        canvas.coords(rectangulo, 175, 0, 225, 50)
        ##leer_valores()  # Llamar a leer_valores después de mover el
rectángulo

# AZUL
def mover_rectangulo_azul():
    move_rectangulo_abajoB()

def move_rectangulo_abajoB():
    coords = canvas.coords(rectangulo4)
    y1 = coords[1]  # Coordenada y de la parte superior
    y2 = coords[3]  # Coordenada y de la parte inferior
    if y2 < 400:
        canvas.move(rectangulo4, 0, 2)
        root.after(10, move_rectangulo_abajoB)
    else:
        canvas.coords(rectangulo4, 110, 0, 160, 50)
        leer_valores()  # Llamar a leer_valores después de mover el
rectángulo

def move_rectangulo_abajo_decisionB():
    coords = canvas.coords(rectangulo4)
    y1 = coords[1]  # Coordenada y de la parte superior
    y2 = coords[3]  # Coordenada y de la parte inferior
    if y2 < 400:
        canvas.move(rectangulo, 0, 2)
        root.after(10, move_rectangulo_abajoB)
    else:
        canvas.coords(rectangulo, 110, 0, 160, 50)
        #leer_valores()  # Llamar a leer_valores después de mover el
rectángulo
```

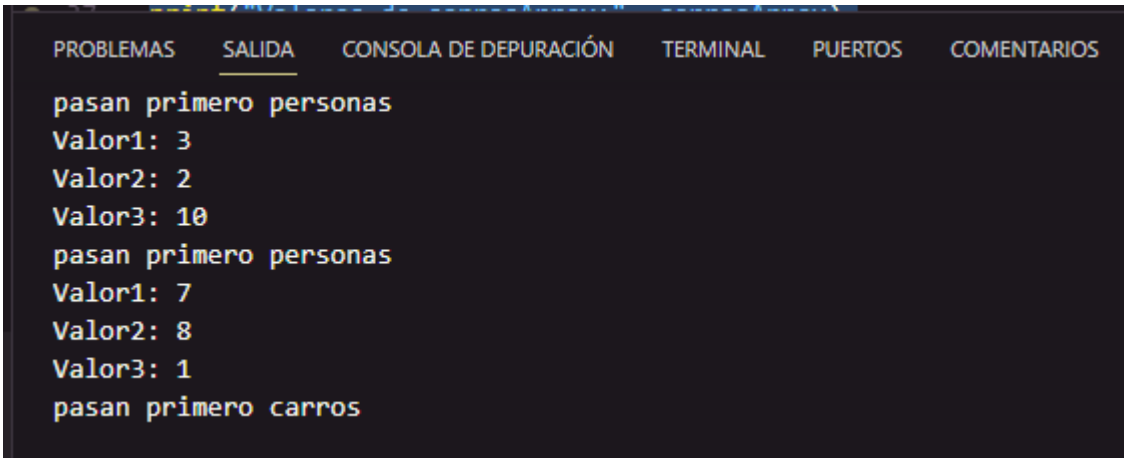
```
##VERDE
def mover_rectangulo_verde():
    move_rectangulo_derecha()

def move_rectangulo_derecha():
    coords = canvas.coords(rectangulo3)
    x1 = coords[0]  # Coordenada x de la parte izquierda
    x2 = coords[2]  # Coordenada x de la parte derecha
    if x2 < 400:
        canvas.move(rectangulo3, 2, 0)  # Mover hacia la derecha
        root.after(10, move_rectangulo_derecha)
    else:
        canvas.coords(rectangulo3, 0, 200, 50, 250)
        leer_valores()  # Llamar a leer_valores después de mover el
rectángulo verde

def mover_rectangulo_morado():
    move_rectangulo_izquierda()  # Llamar al movimiento del rectángulo
morado


def move_rectangulo_izquierda():
    coords = canvas.coords(rectangulo2)
    x1 = coords[0]  # Coordenada x de la parte izquierda
    x2 = coords[2]  # Coordenada x de la parte derecha
    if x1 > 0:
        canvas.move(rectangulo2, -2, 0)  # Mover hacia la izquierda
        root.after(10, move_rectangulo_izquierda)
    else:
        canvas.coords(rectangulo2, 350, 300, 400, 350)
        ##leer_valores()  # Llamar a leer_valores después de mover el
rectángulo morado


root = tk.Tk()
root.title("Laboratorio 07")

canvas = tk.Canvas(root, width=400, height=400)
canvas.pack()

# Creación de los rectángulos
rectangulo = canvas.create_rectangle(175, 0, 225, 50, fill="red")
rectangulo2 = canvas.create_rectangle(350, 300, 400, 350, fill="purple")
rectangulo3 = canvas.create_rectangle(0, 200, 50, 250, fill="green")
rectangulo4 = canvas.create_rectangle(110, 0, 160, 50, fill="blue")
```

```
# Creación del label
label = tk.Label(root, text="")
label.pack()

leer_valores()

root.mainloop()
```

Capturas de ejecución







Video de la ejecución:
https://drive.google.com/file/d/1oCU98HitBzMTzJIC5Y1frFWnBxvZgFC6/view?usp=share_link