Abbottabad University Of Ssience And Technalogy

Traffic Control Simulation

Version 1.0

By

[Waleed Babar 14830]

Using python

CODE OF PROJECT

import time

import random

import heapq

import networkx as nx

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from tkinter import Tk, Label, Button, Entry, Frame, messagebox

# Graph representation

traffic\_graph = {

1: {2: 0, 3: 0},

2: {1: 0, 3: 0},

3: {1: 0, 2: 0}

}

car\_positions = []

# Dijkstra's algorithm to find the optimal lane

def dijkstra(graph, start):

pq = [(0, start)]

distances = {node: float('inf') for node in graph}

distances[start] = 0

while pq:

current\_distance, current\_node = heapq.heappop(pq)

for neighbor, weight in graph[current\_node].items():

distance = current\_distance + weight

if distance < distances[neighbor]:

distances[neighbor] = distance

heapq.heappush(pq, (distance, neighbor))

return min(distances, key=distances.get)

# Update traffic graph weights

def update\_traffic\_graph(graph):

for lane in graph:

for neighbor in graph[lane]:

graph[lane][neighbor] = random.randint(1, 20)

# Initialize random car positions

def initialize\_cars():

global car\_positions

car\_positions = [(random.choice(list(traffic\_graph.keys())), random.choice(list(traffic\_graph.keys()))) for \_ in range(5)]

# Animate car movement

def animate(i):

plt.clf()

G = nx.DiGraph()

for node, neighbors in traffic\_graph.items():

for neighbor, weight in neighbors.items():

G.add\_edge(node, neighbor, weight=weight)

pos = nx.circular\_layout(G)

edge\_labels = nx.get\_edge\_attributes(G, 'weight')

# Define the current lane with green light (from control\_traffic function)

global green\_lane

edge\_colors = []

for u, v in G.edges():

if u == green\_lane or v == green\_lane:

edge\_colors.append('green') # Green for emergency lane

else:

edge\_colors.append('red') # Red for other lanes

nx.draw(G, pos, with\_labels=True, node\_color='lightblue', node\_size=1000, font\_size=10, edge\_color=edge\_colors)

nx.draw\_networkx\_edge\_labels(G, pos, edge\_labels=edge\_labels)

global car\_positions

for idx, (current, target) in enumerate(car\_positions):

next\_target = random.choice(list(traffic\_graph[current].keys()))

car\_positions[idx] = (target, next\_target)

car\_x, car\_y = pos[current]

plt.scatter(car\_x, car\_y, color='red', s=100) # Represent car as a red point

plt.title("Traffic Graph with Cars")

# Non-blocking traffic light control

def control\_traffic(lane, status\_label):

global green\_lane

green\_lane = lane # Set the green lane to the selected lane

green\_time = 10

status\_label.config(text=f"Green In Lane {lane} Active\nRed In Other Lanes Active")

root.after(green\_time \* 1000, lambda: status\_label.config(text=f"Lane {lane} Green Time: {green\_time} seconds"))

# Main simulation start function

def start\_simulation(emergency\_lane\_entry, status\_label):

try:

emergency\_lane = int(emergency\_lane\_entry.get())

except ValueError:

messagebox.showerror("Invalid input", "Enter a valid lane number (1, 2, 3 or 0 for none).")

return

update\_traffic\_graph(traffic\_graph)

initialize\_cars()

optimal\_lane = emergency\_lane if emergency\_lane in [1, 2, 3] else dijkstra(traffic\_graph, start=1)

status\_label.config(text=f"{'Emergency' if emergency\_lane in [1, 2, 3] else 'Optimal'} Lane: {optimal\_lane}")

control\_traffic(optimal\_lane, status\_label)

ani = FuncAnimation(plt.gcf(), animate, interval=1000)

plt.show()

# Create the main Tkinter window

root = Tk()

root.title("Traffic Control Simulation")

root.geometry("800x600")

# Frame for user controls

control\_frame = Frame(root)

control\_frame.pack(pady=20)

# Label and Entry for emergency lane input

Label(control\_frame, text="Enter Emergency Lane (1, 2, 3 or 0 for none):").grid(row=0, column=0, padx=10)

emergency\_lane\_entry = Entry(control\_frame)

emergency\_lane\_entry.grid(row=0, column=1, padx=10)

# Button to start the simulation

Button(control\_frame, text="Start Simulation", command=lambda: start\_simulation(emergency\_lane\_entry, status\_label)).grid(row=1, columnspan=2, pady=10)

# Status Label to show the current state

status\_label = Label(root, text="Traffic Light Status", font=("Arial", 14))

status\_label.pack(pady=20)

# Initialize green\_lane variable to track which lane is green

green\_lane = 0

# Run the Tkinter event loop

root.mainloop()