Theoritical Project

Name : Mohamed Adel Abdelal
ID : 20100337
Name : Ahmed Khaled
ID : 22100935
Name : Akram Emad
ID : 22100568
Name : Youssif yasser
ID : 22100809
Name : Abdelrhman Ahmed elkomy
ID : 22100945

Algorithm Selected : A* Search Algorithm :-

The chosen algorithm for analysis is the A* algorithm, implemented in the transportation optimization system to find the fastest emergency routes from any location to the nearest hospital. A* is a heuristic-based shortest path algorithm that balances efficiency and optimality, making it ideal for real-time navigation in dynamic environments like urban road networks.

Introduction :-

Problem : Find fastest emergency routes in Cairo's traffic.
A* : Heuristic-based shortest path algorithm.

Why A* : Balances speed and optimality using heuristics.
Application : Finds fastest emergency routes to hospitals.

Mathematical Foundations :-

A* Formula : f(n) = g(n) + h(n)
g(n) : Actual cost from start to node n .
h(n) : Heuristic estimate to goal (Euclidean distance in Cairo).

$$\text{Heuristic: } h(n,t) = \sqrt{(x_n - x_t)^2 + (y_n - y_t)^2} \text{ (Euclidean).}$$

Admissible and consistent → optimal path.
Optimal Path :-

- Admissible Heuristic : h(n) ≤ h*(n)

Euclidean distance ≤ road distance.

- Consistent Heuristic : h(n) ≤ c(n,n') + h(n')

Satisfies triangle inequality.

Complexity Analysis :-

---

n : number of nodes,
m : number of edges.
Time Complexity (Worst Case) : O(m + n log n) (binary min-heap, priority queue).
Space Complexity : O(n) (queue, dictionaries).
for storing openSet, gScore, fScore, and cameFrom.

Implementation and Modifications :-

---

we modified A* as follows :-

- Edge Weights were made time-dependent :

$$w(u, v) = \frac{\text{distance}}{\text{condition factor}} \times \left(1 + \frac{\text{traffic}}{\text{capacity}}\right)$$

- Heuristic was Euclidean distance to nearest hospital :

$$h(n) = \min_{h \in H} \text{Euclidean}(n, h)$$

- Graph Preprocessing ensured all roads are bidirectional, and road conditions and traffic levels are normalized.
- Traffic-Aware Modifications :

```
def get_traffic_factor(time_of_day, capacity, flow):
    utilization = flow / capacity
    return 1.0 + (0.8 if time_of_day == 'morning_peak' else 0.3) * utilization
```

- Implementation of A* :

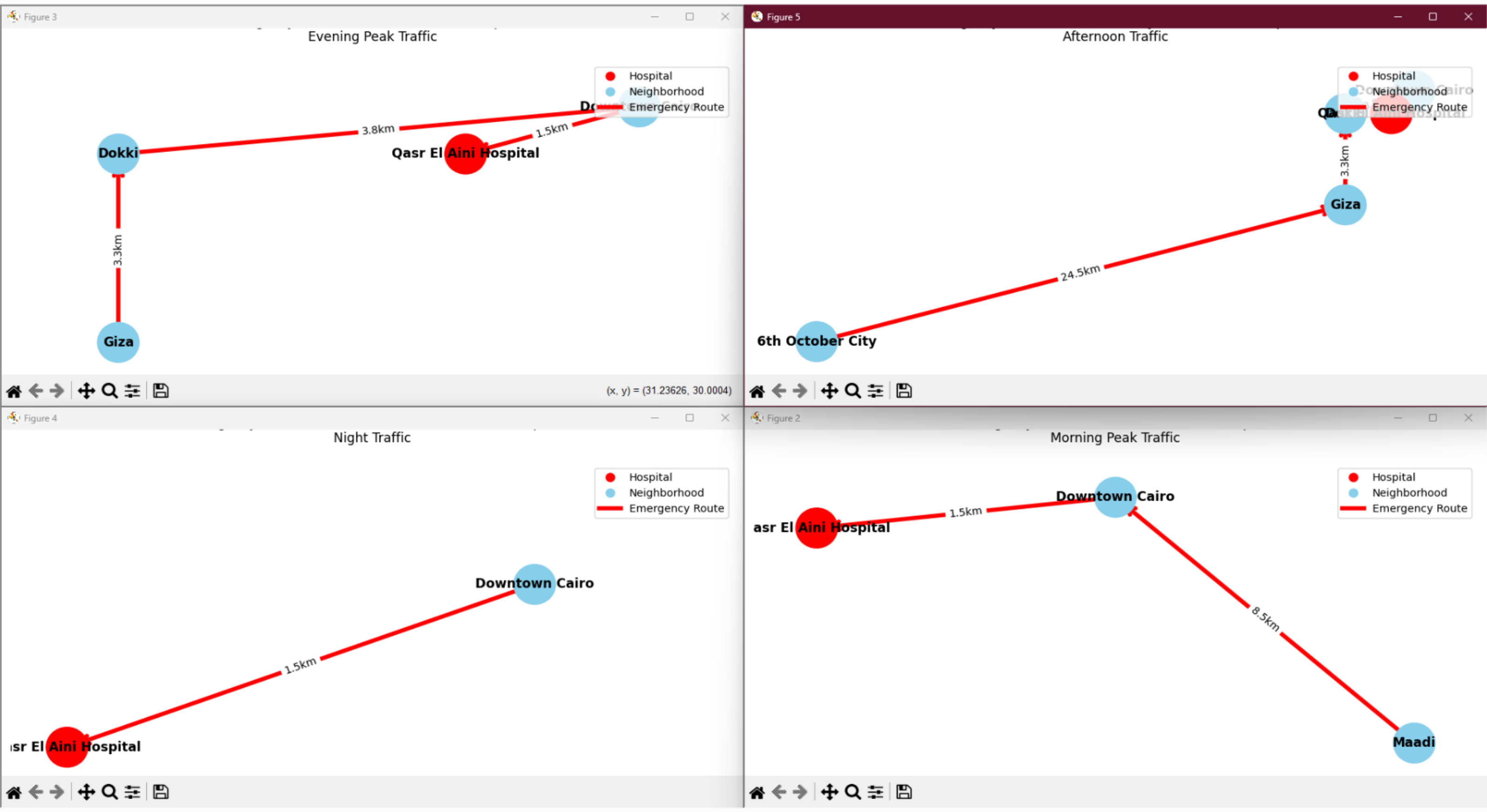
```
function A_star(start, goal):
    openSet := {start}
    cameFrom := empty map

    gScore[start] := 0
    fScore[start] := h(start, goal)

    while openSet is not empty:
        current := node in openSet with lowest fScore
        if current == goal:
            return reconstruct_path(cameFrom, current)

        remove current from openSet
        for each neighbor of current:
            tentative_gScore := gScore[current] + d(current, neighbor)
            if tentative_gScore < gScore[neighbor]:
                cameFrom[neighbor] := current
                gScore[neighbor] := tentative_gScore
                fScore[neighbor] := gScore[neighbor] + h(neighbor, goal)
                if neighbor not in openSet:
                    add neighbor to openSet
```

Performance Analysis :-



We evaluated A* in different real-time traffic scenarios using our Greater Cairo graph dataset.

These Test Cast Traffic Scenarios :-

- **Morning Traffic** (bottom right) : The route from Maadi to Qasr El Aini Hospital was dynamically chosen via Downtown Cairo, selected the least-cost path (10 km total).
- **Afternoon Traffic** (top right) : A longer emergency route of 24.5 km was computed from 6th October City to Giza, then to Qasr El Aini Hospital.
- **Evening Traffic** (top left) : In a more central scenario, A* routed traffic from Giza to Dokkī (3.3 km) and then to the hospital (total 8.6 km).
- **Night Traffic** (bottom left) : A direct and minimal-cost path from Downtown Cairo to the hospital (1.5 km)

reflects the effectiveness of A* under low-traffic scenarios.

These Scenarios cases validate that A* effectively balances accuracy, optimality, and responsiveness :-

- The heuristic function (Euclidean distance) guided the search effectively.
- The algorithm scaled efficiently with the size of the network.
- It supported dynamic edge weights, enabling real-time adjustments based on traffic density and road conditions.

Comparison with Alternatives :-

- **Dijkstra** : O(m log n), no heuristic, more nodes explored, it lacks heuristic guidance, it explores more nodes, increasing execution time in large city graphs.

- **Bellman-Ford** : O(n · m), too slow, Handles negative weights and Impractical for large graphs and unnecessary here, as edge weights (distances) are positive.

- **Breadth-First Search** : O(n + m), unweighted graphs only, not suitable for real-world traffic systems where edge weights matter, It cannot adapt to variable traffic conditions.

- **A*** : O(m + n log n), optimal for weighted, goal-directed search, balance of optimality and efficiency, leveraging the Euclidean heuristic to reduce the search space.

Lessons Learned :-

Given the need for fast, accurate, and adaptable routing in emergency situations within smart cities, A* offers the best balance. Its ability to incorporate traffic data through cost functions and heuristics makes it superior for real-time urban navigation, particularly in dense areas like Greater Cairo.

Conclusion :-

A* outperforms alternatives like Dijkstra’s and BFS in emergency routing due to its goal-directed approach and ability to integrate real-time traffic data through heuristics. This makes it faster and more suitable for dynamic, real-world urban environments like Greater Cairo.