City Network Path Analysis – Final Project Document

1. Project Overview

This project models a city as a graph and applies classical AI search algorithms to find optimal paths between two known locations. The city is represented as **40 virtual locations** with streets that have **distance, speed limits, traffic delays, accidents, closures, and weather impacts**. The main goal is to compare multiple search strategies using **actual travel time (g(n))** as the primary cost while considering realistic city constraints.

---

2. Feature Specification (System Inputs)

2.1 Locations

- 40 virtual city locations representing intersections, buildings, parks, and other points of interest
- Modeled as **nodes** with coordinates for heuristic calculation
- Examples: Central Hospital, High School, Main University, City Mall, Residential Blocks, Industrial Zones, etc.

2.2 Streets (Road Network)

- Connect locations in a realistic manner (2–4 streets per location)
- All streets are **two-way**
- Modeled as **edges** containing:
- Distance
- Average speed
- Speed limit
- Traffic delays (traffic lights)
- Accidents
- Closures
- Weather impact (Clear, Rain, Fog, Snow)

2.3 Distance

- Physical length of each street segment
- Used in **g(n)** (actual travel time) and **h(n)** (heuristic estimate)

2.4 Speed Limits

- Maximum allowed speed on each street
- Combined with average speed to compute effective travel time

2.5 Average Speed

- Represents typical traffic conditions on the street
- Effective speed = min(avg_speed, speed_limit)

2.6 Actual Travel Time (g(n))

- Computed per street as:

```
if closed or weather == 'Snow': cost = inf
speed = min(avg_speed, speed_limit)
adjust speed for weather and accidents
travel_time = distance / speed * 60 + traffic_light_delay
```

- Accumulates along the path for UCS and A*

2.7 Estimated Travel Time (h(n))

- Heuristic for remaining path: straight-line distance / max speed
- Must be **admissible** for A* to guarantee optimality

2.8 Traffic Lights

- Fixed delays at intersections, added to g(n)
- Can be extended to time-dependent delays

2.9 Accidents / Road Closures

- Streets can be blocked or slowed due to accidents or closures
- Modeled in g(n) as speed reductions or infinite cost

2.10 Weather Impacts

- Rain → reduces speed by 20%
- Fog → reduces speed by 30%
- Snow → drastically reduces speed or blocks road

2.11 Energy / Fuel Consumption (Optional)

- Derived from distance, speed, and stop-and-go conditions
- Used for analysis but not primary optimization metric

---

3. Selected Search Algorithms

3.1 A* Search (Primary Algorithm)

- Uses f(n) = g(n) + h(n)

- Guaranteed optimal when heuristic is admissible
- Efficient for realistic city maps with g(n) incorporating delays, accidents, and weather

3.2 Uniform Cost Search (UCS)

- Uses g(n) only
- Finds optimal path without heuristic guidance
- Serves as baseline for comparison

3.3 Greedy Best-First Search

- Uses h(n) only
- Fastest, but may choose suboptimal paths
- Demonstrates tradeoff between speed and optimality

3.4 Bidirectional Uniform Cost Search

- Runs UCS simultaneously from start and goal
- Reduces search space for large maps
- Requires known goal, two-way streets, and static costs

---

4. Code Implementation Plan

4.1 Data Structures

- **Locations:** dictionary mapping name → Location(x, y)
- **City Graph:** dictionary mapping location → list of Street objects
- **Street Object Attributes:** street_name, to, avg_speed, distance, speed_limit, traffic_light_delay, accident, closed, weather

4.2 Cost Calculations

- **g(n):** actual travel time along a street
- **h(n):** straight-line distance to goal / max speed
- Street speed adjusted by avg_speed, speed_limit, weather, and accidents

4.3 Algorithm Modules

- Separate implementations for:
- A* Search
- Uniform Cost Search
- Greedy Best-First Search
- Bidirectional UCS
- All use the same graph and cost functions but different evaluation strategies

4.4 Execution Flow

1. Load city graph and features

2. Select start and goal locations
3. Choose algorithm
4. Run search
5. Record:
6. Path found
7. Total travel time
8. Number of nodes expanded
9. Compare algorithm results

---

5. Evaluation Criteria

- Path optimality
- Total travel time
- Efficiency (nodes expanded)
- Heuristic guidance effectiveness

---

6. Expected Outcome

- **A*** → best balance of optimality and efficiency
- **UCS** → guarantees optimality but explores more nodes
- **Greedy** → fastest but may choose suboptimal paths
- **Bidirectional UCS** → reduces search space in static environments

---

7. Conclusion

This project demonstrates the performance of informed and uninformed search algorithms under realistic city navigation constraints, highlighting how **travel time, accidents, weather, traffic, and road closures** affect route selection. It emphasizes the importance of **admissible heuristics** and well-modeled edge costs in intelligent path-finding systems.