Coursework 1: Route Planning with Search Algorithms

Course: CS3317: Artificial Intelligence (Search Algorithms)
University: Shanghai Jiao Tong University

Overview

You will implement classical search algorithms for **route planning** on a graph of cities in China.

- Each **city** is a node.
- Each **road** is an **undirected weighted edge** with distance in kilometers.
- City coordinates are provided for the heuristic in **A***.

Algorithms in this coursework

- Breadth-First Search (BFS) — *provided as an example implementation*
- Depth-First Search (DFS) — *you implement*
- Uniform Cost Search (UCS) — *you implement*
- A* Search (A*) — you implement*

Learning Objectives

By the end of this coursework you should be able to:

1. Formulate route planning as a graph search problem.
2. Implement DFS, UCS, and A* correctly.
3. Use a coordinate-based heuristic for A*.
4. Explain and observe differences in **optimality** and **efficiency** between algorithms.

Dataset

You are given two text files.

`cities.txt`

Format:

```
CityName latitude longitude
```

`roads.txt`

Format:

```
CityA CityB distance_km
```

Notes:

- The graph is **undirected**.

Why this dataset highlights differences

The road network includes:

- **distractor branches** (extra cities/roads) that can make DFS explore more nodes
- an **expensive shortcut** (low-hop but high-cost edge) so BFS can return a route with **fewer edges but higher total distance** than UCS/A*

This is intentional: you should see different behavior across BFS/DFS/UCS/A*.

Tasks

Task 1 — Implement DFS

Implement `dfs(graph, start, goal)` in `route_planning.py`.

Task 2 — Implement UCS

Implement `ucs(graph, start, goal)` in `route_planning.py`.

Task 3 — Implement A* Search

Implement `astar(graph, coords, start, goal)` in `route_planning.py`.

Heuristic:

- h(n) = straight-line distance (km) from city (n) to the goal city
- compute from coordinates in `cities.txt` (already provided in the starter code)

Task 4 — Run required tests and compare algorithms

Run all algorithms on the required queries (below), collect outputs.

Required Queries

You must run the following queries:

- Beijing → Shanghai
- Beijing → Shenzhen
- Shanghai → Chengdu
- Guangzhou → XiAn
- Chengdu → Hangzhou

Provided Programs

1) `route_planning.py` (starter code)

- BFS is already implemented (example).
- You implement DFS/UCS/A*.
- After each run, the program generates a **standalone HTML** visualization (SVG).

Run a single query:

```
        python route_planning.py --algorithm astar --start Beijing --goal
Shenzhen
```

Specify output file:

```
        python route_planning.py --algorithm bfs --start Beijing --goal
Shanghai --output outputs/route_result.html
```

2) `run_all_tests.py` (batch runner)

Runs all algorithms on all required queries and stores results in a directory.

```
        python run_all_tests.py --outdir test_results
```

It produces:

- `test_results/summary.csv`
- `test_results/summary.json`
- one folder per algorithm and query, each containing:
  - `result.json`
  - `visualization.html`
  - `traceback.txt` (only if errors occur)

What you should see if everything is correct (expected patterns)

This coursework is designed so that the outputs clearly differ.

1) BFS vs UCS/A* (optimality)

- **BFS** minimizes the **number of edges** (fewest hops), not the total distance.
- **UCS** and **A*** minimize the **total distance**.

Therefore:

- On at least one query, you should observe:

- BFS returns a route with **fewer hops** but **higher total distance** than UCS/A*.

2) DFS behavior (not optimal, can be inefficient)

- DFS is **not optimal** (for either hops or distance).
- DFS can expand many nodes depending on branching and neighbor order.

Therefore:

- You may see DFS produce:
  - a longer / higher-cost route
  - **more expanded nodes** than BFS/UCS/A*

3) UCS vs A* (efficiency)

- Both UCS and A* are optimal (with an admissible heuristic).
- A* should usually expand **fewer nodes** than UCS because it uses the heuristic.

Therefore:

- On most queries, you should see:
  - A* expands **fewer nodes** than UCS
  - and often runs faster (see `runtime_ms` in summaries)

Submission Requirements

Submit exactly:

1. Your implemented `route_planning.py`
2. The entire test output directory produced by:

   ```
   python run_all_tests.py --outdir test_results
   ```

   i.e., submit the whole `test_results/` folder.

No report is required.

Grading Rubric (100 points)

Component	Points
DFS implementation	20
UCS implementation	30
A* implementation	35
Running tests + correct outputs directory structure	10
Code clarity (readable, well-structured)	5
Total	**100**

Academic Integrity

- You may discuss high-level ideas.
- You must write your own code.
- Do not copy implementations from other students or online sources.

Appendix: Dataset examples

`cities.txt` (snippet)

```
Beijing 39.9042 116.4074
Shanghai 31.2304 121.4737
XiAn 34.3416 108.9398
...
```

`roads.txt` (snippet)

```
Beijing Tianjin 120
Nanjing Shanghai 300
Hohhot Shanghai 1500
...
```