Lab: Greedy Best-First Search (GBFS) vs A*

Implement, analyze, and compare informed search with real maps and heuristics

Learning Goals

- • Implement GBFS and A* using a priority queue.
- • Design and test heuristics (admissible vs. inconsistent).
- • Compare behavior: optimality, node expansions, time/space.
- • Handle ties, unreachable goals, and repeated-state detection.

Problem Scenario

You're routing a drone from a start city to a goal city. You will run both GBFS (minimize h(n)) and A* (minimize f(n)=g(n)+h(n)) on two small maps:
• Map 1 (EuclidTown): coordinates are provided , use straight-line distance to the goal as h(n).
• Map 2 (TrapVille): same heuristic, but edges are tweaked so GBFS is tempted down an expensive path.

Data:

Map 1: EuclidTown

Nodes & coordinates (for h to goal G):

```
A:(0,0)  B:(2,1)  C:(4,0)  D:(1,3)
E:(3,4)  F:(5,3)  G:(6,1)
```

Undirected edges with costs:

```
A-B:2.2  A-D:3.2
B-C:2.2  B-E:3.6
C-F:3.6
D-E:2.4
E-F:2.4
F-G:2.2
```

Start = A, Goal = G

Reference results (for instructor checking):

```
Optimal path (by Dijkstra): A → B → C → F → G
Optimal cost: 10.2
With straight-line h to G, A* is optimal; GBFS also finds the same path here.
```

Map 2: TrapVille

Nodes & coordinates:

```
S:(0,0)  X:(2,2)  Y:(2,-2)  Z:(4,0)  G:(6,0)
```

Undirected edges with costs:

```
S-X:2.9  S-Y:2.9
X-Z:8.0  Y-Z:2.6
Z-G:2.0
```

Start = S, Goal = G

Reference results:

```
True optimal: S → Y → Z → G, cost 7.5
GBFS (typical ties) is lured into: S → X → Z → G, cost 12.9 (suboptimal)
A* returns the optimal 7.5
```

Required Functions:

- • priority_queue_push(pq, priority, state, g_cost, parent)
- • priority_queue_pop(pq) -> (priority, state, g_cost, parent)
- • gbfs(start, goal, graph, heuristic) -> (path, cost, expansions, visited_order) — priority = h(n)
- • astar(start, goal, graph, heuristic) -> (path, cost, expansions, visited_order) — priority = g(n)+h(n) and maintain best-g
- • reconstruct_path(parents, goal) -> list_of_nodes
- • euclidean_heuristic(node, goal, coords) -> float
- • is_consistent(graph, heuristic) -> (bool, violating_edges[]) — check h(u) ≤ cost(u,v)+h(v) for all edges
- • is_admissible(all_nodes, goal, graph, heuristic) -> (bool, counterexamples[]) — compare to true shortest path

Note: If heaps are confusing, you may use a sorted list/array (slower but acceptable for small graphs). Document your choice.

What to Submit

Part A Correctness (EuclidTown)

- • Run A* on Map 1: report path, total cost, nodes expanded, visited order.

- • Run GBFS on Map 1: same reports.
- • Verify both return A→B→C→F→G with cost ≈10.2 (minor float error OK).
- • Show is_consistent == True and is_admissible == True for the heuristic.

Part B GBFS Trap (TrapVille)

- • Run A* on Map 2 expect S→Y→Z→G, cost ≈7.5.
- • Run GBFS on Map 2 likely S→X→Z→G, cost ≈12.9 (explain tie-breaking).
- • Compare expansions and explain why GBFS fails (it ignores g(n)).

Part C Tie-   Breaking & Variants

- • Change tie-break rule (lexical vs FIFO vs LIFO on equal priority) and re-run GBFS on Map 2. Did results change? Why?
- • (Optional bonus) Add a slightly noisy heuristic h'(n)=h(n)*k where k∈[0.95, 1.05] and show sensitivity for GBFS vs A*.

Marking Rubric (30 Marks)

Implementation (12)	GBFS priority logic (3); A* with best-g & open/closed (6); Path reconstruction & I/O (3)
Heuristic Checks (6)	is_consistent (3); is_admissible (3)
Experiments & Results (8)	Map 1 correct & explained (3); Map 2 trap & analysis (3); Tie-breaking (2)
Clarity & Code Quality (4)	Comments, modularity, clean logs
Optional Bonus (+3)	Noisy-heuristic sensitivity with concise charts/tables