United International University

CSE 3812: Artifical Intelligence lab

Lab 2: Searches

18/08/25

Contents

1 Question 1: Heuristic Admissibility Checker

Problem: In A* search, a heuristic $h(n)$ is **admissible** if it never overestimates the true shortest distance $h^*(n)$ to the goal.

Write a function:

```
def is_admissible(graph, heuristic, goal):
    ...
```

that checks whether a given heuristic is admissible for all nodes in the graph.

Input:

- `graph`: dictionary of dictionaries, where edges are weighted.
- `heuristic`: dictionary mapping nodes to heuristic estimates.
- `goal`: the target node.

Output:

- Return $(\text{True}, [\,])$ if the heuristic is admissible.
- Return $(\text{False}, \text{violating\_nodes})$ if some nodes violate admissibility.

—

Test Case 1 (Overestimation, Not Admissible):

```
graph = {
    'A': {'B': 1, 'C': 4},
    'B': {'A': 1, 'C': 2, 'D': 6},
    'C': {'A': 4, 'B': 2, 'D': 3},
    'D': {'B': 6, 'C': 3}
}
heuristic = {'A': 7, 'B': 6, 'C': 2, 'D': 0}
goal = 'D'

# Shortest distances: A→D=6, B→D=5, C→D=3, D→D=0
# h(A)=7 > 6 and h(B)=6 > 5 → not admissible
Expected: (False, ['A','B'])
```

—

Test Case 2 (Admissible, All Exact):

```
graph = {
    'A': {'B': 2},
    'B': {'A': 2, 'C': 2},
    'C': {'B': 2}
}
```

```
heuristic = {'A': 4, 'B': 2, 'C': 0}
goal = 'C'

# Shortest distances: A→C=4, B→C=2, C→C=0
# h(A)=4, h(B)=2, h(C)=0 → admissible
Expected: (True, [])
```

—

Test Case 3 (Admissible, Underestimates):

```
graph = {
    'S': {'A': 2, 'B': 5},
    'A': {'S': 2, 'B': 2, 'G': 5},
    'B': {'S': 5, 'A': 2, 'G': 1},
    'G': {'A': 5, 'B': 1}
}
heuristic = {'S': 3, 'A': 1, 'B': 0, 'G': 0}
goal = 'G'

# True shortest distances: S→G=5, A→G=3, B→G=1, G→G=0
# All heuristic values  true cost → admissible
Expected: (True, [])
```

2 Question 2: Grid A* with Manhattan Heuristic

Problem: Implement the **A* search algorithm** to find the shortest path in a 2D grid maze.

- The grid is represented as a 2D list of integers:
  - 0 = free cell
  - 1 = wall (blocked)
- Movements are allowed in 4 directions: up, down, left, right.
- Each move has a cost of 1.
- The heuristic function should be the **Manhattan distance**:

$$h(x, y) = |x - x_{\text{goal}}| + |y - y_{\text{goal}}|$$

Function Signature:

```
def a_star_grid(grid, start, goal):
    ...
```

The function should return a tuple $(path, cost)$ where:

- $path$ is a list of (row, col) coordinates from $start$ to $goal$ (inclusive),

- *cost* is the total number of steps in the path (or ∞ if no path exists).

Sample Test Case:

```
grid = [
    [0,1,0,0,0],
    [0,1,0,1,0],
    [0,0,0,1,0],
    [1,1,0,1,0],
    [0,0,0,1,0],
]
start = (0,0)
goal  = (4,4)

path, cost = a_star_grid(grid, start, goal)
print("Path:", path)
print("Cost:", cost)
```

Expected Output (path may vary, cost must match):

```
Path: [(0,0),(1,0),(2,0),(2,1),(2,2),(3,2),(4,2),(4,3),(4,4)]
Cost: 8
```

—