Northeastern University
CS5100 Foundations of Artificial Intelligence
Summer 2025
Professor Sarita Singh
Erdun E
May 26, 2025

Assignment 2 Answer

Question 1

- Breadth-first search
  - Complete:
    - Yes
    - Because the BFS explores all node level by level, so it is finally to find a solution if one exists
  - Optimal:
    - Yes
    - Provided that all steps costs are equal, so BFS would find the least cost solution first
  - Informed:
    - No
    - BFS does not use any heuristic, it just blindly explores the tree level by level
- Depth-first search
  - Complete:
    - No
    - DFS may go down an infinite path in an infinite or cyclic graph and never return
  - Optimal:
    - No
    - The DFS found first solution may not the lowest cost or shortest path solution
  - Informed:
    - No
    - DFS does not use any heuristic, it just explores deeper at will!
- Greedy best-first search
  - Complete:
    - No

    - Greedy search can get stuck in loops or ignore longer paths that lead to the goal
  - Optimal:
    - No
    - It chooses the node that appears closest to the goal but my miss the lowest cost path
  - Informed:
    - Yes
    - It uses a heuristic function to estimate the cost from n to the goal

- Uniform-cost search
  - Complete:
    - Yes
    - It explores nodes in increasing order of path cost and finally will find the goal if one exists
  - Optimal:
    - Yes
    - It's always expands the least cost node first, so it ensure to find the optimal path
  - Informed:
    - No
    - It does not use a heuristic, it only considers cumulative path cost

Question 2

- Tree Search
  - Tree search does not remember which states it has already visited. This means it can visit the same state many times. In some cases, this can make the search slow or even get stuck in a loop.
- Graph Search
  - Graph search remembers visited states using a closed set. It will not visit the same state again. This helps avoid loops and saves time.
- Key difference
  - Graph search avoids repeating states, but tree search does not.

Question 3

- Explanation
  - A heuristic is a rule that estimates how close a state is to the goal. It helps the search algorithm choose better paths first. This makes the search faster and more efficient.

- Example 1 Misplaced Tiles
  - This heuristic counts how many tiles are not in their goal position.
  - It gives a rough idea of how far the puzzle is from the goal. The more misplaced tiles, the more moves are likely needed.
- Example 2 Manhattan Distance
  - This heuristic adds up the total number of horizontal and vertical moves each tile is away from its goal position.
  - It gives a better estimate than misplaced tiles because it considers how far each tile needs to move, not just whether it is wrong.

Question 4(a)

- Greedy search
  - Correct route
    - Mehadia → Lugoj → Timisoara → Arad → Sibiu → Fagaras → Bucharest
  - Detailed steps taken (Node/costs at each step)
    - Mehadia → Lugoj (70)
    - Lugoj → Timisoara (111)
    - Timisoara → Arad (118)
    - Arad → Sibiu (140)
    - Sibiu → Fagaras (99)
    - Fagaras → Bucharest (211)
    - Total = 70 + 111 + 118 + 140 + 99 + 211 = 749
  - Explanation
    - Greedy search is fast but not optimal, because it does not look at path cost, only the distance to the goal.
- Uniform-cost search
  - Correct route
    - Mehadia → Drobeta → Craiova → Pitesti → Bucharest
  - Detailed steps taken (Node/costs at each step)
    - Start at Mehadia
      - Put into frontier Lugoj (cost=70)
      - Put into frontier Drobeta (cost=75)
    - Expand Lugoj (cost=70)
      - Add Timisoara (70 + 111 = 181) to frontier
    - Expand Drobeta (cost=75)

      - Add Craiova (75 + 120 = 195)
    - Expand Timisoara (181)
      - Add Arad (181 + 118 = 299)
    - Expand Craiova (195)
      - Add Pitesti (195 + 138 = 333)
      - Add Rimnicu Vilcea (195 + 146 = 341)
    - Expand Arad (299)
      - Add Sibiu (299 + 140 = 439)
      - Add Zerind (299 + 75 = 374)
    - Expand Pitesti (333)
      - Add Bucharest (333 + 101 = 434)
    - Total = 75 + 120 + 138 + 101 = 434
  - Explanation
    - Uniform-cost search always chooses the path with the lowest total cost. It does not use heuristics, but finds the optimal path by expanding nodes in order of increasing cost.

- A* search.
  - Correct route
    - Mehadia → Drobeta → Craiova → Pitesti → Bucharest
  - Detailed steps taken (Node/costs at each step)
    - Start at Mehadia (g=0, h=241, f=241)
      - Add Lugoj (g=70, h=244, f=314)
      - Add Drobeta (g=75, h=242, f=317)
    - Expand Lugoj (f=314)
      - Add Timisoara (g=181, h=329, f=510)
    - Expand Drobeta (f=317)
      - Add Craiova (g=195, h=160, f=355)
    - Expand Craiova (f=355)
      - Add Pitesti (g=333, h=100, f=433)
      - Add Rimnicu Vilcea (g=341, h=193, f=534)
    - Expand Pitesti (f=433)
      - Add Bucharest (g=434, h=0, f=434)
    - Total = 75 + 120 + 138 + 101 = 434
  - Explanation
    - A* search uses both actual path cost and estimated distance to the goal (f = g + h). It is both efficient and optimal when the heuristic is admissible.

Question 4(b)

- A* search from Sibiu to Bucharest
  - Correct route
    - Sibiu → Rimnicu Vilcea → Pitesti → Bucharest
  - Detailed step-by-step calculation
    - Route 1: Sibiu → Fagaras → Bucharest
      - g = 99 + 211 = 310, h = 0, f = 310
    - Route 2: Sibiu → Rimnicu Vilcea → Pitesti → Bucharest
      - g = 80 + 97 + 101 = 278, h = 0, f = 278
    - Route 3: Sibiu → Rimnicu Vilcea → Craiova → Pitesti → Bucharest
      - g = 80 + 146 + 138 + 101 = 465, h = 0, f = 465
  - Explanation
    - A* search chooses the path with the lowest f(n) = g(n) + h(n). The second route has the smallest total cost (278), so it is selected as the optimal path.
- Greedy Best-First Search from Sibiu to Bucharest
  - Correct route
    - Sibiu → Fagaras → Bucharest
  - Step-by-step heuristic comparison
    - Start at Sibiu (h=253)
    - Neighbors: Fagaras (h=176), Rimnicu Vilcea (h=193)
    - Choose Fagaras because h=176 is lower
    - From Fagaras, go to Bucharest (h=0)
  - Explanation
    - Greedy best-first search always chooses the node with the lowest heuristic value h(n). It ignores path cost, so it selects Sibiu → Fagaras → Bucharest since Fagaras has the lowest h(n) among neighbors.

References

- Russell, S., Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3rd ed.). Pearson.