COSC 4368: Fundamentals of Artificial Intelligence
Problem Set 1
Fall 2024

Task 1
Peter's Journey to Solve City and Network Problems

Background:

Peter Parker is a computer science intern student working on a project in Stark industry. Professor Bruce Banner has tasked Peter with solving real-world problems using Breadth-First Search (BFS) and Depth-First Search (DFS).

City Road Network - Finding Shortest Paths

Peter lives in a large city (Queens) with a complex road network. Their challenge is to help the city's transportation department find the shortest routes from a central location to every other important spot in the city. This is crucial for optimizing delivery routes for local businesses.

Problem:

The city's road intersections are nodes, and roads between intersections are edges. Peter needs to use BFS and DFS to determine the shortest path from the central intersection to all others and compare the both algorithms.

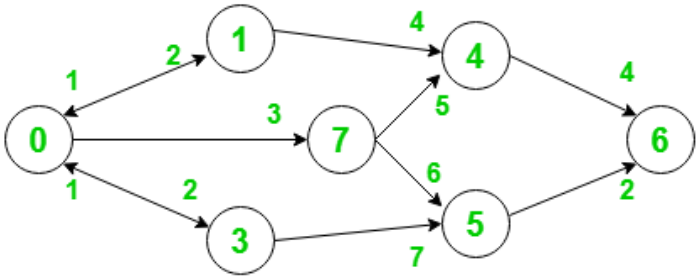
Peter's Plan:

- To calculate the shortest paths in the city road network.

Solving the City Road Network Problem with BFS & DFS

Peter models the city as a graph, where each intersection is a node and roads between intersections are edges. The task is to find the shortest path from the city center (node 0) to 7,5 & 6 intersections.

Graph Input:



Peter should use the BFS and DFS approaches to find the most efficient routes for 7,5 & 6 intersections (edges).

Conclusion:

Through this project, Peter learned how to use BFS and DFS for different real-world applications. BFS helped find the shortest paths in a city road network, while DFS was essential for exploring the university's computer network for connectivity. They tested different scenarios, from small to complex graphs, ensuring their algorithms worked efficiently.

Task 2
Optimal Delivery Route Planning

Scenario:
Imagine you're working for a delivery company named Wayne Enterprise, and your task is to deliver packages to multiple locations in Gotham city. The city is represented as a grid, where each intersection is a node and each road segment between intersections is an edge. The roads have varying travel times due to factors like traffic, construction, or road quality. The goal is to find the optimal delivery route from the company's warehouse to a customer's home while minimizing the travel time.

Problem Setup:
- Nodes: Each intersection in the city grid (represented by coordinates).
- Edges: The roads between intersections with travel times (weights) associated with them.
- Start Node: The warehouse location.
- Goal Node: The customer's home location.
- Cost: The time taken to travel between intersections (edge weights).
- Heuristic (h): The straight-line distance (Euclidean distance) between the current location and the goal location (customer's home).

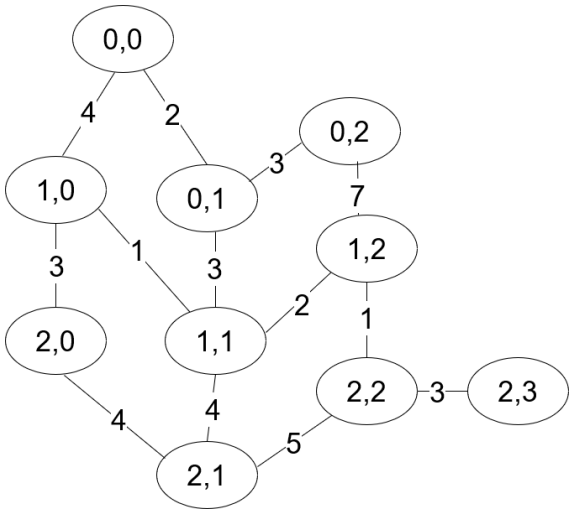
Steps:

1. Formulate the Problem:
   - Create a grid (e.g., 5x5) where each intersection is a node.
   - Assign a travel time (cost) to each road segment between intersections. Some roads may have higher travel times due to factors like traffic or construction.

2. A* Algorithm:
   - Use the A* search algorithm to find the shortest time path from the warehouse (start node) to the customer's home (goal node).
   - The algorithm should prioritize routes that minimize both the travel time and the estimated distance to the goal.

Example Grid Layout:

Consider the grid below, where each node represents an intersection and the numbers between nodes represent travel times (in minutes).



- The warehouse is at (0, 0) and the customer's home is at (2, 3).
- Travel time (cost) between nodes is shown between the intersections.

Task for You:

1. Implement the A* search algorithm to find the fastest route from the warehouse (0, 0) to the customer’s home (2, 3) using travel times as the cost and Euclidean distance as the heuristic.

2. Test the algorithm by simulating different traffic conditions:
   - Increase or decrease the travel times between intersections to simulate high traffic on certain roads.
   - Test how the route changes based on the updated traffic conditions.

3. Output:
   - The optimal path from the warehouse to the customer’s home.
   - The total travel time along the optimal path.
   - A comparison of travel times with different traffic conditions.

Task 3
Randomized Hill Climbing (RHC) for Resource Allocation

Problem Description:

You are tasked with optimizing the allocation of resources to various projects within a company named Daily Planet. Each project has specific requirements and constraints, and you need to maximize or minimize the overall benefit while adhering to the constraints.

Assignment Tasks:

1. Implement Randomized Hill Climbing

- **Implement** the Randomized Hill Climbing algorithm in Python.
- **Test** your implementation on a simple function optimization problem to ensure it works correctly.

Example Objective Functions:

- **Minimization Function:** A common function used in optimization problems for minimization is a quadratic function like where x is an integer:
  $$f(x)=(x-3)^2$$
  This function has a global minimum at x=3.
- **Maximization Function:** A common function for maximization is a negative quadratic or a simple sine function like where x is an integer:
  $$f(x)=-x^2+5$$
  This function has a global maximum at x=0.

2. Resource Allocation Problem

You are given multiple test cases for resource allocation. For each test case, you need to apply RHC to determine the best allocation of resources to maximize or minimize overall benefit while respecting the constraints.

Test Case Description:

- **Resources**: A total of 100 units of resource available.
- **Projects**: Each test case includes a list of projects with the following attributes:
  - **Project ID**
  - **Resource Requirement** (units of resource required for the project)
  - **Benefit/ Est. Time** (value of benefit or estimate time if the project is allocated the required resources)

Constraints:

- You can allocate resources to a project only if you have enough resources available.
- The total amount of resources allocated must not exceed the total available (100 units).

Test Cases:

1. **Test Case 1:**
    - Projects:
        - Project 1: Requires 20 units, Benefit = 40
        - Project 2: Requires 30 units, Benefit = 50
        - Project 3: Requires 25 units, Benefit = 30
        - Project 4: Requires 15 units, Benefit = 25
    - **Objective**: Maximize the total benefit.
2. **Test Case 2:**
    - Projects:
        - Project A: Requires 10 units, Est. Time = 15
        - Project B: Requires 40 units, Est. Time = 60
        - Project C: Requires 20 units, Est. Time = 30
        - Project D: Requires 25 units, Est. Time = 35
        - Project E: Requires 5 units, Est. Time = 10
    - **Objective**: Minimize the total Est. Time.
3. **Test Case 3:**
    - Projects:
        - Project X: Requires 50 units, Benefit = 80
        - Project Y: Requires 30 units, Benefit = 45
        - Project Z: Requires 15 units, Benefit = 20
        - Project W: Requires 25 units, Benefit = 35
    - **Objective**: Maximize the total benefit.

Submission Guidelines:

1. Implemented all the three tasks in Python Language and submitted three .py or .ipynb files.
2. **Brief Report** (1-2 pages):
    - Explain your findings.
    - Add SS of your implementations and results.
    - Submit in PDF file.
3. Submit in the CANVAS.
4. **Failure to follow all instructions will lead to point deductions.**