COIT20277 Introduction to Artificial Intelligence

Week 5

- Heuristic Search Techniques

BE WHAT YOU WANT TO BE
cqu.edu.au

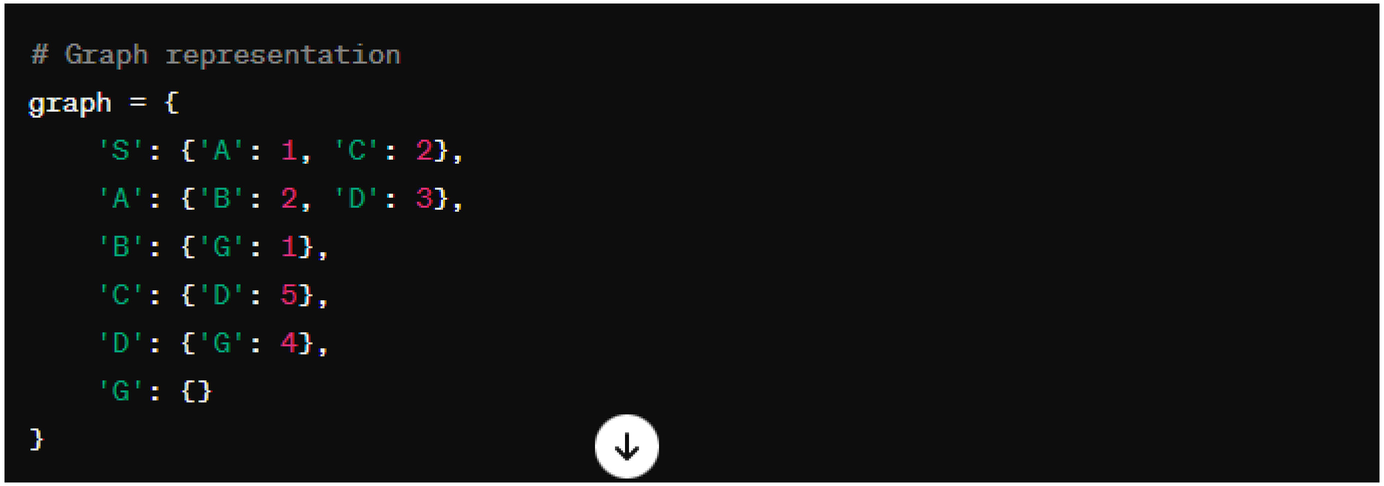
Acknowledgement of Country

I respectfully acknowledge the Traditional Custodians of the land on which we live, work and learn. I pay my respects to the First Nations people and their Elders, past, present and future

Heuristic Search

- You will practice implementing DFS (Uninformed Search) and A* (Informed Search) to find the shortest path between the start node and the goal.
- The objective is to understand how the search techniques work, and their characteristics.
- A Python program will be developed step-by-step through this tutorial. Your tasks include:
  - Input and execute the code in Jupyter Notebook.
  - Answer the accompanying questions.

Step 1: Implement the Graph Representation

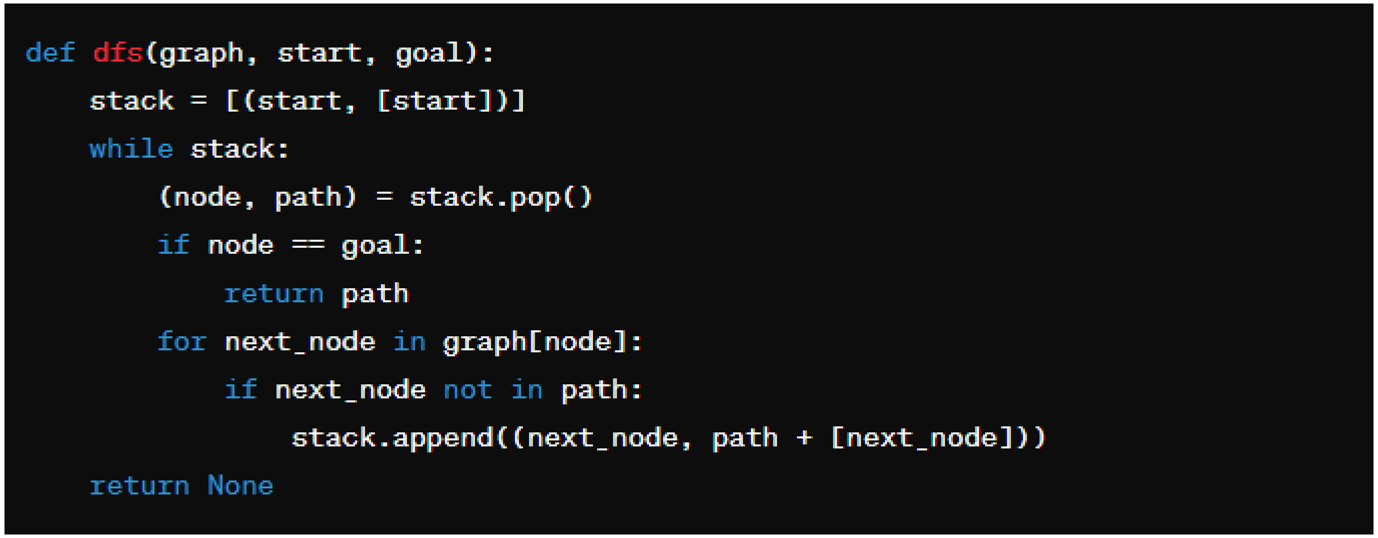


```python
# Graph representation
graph = {
    'S': {'A': 1, 'C': 2},
    'A': {'B': 2, 'D': 3},
    'B': {'G': 1},
    'C': {'D': 5},
    'D': {'G': 4},
    'G': {}
}
```

- Define nodes and their neighbors along with the cost of traversal between them.

- **Question 1:** What data structure did we use to represent the graph in Python?

Step 2: Implement Depth-First Search (DFS)

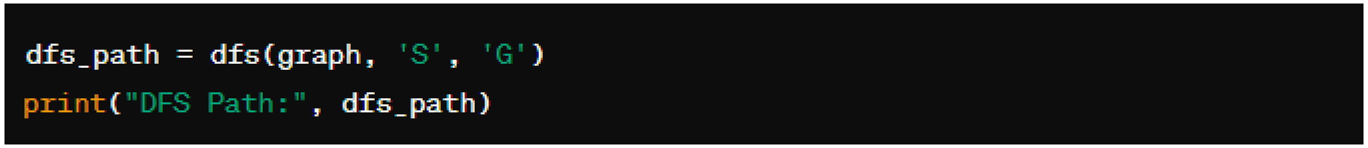


```
def dfs(graph, start, goal):
    stack = [(start, [start])]
    while stack:
        (node, path) = stack.pop()
        if node == goal:
            return path
        for next_node in graph[node]:
            if next_node not in path:
                stack.append((next_node, path + [next_node]))
    return None
```

- Define a function to perform Depth-First Search.

- **Question 2:** What data structure did we use to implement DFS?

Step 3: Test Depth-First Search (DFS)



```
dfs_path = dfs(graph, 'S', 'G')
print("DFS Path:", dfs_path)
```

- Call the DFS function to find the path from 'S' to 'G’.

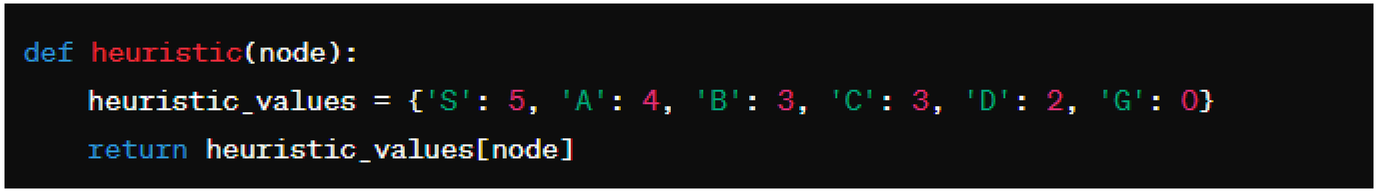
- **Question 3:** What is the output of the DFS function for the given graph?

Step 4: Implement A* Search

```
def astar(graph, start, goal, heuristic):
    open_list = [(0 + heuristic(start), 0, start, [])]
    closed_list = set()
    while open_list:
        open_list.sort()
        (f, g, node, path) = open_list.pop(0)
        if node == goal:
            return path
        closed_list.add(node)
        for next_node in graph[node]:
            if next_node not in closed_list:
                new_g = g + graph[node][next_node]
                new_f = new_g + heuristic(next_node)
                open_list.append((new_f, new_g, next_node, path + [next_node]))
    return None
```

- Define a function to perform A* search.
- **Question 4:** What data structure did we use to implement A* search?

Step 5: Define Heuristic Function

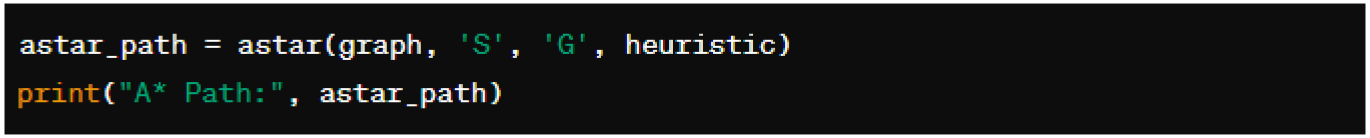


```python
def heuristic(node):
    heuristic_values = {'S': 5, 'A': 4, 'B': 3, 'C': 3, 'D': 2, 'G': 0}
    return heuristic_values[node]
```

- Define a simple heuristic function to estimate the distance from each node to the goal node.

- **Question 5:** What is the purpose of the heuristic function in A* search?

Step 6: Test A* Search



```
astar_path = astar(graph, 'S', 'G', heuristic)
print("A* Path:", astar_path)
```

- Call the A* search function to find the path from 'S' to 'G' using the heuristic function.

- **Question 6:** What is the output of the A* search function for the given graph using the defined heuristic function?

THANK YOU

TIME FOR DISCUSSION & QUESTIONS