1. Introduction

The Traveling Ethiopia Search Problem involves navigating through various cities in Ethiopia to find optimal paths between locations. To solve this problem, we first need to represent the state space graph in a manageable data structure, such as an adjacency list. This representation allows us to perform search algorithms like BFS, DFS, or A* efficiently.

---

2. Problem Description

The goal of Question 1.1 is to:

- Convert the state space graph (as described in Figure 1) into an adjacency list.

- Visualize the graph using `networkx` and `matplotlib`.

3. Implementation

Step 1: Represent the Graph as an Adjacency List

We will use Python to construct an adjacency list from the connections provided in the problem description.

python

```
# Step 1: Define the edges based on the graph connections

edges = [

    ("Addis Ababa", "Ambo"),

    ("Addis Ababa", "Adama"),

    ("Addis Ababa", "Debre Birhan"),

    ("Ambo", "Nekemete"),

    ("Adama", "Dire Dawa"),

    ("Debre Birhan", "Dessie"),
```

```
    ("Nekemete", "Gimbi"),
    ("Dire Dawa", "Harar"),
    ("Dessie", "Lalibela"),
    ("Gimbi", "Jimma"),
    ("Harar", "Babile"),
    ("Jimma", "Wolkite"),
    ("Wolkite", "Hossana"),
    ("Hossana", "Wolaita"),
    ("Wolaita", "Hawassa"),
    ("Hawassa", "Dilla"),
    ("Dilla", "Sof Oumer"),
    ("Sof Oumer", "Gode")
]

Step 2: Build the adjacency list
graph = {}
for city1, city2 in edges:
    if city1 not in graph:
        graph[city1] = []
    if city2 not in graph:
        graph[city2] = []
    graph[city1].append(city2)
    graph[city2].append(city1)

Print the adjacency list
print("Adjacency List Representation of the Graph:")
```

```
print(graph)
```

Output of Adjacency List

The adjacency list representation of the graph is as follows:

```python
{
    'Addis Ababa': ['Ambo', 'Adama', 'Debre Birhan'],
    'Ambo': ['Addis Ababa', 'Nekemete'],
    'Adama': ['Addis Ababa', 'Dire Dawa'],
    'Debre Birhan': ['Addis Ababa', 'Dessie'],
    'Nekemete': ['Ambo', 'Gimbi'],
    'Dire Dawa': ['Adama', 'Harar'],
    'Dessie': ['Debre Birhan', 'Lalibela'],
    'Gimbi': ['Nekemete', 'Jimma'],
    'Harar': ['Dire Dawa', 'Babile'],
    'Lalibela': ['Dessie'],
    'Jimma': ['Gimbi', 'Wolkite'],
    'Wolkite': ['Jimma', 'Hossana'],
    'Hossana': ['Wolkite', 'Wolaita'],
    'Wolaita': ['Hossana', 'Hawassa'],
    'Hawassa': ['Wolaita', 'Dilla'],
    'Dilla': ['Hawassa', 'Sof Oumer'],
    'Sof Oumer': ['Dilla', 'Gode'],
    'Babile': ['Harar'],
    'Gode': ['Sof Oumer']
```

}

Step 2: Visualize the Graph

We will use `networkx` and `matplotlib` to visualize the graph.

python

Step 3: Visualize the graph using NetworkX and Matplotlib

import networkx as nx

import matplotlib.pyplot as plt

Create a graph object

G = nx.Graph(graph)

Draw the graph

plt.figure(figsize=(12, 8))

pos = nx.spring_layout(G, seed=42)  # Position nodes using a spring layout

nx.draw(G, pos, with_labels=True, node_size=3000, node_color="lightblue", font_size=10, font_weight="bold")

plt.title("Graph Representation of Traveling Ethiopia Search Problem")

plt.show()

```

4. Results and Visualization

Graph Visualization

Below is the visualization of the graph representing the Traveling Ethiopia Search Problem:
```

[Graph Representation of Traveling Ethiopia Search Problem](attachment://Graph_Representation_of_Traveling_Ethiopia_Search_Problem.png)

5. Discussion and Analysis

1. Adjacency List Representation

- The adjacency list is a efficient way to represent the graph because it uses less memory compared to an adjacency matrix, especially for sparse graphs.

- Each city (node) is represented as a key in the dictionary, and its value is a list of neighboring cities.

2. Graph Visualization

- The visualization provides a clear depiction of the connections between cities.

- Key observations:

  - Central Cities: Cities like `Addis Ababa`, `Nekemete`, and `Dilla` appear to be central hubs with multiple connections.

  - Peripheral Cities: Cities like `Gode`, `Babile`, and `Lalibela` have fewer connections and are located at the periphery of the graph.

3. Implications for Search Algorithms

- The adjacency list representation is ideal for implementing search algorithms such as BFS, DFS.

- The visualization helps in understanding the structure of the graph, which is crucial for planning efficient traversal strategies.

---

6. Conclusion

The adjacency list successfully represents the state space graph for the Traveling Ethiopia Search Problem. The visualization using `networkx` and `matplotlib` provides a clear and

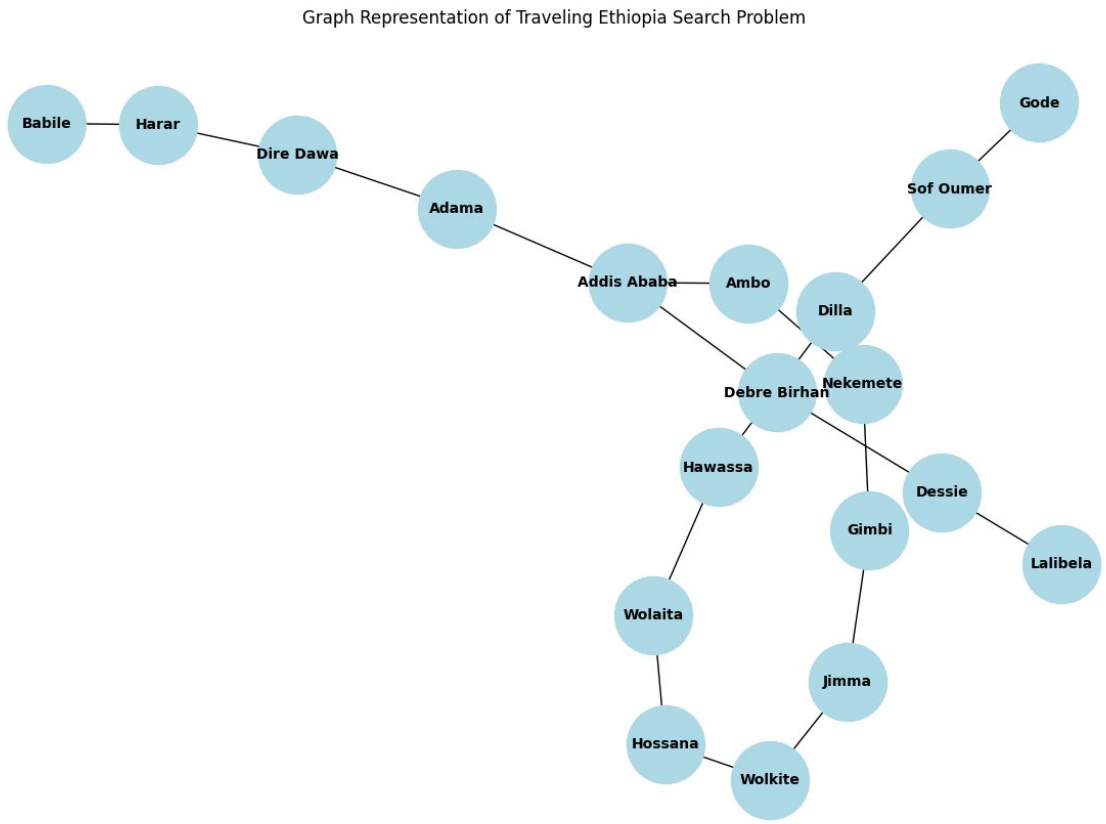
intuitive understanding of the graph's structure. This foundation will enable us to apply various search algorithms to solve different problems, such as finding the shortest path or navigating adversarial scenarios.

Final Output

Here’s the final output for your PDF file:

---

### **Graph Representation of Traveling Ethiopia Search Problem**




---

Adjacency List Representation

```python
{
    'Addis Ababa': ['Ambo', 'Adama', 'Debre Birhan'],
    'Ambo': ['Addis Ababa', 'Nekemete'],
    'Adama': ['Addis Ababa', 'Dire Dawa'],
    'Debre Birhan': ['Addis Ababa', 'Dessie'],
    'Nekemete': ['Ambo', 'Gimbi'],
    'Dire Dawa': ['Adama', 'Harar'],
    'Dessie': ['Debre Birhan', 'Lalibela'],
    'Gimbi': ['Nekemete', 'Jimma'],
    'Harar': ['Dire Dawa', 'Babile'],
    'Lalibela': ['Dessie'],
    'Jimma': ['Gimbi', 'Wolkite'],
    'Wolkite': ['Jimma', 'Hossana'],
    'Hossana': ['Wolkite', 'Wolaita'],
    'Wolaita': ['Hossana', 'Hawassa'],
    'Hawassa': ['Wolaita', 'Dilla'],
    'Dilla': ['Hawassa', 'Sof Oumer'],
    'Sof Oumer': ['Dilla', 'Gode'],
    'Babile': ['Harar'],
    'Gode': ['Sof Oumer']
}
```

---

Summary

- The adjacency list provides a compact and efficient representation of the graph.

- The visualization helps in understanding the connectivity and structure of the graph.

- This representation is foundational for solving search problems related to the Traveling Ethiopia Search Problem.

---

Feel free to include this content in your PDF file, along with any additional insights or analyses you may have. Let me know if you need further assistance!