Team Project – Data Structures
Spring 2024

Deliveries

- For this project you are allowed to submit any number of header files and implementation files.
- Write necessary comments in a way that it should be clear and concise, providing meaningful explanations of the code's purpose or functionality.
- Do not upload a code snippet.
- Do not submit any zip or compressed files, binary files, or any other generated files.
- Do not put the code inside the PDF file.
- Submission of unnecessary files or binary files in addition to source files will make you lose the points of the assignment.
- You can have a minimum of two members and a maximum of three members in your team.
- You must submit as a team. With the following required files.
    - Source files of your implementation.
    - A report of the project. The report should include the description of classes used in the project. The description and time complexity analysis of the implemented algorithms. Code sample runs (including edge cases).
    - Each team should prepare a short 3 to 4 minutes presentation. Students are required to execute their code and demonstrate its functionality during their presentation.
- We evaluate your code using the following C++ online compiler. You will not get points if your assignment does not compile with this. https://www.onlinegdb.com/online_c++_compiler

Learning Outcomes

- **Graph Theory and Data Structures**: You will learn about graph data structures, how to represent them in code, and how they can model real-world problems. You'll understand concepts like nodes (vertices) and edges, and how to store them using adjacency lists or matrices.
- **Algorithm Implementation:** Implementing graph algorithms like Dijkstra's algorithm , DFS, BFS, Prim's, Kruskal's will deepen your understanding of these algorithms, their time complexities, and their applications in finding the shortest or most efficient paths.
- **Data Processing and Manipulation:** Working with datasets, such as airport and flight information, will give you experience in data processing, filtering, and transformation, which are valuable skills in data analysis and software development.

Project Title:

Graph-Based Airport Connectivity and Flight Route Optimization System

Project Description

This project aims to develop a comprehensive system for modeling and optimizing airport connectivity and flight routes. The core of the system is a graph-based representation, where each node represents an airport, and edges represent direct flights between airports. This model allows for the application of various graph algorithms to analyze and optimize the network of flights.

Tasks:

1. **Airport Connectivity Map**: The system constructs a graph where airports are nodes, and direct flights are edges. This map serves as the foundation for analyzing the connectivity of the airport network, enabling tasks such as finding the shortest path between airports.
2. **Flight Route Optimization**: Building on the connectivity map, the system employs graph algorithms like DFS, BFS, Dijkstra's, Prim's and Kruskal's to find the most efficient routes between airports. This feature helps in identifying optimal routes for passengers and cargo, potentially reducing travel time and costs.
3. **Efficiency Check/ Evaluation of the Algorithms:**

Functionality:

Here is the list of things your project should support:

1) Read the information from the dataset (a csv file) and create a weighted **directed** graph G. Note that you need to consider two weights for each edge. One is the *Distance* and the other is *Cost.*
2) Find the shortest path between the given origin airport and destination airport. The algorithm should output both the path and the length of the path. The algorithm should provide the appropriate message if such path doesn't exist.
3) Find all shortest paths between the given origin airport and all the airports in the destination state. The algorithm should output all the paths and their lengths. The algorithm should provide the appropriate message if no such paths exist.
4) Find the shortest path between the given origin airport and destination airport with a given number of stops. The algorithm should provide the appropriate message if such path doesn't exist.
5) Count and display total direct flight connections to each airport. You should consider both outbound and inbound flights. For instance, if you can directly fly to Tampa airport only from Miami, Orlando, and Atlanta, the inbound count for Tampa airport would be 3. If you can directly fly from Tampa airport only to New York, the outbound count for Tampa airport is 1. The total number of direct flight connections for Tampa airport is 4. The list of airports should be sorted based on the total direct flight connections count, starting with the airports having the highest number of direct flight connections.
6) Create an undirected graph G_u from the original directed graph G using the following rules:
    a. For each pair of vertices u and v, if there is only one directed edge(either (u,v) or (v,u)) between them, you keep that single edge with its corresponding *cost* as an undirected weighted edge. You can ignore the *distance* on that edge.
    b. For each pair of vertices u and v, if there are two directed edges (u,v) and (v, u) between them, you keep the one with the minimum *cost* value as an undirected weighted edge. You can ignore the *distance* on that edge.
7) Generate a Minimal Spanning Tree utilizing Prim's algorithm on G_u that you created in the previous step. The algorithm will output both the content of the constructed MST and its total cost. In this step, for each edge you need to consider the *cost* as weight to minimize the total cost. In the event of a disconnected graph, the algorithm will appropriately notify that an MST cannot be formed. Note: A connected graph is defined as one where there exists a path between every pair of vertices.
8) Generate a Minimal Spanning Tree using Kruskal's algorithm on G_u that you created in the previous step. The algorithm will output both the content of the constructed MST and its total cost. In this step, for each edge you need to consider the *cost* as weight to minimize the total cost. If the graph is disconnected the algorithm should provide minimum spanning forest consisting of a minimum spanning tree for each connected component.

Dataset

The dataset, named "airports.csv," consists of four columns: Origin_airport, Destination_airport, Origin_city, Destination_city, Distance and Cost. This dataset is required for implementing all the tasks mentioned above. The Origin_airport and Destination_airport columns use the abbreviated format for airport names, such as PIT, ATL, ORD, etc. The Origin_city and Destination_city columns provide the actual names of the cities and states. The dataset contains 384 entries and includes flight information for 140 different airports.

Implementation Details

The implementation of the team must be done using **any data structures**, and you cannot utilize any STL library (except for the vector class), to implement any data structures. You must implement the project as a team.

Sample Test Cases

Sample output of Task 2 (illustrative example, does not contain actual airport data)

```
Shortest route from IAD to MIA: IAD -> ORD -> MIA. The length is 765. The cost is 234.

Shortest route from PIT to ACT: None
```

Sample output of Task 3 (illustrative example, does not contain actual airport data)

```
Shortest paths from ATL to FL state airports are:

Path            Length      Cost
ATL->MIA        595         123
ATL->TPA        398         213
ATL->JFK->MCO   1450        456
```

Sample output of Task 4 (illustrative example, does not contain actual airport data)

```
Shortest route from IAD to MIA with 3 stops: IAD -> ORD -> DFW-> ATL-> MIA. The length is
2065. The cost is 544.

Shortest route from PIT to ACT with 2 stops: None
```

Sample output of Task 5 (illustrative example, does not contain actual airport data)

Airport	Connections
ATL	12
ORD	11
MIA	11

Sample output of Tasks 7 and 8 (illustrative example, does not contain actual airport data)

Minimal Spanning Tree:

Edge	Weight
ATL - TPA	2
ORD - MIA	5
JFK - LAX	6

Total Cost of MST: 13

Tentative Rubric:

Report 20%
Presentation 10%
Implementing graph functionality correctly: 70%