Project 2
CSE 310: Data Structure and Algorithms
Professor : Yiran Luo

Team name: Mumbaikars
Team Members: Devendra Janyani and Jai Patel
ASU IDs: 1225359265 & 1226189018

Problems Encountered

- Ensuring the program handled various edge cases, such as graphs with no edges, all vertices having odd degrees, or disconnected components, required thorough testing and adjustments.

Known Bugs or Incomplete Implementations

- Disconnected Graphs: The current implementation of Dijkstra's algorithm may not handle disconnected graphs correctly, as it assumes all vertices are reachable from any starting vertex. If a graph is disconnected, the unreachable vertices will show incorrect shortest path values.
- Error in template file of vertex.cpp file with capital I in index line 9.

External References

- https://stackoverflow.com/questions/40247747/understanding-dijkstra-algorithm
- https://www.geeksforgeeks.org/python-program-for-dijkstras-shortest-path-algorithm-greedy-algo-7/

Design Decisions

The graph was represented using a 2D array to store the adjacency matrix, which simplified the process of edge insertion and checking connections between vertices. Although this choice increases memory usage for sparse graphs, it allows constant time complexity for accessing edge weights. The Vertex class encapsulates properties like the vertex index and degree. The Edge class is designed to manage properties of edges, including endpoints and weights. For this project, the weight was constant but could be adapted for variable weights. Dijkstra's Algorithm: Implemented manually to compute shortest paths from a source vertex to all other vertices. The algorithm was adjusted to handle direct array manipulations for maintaining the open set of vertices. Dynamic arrays were used for storing vertices and adjacency matrices, in line with project restrictions. Manual array management required careful attention to memory allocation and deallocation to prevent leaks.