Project Statement: Top K Shortest Path Problem with MPI and OpenMP

In this project students will tackle the K Shortest Path Problem using a combination of MPI for distributed computing and OpenMP for shared memory parallelization within MPI processes. The goal is to find the Kth shortest path among all nodes of a given graph.

- **Kth Shortest Path Problem:** The Kth Shortest Path Problem involves finding the Kth shortest path between two nodes in a graph. Unlike the standard shortest path problem, which aims to find the shortest path, the Kth shortest path problem seeks the Kth shortest path, which may not necessarily be unique.

Further details available at the following link
https://www.geeksforgeeks.org/1st-to-kth-shortest-path-lengths-from-node-1-to-n-in-given-graph/

- **Problem Description:** Your task is to implement a parallel solution for the Top K Shortest Path Problem using MPI and OpenMP. Here's an outline of the steps to follow:
  - **Input:** Read the graph representation from a file. The graph can be a weighted graph. You may need to perform pre-processing to adapt it according to the requirements of the code given in the above link.
  - **Initialization:** Initialize a distance matrix with the weights of the graph edges. Initialize the diagonal elements to zero (distance of a vertex to itself) and infinity for unreachable vertices.
  - **Parallelization Strategy:** Divide the computation of the Kth shortest path among MPI processes. Each MPI process will be responsible for exploring a subset of paths. Within each MPI processes the loops shall be parallelized using Openmp.
- **Testing with Provided Graphs:**
  - Utilize the graphs provided in the links below.
    1. Doctor Who
    2. Enron Email
    3. EU Email
  - You are required to execute the given code to compute Top K shortest path between 10 randomly selected pair of nodes and show the results. This will be followed by execution of the MPI+OpenMP code for the same 10 pairs of nodes and show the results.
- **Reporting Requirements:**
  - Discuss any challenges faced during preprocessing, implementation and testing, as well as any optimizations applied to improve performance.
  - Report the results of the experiments for each of the provided graphs, including the execution time and speedup achieved with different configurations.
- **Submission Requirements:**
  - Submit the source code of your MPI and OpenMP implementation along with a README file containing instructions for compilation and execution.
  - Include a detailed report documenting the experimental setup, results, analysis, and insights.