

**Project Report**

**Subject: PDC**

**Section: DS-N**

**Submitted to: Ma’am Parisa Salma**

**Fatima Azfar (20i-2658)**

**Raffay Ahmed (21i-1674**)

## Explanation:

### Data pre-processing:

* **tokenize**: Splits a string based on a given delimiter and returns a vector of strings. This is typically used for parsing input data from files

### Class Network Graph

* **Data Structure**: Uses a nested **unordered\_map** to store graph connections, where each node maps to another map that holds connected nodes and the weights of the edges between them.
* **Insert Edge:** Adds bidirectional (undirected) edges between nodes with a specified weight.
* **Initialize Distance Matrix**: Creates a distance matrix from the graph data, initializing direct distances from the connections and setting other distances to INT\_MAX. Diagonal elements (distance from a node to itself) are set to zero.
* **RetrieveKShortestPaths**: Uses a priority queue to find up to k shortest paths from a source node to a target node. It uses OpenMP for parallel processing to explore paths concurrently.

### MPI:

The master node (rank 0) performs the initial file reading and graph setup

### OPENMP:

Utilized within the retrieve KShortestPaths method to parallelize the exploration of different paths using multiple threads. This can accelerate the process of finding shortest paths by working on multiple possibilities concurrently

## Challenges faced during implementation: 1- Reading Graph

* Maintaining scalability
* Load Balancing
* Segmentation fault

## Optimizations done:

1. Optimized the graph structure
2. Finetuned MPI (by adjusting the number of threads and parallelizing tasks)

Execution time and speedup received:   
3 secs to execute (sequential time of the working algorithm)

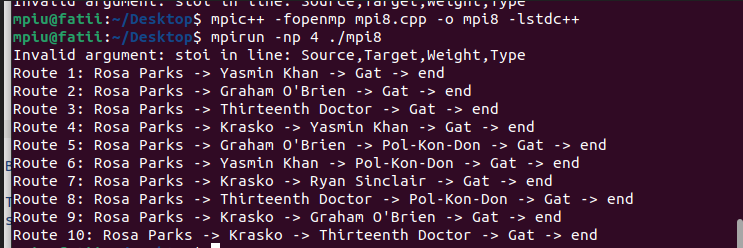
# Distance Matrix:

## K-shortest path:

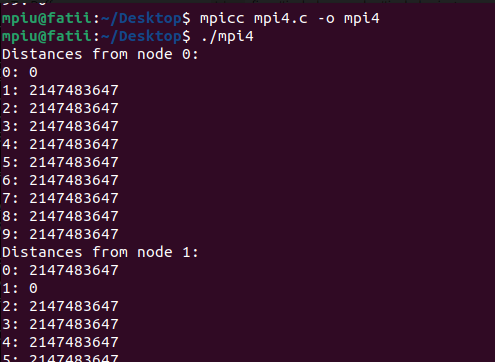
## Parallelization:

## Critical section:

## Output: (for doctorwho.csv)



## For enron-email:



## For EU-Email:

