HOMEWORK 1: MATRIX MULTIPLICATION (5 POINTS)

Parallelise the matrix multiplication algorithm using MPI as follows:

1. Initialise the two matrices with random numbers;
2. Scatter the first matrix to all the processes so that each process is responsible for multiplying an (approximately) equal amount of rows (use MPI_Scatter);
3. Broadcast the second matrix to all processes (use MPI_Bcast);
4. Multiply the matrix partitions in parallel;
5. Gather the product rows and build the result matrix (use MPI_Gather).

HOMEWORK 2: DIJKSTRA ALGORITHM (5 POINTS)

Parallelise the Dijkstra's algorithm using MPI as follows:

1. Initialise the graph with random edges and weights, select a random source node, and broadcast them to all processes (use MPI_Bcast);
2. Set the current node to the source node in all processes and partition the graph (i.e. select the rows in the adjacency matrix based on the rank) across all processes so that every process is responsible for computing the distance of approximately $\frac{|V|}{|P|}$ (own local) nodes;
3. One MPI process (e.g. rank 0, called in the following "master") iterates over the node list starting with the source node;
4. Each process (including the master) finds in its own local graph partition the node with the minimum distance to the current node and sends it to the master (use MPI_Reduce);
5. The master computes the global minimum and broadcasts it to all processes;
6. All processes set their current node to the global minimum, update the minimum distance to their neighbours, and go back to step 4;
7. The master collects the distance vectors from all processes.

HOMEWORK 3: ERATOSTHENES SIEGE (5 POINTS)

Parallelise the Eratosthenes Siege algorithm using MPI as follows:

1. All processes share the interval $\left[1, \sqrt{N}\right]$ and equally partition the remaining interval $\left[\sqrt{N}+1, N\right]$;
2. All processes iterate the shared interval $\left[1, \sqrt{N}\right]$ and eliminate the composite (i.e. non-prime) numbers;
3. For each prime number in the interval $\left[1, \sqrt{N}\right]$, each process eliminates the composite numbers greater than $\sqrt{N}$ in its own partition.

HOMEWORK 4: PERFORMANCE ANALYSIS (5 · 3 POINTS)

1. Choose a large problem size (i.e. matrix size, graph nodes, highest number) and execute each algorithm for 1, 2, 4, and 16 parallel processes;
2. Compute the speedup and efficiency for each algorithm;
3. Repeat the steps 1 and 2 until you find a problem size that gives you good speedup and efficiency results.