**Parallel Processing: Assignment 3**

**Algorithm:**

For this assignment, our goal was to parallelize the Dijkstra’s algorithm used to find the shortest paths in a graph from a given start node to every other node. To achieve such parallelization, our idea was to split the computation in the number of processes passed to the function. This way, each process finds its local minimum and locates the position of the minimum value in the graph. After each process finds their local values, they start forwarding it to the next process which then compares and updates their own local least and least position until it reaches the last process, which performs the last comparison and obtains the global least and least position. Lastly, the last process broadcasts the values back to the other processes so that they can also finish their shortest path computations.

**Implementation:**

Given *np* processes, the process with id 0 finds its local minimum by searching the graph from 0 until , where *n* = number of vertices in the graph. Then, process with id 1 finds its local values from until , and so on…

To scale this, we used the following condition:

For each process, we save their local *least* and *leastPos* inside the array *weights*, which gets forwarded to the next process. Apart from process with id 0, all the other processes receive the *weights* from the previous process, which gets stored in the array *receivedWeights*. Before sending their own local *weights* array, they first compare the values of *receivedWeights* with their found local *weights*. If the values received from the previous process are smaller than the ones the current process found locally, then their *weights* array gets updated, which then gets forwarded to the next process. When it reaches the last process that has id *np – 1*, it performs the last comparison, just like the ones before, but it does not send it back to the next process. Instead, the last process broadcast the *weights* – which are now the global minimum values – back to all the process that precedes it. After all the process contains the same information that reached the last process, they all finally compute and populate the *distance* array with the shortest paths from the graph based on the optimized global minimum *least* and *leastPos* values.