Consider the problem of finding “all-pairs-shortest-paths” in a weighted undirected graph $G = (V, E, w)$. Start with Dijkstra’s algorithm for finding the shortest paths from a single source (the vertex $s \in V$) to all other vertices:

$V_T \leftarrow \{s\}$

for all $v \in V - V_T$ do
  if $(s, v) \in E$ then $\ell[v] \leftarrow w(s,v)$
  else $\ell[v] \leftarrow \infty$
end for

while $V_T \neq V$ do
  find $u \in V - V_T$ such that $\ell[u] = \min \{\ell[v] : v \in V - V_T\}$
  $V_T \leftarrow V_T \cup \{u\}$
  for all $v \in V - V_T$ do
    $\ell[v] \leftarrow \min \{\ell[v], \ell[u] + w(u,v)\}$
  end for
end while

To find “all-pairs-shortest-paths” run Dijkstra’s algorithm $n$ times, each time for a different vertex as the source.

**Question 1**

Write in detail the serial Dijkstra’s algorithm for “all-pairs-shortest-paths”. Express the time needed for the serial algorithm to complete ($T_S$) as a function of the number of vertices $n$. Recall that the problem size is defined by the serial time $W = T_S$.

**Question 2**

Consider a homogeneous parallel machine with $p$ processors. Develop (and present in detail) a parallel version of Dijkstra’s single-source shortest path algorithm; make sure that you include all the needed communication steps.

**Question 3**

Consider a homogeneous parallel machine with $p$ processors. Develop (and present in detail) a parallel version of Dijkstra’s “all-pairs-shortest-paths” algorithm; make sure that you include all the needed communication steps.
Question 4

Assume that the communication time needed to transfer a block of \( M \) words between two processors is

\[ t_{\text{comm}} = t_{\text{startup}} + M \cdot t_{\text{word}}. \]

This communication time is formed by a fixed startup overhead (needed to create the message header etc.) plus the physical transfer time of the data “over the wire” (the time to transfer one word, \( t_{\text{word}} \), can be thought of as the inverse of the bandwidth of the interconnection network).

Perform an analysis of the parallel Dijkstra’s algorithms (both the single source and the all pairs shortest paths) and derive:

- The parallel time \( T_P \) as a function of the matrix size \( n \)
- The speedup \( S \) and the efficiency \( E \)
- The isoefficiency function \( n(p) \): how do we need to scale the graph size when the number of processors increases such that the efficiency of the calculation is kept constant?

Question 5

Implement in MPI the parallel Dijkstra’s “all-pairs-shortest-paths” algorithm. Run the algorithm on 1, 4, 9, and 16 processors and measure the time to completion. Draw the speedup versus number of processors curve and the efficiency versus the number of processors curve. Comment on what these curves mean from the point of view of scalability.