Dijkstra's Parallel Algorithm

We assume

- Graph, G = (N,E).
- Each edge (i,j) is associated with a cost w[i][j] that is stored in an adjacency matrix, Adj.

Step 1

- Set number of local vertices: N/P
  - We assume one process per processor
- Make a column partition of the adjacency matrix for the local vertices. Size of the adjacency matrix is also N/P
- Create arrays of size N/P:
  - Dist: stores the final length of the shortest path from the source to the destination
  - Pred: stores the predecessor
  - Marker: indicates if the local vertices have been permanently labeled or not. Initially it is set to 1

Step 2

- Initialize the distance from the source,s, to all it adjacent vertices.
  - This can be done by looking at the column of each local vertex. This indicates the incoming edges.
  - If adj[i][0] != 0 then
    - Set Dist[i] = Adj[i][0]

Step 3

If process, j, contains source vertex, s, then mark the source as permanently labeled.

Indicate this in the marker array by setting the corresponding index position of the source to 0.

Step 4

- Iterate N-1 times
- Iter ← 1. Got to Step 5.

Step 5

- Find the local minimum.
  - Look at the local vertex, i, and get the minimum shortest distance from source to i. This can be obtained by looking at the Dist array.
  - Also get the vertex that produced the minimum distance.

Step 6

- Compute the global minimum
  - One way, send local minimum to all processes and let the processes individually compute the global minimum.
- Now every process has
  - the global minimum vertex, u.
  - The global minimum distance, udist.
- The process that has u marks it as seen.

Step 7

- If Adj[i][u] != 0, then u is an incoming edge of i. Update the distance if necessary.
- That is,
  - Dist[i] > udist+Adj[i][u] then
    - Dist[i] = udist+Adj[i][u]

Step 8

- Increment iter
  - iter = iter+1.
- If iter < N, go to step 5.