

## Assignment-2

Karthik konyala - R11901409

Pavan Kalyan Anchula-R11904727

Gowtham Edumudi -R11912904

Harshad Hrithik Lanka- R11890176

### **Question:**

This is the non-parallelized version of Dijkstra's algorithm, which computes shortest paths sequentially.

### **Parallelization of Dijkstra's Algorithm using MPI**

Main aim to implement parallel dijkstra's algorithm makes it more effective to improve the performance of shortest path computation.

- 1.**MPI\_Bcast**. To ensure that all the procedures are using the same data to work.
2. Calculate the Local Minimum Distance of each process independently and also the corresponding node using the first for loop which iterates over all the nodes.
3. Using **MPI\_Allreduce** for global synchronization for the minimum distance across all the processes to make sure all of the processes work with the same global minimum.
4. **MPI\_Allgather** is used to collect local least positions and store them in an array which determines which node is corresponding to the global minimum distance.
5. **Updating Parallelized Distances**: Parallelizing distance update loop by distributing the nodes in the second for loop. Processors are tasked with updating the distances of the specific nodes respectively.
6. **MPI\_Allreduce** synchronizes and updates the global distance array in each iteration to make sure all processes have the updated shortest paths.
6. Original code is the non-parallelized version of dijkstra's algorithm, now we distribute the update of the nodes and minimize the global communication by which we can achieve speed computation when compared to the original code.
7. Each process only handles a set of nodes which reduces the overall computational time and makes the solution more scalable and efficient for large graphs.

### Compile:

```
mpicc -o assign_2 assign_2.c
```

### Run:

```
mpirun -np 4 ./assign_2
```

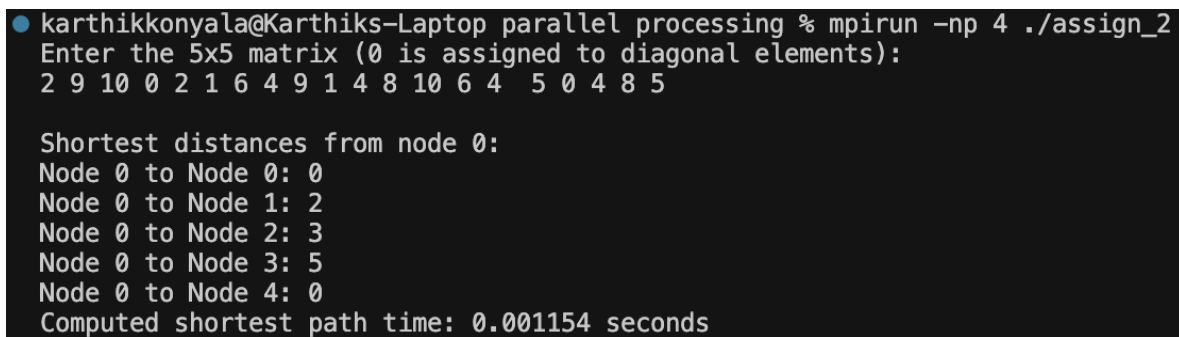
### testcase: (all the non-diagonal elements)

```
2 9 10 0 2 1 6 4 9 1 4 8 10 6 4 5 0 4 8 5
```

### output:

```
0 2 3 5 0
```

### Screenshot of the output



```
● karthikkonyala@Karthiks-Laptop parallel processing % mpirun -np 4 ./assign_2
Enter the 5x5 matrix (0 is assigned to diagonal elements):
2 9 10 0 2 1 6 4 9 1 4 8 10 6 4 5 0 4 8 5

Shortest distances from node 0:
Node 0 to Node 0: 0
Node 0 to Node 1: 2
Node 0 to Node 2: 3
Node 0 to Node 3: 5
Node 0 to Node 4: 0
Computed shortest path time: 0.001154 seconds
```