

**Final Project Report**

**Parallel Programming**

The Parallel Version Of Floyd Algorithm With Time Complexity

Of O(N Log\_2 P / Sqrt(P)) Using MPI And OpenMP

**Guidance**

Dr. Yu Zhuang

**Project by -**

Sangam Sahai

**Introduction**

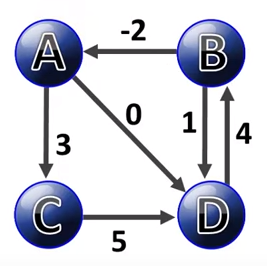
In computer science, the Floyd–Warshall algorithm is an algorithm for finding shortest paths in a weighted graph with positive or negative edge weights (but with no negative cycles).

The original time complexity of this Algorithm is n to the power of 3.

We can reduce the time complexity by the power of parallel Processing.

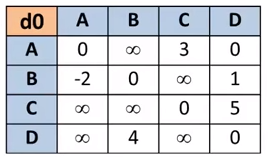
**Problem Instance**

Let us assume the input graph is as follows-

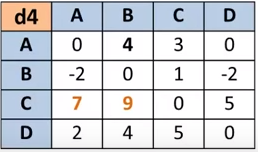


We will call this input graph as d0.

The way we represent this graph in the form of adjacency matrix is –



The expected output after the 4 th iteration is as follows (d4) –



**Methodology**

**This is a hybrid approach in which we use MPI for inter process communication and OpenMP for Multithreading. MPI is used for communication between the nodes. And OpenMP facilitates the multithreading on different cores of the same node, which have shared memory architecture.**

Instead of splitting the matrix on the basis of rows, we can split the bigger matrix into *smaller matrices or* ***grids***.

If the size of the original matrix is n\*n and if we are running p processes parallel, then the size of the smaller grids would be (n/sqrt(p) \* n/sqrt(p)).

Now, each of the p processes is assigned one such grid and then they work on the grids assigned to them simultaneously.

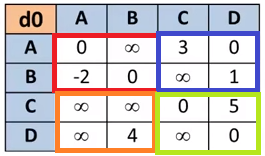
And further, by using the power of multithreading (*using OpenMP*), we will actually create more parallelism.

In all smaller grids , each process spawns n number of threads where n is the number of rows in that small grid assigned to the process.

And then those n threads work parallely to compute on the rows of that smaller grid.

If we run this code on 4 parallel processes, then we get 4 smaller grids of size (2\*2).

The grid breakup is as follows –



The process with rank 0 gets the grid colored in red.

The process with rank 1 gets the grid colored in blue.

The process with rank 2 gets the grid colored in orange.

The process with rank 3 gets the grid colored in green.

If we run the code on 4 parallel processes , then we get the following output .

However , please not the following points -

Here , the value *2147483647* represents *infinity*. In the code , it has been represented as *INT\_MAX* which is the maximum integer value in C.

And in the original screenshot of the output, I have highlighted the output of individual processes in respective colors which represent the smaller grids after the computation has been done.

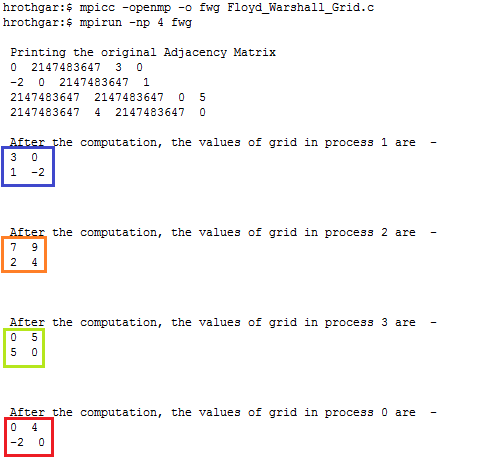
The command used for *compiling* is –

 mpicc -openmp -o fwg Floyd\_Warshall\_Grid.c

The command used for the *running* the code with 4 parallel processes is -

mpirun -np 4 fwg

The output screen shot with highlighted output is as follows -



These results clearly represent our expected output which is –



**Hence the desired output is achieved with a time complexity of –**

**O(N log\_2 P / sqrt(P)).**