CSC 346/646 Parallel Computation Spring 2014

Floyd-Warshall's Algorithm (All Points Shortest Path in a Directed Graph)

For our final programming project this semester, we will implement parallel Floyd-Warshall algorithm using MPI. This version is based on a 2-D block decomposition of the cost matrix. For simplicity, let us assume the number of processors is a perfect square (e.g., 1, 4, 9, 16, 25, 36, 49, 64), so that one processor can be assigned to each block. Further let us assume the number of graph vertices n is divisible by the square root of the number of processors. This assumption ensures that all blocks are square and of the same size.

A pseudocode description of the algorithm is given below:

Input: A weighted graph $G = (V, E)$ with weight function $f : V \times V \to \mathcal{R}^+ \cup \{+\infty\}$.
Let n denote the number of vertices in V. Number the vertices $V = \{v_1, v_2, v_3, ..., v_n\}$.

Output: An $n \times n$ matrix C such that $C_{i,j}$ is the cost of the shortest path from v_i to v_j.

Key Idea: Define a matrix $C^{(k)}_{i,j}$ as the cost of the shortest (restricted) path from v_i to v_j that goes through intermediate vertices numbered no higher than k.

Method:

Check that the number of processes p is a perfect square.
Process rank 0 reads the $n \times n$ input matrix ; check that n is divisible by $\sqrt{p}$.
Block-distribute each $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$ part of the input matrix

Initialize the $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$ block portion of the cost matrix $C^{(0)}$

// Let $P_{i,j}$ denote the processor in block-row i and block column j

for $k = 1$ to n do
{
each $P_{i,j}$ that has a segment of the kth row of $C^{(k-1)}$ broadcasts it to $P_{*,j}$
each $P_{i,j}$ that has a segment of the kth column of $C^{(k-1)}$ broadcasts it to $P_{i,*}$
each process $P_{i,j}$ computes its $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$ part of $C^{(k)}$
}

Process rank 0 collects the $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$ parts of $C^{(n)}$ and writes $C^{(n)}$ to a file.

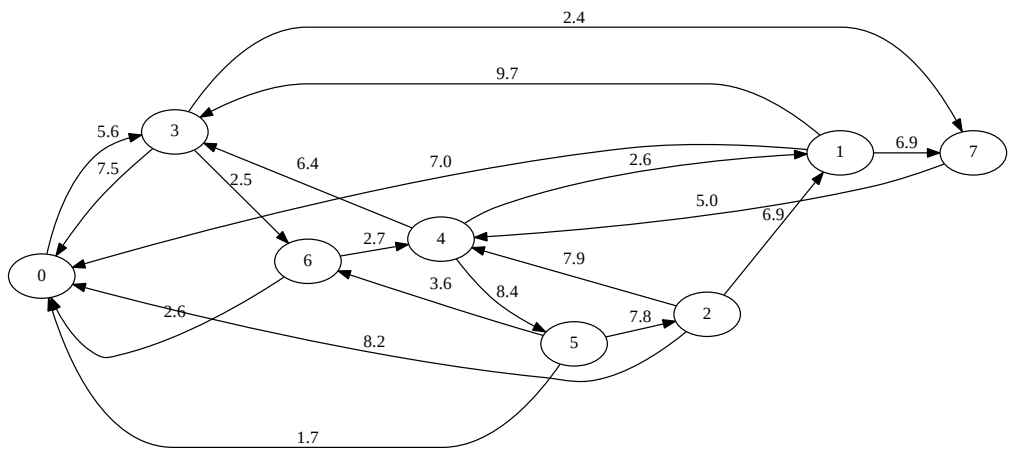
Input Data Format The input file represents a graph $G = (V, E)$ and a weight function: $f : V \times V \to \mathcal{R}^+ \cup \{+\infty\}$.

The input file is an ordinary text (ASCII) file. The first line of the input file is the number of nodes n in the graph. The subsequent n lines each contain n real numbers representing a matrix defining the weight function f. The edge weights of our graph are generally positive real numbers, but we also need a way to represent $+\infty$. We adopt the convention that

a negative number (usually -1.0) indicates $+\infty$. Also, the weight (cost) of a node going to itself is assumed to be zero. For example, a graph with 8 nodes could be given by the following file contents:

```
8
 0.0   -1.0   -1.0    5.6   -1.0   -1.0   -1.0   -1.0
 7.0    0.0   -1.0    9.7   -1.0   -1.0   -1.0    6.9
 8.2    6.9    0.0   -1.0    7.9   -1.0   -1.0   -1.0
 7.5   -1.0   -1.0    0.0   -1.0   -1.0    2.5    2.4
-1.0    2.6   -1.0    6.4    0.0    8.4   -1.0   -1.0
 1.7   -1.0    7.8   -1.0   -1.0    0.0    3.6   -1.0
 2.6   -1.0   -1.0   -1.0    2.7   -1.0    0.0   -1.0
-1.0   -1.0   -1.0   -1.0    5.0   -1.0   -1.0    0.0
```

The following diagram illustrates the corresponding graph:



Output Your program must gather the distributed cost matrix and write it to a file. Use the same format as the input file.

Sample Data

Two sample problems may be downloaded from *http://menehune.opt.wfu.edu/csc346*

1. A small 16×16 problem (for development and testing) named `A_16.dat`
2. A large 3360×3360 problem for doing run-time measurements in a file named `A_3360.dat`. Exclude file I/O time from your measurement. To obtain timing measurements, run your code with the number of processors taken from the set $\{1, 4, 9, 16, 25, 36, 49, 64\}$. Notice that 3360 is divisible by the square root of each of these numbers.

The shortest path cost matrix is also available on *menehune.opt.wfu.edu/csc346* so that you can check the correctness of your answer.

Turn In

1. Your source code
2. Plots of both the speedup and efficiency for the problem size 3360×3360.