Exam 2
COSC 420: High-Performance Computing

Dr. Joseph Anderson

Write your name on the exam. Staple all scratch paper before submission. No calculators or electronics other than your lab PC may be used during this exam. You may refer to any notes or previously written source code. **No browser windows may be open at any time, except for submission.**

Consider a graph with nodes labeled $1, 2, 3, \ldots, 8$. You are given the following adjacency matrix for a directed graph:

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

where $A_{i,j} = 1$ if there is an edge from i to j, and 0 otherwise.

Working in groups of at most two, use the MPI library to make as much progress as you can on the following tasks:

1. Sketch the graph with adjacency matrix A (on paper or in a digital format).
2. Using code from Lab, use the power method to compute the dominant eigenvalue of A and its corresponding eigenvector. State what both of these are.
3. In a distributed fashion, perform random walks on the graph as follows. Be sure to document what parallel mechanisms (communication, file sharing, etc.) will be used in your solution.
   (a) Select a random starting node, s
   (b) For a set number of iterations, k, start a random walk from s and step k times
   (c) Record which node the walk terminates at, and keep a count of which nodes are the endpoint of the random walk
   (d) Repeat steps (a)-(c) for up to 1000 times, on a number of processors, aggregating a vector v on a master node. The component $v[i]$ is the number of times that one of the random walks landed on node i.
      i. After each iteration, normalize this vector to have a sum of 1, so that it can be interpreted as a probability distribution on the vertices.
      ii. Does your code encounter any issues with numerical precision? Overflow? Underflow? How can you try to detect this programatically or otherwise?
      iii. Does it seem to converge with more iterations?
   (e) The v computed above can be interpreted as a probability distribution saying that if you randomly walk the graph, each coordinate is the probability that you end on the corresponding vertex. Which nodes are most and least likely? Does this align with the intuition by looking at the structure of the graph?

(f) Compare this vector with the eigenvector computed in task 1. Is there a relationship between the two?

(g) Submit your source code, documentation, and printouts of the executions that generate the final results (trim out extraneous output statements before you do this)

4. (Bonus, 10 pts): In the random walk, use the master node to generate a pseudo-random list of seeds for each worker node, and scatter this among the worker processes.