Lab Session 10

Submission deadline: May 7, 11:59pm

Description:

In this lab session, you will parallel the Prim's algorithm to find a minimum spanning tree.

Step 1: Generate a random graph

This java program (https://www.unf.edu/~wkloster/Algorithms/chapter4/randomgraph.java) can generate a random graph represented with an adjacency matrix. Slightly change the program such that it can generate a weighted graph. The weights are random numbers between 0 and 100.

Use your new program to generate a weighted random graph. The graph should have 100 vertices and use edge probability of 50. Use the system clock as the seed to generate random numbers. Save your adjacency matrix into a file.

Step 2: Implementation of a serial Prim's algorithm

Check the code shown in Slide 15 (see cse179_April11_graph) and implement the Prim's algorithm. You can find a sample implementation here. (http://www.geeksforgeeks.org/greedy-algorithms-set-5-prims-minimum-spanning-tree-mst-2/). Your implementation should be based on the sample implementation, and read the adjacency matrix from the file generated in Step 1.

Step 3: Parallelize the serial Prim's algorithm with OpenMP

As we discussed in the class, there are three loops in the Prim's algorithm. Use OpenMP to parallel these loops. Change different number of threads and measure performance.

Note that you must make sure your OpenMP code can execute correctly. In particular, the output minimum spanning tree of your OpenMP code should be the same as the output of your serial code.

Submit your solution code, including the serial code and parallel code, and your input adjacency matrix. Write a one-page report on how you parallel the code and what is the performance.