CSCE 435 Fall 2015

HW 7: Graph Algorithms

Due: 11:59pm Monday, December 7, 2015

Given a fully connected undirected weighted graph G = (V,E,w) with **positive** edge weights, the minimum spanning tree (MST) is defined as a spanning tree with minimum weight. A spanning tree of an undirected graph is a tree containing all nodes of the graph. The weight of a spanning tree is the sum of weights of all the edges in the tree.

You are provided a file called `mst.c` that has the following routines:

- `init_adj_matrix`: generates the adjacency matrix A of an undirected weighted graph where the weights between edges are selected randomly;
- `minimum_spanning_tree`: computes the MST for A using Prim's algorithm.

The code uses two data structures:

- `ADJ_MATRIX_t`: stores the adjacency matrix as a two-dimensional array. `A[i][j]` stores the weight of the edge between nodes i and j. The number of nodes in the graph is stored in `n`.
- `MST_t`: stores the MST in the form of two arrays called `node` and `weight`.

Prim's algorithm grows the MST one node at a time starting from a root node. A node u is included in the current MST if the weight of the edge from u to an MST node, say v, is the smallest weight edge among all edges between MST nodes and non-MST nodes. The array element `node[u]` stores v and `weight[u]` stores the weight of the edge (u,v). Thus, the spanning tree is represented by the pairs `(i,node[i])` for i = 0,..,n-1 (i ≠ root), and the corresponding edge weights are stored in `weight[i]`.

1. (80 points) You need to develop a parallel implementation of the minimum spanning tree algorithm that computes the MST of A. You are allowed to select one of the following approaches to parallelize the code:
   a) pthreads-based implementation on ADA that uses all the cores on a single ADA node; or
   b) CUDA-based implementation on ADA that uses a GPU device.

   A total of 20 points are reserved for performance of the code: speedup obtained by the parallel code over the serial code in `mst.c`.
2. (20 points) Execute the code for various values of n to demonstrate the parallel performance of your code. Include material such as plots and arguments as appropriate to convince the reader that your implementation delivers high parallel performance.

Submission: You need to upload the following to ecampus:

1. Problem 1: submit the file `mst.c`.
2. Problem 2: submit a single PDF or MSWord document with your response.