Project Proposal

Name of the project

Designing minimum-cost transportation network in rural area

Group Member

Qi Zheng
Xun Cao

Descriptions

I. Background

Our project is to help a non-profit institution named ***Habitat for Humility*** to design paved roads among a number of scattered villages in a large rural area. The goal of our project is to connect the villages by a simple transportation network. However, since the budget for this project is very tight, we want to minimize the total cost of paving the roads, while still ensure that local residents from any village can travel to any other village via paved roads.

The pavement work will be done on the existing natural dirt roads. As a result of differences in topography and distances, the estimated costs of paving different dirt roads would vary a lot. In our case, the construction cost for each dirt road is estimated by the joint effort of local governments, engineering consulting firms, and NGOs. This information has been provided to us.

To design this transportation network, we will implement Prim's algorithm to find a Minimum Spanning Tree (MST) of an undirected connected graph. The input file is a text file representing a graph (RMAT graph with power-law degree distributions and small-world characteristics) in linked list representation. Each node in the graph represents a village, and each edge in the graph corresponds to an existing dirt road. The estimated cost of each potential road is also provided as the weight of each edge. Our software will output a subgraph/tree which would yield the minimal total cost. The estimated total cost will also be calculated at the end of the simulation.

II. Implementation:

We would use Prim's algorithm to find the MST for solving this problem. Prim's algorithm is a type of Greedy Algorithm. To implement this algorithm, we would start from a random root node s, and greedily "grow" a tree T from s outward. At each step, we will add the cheapest edge e to T which has exactly one end point in T.

1. Write the parseEdges to parse edges from graph file to create our graph in C.
2. Write primMst function to compute the minimum spanning tree for the given graph.
   computeMst includes: Priority Queue and other implementations. D.

Pseudocode:

```
/* ***************************************************************************
*Basic idea: maintain set of explored nodes S. For each unexplored node v, maintain attachment*
* cost a[v] = cost of cheapest edge v to a node in S                                   *
****************************************************************************/
Prim(G, c) {
    //a[v]: the shortest edge between v and a node in explored set S
    for each (v ∈ V)
        a[v] ← ∞
    Initialize an empty priority queue Q
    For each (v ∈ V)
        insert v onto Q
    Initialize set of explored nodes S ← Ø

    while (Q != Ø) {
        u ← delete min element from Q
        S ← S U { u }
        For each (edge e = (u, v) incident to u)
            if ((v ∉ S) and ($c_e$ < a[v]))
                decrease priority a[v] to $c_e$
    }
}
```

III. Key data structures:

- For each unexplored node v in set V, the edge distance of it to any nodes in explored node set S will be stored as a heap data structure (a priority queue). The minimum a[v] will be popped out and add it to explored set S. Nodes will be added to the minimum spanning tree.
- Each minimum distance a[v] will be stored in a priority queue using heap as well.

3. Write outputMst function to output the edges chosen and the total cost. In the output file, the first line contains total cost, while the following lines list chosen paths.

IV. Workload Division:

Two group members will work together. And the work will be divided into a software

library and the simulation portion, each will be conducted by an individual.
-- the software library will include the algorithm to find MST with associated data structure.
-- the simulation part will assemble the graph, generate input files for MST algorithm, call the MST algorithm function to find the solution, do the post processing, output the results and write into text files.

Reference

Chakrabarti, Deepayan, Yiping Zhan, and Christos Faloutsos. *"R-MAT: A Recursive Model for Graph Mining."* SDM. Vol. 4. 2004.

Kleinberg, Jon, and Éva Tardos. *Algorithm Design*. Boston: Pearson/Addison-Wesley, 2006. Print.