PROJECT EULER

MINIMAL NETWORK

Sépage

PARIS

YONGWE Jean-Luc

24 février 2017

Université Lyon 2 , département DIS , master 2 SISE

Introduction

The *MINIMAL NETWORK* problem more known as the *MINIMUM SPANNING TREE* (MST) problem is a subset of the edges of a connected , edge-weighted undirected graph that connects all the vertices together , without any cycles and minimum possible total edge weight.
That is, it is a spanning tree whose sum of edge weights is as small as possible. More generally, any undirected graph (not necessarily connected) has a minimum spanning forest, which is a union of the minimum spanning trees for its connected components.

Applications

MST is fundamental problem with diverse applications.

- Cluster analysis.
- Real-time face verification.
- Find road networks in satellite and aerial imagery.
- Autoconfig protocol for Ethernet bridging to avoid cycles in a network.
- Approximation algorithms for NP-hard problems (e.g., TSP, Steiner tree).
- Network design (communication, electrical, hydraulic, computer, road).

Given a text file of graph weighted , the goal was to find out what is the best MST that could be built.
But in the building process there is some details we had to take care of..

All the following methods have been implemented with a scala 2.11 kernel with a jupyter notebook nad have deployed with map , zip .. distribution functions

First of all we had to get the data from the text file *'minimal network'* to the notebook and do the same for the artificial nodes that has been created on the fly , using the package *scala.io*

Convert '-' to 0

Basically the submitted graph is under the form of an Array of arrays of Array of Lists.
We choose to define a method called *replace* that replace all the occurencies of a character by another.
The result is a lists of String we had cut them by elements and convert each one of them into an Integer.

Graph definition

From this moment forward in order to move and work on a graph we had to define from the beginning , we start by definition the *nodes* by giving the name (String) store in the node with name file , we follow it up with the *edges* that built with a combine paired nodes (ex : *edge(Node("a") , Node("b")) , edge(Node("a") , Node("z"))..*)

Connected graph

We have define a method *extractnotconnected* that only collect *edgesweihted* where the weight is higher than zero

MST

We created a method *minweightededge* that return the lightest edge based on his weight , we also built a method *mw* that compare the weight of a given edge and compare it to all the edges of a subset of edges with the same node and return the ligthest of them (in case there is an edge in the subset ligther than the given weighted edge).

mst a tail-recursive method that progressively extract and store the lightest edges on every subset surrounding a particular node for all the distinct nodes.
The end-result is a graph with all the edgesweighted linked with the lightests weights.

Big O

The big O of my main method will be based on the Big O of others methods that it depends on..
every step of the way we have : one addition to the list that implide n comparison for the minimum of a certain subset edges that get into n comparisons between a certain weighted edge and all the edges of the subset

we ends up with : (n = search of the minimum weight , m = comparison current minimum weighted edge and ohters p = addition to the list)
$O(n * m * p) = O(n^3)$
Because we operate on the same amount of data every single time.

Conclusion (personal)

I really enjoy this challenge but I believe there was a way to go faster and have a better solution , my solution was based on linking one node to nearest and lightest in term of weight neighbor (this is a solution I came up with by drawing the graph on a paper , the web solutions were not easy to emulate starting with differents structures) .. to validate we may have try to drawn a really graph in 2Dimensions

REFERENCES

Wikipedia : $https : //en.wikipedia.org/wiki/Minimum_spanning_tree$
princeton.edu : $http : //algs4.cs.princeton.edu/lectures/43MinimumSpanningTrees.pdf$