Jonathan Steininger
Programming Assignment 6
Assignment6
4/29/2017

PROBLEM STATEMENT

The problem was to determine the minimum spanning tree between various cities.

METHODOLOGY

This program was implemented by first creating an easily searchable matrix containing the relevant distances between each of the cities. Then, the program "starts" at one city and looks at the distances between it and all other cities and chooses the to "travel" to the "nearest" city. The process is then repeated at the new city, but it refrains from looking at or attempting to travel to any cities that have previously been visited. It continues to travel to the next nearest city until it has traveled to all of the available cities. Additionally, it prints out the identity of each city as it visits them.

OBSERVATIONS

This program is markedly faster than the related traveling salesperson problem. In this problem, the program is only interested in finding the shortest path between any two points currently available to it. It does not necessarily provide the shortest overall path like tsp. As a result, this program is many times faster and more efficient than tsp (although it is important to reiterate that this program does not perform the same function as tsp – they have entirely different purposes, and they are only superficially similar). The recurrence relation of this algorithm is $T(n) = c(n-1) + c(n-2) + \ldots + c(n-(n-1))$, which implies $O(n^2)$. While $O(n^2)$ is not an ideal time complexity is vastly more preferable than tsp's $O(n!)$. The tsp program took 2 seconds for 13 cities, 12 minutes for 16 cities, and after 331 minutes (5.5 hours) it had not finished computing the answer for 19 cities, this program took less than 1 second to do 12, 13, 14, 15, 16, 19, and 29 cities combined.

RESULTS

run:
Number of cities: 12
Path of minimum spanning tree (city, cost):
(0, 0) (5, 60) (3, 39) (8, 60) (4, 42) (1, 71) (11, 59) (6, 78) (7, 78) (10, 76) (9, 23) (2, 129)
Total Cost: 715

Number of cities: 13
Path of minimum spanning tree (city, cost):
(0, 0) (5, 60) (3, 39) (8, 60) (4, 42) (1, 71) (11, 59) (6, 78) (7, 78) (10, 76) (9, 23) (2, 129) (12, 89)
Total Cost: 804

Number of cities: 14
Path of minimum spanning tree (city, cost):
(0, 0) (5, 60) (3, 39) (8, 60) (4, 42) (1, 71) (13, 34) (11, 121) (6, 78) (7, 78) (10, 76) (9, 23) (2, 129) (12, 89)
Total Cost: 900

Number of cities: 15
Path of minimum spanning tree (city, cost):
(0, 0) (5, 60) (3, 39) (8, 60) (4, 42) (1, 71) (13, 34) (14, 16) (12, 45) (2, 89) (9, 129) (10, 23) (7, 76) (6, 78) (11, 78)
Total Cost: 840

Number of cities: 16
Path of minimum spanning tree (city, cost):

(0, 0) (5, 60) (11, 46) (8, 68) (4, 42) (1, 71) (9, 65) (3, 39) (14, 34) (13, 61) (10, 64) (15, 143) (12, 57) (7, 104) (6, 178) (2, 386)
Total Cost: 1418

Number of cities: 19
Path of minimum spanning tree (city, cost):
(0, 0) (5, 60) (11, 46) (8, 68) (4, 42) (1, 71) (9, 65) (3, 39) (14, 34) (18, 49) (15, 64) (12, 57) (7, 104) (6, 178) (10, 147) (13, 64) (17, 32) (16, 61) (2, 274)
Total Cost: 1455

Number of cities: 29
Path of minimum spanning tree (city, cost):
(0, 0) (27, 34) (5, 52) (11, 46) (8, 68) (4, 42) (20, 50) (1, 41) (19, 49) (9, 25) (3, 39) (14, 34) (18, 49) (24, 52) (6, 72) (22, 111) (26, 74) (23, 38) (7, 48) (15, 84) (12, 57) (17, 128) (13, 32) (21, 36) (16, 47) (10, 106) (28, 272) (25, 36) (2, 78)
Total Cost: 1800

BUILD SUCCESSFUL (total time: 0 seconds)