**Project 3 – Closest Edge Insertion Heuristic**

Zachary Shumate

[CECS]

Speed School of Engineering

University of Louisville, USA

znshum01@louisville.edu

1. **Introduction** (What did you do in this project and why?)

This project had the goal of solving the Traveling Salesman Problem using a variant of the greedy heuristic. The variant that I used was the Closest Edge Insertion Heuristic that started with a trivial number of cities connected and created connections to new cities that replaced the closest edge to said city with new edges passing through it. This is the next step towards developing an efficient algorithm that can find the shortest possible path, Hamiltonian Cycle, between cities in a list.

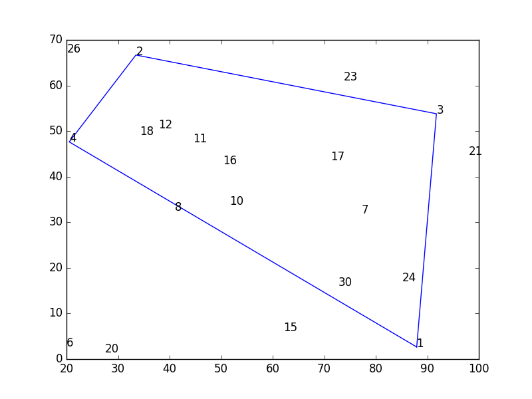
1. **Approach** (Describe algorithm you are using for this project)

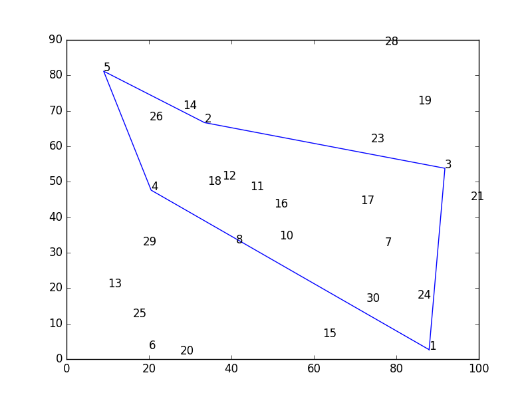
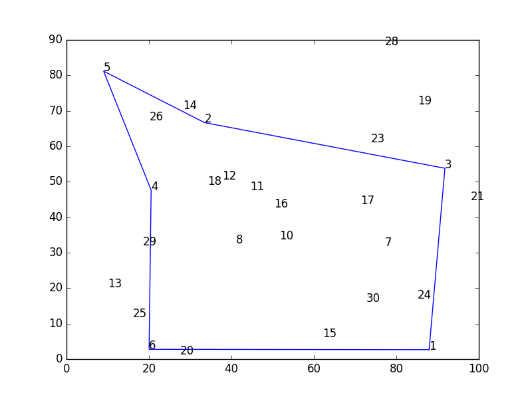
My Python script starts by accepting the files with the cities and coordinates that are put into a list. I started with the first three cities connected, which is trivially solvable. Looking at the remaining cities in order, the algorithm found the closest existing edge and then replaced it with two edges that connected the new city to the cities connected by the replaced edge. The closest edge is determined by calculating the lengths of the new edges that would replace it, the shortest total length belonging to the closest edge.

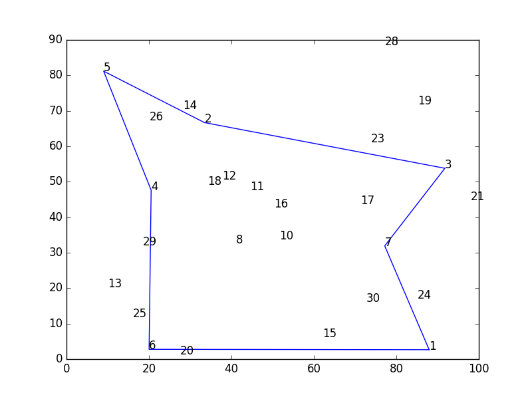
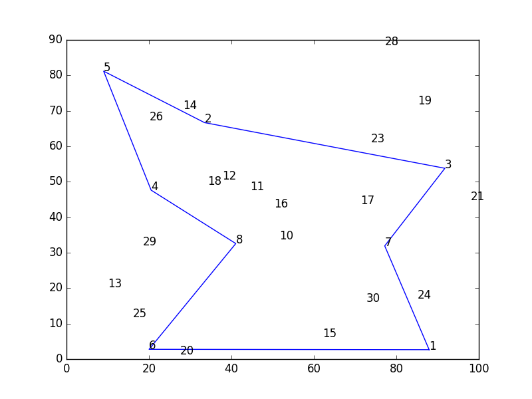
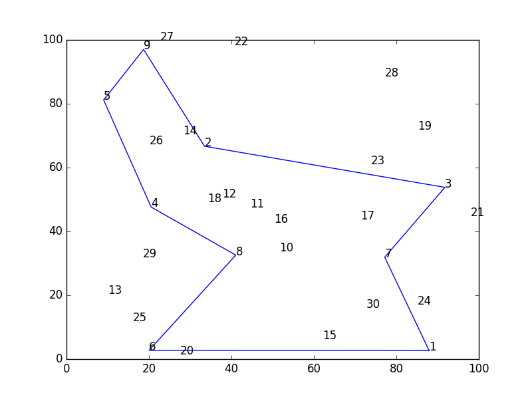
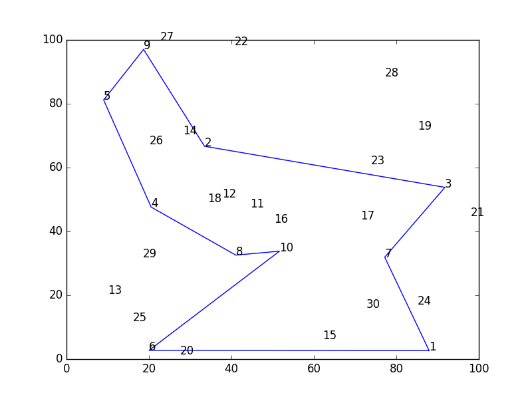
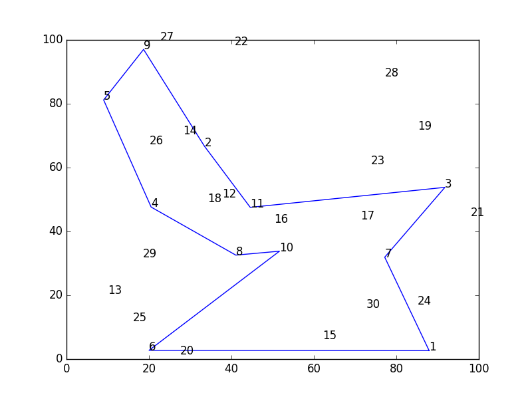
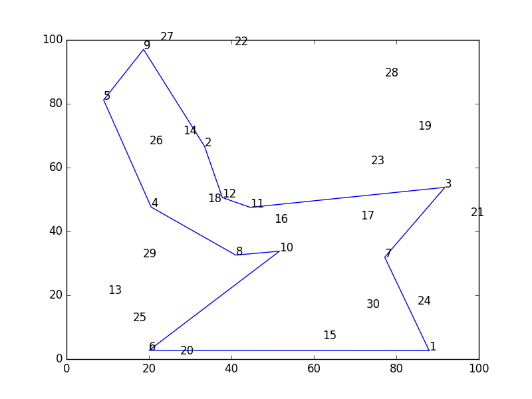
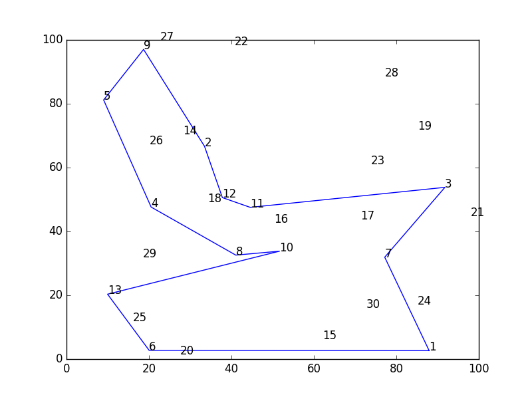
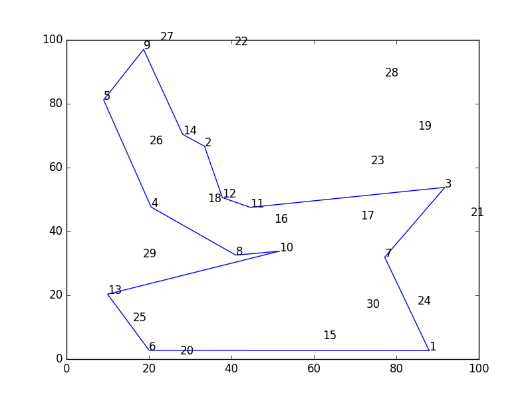
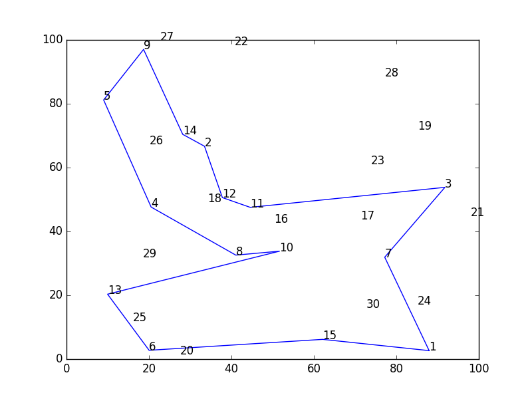
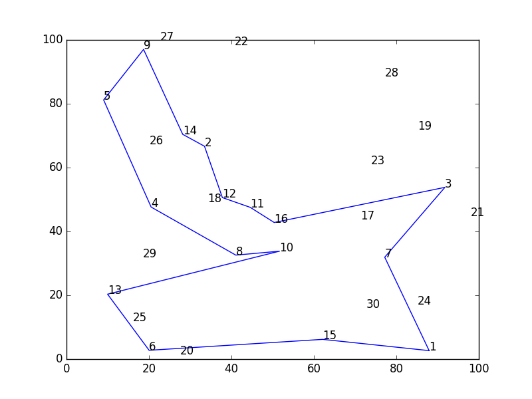
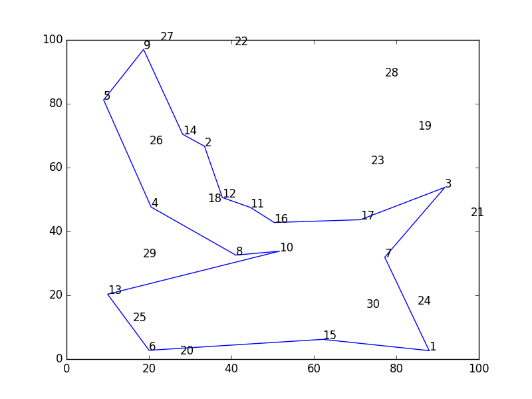
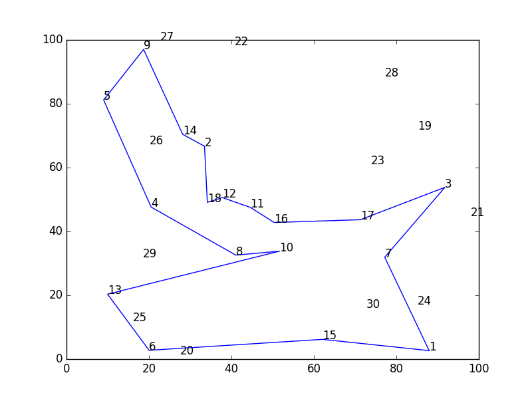
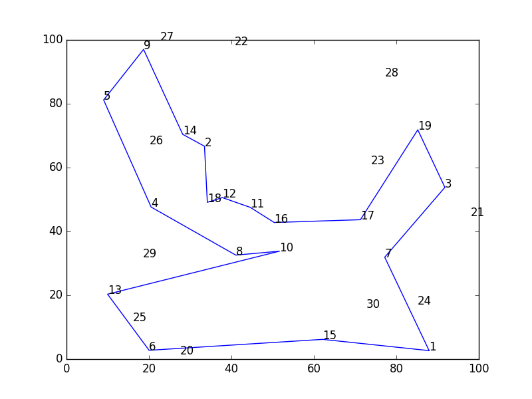
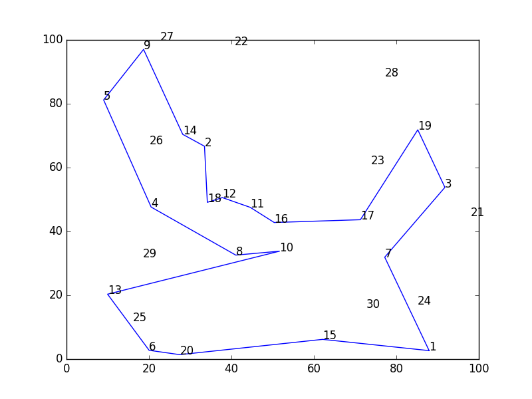
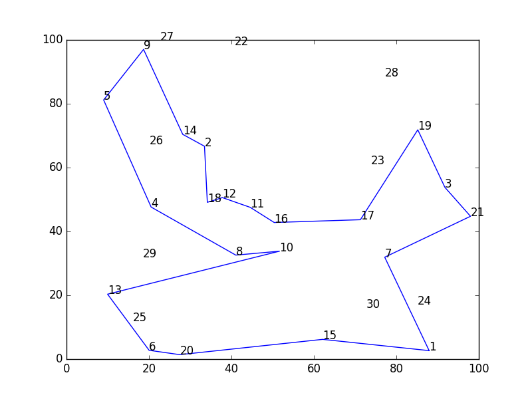
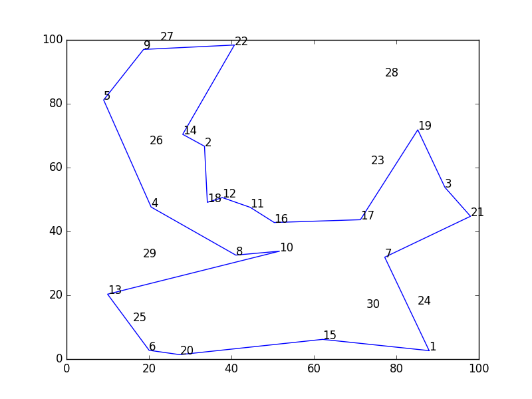
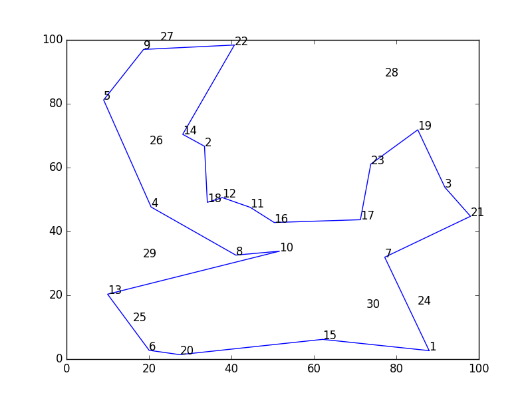
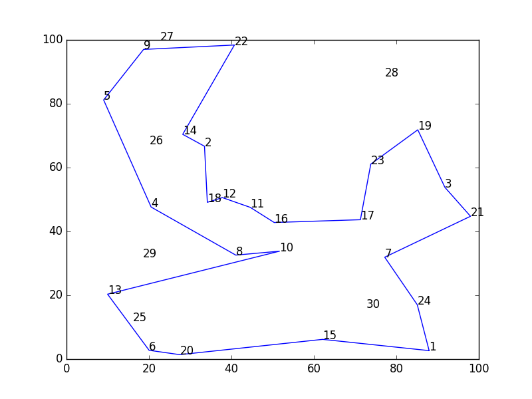
1. **Results** (How well did the algorithm perform?)
   1. **Data** (Describe the data you used.)

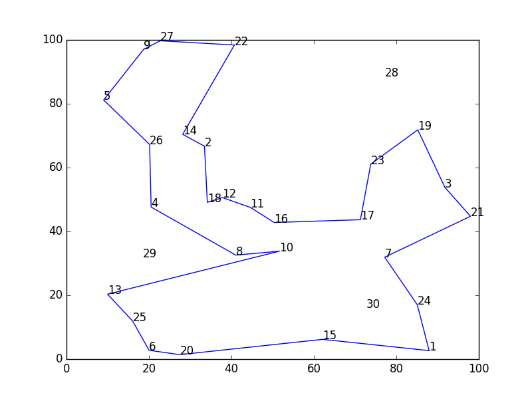
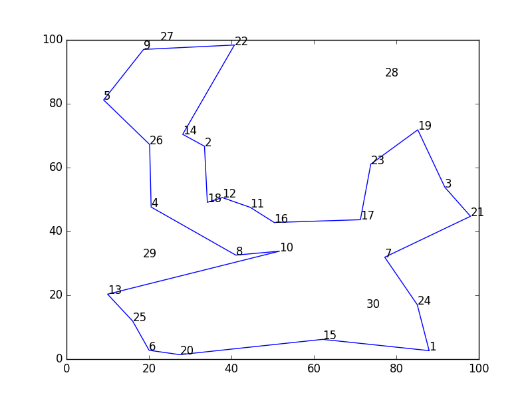
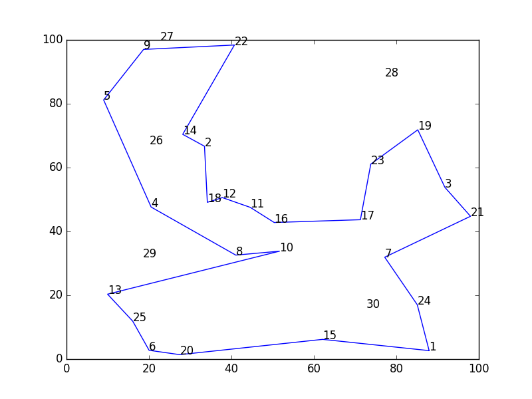
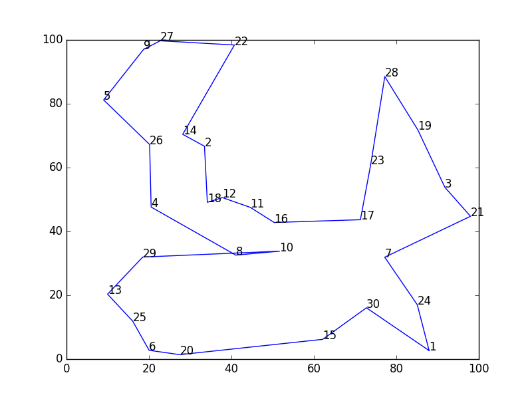
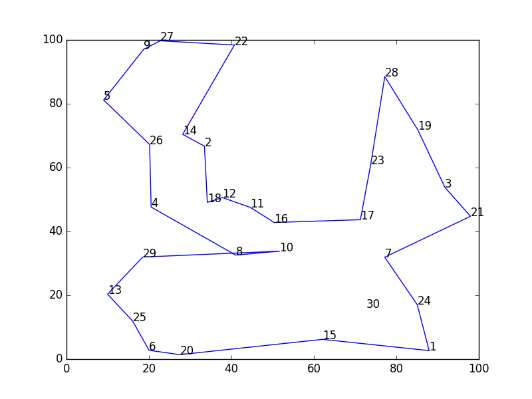
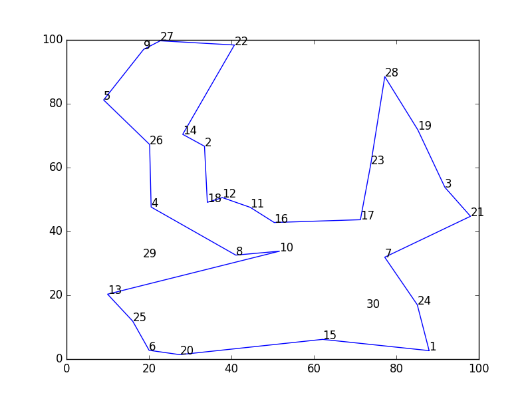
The data used for this project consisted of the provided thirty and forty city files. The forty city file was just the thirty city file with ten additional cities. The x and y coordinates for each city where unique.

* 1. **Results** (Numerical results and any figures or tables.)

(Pictures showing 4 cities to 30 cities)

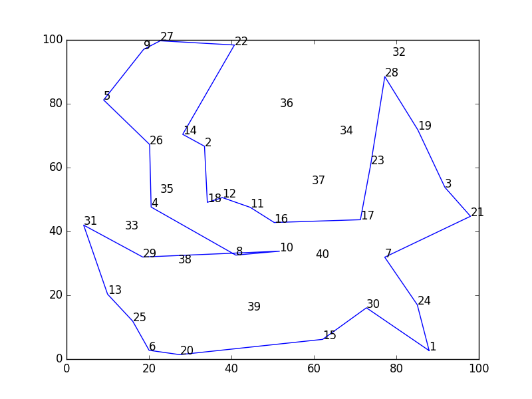
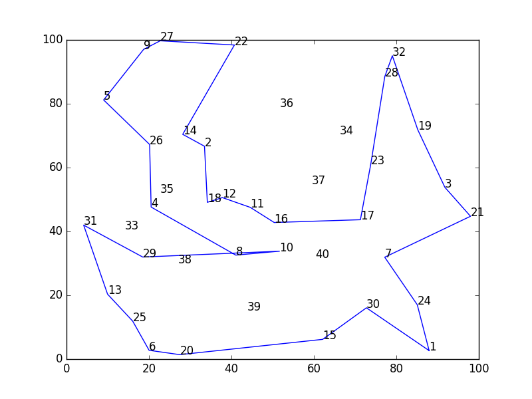
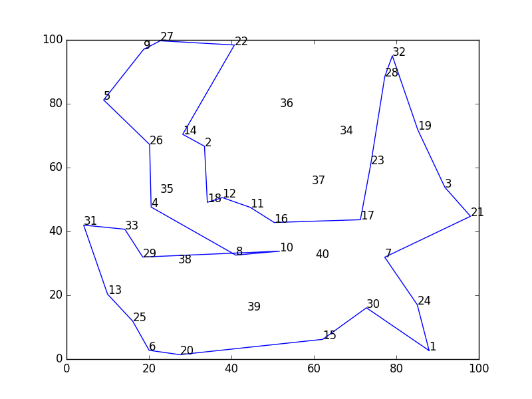


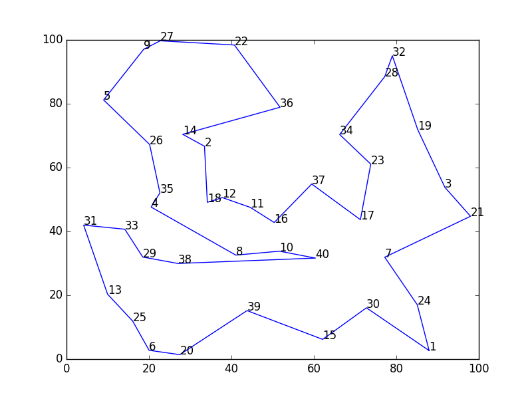
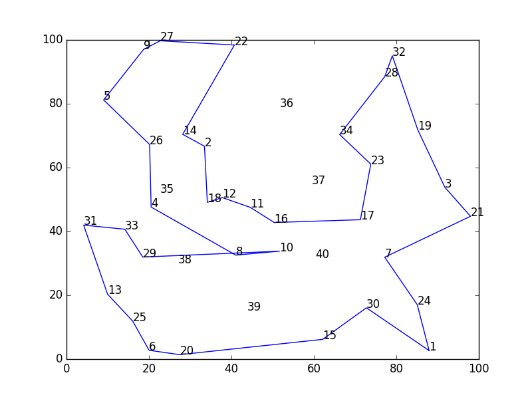
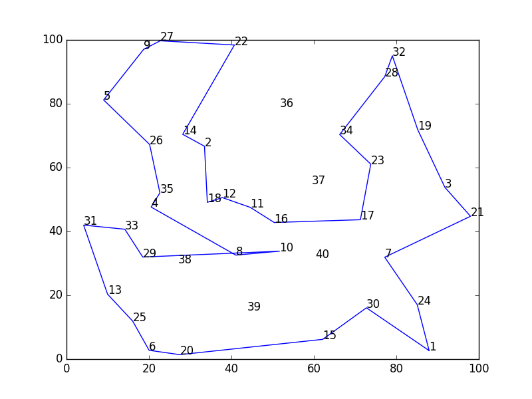
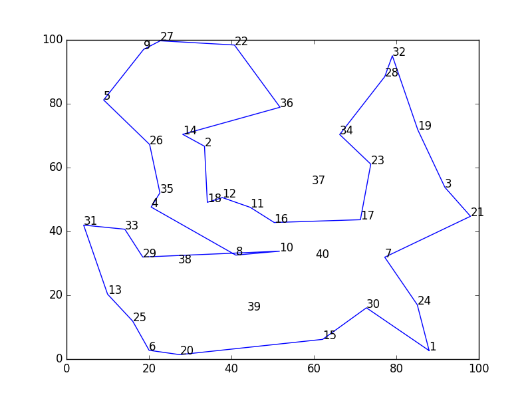
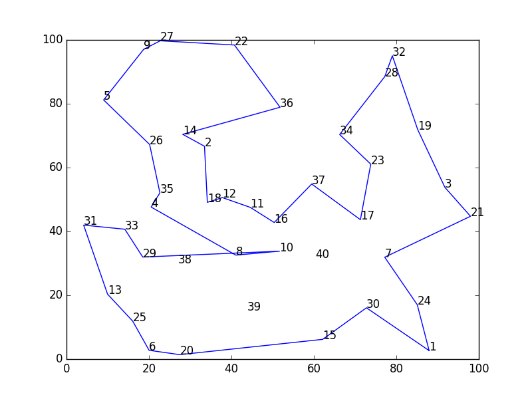
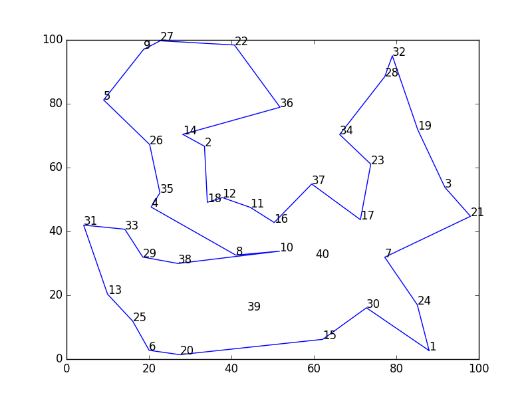
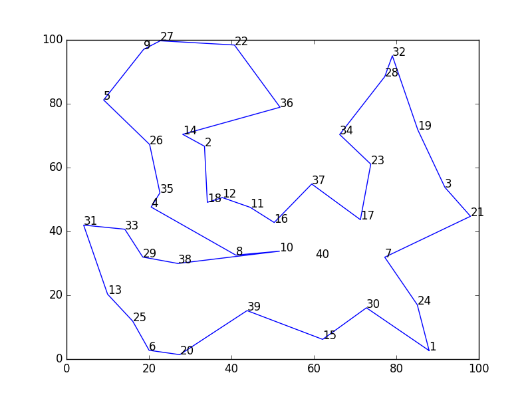


**30 cities**: {'**path**': [{'y': '2.658162', 'x': '87.951292', 'num': '1'}, {'y': '16.071047', 'x': '72.743919', 'num': '30'}, {'y': '6.183050', 'x': '62.129582', 'num': '15'}, {'y': '1.394696', 'x': '27.466659', 'num': '20'}, {'y': '2.761925', 'x': '20.032350', 'num': '6'}, {'y': '11.899167', 'x': '16.052736', 'num': '25'}, {'y': '20.337535', 'x': '9.961241', 'num': '13'}, {'y': '31.971191', 'x': '18.494217', 'num': '29'}, {'y': '33.808405', 'x': '51.658681', 'num': '10'}, {'y': '32.578509', 'x': '41.059603', 'num': '8'}, {'y': '47.633290', 'x': '20.526749', 'num': '4'}, {'y': '67.238380', 'x': '20.160527', 'num': '26'}, {'y': '81.185339', 'x': '9.006012', 'num': '5'}, {'y': '97.015290', 'x': '18.692587', 'num': '9'}, {'y': '99.725333', 'x': '22.730186', 'num': '27'}, {'y': '98.400830', 'x': '40.730003', 'num': '22'}, {'y': '70.415357', 'x': '28.186895', 'num': '14'}, {'y': '66.682943', 'x': '33.466597', 'num': '2'}, {'y': '49.113437', 'x': '34.156316', 'num': '18'}, {'y': '50.599689', 'x': '37.806330', 'num': '12'}, {'y': '47.541734', 'x': '44.563128', 'num': '11'}, {'y': '42.796106', 'x': '50.376904', 'num': '16'}, {'y': '43.671987', 'x': '71.285134', 'num': '17'}, {'y': '61.076693', 'x': '73.799860', 'num': '23'}, {'y': '88.503677', 'x': '77.196570', 'num': '28'}, {'y': '71.837519', 'x': '85.201575', 'num': '19'}, {'y': '53.807184', 'x': '91.778314', 'num': '3'}, {'y': '44.746239', 'x': '97.985778', 'num': '21'}, {'y': '31.922361', 'x': '77.181310', 'num': '7'}, {'y': '17.029328', 'x': '85.076449', 'num': '24'}, {'y': '2.658162', 'x': '87.951292', 'num': '1'}], '**distance**': 504.7628346387057}

Script completed in 18.4922430515 seconds

(Pictures showing 31 cities to 40 cities)

****

****

**40 cities**: {'**path**': [{'y': '2.658162', 'x': '87.951292', 'num': '1'}, {'y': '16.071047', 'x': '72.743919', 'num': '30'}, {'y': '6.183050', 'x': '62.129582', 'num': '15'}, {'y': '15.189873', 'x': '43.790850', 'num': '39'}, {'y': '1.394696', 'x': '27.466659', 'num': '20'}, {'y': '2.761925', 'x': '20.032350', 'num': '6'}, {'y': '11.899167', 'x': '16.052736', 'num': '25'}, {'y': '20.337535', 'x': '9.961241', 'num': '13'}, {'y': '41.981262', 'x': '4.153569', 'num': '31'}, {'y': '40.690329', 'x': '14.145329', 'num': '33'}, {'y': '31.971191', 'x': '18.494217', 'num': '29'}, {'y': '29.957806', 'x': '27.042484', 'num': '38'}, {'y': '31.645570', 'x': '60.375817', 'num': '40'}, {'y': '33.808405', 'x': '51.658681', 'num': '10'}, {'y': '32.578509', 'x': '41.059603', 'num': '8'}, {'y': '47.633290', 'x': '20.526749', 'num': '4'}, {'y': '52.076785', 'x': '22.656941', 'num': '35'}, {'y': '67.238380', 'x': '20.160527', 'num': '26'}, {'y': '81.185339', 'x': '9.006012', 'num': '5'}, {'y': '97.015290', 'x': '18.692587', 'num': '9'}, {'y': '99.725333', 'x': '22.730186', 'num': '27'}, {'y': '98.400830', 'x': '40.730003', 'num': '22'}, {'y': '78.902954', 'x': '51.797386', 'num': '36'}, {'y': '70.415357', 'x': '28.186895', 'num': '14'}, {'y': '66.682943', 'x': '33.466597', 'num': '2'}, {'y': '49.113437', 'x': '34.156316', 'num': '18'}, {'y': '50.599689', 'x': '37.806330', 'num': '12'}, {'y': '47.541734', 'x': '44.563128', 'num': '11'}, {'y': '42.796106', 'x': '50.376904', 'num': '16'}, {'y': '54.852321', 'x': '59.477124', 'num': '37'}, {'y': '43.671987', 'x': '71.285134', 'num': '17'}, {'y': '61.076693', 'x': '73.799860', 'num': '23'}, {'y': '70.360424', 'x': '66.258736', 'num': '34'}, {'y': '88.503677', 'x': '77.196570', 'num': '28'}, {'y': '95.034639', 'x': '79.027680', 'num': '32'}, {'y': '71.837519', 'x': '85.201575', 'num': '19'}, {'y': '53.807184', 'x': '91.778314', 'num': '3'}, {'y': '44.746239', 'x': '97.985778', 'num': '21'}, {'y': '31.922361', 'x': '77.181310', 'num': '7'}, {'y': '17.029328', 'x': '85.076449', 'num': '24'}, {'y': '2.658162', 'x': '87.951292', 'num': '1'}], '**distance**': 603.0734633454522}

Script completed in 26.0532128811 seconds

1. **Discussion** (Talk about the results you got and answer any specific questions mentioned in the assignment.)

This algorithm found the resulting path for the provided datasets quickly, when you consider that much of the time the script took to run was from generating the graphs. The previous projects would have taken much longer to complete a comparably sized dataset, especially brute forcing it. This method, though quick, does not generate the most efficient path, since this greedy approach occasionally replaces an edge that results in overlap of edges.

1. **References** (If you used any sources in addition to lectures please include them here.)

I did not use any additional resources. All of the code was written by me.