# CSC 340 Project 2

14 February 2018

Due: 11:59 PM, 15 March 2018

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Instructions

This project retains all the rules of a take home test. You may use your book and software that you have developed yourself for this course. If you find it convenient to do so, you may also use a plotting tool such as Excel, SAS, or gnuPlot to render graphics. You ***may not*** use commercial software or statistical tools, such as those included in Excel or SAS, or MatLab, to generate the results you report. You may not use the built-in or library matrix or vector operations of any programming language to find matrix inverses or determinants, or to perform matrix or vector operations such as addition, subtraction, multiplication, or scalar multiplication. ***The results that you report must arise from algorithms that you yourself have implemented.*** (You may use commercial software ***only*** to check your work.) ***Again, the computational results required for this project must arise from your own personal implementation of the required algorithms in a programming language such as Python, Java, C, or C++.*** You are required to do your own work in implementing algorithms, creating supporting code, and applying your software to these problems. You must submit your source code together with the examination results.

You are expected and required to ***write*** your findings as you would document the results of laboratory experiments. In order to assure that all questions receive complete responses, please embed your responses into the Problems section of this project document itself (and prior to submission, please delete both the **Instructions** and the **Timeliness** sections). Be sure to include in the report any response that you wish to have considered for credit; items linked elsewhere may not be accepted for credit. ***In order to receive credit for any given problem, you must also be prepared to provide a live demonstration of your implementation of the associated algorithms and to explain any component of the source code that you provide.*** Your submission document may contain a mixture of program output, charts, or written work, as appropriate.

## Timeliness:

In order to receive full credit, the project report and supporting documentations or codes must be delivered by the due date and time. Late submissions will accrue a penalties as follows: for the first 24-hour period immediately after the due date, 10%; for the next 24-hour period, 20%; for the next 24-hour period, 40%; for the next 24-hour period, 80%; thereafter, no credit.

## Problems:

Consider the data in the file “eigendata.txt” which is posted to the course Web site. The data represent measurements describing the locations of the objects in the class illustrated in Fig. 1.

Fig. 1. A visualization of the measurements in the “eigendata.txt” file

1. Eigenvectors and eigenvalues (30 points)
   1. For the class data given in the “eigendata.txt” file, find and report:
      1. The ***mean*** vector and the ***covariance*** matrix. (5 points)
      2. The ***trace*** of the covariance matrix. (5 points)
      3. The ***determinant*** of the covariance matrix. (5 points)
      4. All ***eigenvalues*** for the covariance matrix. (5 points)
      5. A ***unit length*** eigenvector for each of the eigenvalues. (5 points)
   2. ***One a single chart, plot the data and the class mean*** for the class, as well as ***the eigenvectors*** drawn emanating from (with their tails located at) the class mean and their heads translated (in the mathematical sense) accordingly. You should rescale the eigenvectors to convenience lengths so that they can be seen easily in the plot. (5 points)
2. More on eigenvalues and eigenvectors (20 points):   
   Estimate the roots of the polynomial:  
   p\*(x) = 30x5 – 139 x4 – 1689 x3 + 4903 x2 - 2733 x – 756, i.e., find and report all values of r such that p\*(r) = 0. Note that p\*(x) is NOT a monic polynomial. You must:
   * 1. Find a monic polynomial p(x) that has the same roots as p\*(x), and then write the companion matrix **A** for p(x). (5 points) (Hint: This is NOT a programming problem. Simply write down the companion matrix, appropriately labeled.)
     2. Use your implementation of ***Leverrier’s algorithm*** to ***find*** the coefficients for and ***report*** the characteristic equation for the matrix **A**. (5 points)
     3. Use your implementation of the ***direct power method*** to find an estimate for the largest eigenvalue for the matrix **A**. (5 points)
     4. Repeat steps i-iii with the deflated polynomials and corresponding companion matrices to find estimates for the other roots of *p*? (5 points)
3. A Traveling Salesperson Problem (50 points)

Fig. 2, A TSP map

Consider a collection of cities, labeled A through N, as indicated in Fig. 2, with coordinates given below in the TSP Data section of this project. Find an ordering (a permutation of the city labels) for taking a least cost, round trip that visits each of the cities, except the starting city, exactly once. The cost of the trip will be represented by the cumulative distance traveled and the trip cost must include the cost of returning to the starting city.  
  
You are to compare the relative merits of four alternative methods of finding or estimating a least cost trip. Recall that a [permutation](file:///C:\Users\tagliarinig\Desktop\fromE\MyStuff\CSC\Courses\340\532\PermutationTester.java) is just a ***one-to-one*** function of a set S ***onto*** itself; for example, if the cities were labeled 1,…,n, then any bijective function, *p*: {1, 2, …, n} →{1, 2, …, n} would permute the city labels.

* 1. Exhaustive search (10 points)
     1. Generate all solutions for the given problem instance.
     2. Find and report the mean and standard deviation of this distribution, as well as the length and trip order for both the longest and the shortest trips.
     3. Collect data for a histogram of this distribution of solutions using at least 100 trip length bins and use some tool, such as Excel®, to ***plot the histogram of the distribution***. (You may actually wait to do this until part “e” of the question.)
     4. How long did the exhaustive search take?
     5. How long would you expect the algorithm to take if the number of cities, *n*, were to increase by one?
     6. What is the time complexity of the exhaustive search algorithm used?
  2. Random search (10 points)
     1. Generate data for a histogram of the costs of 1,000,000 ***randomly*** generated solutions for the given TSP problem.
     2. Find and report the mean, extreme values (the maximum and minimum) and trip orders, and standard deviation of this distribution of solutions.
     3. Organize data for a histogram of this distribution of solutions using at the same 100 bins as in part “a” and use some tool to ***plot the histogram of the distribution***. (You may actually wait to do this until part “e” of the question.)
     4. What is the time complexity of the random search algorithm?
  3. Genetic algorithm (10 points)
     1. Create a genetic algorithm to find good solutions for the problem instance.
     2. Find and report the mean, extreme values (the maximum and minimum) and trip orders, and standard deviation of this distribution of solutions.
     3. Use your genetic algorithm to find and report a histogram for at least 50 solutions for the problem using the same bins as before. (You may actually wait to do this until part “e” of the question.)
     4. What is the time complexity of the genetic algorithm?
  4. Simulated annealing (10 points)
     1. Create a simulated annealing algorithm to find a good solution for the problem instance.
     2. Find and report the mean, extreme values (the maximum and minimum), and standard deviation of this distribution of solutions.
     3. Use your simulated annealing algorithm to find and report a histogram for at least 50 solutions for the problem using the same bins as before. (You may actually wait to do this until part “e” of the question.)
     4. What is the time complexity of the simulated annealing algorithm?
  5. ***Compare*** (10 points)
     1. Scale each of the histograms by dividing each count in each bin by the maximum frequency count for that histogram.
     2. ***On a single chart***, plot all four of the scaled histograms.
     3. What fraction of the distribution of possible solutions is better than your best solution by random searching?
     4. What fraction of the distribution of possible solutions is better than your best solution by using the genetic algorithm?
     5. What fraction of the distribution of possible solutions is better than your best solution by using the simulated annealing algorithm?
     6. What are the relative merits of each of the approaches?

# TSP Data

|  |  |  |
| --- | --- | --- |
| TSP |  |  |
| X | Y | Label |
| 0.725228831 | 0.028301616 | A |
| 0.632613331 | 0.879012261 | B |
| 0.085878084 | 0.352754499 | C |
| 0.880437853 | 0.852414005 | D |
| 0.725231388 | 0.382031121 | E |
| 0.74550796 | 0.32391051 | F |
| 0.166612034 | 0.10383117 | G |
| 0.637364579 | 0.962848092 | H |
| 0.136222778 | 0.908730558 | I |
| 0.078296453 | 0.248218346 | J |
| 0.396853572 | 0.962644387 | K |
| 0.390728772 | 0.703886694 | L |
| 0.276887516 | 0.354859612 | M |
| 0.974681576 | 0.344309411 | N |