HOMEWORK 6

Due: Tuesday, May 15, 2018, 11:55pm

1. **The Shared Birthday Problem.** Write a Monte Carlo simulation to answer the following question: *how many people are required in a group before it is more likely than not that any two of the their birthdays occur in the same week?* Conceptually speaking, your algorithm should model starting with an empty group and introducing new people with random birthdays, one at a time, until a pair of birthdays occurs in the same week.

   Assume that a year has 365 days and that all birthdates are equally likely. Our definition of "same week" for this problem will be if the birthdays are separated by less than 7 days in either direction (e.g., birthdates of 8 and 2 occur in the same week, while birthdates of 9 and 2 do not). If the Nth person added has the birthday that produces the match, then your answer for that single trial is N. In the following example

group $= [124]$,	No match, add new person
group $= [124, 102]$,	$\vdots$
group $= [124, 102, 217]$,	$\vdots$
group $= [124, 102, 217, 26]$,	$\vdots$
group $= [124, 102, 217, 26, 106]$,	Match found!

   The required number of people was 5. Your solution must account for the fact that a birthday near the end of the calendar year and another birthday near the beginning can produce a match if the effective separation is less than 7 days. For example

...	362	363	364	365	1	2	3	4	...

   birthdates of 364 and 2 occur in the same week, while birthdates of 362 and 4 do not. Repeat this procedure for 1×10^4 trials, storing the N-value for each trial in a 1×10^4-element vector, and report the median number of people required using the exact formatting shown below:

   ```
   Median Number of People = 00
   ```

   Where `00` will be replaced with your answer rounded up to the nearest integer. To get a sense for the distribution of N-values, plot a histogram using the built-in function `histogram`. Consider what this median indicates: if you are in a group larger than this, then there is $> 50\%$ chance of two people having a birthday in the same week. Does this number seem reasonable to you? Would you expect this value to increase or decrease if we accounted for the fact that not all birthdates are equally likely?

   Note: You are free to use the built-in `median` function, but do not use the functions `find`, `ismember`, etc., in your solution.

2. **Simulated Annealing.** In lecture, we saw how we can solve a very complicated problem like the traveling salesperson by the controlled injection of random exploration. In this problem, you'll develop your own simulated annealing algorithm and use it to find the shortest, fully-closed path between `N` randomly-positioned points on a 2D grid.

   (a) Start by constructing two, `[1xN]` vectors containing uniformly distributed random values in the range $(0, 1)$ to represent the x and y positions of each node (city) that will eventually be visited. These values won't change during the course of our algorithm; we'll only be updating the order in which we visit them. To construct the initial visitation order, we'll create a single random permutation of the values `[1, 2, ..., N]` using `path = randperm(N)`.

   (b) Next, we'll determine the length of the initial path by writing a function, `getPathDistance`. This function must use *exactly* the header shown below

   ```
   function dist = getPathDistance(x, y, path)
   ```

   where x and y are the `[1xN]` vectors describing the location of each node and `path` is a permutation of the values `[1, 2, ..., N]` describing the order of visitation. The output, `dist`, represents the sum of the Euclidean distances between successive nodes in the path. For example, if `path = [3, 1, 4, 2]`, then `dist` = $\overline{31} + \overline{14} + \overline{42} + \overline{23}$, where each segment in the trip can be computed according to $\overline{ab} = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$

   Note: Since we're assuming the path forms a closed loop, `dist` must also include the separation between the first and last nodes in path, $\overline{23}$ in the above example.

   (c) With the problem setup finished, we can begin the simulated annealing process. Begin by defining an initial temperature $T = 1000$. We'll continue iterating as long as $T > 1$ and at the end of each step in the iteration "cool" the simulation according to $T = T(1 - \lambda)$, where the decay rate, $\lambda = 0.005$. At every step in the simulation, carry out the following operations:

      i. Swap exactly 2 randomly-chosen nodes in `path` to form `pathNew`
      ii. Call your function from Part 2b to calculate `distNew`, the length of the new path.
      iii. Calculate the difference in path length, ΔL `= distNew - dist`.
      iv. If the new path is shorter than the old path, ΔL `< 0`, or $p >$ `rand`, then update the `path` vector and `dist` value to reflect the new route.

   where, in the final step, p is our probability of exploration metric, $p = \exp(-c\Delta L/T)$, with $c = 1000$ and `rand` is a uniformly distributed random value in the range $(0, 1)$. As discussed in lecture, the above algorithm gives us *two* potential routes by which we can replace the current path with the new path. The first (ΔL `< 0`) takes effect only if the new distance is found to be shorter, and the second ($p >$ `rand`) gives us a certain probability of exploring solution areas that might otherwise seem detrimental to the overall solution, with the latter more likely to occur during the initial portion of the search when the temperature, T, is still high.

   (d) Using a value of `N = 10`, generate the following two plots to showcase your method: (1) the distribution of nodes (markers) and the final, completely-closed path taken between them (lines) and (2) a plot of the distance corresponding to the current path at the end of each iteration as a function of iteration number.

   Note: You can plot both endpoints *and* lines with a line option like `plot(..., ..., 'o-')` (`help plot` for more options). We're primarily interested in the perimeter of the final path shown in your first figure, not the handedness (clockwise vs counterclockwise) or start/stop points, so don't worry about distinguishing these features.

(e) Does your solution always find the shortest path (since we don't always know what the best path is, you'll have to use your best judgement here or try to find a counterexample that clearly shows the shortest path isn't found)? As you increase the value of `N`, what happens to the solution produced by your method? Can you find a better combination of λ and c that improves the behavior when `N > 10`?

Note: Although you're encouraged to modify your code to generate results for your report and discussion, please submit the version outlined above, where `N = 10`, $\lambda = 0.005$, $c = 1000$, and initial temperature $T = 1000$.

Using the naming convention presented in the syllabus, submit **two** separate files to the CCLE course website: (1) a .pdf of your written report and (2) a .zip file containing all of the MATLAB files written for the assignment. Remember to use good coding practices by keeping your code organized, choosing suitable variable names, and commenting where applicable. Any of your MATLAB .m files should contain a few comment lines at the top to provide the name of the script, a brief description of the function of the script, and your name and UID.