CS 430 Introduction to Algorithms	**Fall Semester, 2018**
Project - version 1.01	
Assigned: Nov. 6	*Due: Nov. 27*

The goal of the project is to implement heuristics/algorithms for the following problem, whose solution is useful in broadcasting protocols for ad-hoc wireless networks:

Broadcast Step Problem

Input: A set of points C, called centers, all inside the disk centered at the origin (the point with both coordinates 0) of radius 1, and a set P of points, all outside the disk centered at the origin of radius 1.
Output: A subset of centers $Q \subseteq C$, such that for every point in P there is a center in Q at distance at most 1. (we say that the center "covers" the point.)
Measure: Minimize the size of Q.

Note that, for the problem to have a solution, we assume that for every point in P there is a center in C at distance at most 1.

The Broadcast Step problem was studied by the instructor, and
`http://www.cs.iit.edu/~calinesc/forwarding.pdf` describes a polynomial-time algorithm guaranteed to return a solution at most three times the optimum solution. This is a complicated algorithm and simpler heuristics might do better in practice.

If one searches the papers citing the paper above, one would find approximation algorithms which guarantee a solution closer to the optimum, as well as dynamic-programming-based exact algorithms. These are also fairly complicated, so we'll settle for two approximation algorithms that may be useful as they are simpler and likely faster.

The first heuristics you are asked to implement is the following local-optimization procedure. Start with an arbitrary feasible solution. Try all combinations of two centers from the current feasible solution, and another center. If the removal of the two centers followed by the addition of the other center results in another feasible solution, then proceed and change the current feasible solution. Repeat trying all combinations, until no combination leads to another feasible solution. Procedures similar to this are used by the meta-heuristic method Simulated Annealing, as well as by one of the approximation algorithms mentioned before.

The second heuristic is the following adaptation of the Greedy algorithm described in Section 35.3 of the textbook (with a variant discussed in class). Start with S the empty set. As long as there are points that are not covered by centers in S, find a center which covers the largest number of points which were not previously covered. Add this center to S and repeat.

The Project

Teams of two students will work on each project submission. If you cannot find a team-mate, select and implement one of the algorithms mentioned above.

The programs must be in C/C++, Java or Python, or you must pe prepared to do a demo in Stuart Building (for example, by bringing your laptop). The input and output format is given below and is very strict - we plan to do automatic grading.

In addition to submitting the complete source code on Blackboard, each team must turn in written (hard-copy) document with:

1. Pseudocode for the algorithm(s)
2. Analysis of the running time. Do not exceed $O(n^5)$ for the local optimization procedure and $O(n^3)$ for the Greedy algorithm, or your implementation will be too slow. Each of these running times can be improved by a factor of n.
3. One instance for each algorithm on which the algorithm fails to return the optimum solution. Show the run of the algorithm on the instance, and show a better solution.
4. A project report, on which you compare the quality (that is, size) and running time (that is, milliseconds) of the implemented algorithms for all the instances posted on the web.

Outside Sources

Any human help (including collaboration in between teams) is strictly prohibited.

You can use any other source for both ideas and code, provided you clearly mark when the idea or code is borrowed, and give full acknowledgement to the source. But make sure you are not breaking any laws.

Input and Output Formats

The programs should read the input from a sequence of files called "instance01" to at most "instance99", and output solutions in the files "solution01" to "solution99". Instances will have at most 2000 centers and at most 6000 points.

Each input file starts with an integer, the number of centers, and then a newline character. Next, on every line, there are the two coordinates of every center, given as floating-point numbers. The next line has another integer, the number of points followed by a newline character. Next, on every line, there are the two coordinates of every point, given as floating-point numbers. For example:

99
-0.12 0.55
0.01 0.98
.....
123
1.12 -0.5
.....

The output format is the following: start with the name of the algorithm, followed after a blank by the size of the solution and a newline character. Then, on every line, one integer in between 1 and the number of centers, indicating the index of every center selected by the

algorithm. The centers are numbered starting from 1, and their coordinates are given in order in the input file.

For the Greedy algorithm, output the centers in the order they are selected by the algorithm.

Note: The outputs are not unique, as there is freedom in both algorithms with respect to breaking ties, or which combination of three lines to pick. The grading program can still check if the output is the result of the algorithm.

Misceleneus

I will test your code on additional instances, not posted on the web. More examples of input and output files are posted on the web. Please follow carefully the format since we are going to automatically check the output of your program for correctness.

If you would rather try an idea of yours, write down pseudocode and discuss in person your idea with the instructor whose approval will be required before your algorithm is implemented.