CS 430 Introduction to Algorithms	**Spring Semester, 2021**
Project - version 1.1	
Assigned: April 9	*Due: May 7*

The goal of the project is to implement several algorithms for the following problem: SEPARATING POINTS BY AXIS-PARALLEL LINES.

Input: Set of n points in the two-dimensional plane, point i given by coordinates x_i and y_i. No two points have the same x- or y-coordinate.

Output: Set S of vertical or horizontal lines, each given by the direction and one coordinate (where it intersects the axis), such that any two points of the input are separated by a line of S.

Measure: Minimize $|S|$.

SEPARATING POINTS BY AXIS-PARALLEL LINES has applications to fault-tolerant multimodal sensor fusion in the context of embedded sensor networks. In the following, by *line* we mean axis-parallel line.

This problem is so-called NP-hard, and therefore polynomial-time algorithms are very unlikely to exist. We have to settle for approximate solutions - that is our algorithms are not guaranteed to return the optimum solution. Such algorithms are called heuristics, and the project asks you to implement two such algorithms.

SEPARATING POINTS BY AXIS-PARALLEL LINES was studied by the instructor, and `http://www.cs.iit.edu/~calinesc/separating.pdf` describes a polynomial-time algorithms guaranteed to return a solution at most twice the optimum solution. This is a complicated algorithm and simpler heuristics might do better in practice.

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 lines from the current feasible solution, and another line. If the removal of the two lines followed by the addition of the other line 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. Such procedure is used by the meta-heuristic method Simulated Annealing.

The second heuristic is the following adaptation of the Greedy algorithm described in Section 35.3 of the textbook. Start with S the empty set. As long as there are pairs of points non-separated, find a line which separates the largest number of pairs of points which were not previously separated. Add this line 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. 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 $O(n^6)$ for the local optimization procedure and $O(n^4)$ for the Greedy algorithm, our your implementation will be too slow. As a matter of fact, 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.

Outside Sources

Any source from books or the open Internet is allowed, but you must clearly mark the ideas and/or code that is borrowed, and give full acknowledgement to the source.

No other collaboration is allowed, and specifically collaboration in between teams is strictly prohibited. Sharing a solution is also considered dishonesty.

Late penalty policy

The penalty is 1% per day being late, up to a maximum of five days. No further extensions will be given so that we have time to score the submissions before the grades are due.

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 "local_solution01" to "local_solution99", and "greedy_solution01" to "greedy_solution99". Instances will have at most 100 points.

Each input file starts with n, the number of points, followed by n lines, each containing two integers: the x and y coordinates of the point. The points are sorted by the x coordinates. For example:

`intance01`:

5
1 1
2 5
3 3
4 2
5 4

The output file should contain in the first line m, the number of axis-parallel lines used by the solution, follow by another m lines, each describing one axis-parallel line as follows: the characters 'h' or 'v' are used to describe if the axis-parallel line is horizontal or vertical, and a floating point gives the coordinate where this line crosses the axis. For example:

`local_solution01`:

3
v 2.5
v 3.5
h 2.5

For the Greedy algorithm, output the lines in the order they are found by the algorithm.

For the Local algorithm, output first the horizontal and then the vertical lines, and output them in increasing order of coordinates.

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.

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.

The submission process

Submit on Blackboard one single zip file with the full source code and the document with pseudocode and analysis. For teams of two, the document should include both names, and then one student can just submit the name of his or her team-mate.

Teams of three have to "invent" a third algorithm and submit the idea to the instructor one week before the project is due. This third algorithm should terminate in one minute for an instance with 30 points, and be "essentially different" from the existing two algorithms.

If you submit two days earlier we may have time to compile and run your code and let you know what mistakes we found and give you a chance to make corrections.