Data Structures and Algorithms

COMP 251, Winter 2013

Assignment 3

Due date: Monday, March 11, 2013
6pm

All coding for the assignment must be in Java. All code should be well-commented, in a professional style, with appropriate variables names, indenting, etc. Your code must be clear and readable. **Marks will be very generously deducted for bad style or lack of clarity.**

1. A *dependency graph* is a DAG that describes necessary precedence between individual tasks: each task is represented by a node, and if t_1 must be completed before t_2 can be started, then there is a directed edge from the node representing t_1 to the node representing t_2.

   Assume you have n tasks to perform, arranged as such as a dependency DAG $G = (V, E)$. For simplicity assume each task takes the same amount of time to perform. You have p people available to perform tasks, each of whom can do at most one task at a time. The goal is to give your workers tasks so as to minimize the total time taken to complete all tasks, while respecting the dependencies given by the dependency DAG.

   (a) Is there any lower bound constraint(s) on the solution (minimum time to do all tasks) for an arbitrary n, p, and dependence DAG? Explain. **4**

   (Note that this question is asking about the solution itself, not the running time of the algorithm generating the solution!)

   (b) Describe a greedy solution to perform all tasks in as little total time as possible. Your solution does not need to be optimal, but it should be as good as you can make it. **6**

   Note that the running time of your algorithm is not a great concern, but it should be efficient enough in practice to scale up to at least thousands of nodes in the dependency DAG and or workers.

   Give a clear pseudo-code description of your proposed algorithm. Is your solution optimal? Justify your answer.

   (c) You do not need to actually implement the above algorithm (although it should be clearly implementable). However, testing such an algorithm would require some input dependency DAGs. Design and implement an algorithm that accepts 1 or 2 command-line parameters: n, and an optional seed to a random number generator. A template is provided in *MyCourses*. **10**

   Your program should emit the graph to standard output in an adjacency list format. For each vertex emit a single line starting with the vertex number followed by the list of vertices to which there are outgoing edges, all separated by spaces. Do not include anything else in the output. For example ($n = 3$):

   ```
   1 3 2
   2 3
   ```

   Your function must be capable of generating any possible DAG on n nodes.

2. Consider a rectangle in the positive quadrant of $\mathcal{R}^2$, with bottom-left corner at the origin. Inside are $n-4$ points, randomly located (ie random coordinates, but strictly within the rectangle). The goal is to add as many edges as possible between points, using straight lines only, and without allowing any edges to intersect.

   (a) Implement this simulation. You may use any approach to placing edges you want, as long as its **10** complexity is in $O(n^k)$ for some k, and it achieves the goal of ensuring no further edge can be added without introducing an intersection.

Your program should take three or four command-line parameters: $n-4$, the width and height of the rectange, and an optional random seed value (for reproducibility). You may assume ($n \geq 4$). Using a random number generator (initialized to the seed value if one is provided), choose random coordinates for the $n-4$ points such that they lie (strictly) within your rectangle.

Connect as many nodes as possible without crossing edges. Emit the total number of edges in the graph.

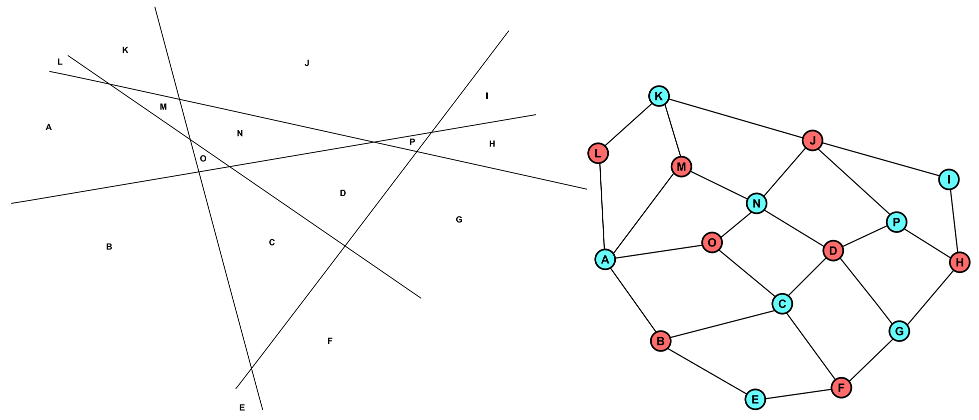
Note: rendering the results visually is not required. However, for debugging it is helpful, so a library to help you draw graphs is provided in *MyCourses*. nb: You do NOT need to include this library in your submission.

(b) What is the worst-case asymptotic complexity of your edge-addition process? Give a brief high-level description of your approach, a suitable (tight) complexity class, and an argument explaining how/why you derived that complexity. **5**

(c) Plot the number of edges added as a function of n. Comment on the results—what do you observe? **5**

3. Drawing a line on the plane bisects the plane into two pieces. Suppose you then draw another line, not parallel to the first. It will further divide the two halves of the plane. You can keep on doing this, each time choosing a new line of arbitrary slope, but not parallel to any other line (also assume no more than 2 lines meet at any point). **10**

   An adjacency graph can be constructed to represent this. Each node represents an undivided portion of the plane, and edges exist between portions that share a boundary edge (as created by your lines). An example is shown below.



   Use induction to prove that the adjacency graph is always a bipartite graph for any such arrangement of lines.

What to hand in

Submit your assignment to *MyCourses*. Note that clock accuracy varies, and late assignments will not be accepted without a medical note: **do not wait until the last minute**. Assignments must be submitted on the due date **before 6pm**.

Where possible hand in only **source code** files containing code you write. Do not submit compiled binaries or .class files. For the written answer questions submit either an ASCII text document or a .pdf file *with all fonts embedded*. Do not submit .doc or .docx files. Images (plots or scans) are acceptable in all common graphic file formats.

Note that for written answers you must show all intermediate work to receive full marks.
This assignment is worth 6% of your final grade. **$\overline{50}$**