Midterm Examination

CS 5114 (Fall 2018)

Assigned: Monday, October 8, 2018.
PDF solutions due on Canvas before the start of class on Monday, October 15, 2018.

Instructions

1. The Graduate Honor Code applies this to homework. In particular, you are not allowed to consult any sources other than your textbook, the slides on the course web page, your own class notes, and the instructor. Do not use a search engine.
2. You may consult the textbook, your notes, the course web site, or solutions to earlier homework assignments to solve the problems in the examination. Of course, I am available to answer your questions.
3. Describe your algorithms as clearly as possible. The style used in the book is fine, as long as your description is not ambiguous. Explain your algorithm in words. A step-wise description is fine. *However, if you submit detailed code or pseudo-code without an explanation, we will not grade your solutions.*
4. For every algorithm you describe, prove its correctness, and state and prove the running time of the algorithm. I am looking for clear descriptions of algorithms and for the most efficient algorithms and analysis that you can come up with. Except for one problem, I am not specifying the desired running time for each algorithm. I will give partial credit to non-optimal algorithms, as long as they are correct.
5. Do not make any assumptions not stated in the problem. If you do make any assumptions, state them clearly, and explain why the assumption does not decrease the generality of your solution.
6. You must also provide a clear proof that your solution is correct (or a counter-example, where applicable). Type out all the statements you need to complete your proof. *You must convince us that you can write out the complete proof. You will lose points if you work out some details of the proof in your head but do not type them out in your solution.*
7. You must prepare your solutions digitally, i.e., do not hand-write your solutions. I prefer that you use LaTeX to prepare your solutions. However, I will not penalise you if you use a different system. To use LaTeX, you may find it convenient to download the LaTeX source file for this document from the link on the course web site. At the end of each problem are three commented lines that look like this:

   ```
   % \solution{
   %
   % }
   ```

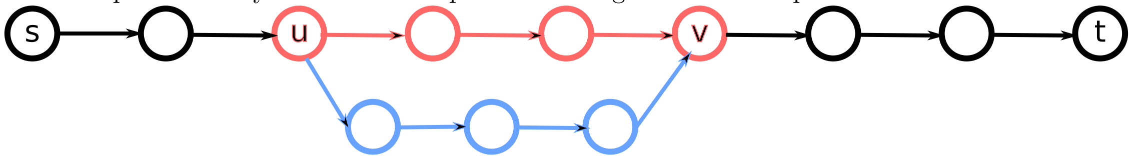
   You can uncomment these lines and type in your solution within the curly braces.

Good luck!

Problem 1 (15 points) Let us start with some quickies. For each statement below, say whether it is true or false. *You do not have to provide a proof or counter-example for your answer.*

1. Every bipartite graph is a tree.
2. $\sum_{i=1}^{n} i \log i = \Theta(n^2 \log n)$.
3. Dijkstra's algorithm may not terminate if the graph contains negative-weight edges.
4. In a directed graph, $G = (V, E)$, each edge has a weight between 0 and 1. To compute the length of the *longest* path that starts at u and ends at v, we can change the weight of each edge to be the reciprocal of its weight and then apply Dijkstra's algorithm. Here, the *length* of a path is the sum of the weights of the edges in the path.
5. Suppose π is the shortest s-t path in a directed graph. Then, there can be two nodes u and v in π such that length of the portion of π connecting u to v is larger than the length of the shortest u-v path in the graph. Here π is the name I am giving to the shortest path between s and t; it is not the mathematical constant.

   In the figure below, π is the path composed of black and red edges. As an example, I have marked two nodes u and v in π and denoted the portion of π connecting u to v using red edges. The blue edges denote another path from u to v in the graph. The figure does not indicate edge costs. If you answer "yes" to this question, you are saying that there are graphs where the length of the red path can be larger than the length of the blue path (knowing that π is the shortest s-t path). If you say "no", then you are saying that for every pair of nodes u and v in π, the length of the red path is always less than or equal to the length of the blue path.



Problem 2 (15 points) A connected, undirected graph G contain k cycles. Each edge in G has a distinct weight. Develop an algorithm to compute the minimum spanning tree of G in $O(mk)$ time. You may assume that $k < \log n$, so that the running time that I am seeking is faster than what you would obtain using Prim's or Kruskal's algorithm.

Problem 3 (30 points) Inspired by Virginia Tech's Flint Water Study, you start thinking about using multiple sensors embedded in different parts of a city to automatically test water quality. Sensors can communicate with each other. However, each sensor has limited resources (e.g., it can contact only "nearby" sensors) and these communications can be unreliable and asymmetric. The goal of each sensor is to relay the information it has collected back to a "master" sensor. After deploying the sensors, you can measure which pairs can communicate and how reliable each link is. Your goal is to determine the maximum number of sensors that can communicate with the master through a reliable series of links.

Fortunately, you have taken CS 5114! You model the problem as follows. You have a directed graph $G = (V, E)$. Each node in V corresponds to a sensor. The master sensor is a specific node s in V. Every edge $e = (u, v)$ directed from node u to node v has a reliability $r(e)$ that is positive and at most 1, i.e., $0 \leq r(e) \leq 1$. Since communications are asymmetric, it is not necessary that $r(u, v) = r(v, u)$. The reliability of a path is the product of the reliabilities of the edges in that path. Given such a graph G and a parameter $q > 0$, the `Sensor Network` problem is to compute the largest set S of nodes in V such that for every node u in S there is a directed path from u to s with reliability at least q. Describe an algorithm for the `Sensor Network` problem.

Note: I have used the problem of placing sensors to inspire this question. I suggest you develop your solution using the language of graphs. In other words, you should be able to present your solution even if you read only the second paragraph of this question.

Hint: You may develop your own algorithm or modify an algorithm we have designed in class. In either case, you must not forget to prove the correctness of your algorithm, as we always require in this class. An alternative you can use is to transform the *input* to the `Sensor Network` problem so that it becomes a suitable input for a different problem X we have studied in class. In this case, you can use an algorithm A for problem X on this transformed input, take the output of algorithm A, and then convert it to a solution to the `Sensor Network` problem. If you use this option, you do not have to prove the correctness of algorithm A (since we have done so in class) but you must show that your transformations yield the correct answer to the `Sensor Network` problem.

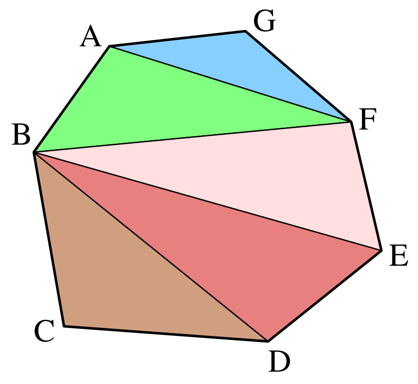
Problem 4 (40 points) A convex polygon is a polygon where very interior angle is less than 180 degrees. A museum is in the shape of a convex polygon with n vertices. The museum is patrolled by guards. The Directory of Security at the museum has the following rules to ensure the most safety in as time-economical a way as possible:

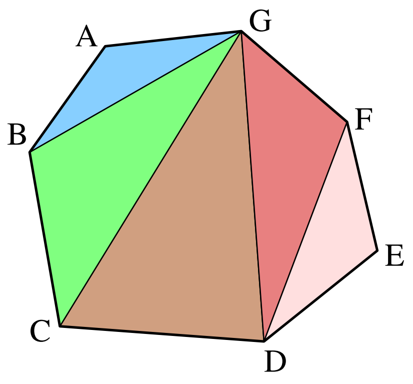
(a) Each guard traverses a path in the shape of a triangle; each vertex of such a triangle must a vertex of the polygon.

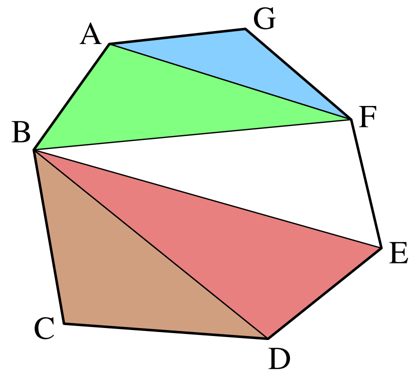
(b) A guard can survey all the points inside his or her triangle and *only* these points; we say that these points are *covered* by the guard.

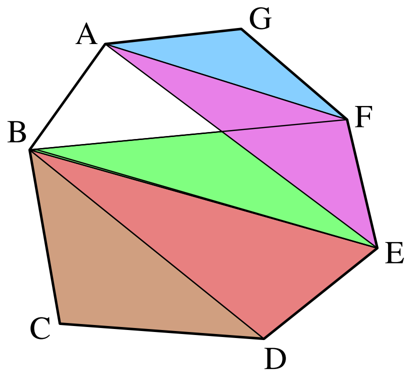
(c) Every point inside the museum must be covered by some guard.

(d) The triangles traversed by any pair of guards do not overlap in their interiors, although they may share a common edge.









Each guard can be¡ specified by the triangle he/she patrols. We call a set of triangles that satisfy these rules *legal*. Above are four figures that illustrate the problem. The museum is the polygon ABCDEFG. Each coloured (shaded) triangle corresponds to a guard and the guard traverses the perimeter of his or her triangle.

- The top two figures show a set of triangles that are legal, since they satisfy the constraints laid down by the Directory of Security. In the top left figure, the guards traverse the boundaries of triangles AFG (blue), ABF (green), BEF (pale red), BDE (light red), and BCD (brown). In the top right figure, the guards traverse ABG (blue), BCG (green), CDG (brown), DFG (light red), and DEF (pale red).

- The bottom two figures show a set of triangles that *do not* satisfy these constraints: in the figure on the bottom left, part of the museum is not covered by any guard (the unshaded triangle BEF) while in the figure on the bottom right, the pink triangle (AEF) and the green triangle (BEF) intersect (note that the part of the green triangle in the image is covered by the pink triangle).

Given these constraints, the *cost* incurred by a guard is the length of the perimeter of the triangle the guard traverses. The *total* cost of a set of legal triangles is the sum of the length of their perimeters. Our goal is to find a legal set of triangles such that the total cost of the triangles is as small as possible. Given the x- and y-coordinates of the vertices of the art gallery and the ordering of these vertices along the boundary of the art gallery, devise an algorithm whose running time is polynomial in n to solve this problem. Note that we are not trying to minimise the number of guards; we want to minimise the total lengths of the routes patrolled by the guards. You may assume that the length of any line segment is the Euclidean distance between the end-points of the line segment and that this length can be computed in constant time.

Since this problem may be quite difficult, let us split up into more digestible pieces. Let the museum be a convex polygon P with n vertices $p_0, p_1, \dots p_{n-1}$ ordered consecutively in counter-clockwise order around P. Except for $p_{n-1}p_0$, which is an edge of the polygon, a line segment p_ip_j is a *diagonal* if p_i and p_j are not adjacent vertices, i.e., $|i - j| \neq 1$. The order of the endpoints does not matter, i.e., the diagonals p_ip_j and p_jp_i are identical. Since P is convex, every point of a diagonal (other than its end-points) lies in the interior of P. Clearly, in any legal set of triangles, every triangle edge is either an edge of P or a diagonal of P. For each of the questions below, you must provide a proof for your answer. Some of the proofs may be short, especially for parts (i) and (iii).

(i) (5 points) How many diagonals does P contain in total?

(ii) (10 points) How many diagonals does any legal set of triangles contain? If you use a proof by induction, please be sure to state the base case, inductive hypothesis, and inductive step clearly.

(iii) (7 points) Given a legal set of triangles, express the total cost of these triangles in terms of the perimeter of P and the lengths of those edges of the triangles that are diagonals of P.

(iv) (18 points) To start formulating the dynamic programming recursion, consider the edge p_0p_{n-1}. This edge must be part of some triangle T in the optimal solution. Think about where the third vertex in T can be. Since the optimal solution contains a legal set of triangles, the edges of the other triangles in the optimal solution cannot intersect the edges of T. Use this fact to derive a recursion to compute the optimal solution. State and prove the running time of the resulting algorithm.