Homework Assignment 3

Give written (and legible) answers to the following questions. Whenever you are asked to give an algorithm for a problem, you should present the following: the algorithm, a justification of its correctness, and a derivation of its running time. Write clear and convincing pseudo-code for your algorithms. All work must be done independently.

**Question 1. (10 points)** Give pseudo-code for an algorithm to compute a 3-coloring of a triangulated simple polygon. The algorithm should run in linear time.

**Question 2. (25 points)** In computer graphics, triangulations are used to represent surfaces. Graphics libraries like OpenGL support a special type of triangulation called a strip, which is defined as follows. For \( n \geq 3 \), given a sequence of vertices \( V = \langle v_1, v_2, \ldots, v_n \rangle \), the strip of \( V \) consists of the \( n - 2 \) triangles \( \langle v_i, v_{i+1}, v_{i+2} \rangle \), for \( 1 \leq i \leq n - 2 \). See Figure 1.

![Figure 1: An x-monotone strip polygon.](image)

Assume that you are given an \( n \)-sided, \( x \)-monotone simple polygon \( P \) as a sequence of vertices in counterclockwise order.

1. Give an algorithm to decide if \( P \) can be triangulated as a strip of the form given above.
2. Give a rigorous proof of correctness and derive the running time of your algorithm.

*Note:* The \( x \)-monotonicity will play a significant role in your algorithm and its proof of correctness. There is a simple linear time algorithm for this problem, but the proof of correctness is not completely straightforward. Your algorithm need not run in linear time. However, a more efficient algorithm will receive more credit.

**Question 3. (20 points)** You are given a pair of vertical lines \( x = x_0 \) and \( x = x_1 \), which define a vertical strip in the plane. You are also given a set \( L \) of \( n \) nonvertical lines, \( l_i : y = a_i x + b_i \). Present an \( O(n \log n) \) time algorithm that counts the number of intersections of the lines of \( L \) within this strip. (Thus, the output consists of a single number.) You may make whatever
general position assumptions are convenient. \textit{Hint:} Keep in mind that these are infinite lines, not line segments, hence each line will intersect both vertical lines that bound the strip. The order of these intersections on each vertical line provides enough information to count the total number of line intersections within the strip.

\textbf{Question 4. (20 points)} This question is about partitioning an orthogonal pyramid into convex quadrilaterals. An \textit{orthogonal polygon} is a polygon in which each pair of adjacent edges meets orthogonally. Without loss of generality, one may assume that the edges alternate between horizontal and vertical.

\begin{center}
\begin{tikzpicture}
\draw (0,0) -- (0,4);
\draw (1,0) -- (1,4);
\draw (2,0) -- (2,4);
\draw (3,0) -- (3,4);
\draw (4,0) -- (4,4);
\draw (5,0) -- (5,4);
\draw (0,1) -- (5,1);
\draw (0,2) -- (5,2);
\draw (0,3) -- (5,3);
\end{tikzpicture}
\end{center}

\textbf{Figure 2: An orthogonal pyramid}

An \textit{orthogonal pyramid} \(P\) is an orthogonal polygon monotone with respect to the vertical that contains one horizontal edge \(h\) whose length is the sum of the lengths of all the other horizontal edges. \(P\) consists of two “staircases” connected to \(h\), as shown in Figure 2.

(a) Prove that an orthogonal pyramid may be partitioned by diagonals into convex quadrilaterals.

(b) Design a linear-time algorithm for finding such a partition.

\textit{Hint:} Use a sweep-line method (sweep a horizontal line from top to bottom of the orthogonal pyramid). You may combine parts (a) and (b) of the question to give a constructive proof. In other words, the proof that we can always decompose an orthogonal pyramid into convex quadrilaterals will also give an algorithm to do the decomposition. The linear time algorithm discussed in class to triangulate an \(x\)-monotone polygon is an example of a constructive proof.

\textbf{Question 5. (25 points)} The objective of this problem is to investigate one approach for an \(O(m + n)\) time algorithm to determine whether one convex polygon \(P\) lies inside of another convex polygon \(Q\). Here \(m\) and \(n\) denote the number of vertices of \(P\) and \(Q\), respectively. There are other algorithms for doing this (for example, the sweepline approach discussed in class to find the intersection of two convex polygons), but here I would like you to consider this particular approach.

(a) Given the leftmost vertex \(v_1\) of \(Q\), show how to use orientation tests to compute the counterclockwise point of tangency of \(P\) relative to \(v_1\), denoted \(t_1\). (More formally, \(P\) lies on and to the left of the directed line \(v_1t_1\). Such a line is called a \textit{supporting line} for \(P\). See Figure 3.) Show that your method runs in \(O(m)\) time.
Figure 3: Tangent points of $P$ relative to $Q$.

(b) Assuming that the CCW tangent point $t_{i-1}$ has already been computed for vertex $v_{i-1}$ of $Q$, show how to compute the tangent point $t_i$ for the next vertex $v_i$ in CCW order about $Q$.

(c) Prove that as the tangent points are being computed, it is possible to determine whether $P$ is not contained in $Q$.

(d) Assuming that $P$ is contained within $Q$, show that all of these tangent points can be computed in $O(m + n)$ time.

You may design your algorithms in (a) and (b) so that they terminate if it is discovered that $P$ is not contained in $Q$.

Please upload your homework solutions on Sakai by 11:55pm on March 24, 2019.