Algorithms and Combinatorics of Geometric Graphs (`Geomgraphs`) 2025-2026

TD2 (exercises)

Luca Castelli Aleardi

october 2nd 2025

Exercise 1 – Schnyder woods and Menger theorem

Schnyder woods lead to a very simple proof of Menger theorem in the planar triangulated case.

Question 1.1 (Menger theorem for triangulations)**.** *Let us consider a plane triangulation $\mathcal{T}$ with $n > 3$ vertices. Show that, given two distinct vertices u and v, there are three disjoint paths from u to v (they only cross at u and v). Devise a linear-time algorithm for computing such paths.*

Question 1.2. *Using previous question, show that a planar triangulation (with $n \geq 4$) is 3-connected* [1].

Exercise 2 – A simple algorithm for embedding maximal planar graphs

A *maximal planar graph* of size n is a planar graph with n vertices and $3n - 6$ edges (it is assumed to be simple: no loops, no multiple edges).

Question 2.1. *Let us consider a maximal planar graph $\mathcal{G}$ with n vertices. Show that it is possible to compute in linear time a non separating cycle C of G.*

Question 2.2. *Let us consider a maximal planar graph $\mathcal{G}$ with n vertices and a cycle $C = \{a, b, c\}$ which is not separating. Show that, that one can compute in linear-time a canonical ordering $\pi = \{v_0 = a, v_1 = b, ..., v_{n-1} = c\}$ of G.*

Question 2.3. *Using previous questions show that there exists a linear-time algorithm for embedding maximal planar graphs. More precisely, if the algorithm receives as input a maximal planar graph $\mathcal{G}$ then it terminates and outputs a planar embedding of $\mathcal{G}$ (a plane triangulation).*

Exercise 3 – Triangulating planar graphs

In this exercise, we will design and analyze an algorithm which, given a planar graph G with n vertices, incrementally triangulates the faces of G: the result is a planar triangulation $\mathcal{T}(G)$ that has the same vertices as G and is *simple* (no loops or multiple edges).

The idea of the algorithm is as follows: we will triangulate the faces of the graph G one after another, in such a way that we never create multiple edges. To do this, we triangulate a face $f = v_1, v_2, \ldots, v_k$ by adding edges, as illustrated in Figure 2. More precisely, let $v_1 \in f$ be a vertex of minimal degree in a face f of degree k: denote by $(v_1, v_2, \ldots, v_{k-1}, v_k)$ the vertices of f listed in clockwise order, and distinguish two cases:

Case 1: if none of the vertices $v_3, v_4, \ldots, v_{k-1}$ is adjacent to v_1, then we add to G the edges

$$(v_1, v_3), (v_1, v_4) \ldots (v_1, v_{k-1})$$

.

Case 2: otherwise, denoting by v_j a neighbor of v_1, we add the zigzag of edges

$$(v_2, v_k), (v_2, v_{k-1}), \ldots, (v_2, v_{j+1}), (v_{j+1}, v_3), \ldots (v_{j+1}, v_{j-1})$$

[1] A graph is 3-connected if one has to remove at least 3 vertices to disconnect the graph (the removal of two arbitrary vertices leave the graph connected).

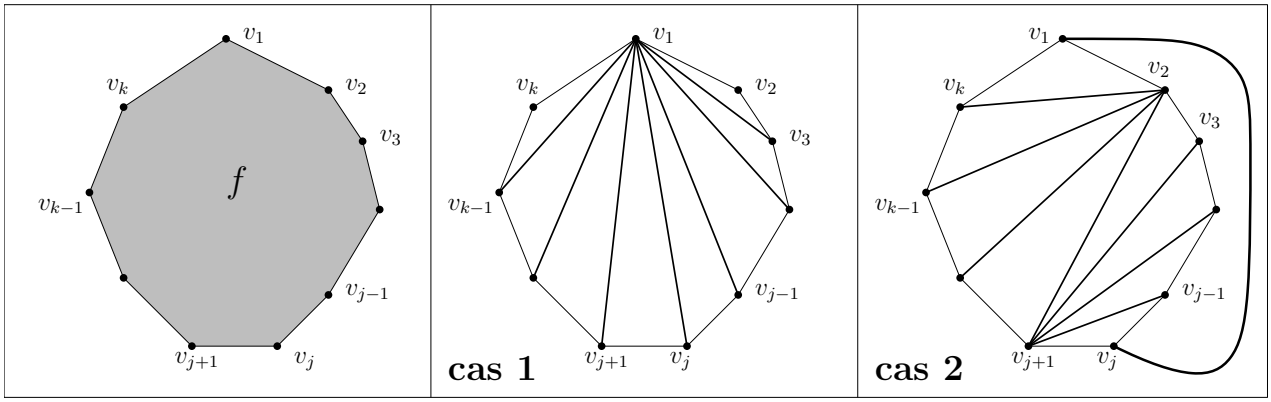

Figure 1: Triangulating a face.

Question 3.1. *Let G be a 3-connected planar graph. Show that the graph G' obtained from G by triangulating a face using the above procedure is simple: it has no multiple edges.*

Question 3.2. *Show that if G is a simple planar graph with n vertices, then:*

$$\sum_{f \in G} \min_{v \in f} \{deg(v)\} = O(n)$$

To triangulate the graph G, it is sufficient to repeat the procedure described above by triangulating each face of degree greater than 3: care must be taken to always choose a vertex v_1 of minimal degree (which is crucial for the algorithm to be efficient).

Question 3.3. *Analyze the time complexity of the algorithm described above (the answer should depend on n, the size of G).*

Hint: You are advised to write the pseudocode of the algorithm and analyze all of its steps. Show that the above algorithm has a complexity $\sum_{f \in G}(deg(f) + \min_{v \in f}\{deg(v)\})$ and deduce from this an upper bound that depends on n.

Question 3.4. *What happens if the initial graph G is 2-connected, but not 3-connected?*

The algorithm analyzed above is simple to implement but has the following drawback: the degree of some vertices in $\mathcal{T}(G)$ could be $O(n)$, even if in the initial graph the vertices have bounded degree.

Question 3.5. *Show that there exist graphs G of size n whose vertices have degree bounded by a constant, but for which the algorithm above produces a triangulation $\mathcal{T}(G)$ with vertices whose degree is linear in n.*