MPRI - GeomGraphs

Exercise sheet 1 (due on november 6th, before 9 am)

Luca Castelli Aleardi

october 16th 2025

The two exercises below can be solved independently and in any order. All arguments should be expressed in a rigorous and clear manner.

Exercise 1 – Efficient algorithms for planar graphs

In this exercice we consider simple planar graphs (no loops, no multiple edges) and we address the problem of efficiently and listing the 4-cliques (complete sub-graphs of size 4). A *triangle* is a cycle consisting of 3 distinct vertices (equivalently, a triangle is a 3-clique, a complete graph on 3 vertices): observe that a triangle does not necessarily define a face in the planar embedding of a graph (refer to Fig. 1). We assume that the input graph has n vertices and is provided with a planar embedding [1].

Listing triangles in (planar) graphs. The goal of this section is to devise a linear-time algorithm that enumerates (or count) all triangles in a planar graph.

Question 1.1 *Show that the algorithm* `CountTriangles` *illustrated in Fig. 1 counts all triangles of an arbitrary graph G (not necessarily planar) and can be implemented in $O(m \cdot n)$ time, where n and m are the number of vertices and edges of G respectively.*

Question 1.2 *Let G a simple planar graph with n vertices. Show that previous algorithm can be used to count (or enumerate) all triangles of G in linear time.*

Hint: *it could be useful to first provide an upper bound on the following sum on the edges of G:*

$$\sum_{(u,v)\in E} \min\{deg(u), deg(v)\}$$

Listing all 4-cliques in linear time. Let un consider a partition of the vertices of G into $k+1$ sets $V_0, V_1, V_2, \ldots, V_k$ obtained by computing a BFS tree (according to a *breadth-first search*) whose root is an arbitrary vertex r. By definition V_j is the set of vertices at distance j from the root r (so $V_0 = \{r\}$). Let us denote by E_j the set of edges $e = (u, v)$ such that $u \in V_{j-1}$ and $v \in V_j$ (an edge belongs to E_j if it is connecting two vertices on levels V_j and V_{j-1}).

[1] In this exercice you may assume that you are provided with a representation of the embedding of the input graph G (e.g. the half-edge representation) and with a data structure for effiently testing in $O(1)$ time whether an edge (u, v) belongs to G (using for example an adjacency matrix or a hash table storing the pairs $\{u, v\}$).

```
procedure CountTriangles(G = (V, E))
  sort the vertices in V according to their degrees (non-increasing order)
  Count := 0;
  for each vertex u ∈ V
     do { mark all vertices which are neighbors of u in G;
          for each marked vertex v ∈ V
             do { for each vertex w which is a neighbor of v in G
                     do if w is marked then Count := Count + 1;
                  unmark vertex v;
          G := G \ {u};
  return Count;
```

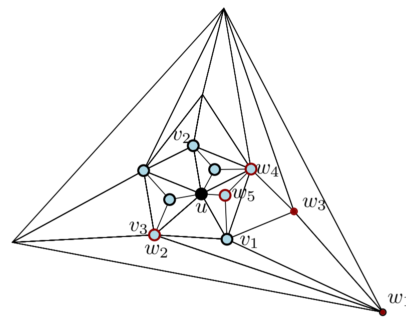

Figure 1: Procedure `CountTriangles` for counting all triangles in a graph G.

Question 1.3 *Consider a 4-clique $Q = \{u, v, w, x\}$ in G. Show that the four vertices u, v, w, x cannot all belong to the same level V_j.*

Question 1.4 *Consider a 4-clique $Q = \{u, v, w, x\}$ in G, and let j be a positive integer $\leq k$.*

- *assume $u \in V_{j-1}$ and $v, w, x \in V_j$. Show that for one of the tree vertices v, w, x the only incident edge lying in E_j has u has other extremity.*
- *assume $u, w, x \in V_{j-1}$ and $x \in V_j$. Show that the edges incident to x lying in E_j are exactly (u, x), (v, x) and (w, x).*
- *assume $u, v \in V_{j-1}$ and $w, x \in V_j$. Show that one of the vertices w, x has exactly two incident edges lying in E_j (whose other extremities are u and v).*

Question 1.5 *Based on the case analysis of previous question, devise* [2] *a linear time algorithm* `enumerate`$(G, \mathcal{L})$ *that allows us to list all 4-cliques of the input planar graph G, provided with the complete list $\mathcal{L}$ of all triangles contained in G (which is assumed to be pre-computed using the algorithm of question 1.2).*

Exercise 2 – Schnyder woods and graph representations

In this section we want to devise a fast and space-efficient representation of the combinatorial structure of a planar graph. For instance, an adjacency list representation uses $O(n)$ memory words (each of size $O(\log n)$ bits) and allows to check whether an edge (u, v) is in a graph G in $O(deg(u) + deg(v))$ time. On the other hand, an adjacency matrix representation allows us to answer this query in $O(1)$ time but it consumes $\Omega(n^2)$ bits to represent the graph.

Question 2.1 (application of Schnyder woods) *Let G be a planar graph with n vertices. Devise a data structure using at most $O(n)$ memory words (each of size $O(\log n)$ bits) and allows us to answer whether $(u, v) \in G$ in worst case $O(1)$ time per query.*

Warning: *the use of hash tables is not allowed.*

[2] You are asked to provide a high level description, as well as the pseudo-code, of your algorithm and to justify its runtime complexity (with respect to the parameter n).