Maximal Planar Graphs & Triangulation

This section investigates the planar graphs with maximal edge desities.

def maximal planar graph: a planar graph that adding any missing edge would make it non-planar.

e.g. $\boxed{K_4}$  (can't add an edge ...)

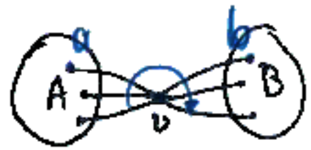
 (add an edge would break the inequality $|E| \leq 3|V|-6$.)

Maximal planar graphs are interesting ~~because~~ for several reasons. First, they are the "worst cases" ~~planar graphs~~ but at the same time valuable examples in understanding the ~~appearance~~ properties of planar graphs. Second, due to the "maximality" restriction, they exhibit very nice structures and could serve as a "normal form" for any planar graphs; we would elucidate this aspect in the following.

Lemma 14

Every maximal planar graph is biconnected.

Proof. Suppose to the contrary that there's a cut vertex v. It separates $G-v$ into at least two components A and B.



We inspect the neighbours of v in circular order.

In the beginning, we meet some neighbours from A, and at some point, we see a neighbour from B. Let $a \in A$ be the guy that we saw just before this point, and $b \in B$ be the guy that we see right now.

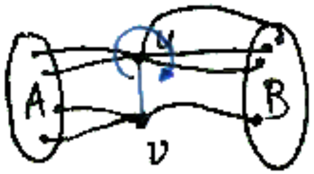
Since v is a cut vertex, $ab \notin E$. However, we could always draw an arc close enough to a-v-b to connect a and b. So the graph $G+ab$ is still planar, contradicting maximality. ■

By a similar argument, we could strengthen Lemma 14:

Lemma 15

Every maximal planar graph is 3-connected.

Proof (Sketch.) Suppose to the contrary that removing u and v would disconnect the graph to A and B.



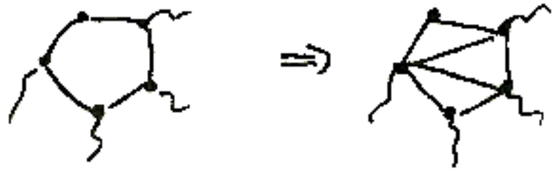
Note that $uv \in E$ since the graph is maximal planar and nothing could prevent us from connecting uv due to our assumption of (<3)-connectivity.

Now we inspect all neighbours of u in circular order as in Lemma 14, starting from $v \in N(u)$ and running clockwise.

Replaying the argument in Lemma 14 would lead us to a contradiction. ■

These results are not so surprising because we expect maximal planar graphs to be denser than a tree or a cycle.

Observing the examples we gave (and maybe other examples you draw yourself), we spot that a maximal planar graph is full of "triangles". Intuitively it's clear as well: if there's a face bounded by ≥ 4 edges, why not add a chord to "triangulate" it?



Applying such intuitive argument needs special care, however. There are at least 2 pitfalls:

(1) A face bounded by ≥ 4 edges doesn't necessarily has a good-looking, cyclic boundary. Perhaps it looks like

(2) Even if it looks nice, we should ensure that we don't add a chord that already exists:

- These pitfalls could be avoided, which we shall do right away.

Lemma 16

Any face in a biconnected plane graph is bounded by a cycle. ↗ in particular, maximal planar graph

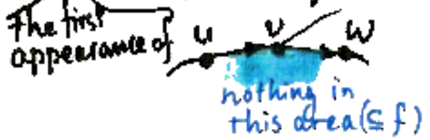
Proof. Suppose there's a face $f \in F(G)$ where ∂f is not a cycle. But ∂f can't be a tree, for otherwise any $e \in \partial f$ has both sides incident to f and thus e doesn't lie on a cycle in G, contradicting biconnectivity.

So ∂f contains at least a cycle. We walk through ∂f and output the sequence of vertices we encounter in each step.

e.g. 

$\Rightarrow (1,4,5,4,6,4,1,2,3)$

If ∂f is not a cycle, then the sequence would have a vertex v appearing twice or more. Let u and w be the predecessor and successor of v respectively.



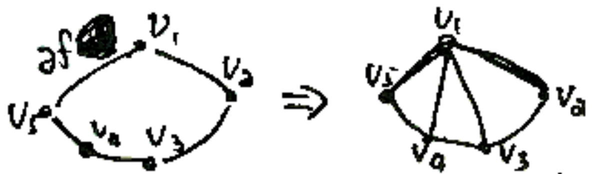
By biconnectivity, $\exists$ a cycle C : uv and vw lie on C. By Jordan Curve Theorem (Theorem 4) C has an interior and an exterior. Say the blue area $\subseteq$ interior of C, then f is enforced to be entirely in the interior of C. But this would tell us that v cannot appear twice in the sequence! ∎

This lemma states something obvious and could be accepted as axiom if you don't want to read the proof. Anyway, it resolve the first pitfall and gurantees that all faces look nice in a maximal planar graph.

Theorem 17

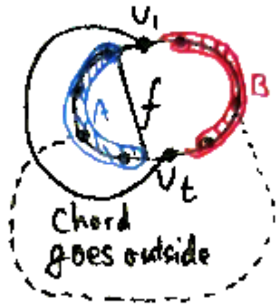
Every face in a maximal planar graph is bounded by a triangle.

proof. Suppose there's a face f where ∂f is not a triangle. Then by Lemma 16, ∂f is a cycle of length ≥ 4. Denote $\partial f =: v_1, \cdots, v_k$ $(k \geq 4)$.

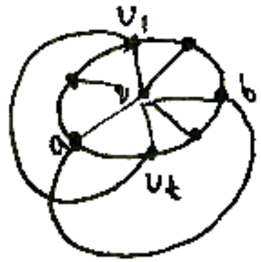


(a) If ~~∂f~~ ∂f contains no chord, then we could safely triangulate it by adding chords from ~~one~~ a fixed vertex to all non-neighboring vertices. Formally, we add edges $v_1 v_i$ for all $3 \leq i \leq k-1$. Since $k \geq 4$ we always add some edge. This contradicts with maximal planarity.

(b) If ∂f contains a chord, say $v_1 v_t$ w.l.o.g., it would cut ∂f into two parts A and B, both non-empty.



We claim that there's no chords between A and B. Suppose there's one, then pictorially, it must intersect with the chord $v_1 v_t$. To make the argument rigorous, we could add a virtual vertex v at the center of f, and connect $v v_i$ $(\forall i)$.



[This operation doesn't change planarity]

One would find out that this graph contains a subdivision of K_5 (with ~~vertices~~ pricipal vertices v, v_1, v_t, a, b), contradicting planarity.

So we know no chord could present be A and B. And it's a routine exercise to see we could triangulate f by adding A-B chords, contradicting the maximal planarity. ■

Remark. The proof is rather constructive and yields an algorithm to augment any non-maximal planar graph to a maximal planar graph. This is usually called "topological triangulation". We will look at an efficient implementation later.

Corollary 18.
The following statements are equivalent:
(1) G is maximal planar.
(2) G is planar and satisfies $|E| = 3|V| - 6$.
(3) G is planar and every face is bounded by a triangle (in a chosen embedding).

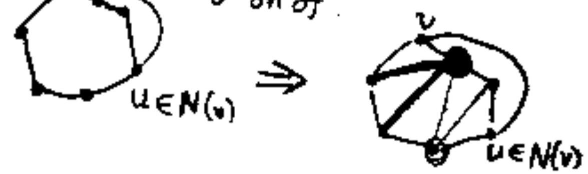
Proof. Exercise. ∎

We conclude the section by presenting a linear algorithm for triangulating a <u>plane</u> graph. By "plane graph" we emphasised that some geometric aspects of the embedding should be given as input.

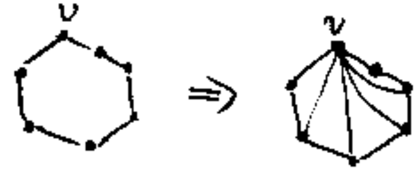
Algorithm: Triangulate.

foreach $f \in F(G)$ **do**
  pick an arbitrary $v \in \partial f$
  let $list[v] := list[v] \cup \{f\}$

foreach $v \in V(G)$ **do**
  mark all $u \in N(v)$.
  foreach $f \in list[v]$ **do**
    go through ∂f clockwise from v
    if bump into a marked vertex u not directly adjacent to v on ∂f **then**



    else



  unmark all $u \in N(v)$

So basically the "if" corresponds to the case where a chord is related to v, and "else" corresponds to the case where v is chordless.

We did ~~the~~ a preprocessing step that

associates each face with a vertex.
This is only for efficiency consideration:
To make each face being accessed exactly
once in the main loop. It should be clear
that the overall running time is linear,
since every edge is traversed only 6 times
(2 for marking, 2 for unmarking, and 2
for traversing faces), every vertex is accessed
once, and every face is processed once.