CS762: Graph-Theoretic Algorithms
Lecture 23: Triangulated Graphs
March 13, 2002

Scribe: Philip Yang

Abstract

Given a planar graph G, there are three ways to make it triangulated, by adding new edges only, by adding new vertices and new edges that are incident to at least one of them, and by adding both edges and vertices. We will give an algorithm for the first method in today's lecture. The algorithm will run in time $O(m)$. Then we define the canonical ordering for a planar graph.

1 Introduction

A simple triangulated graph is a planar graph G such that every face has degree 3, including the outer face. In last lecture we have proved that triangulated graphs are 3-connected. We also discussed that there are three ways to triangulate a planar graph. All can be in done in time $O(m)$. In today's lecture, we are going to discuss the first method in detail.

2 Triangulated planar graphs

In last lecture we briefly mentioned the following algorithm for triangulated planar graphs. Here, we give details, and show that it runs in time $O(m)$.

2.1 Algorithm

The algorithm first finds all faces in a planar graph, which can be done in linear time. One possible way to do it is to use some modified DFS algorithm. Then we skip the faces of degree 3, which are already triangles. For faces of degree greater than 3, we have two choices, see figure 1.

Figure 1(a) shows that v_1 does not connect to any of $v_3, \ldots, v_k - 1$, we add new edges between v_1 and $v_3, \ldots, v_{k-1}$ without introducing multiple edges in the new graph, which is line 8 in our algorithm. Figure 1(b) shows if there is an edge (v_1, v_j) in G but not in face $(v_1, v_2, \ldots, v_k)$, we can use line 11 and 12 in our algorithms to make it triangulated.

We made one change in our algorithm from last lecture: Previously, we let v_1 (Line 4) be an arbitrary vertex on the face, but now we pick a vertex of the minimum degree instead of an arbitrary one. This is needed for the improved time complexity.

2.2 Time Complexity

The running time is not obviously linear. First the algorithm goes through each face of G. The sum of degrees over all faces is twice the number of edges. Now inside each loop, we need to mark

Algorithm 1 Triangulate a Planar Graph G

```
1: fix an arbitrary embedding of a given graph G i
2: for all faces f in this embedding do
3:   if deg(f) $\geqslant$ 4 then
4:     Let $v_1$ be a vertex of minimum degree in $f$
5:     Let $v_2, v_3, \ldots, v_k$ be the remaining vertices of f in clockwise order
6:     Mark $v_1$ and all its neighbors of $v_1$ in the graph
7:     if none of $v_3, \ldots, v_{k-1}$ is marked then
8:       Add edges $(v_1, v_3), (v_1, v_4), \ldots, (c_1, v_{k-1})$
9:     else
10:      {(say $v_j$ is marked)}
11:      Add edges $(v_2, v_{j+1}), (v_2, v_{j+2}), \ldots, (v_2, v_k)$
12:      Add edges $(v_3, v_{j+1}), (v_4, v_{j+1}), \ldots, (v_{j-1}, v_{j+1})$
13:    end if
14:    Unmark $v_1$ and all its neighbors
15:  end if
16: end for
```

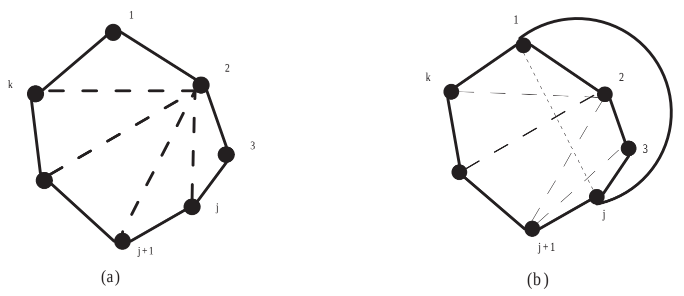

Figure 1: Triangulated graphs (a) and (b)

and unmark the neighbors of a vertex of minimum degree. So the total running time becomes

$$\sum_{f face} \{deg_G(f) + 2min_{v \in f} deg_G(v)\}$$

Because the degree of a vertex can be $O(n)$, the bound for the second term in our equation could be as bad as $O(n^2)$. Before we start our analysis, we introduce some new theorems.

Definition 1 *Let G be a graph. Assume there are k-forests $F_1, F_2, \ldots, F_k$ on the vertices of G such that E(G) $\subseteq E(F_1) \cup E(F_2) \cup \cdots \cup E(F_k)$. We say that G has arboricity a(G) $\leqslant$ k.[NC88]*

We will see the following result in the next lecture.

Theorem 1 *Every simple planar graph has arboricity $\leqslant$ 3*

We can use arboricity for obtaining bounds running time because of the following result:

Lemma 1 *Let G=(V,E) be a graph, then*

$$\sum_{(v,w) \in E} min\{deg_G(v), deg_G(w)\} \leqslant a(G) \cdot 2m$$

Proof: Let F_i $(1 \leqslant i \leqslant a(G))$ be the edge-disjoint forests of G such that E $= \cup_{1 \leqslant i \leqslant a(G)} E(F_i)$. Arbitrarily pick a vertex in each tree of F_i, and regard the tree as rooted at this vertex, with all edges directed from the parent to the child. Thus we can orient F_i such that any vertex v in F_i has $indeg_{F_i} \leq 1$. Now

$$\begin{aligned}
\sum_{(v,w)\in E(F_i)} \min\{\deg_G(v), \deg_G(w)\} &= \sum_{v\to w\in E(F_i)} \min\{\deg_G(v), \deg_G(w)\} \\
&\leq \sum_{v\to w\in E(F_i)} \deg_G(w) \\
&= \sum_{w\in V(F_i)} indeg_{F_i}(w) \deg_G(w) \\
&\leq \sum_{w\in V(F_i)} \deg_G(w) \\
&\leq 2m,
\end{aligned}$$

therefore

$$\sum_{(v,w)\in E(G)} \min\{\deg_G(v), \deg_G(w)\} = \sum_{i=1}^{a(G)} \sum_{(v,w)\in E(F_i)} \min\{\deg_G(v), \deg_G(w)\} \leq \sum_{i=1}^{a(G)} 2m = 2ma(G)$$

as desired. ■

This technique does not give good bounds for any non-planar graph. For example, if a graph G is a complete graph, then $\min\{deg(v), deg(w)\} = n-1$ for any edge (v, w) and $\sum_{(v,w)\in E} min\{deg(v), deg(w)\} \in \theta(n^3)$ since the complete graph has $\theta(n^2)$ edges. But for planar graphs, this gives a linear term.

We finish the analysis of the running time by computing the summation of minimum degrees over all edges in a face. The total running time of our algorithm is

$$\begin{aligned}
\sum_{f face} \{deg_G(f) + 2min_{v\in f} deg_G(v)\} &\leqslant 2m + 2 \sum_{f face} \sum_{(v,w)\in E \text{ on a face } f} min\{deg_G(v), deg_G(w)\} \\
&= 2m + 4 \sum_{(v,w)\in E} \{deg_G(v), deg_G(w)\} \\
&= 2m + 4\cdot 2\cdot 3\cdot m \\
&= O(m) = O(n)
\end{aligned}$$

for a planar graph.

2.3 Correctness

[This section was not actually covered in class and has been added by T. Biedl.]

In the previous lecture, we argued that our algorithm cannot possibly introduce multiple edges, because we check whether we are introducing multiple edges if we add edges from v_1, and there cannot be multiple edges because of planarity in the other case. However, as written, our algorithm might introduce loops. More precisely, assume that v_1 is a cutvertex, with one of its incident biconnected components inside the face. (See also Figure 2.) Then we might accidentally introduce a loop at vertex v_1.

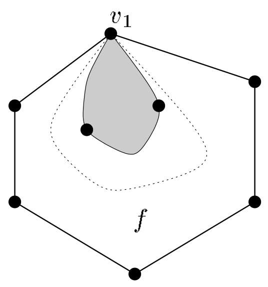

Figure 2: If v_1 is a cut vertex, then adding edges might create a loop.

We could easily test whether this happens at v_1 (it happens if and only if v_1 appears at least twice on the list of incident vertices of the face), but we would have to do this repeatedly for all vertices on the face, which would make linear time complexity difficult (though feasible). Instead, we will go another route.

It is not difficult to observe that we can add a loop only in the case that the endpoint of the loop is a cutvertex. So if we simply remove all cutvertices first (by adding edges appropriately), then we will never add a loop. The algorithm for making the graph biconnected is illustrated in Figure 3 and given as follows:

- Find the cutvertices and biconnected components, and build the block tree.
- For all cutvertices v
  - For any pair of neighbouring biconnected components B_1 and B_2 of v
    - Let v_1 and v_2 be the two neighbours of v that are consecutive (in the cyclic order of edges around v) and are in B_1 and B_2.
    - Add edge (v_1, v_2).

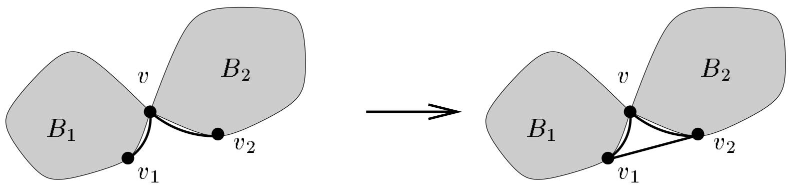

Figure 3: Removing a cutvertex by adding an edge.

Note that adding one such edge will turn the two biconnected components into one biconnected components, so every added edge reduces the number of biconnected components by one, and at the end the graph is biconnected.

Since computing the block tree can be done in $O(m)$ time, and the rest of the algorithm is proportional to the size of the blocktree (which is also $O(m)$), this takes $O(m)$ time. So in all, making a planar graph triangulated takes $O(m) = O(n)$ time.

3 Canonical Ordering

Now we start a new topic and introduce the canonical ordering, which is a useful tool for graph drawing and other applications.

Definition 2 *A canonical ordering is a vertex order $v_1, v_2, \ldots, v_n$ of a triangulated planar graph such that v_1, v_2, v_n are vertices in the outerface, and for all k with $3 \leq k \leq n-1$, we have*

1. *the graph G_k induced by $v_1, v_2, \ldots, v_k$ is 2-connected and internally triangulated,*
2. *the outer face of G_k contains edges (v_1, v_2),*
3. *v_{k+1} is in the outer face of G_k,*
4. *the neighbors of v_{k+1} in G_k are an interval on the outerface of G_k*

The definition was first given in [FPP90], Figure 3 illustrates the concept of a canonical order.

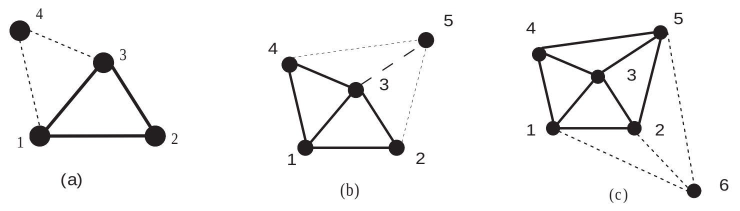

Figure 4: Illustration of a canonical order. (c) is incorrect

Figure 3(a) shows the base case, which is an induced graph on vertices v_1, v_2 and v_3, and we are going to add v_4 into it. By definition any induced subgraph G_k is 2-connected, so v_4 must connect at least two vertices of v_1, v_2 and v_3. Figure 3 (b) shows us if there are edges (v_2, v_5) and (v_4, v_5), then by our definition, (v_3, v_5) must exist in G_5. At the end we need to consider one thing, how to get (v_1, v_2, v_n) as the outer face. Figure 3(c) shows that we can not add a vertex such that (v_1, v_2) is removed from the outer-face.

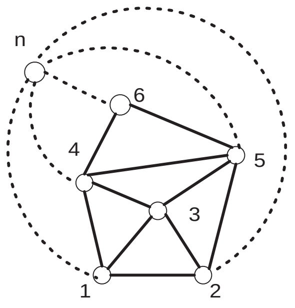

Figure 5: Construct a triangulated graph

Figure 4 shows us that v_n must connect to all vertices on the outer face of G_{n-1}, which is obvious. Otherwise we will have an external face which has degree different from 3.

The next lecture will illustrate how to use the canonical ordering for graph drawing.

References

[FPP90] H.de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica 10*, 1990.

[Mur01] U.S.R Murty. *C&O Graph Theory Course Note.* University of Waterloo, C&O Department, 2001.

[NC88] T. Nishizeki and N. Chiba. *Planar Graphs: Theory and Algorithms.* North-Holland, 1988.