Chapter 7

PLANAR GRAPHS

7.1 BRIDGES AND KURATOWSKI'S THEOREM

Consider a graph drawn in the plane in such a way that each vertex is represented by a point; each edge is represented by a continuous line connecting the two points which represent its end vertices and no two lines, which represent edges, share any points, except in their ends. Such a drawing is called a *plane graph*. If a graph G has a representation in the plane which is a plane graph then it is said to be *planar*.

In this chapter we shall discuss some of the classical work concerning planar graphs. The question of efficiently testing whether a given finite graph is planar will be discussed in the next chapter.

Let S be a set of vertices of a nonseparable graph $G(V, E)$. Consider the partition of the set $V\text{-}S$ into classes, such that two vertices are in the same class if and only if there is a path connecting them which does not use any vertex of S. Each such class K defines a *component* as follows: The component is a subgraph $H(V', E')$ where $V' \supset K$. In addition, V' includes all the vertices of S which are connected by an edge to a vertex of K, in G. E' contains all edges of G which have at least one end vertex in K. An edge $u \overset{e}{-} v$, where both u and v are in S, defines a *singular component* ($\{u, v\}$, $\{e\}$). Clearly, two components share no edges, and the only vertices they can share are elements of S. The vertices of a component which are elements of S are called its *attachments*.

In our study we usually use a set S which is the set of vertices of a simple circuit C. In this case we call the components *bridges*; The edges of C are not considered bridges.

For example, consider the plane graph shown in Fig. 7.1. Let C be the outside boundary: $a \overset{1}{-} b \overset{2}{-} c \overset{3}{-} \cdots \overset{6}{-} g \overset{7}{-} a$. In this case the bridges are: ($\{e, g\}$,$\{8\}$), ($\{h, i, j, a, e, g\}$, $\{9, 10, 11, 12, 13, 14\}$), ($\{a, e\}$, $\{15\}$) and ($\{k, b, c, d\}$, $\{16, 17, 18\}$). The first and third bridges are singular.

Lemma 7.1: Let B be a bridge and a_1, a_2, a_3, three of its attachments. There exists a vertex v, not an attachment, for which there are three vertex

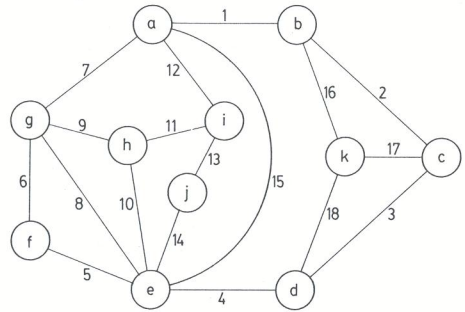


Figure 7.1

disjoint paths in B: $P_1(v, a_1)$, $P_2(v, a_2)$ and $P_3(v, a_3)$. ($P(a, b)$ denotes a path connecting a to b.)

Proof: Let $a_1 \overset{e_1}{-} v_1$, $a_2 \overset{e_2}{-} v_2$, $a_3 \overset{e_3}{-} v_3$ be edges of B. If any of the v_i's ($i = 1$, 2, 3) is an attachment then the corresponding edge is a singular bridge and is not part of B. Thus, $v_i \in K$, where K is the class which defines B. Hence, there is a simple path $P'(v_1, v_2)$ which uses vertices of K only; if $v_1 = v_2$, this path is empty. Also, there is a simple path $P''(v_3, v_1)$ which uses vertices of K only. Let v be the first vertex of $P''(v_3, v_1)$ which is also on P'. Now, let $P_1(v, a_1)$ be the part of P' which leads from v to v_1, concatenated with $v_1 - a_1$; $P_2(v, a_2)$ be the part of P' which leads from v to v_2, concatenated with $v_2 - a_2$; $P_3(v, a_3)$ be the part of P'' which leads from v to v_3, concatenated with $v_3 - a_3$. It is easy to see that these paths are disjoint.

Q.E.D.

Let C be a simple circuit of a nonseparable graph G, and $B_1, B_2, \ldots, B_k$ be the bridges with respect to C. We say that B_i and B_j *interlace* if at least one of the following conditions holds:

(i) There are two attachments of B_i, a and b, and two attachments of B_j, c and d, such that all four are distinct and they appear on C in the order a, c, b, d.
(ii) There are three attachments common to B_i and B_j.

For each bridge B_i, consider the subgraph $C + B_i$. If any of these graphs is not planar then clearly G is not planar. Now, assume all these subgraphs are planar. In every plane realization of G, C outlines a contour which divides the plane into two disjoint parts: its inside and outside. Each bridge must lie entirely in one of these parts. Clearly, if two bridges interlace they cannot be on the same side of C. Thus, in every plane realization of G the set of bridges is partitioned into two sets: those which are drawn inside C and those which are drawn outside. No two bridges in the same set interlace.

Lemma 7.2: If $B_1, B_2, \ldots, B_m$ is set of bridges of a nonseparable graph G with respect to a simple circuit C and the following two conditions are satisfied:

(1) For every $1 \leq i \leq n$, $C + B_i$ is planar,
(2) No two bridges interlace,
then $C + B_1 + B_2 + \cdots + B_m$ has a plane realization in which all these bridges are inside C.

Proof: We shall only outline the proof. As we go clockwise around C there must be a bridge B_i such that we encounter all its attachments in some order: $a_1, a_2, \ldots, a_t$, and no attachment of any other bridge appears between a_1 and a_t on C. Such a bridge can be found by starting from any attachment a of B_1 and going around, say clockwise. If before encountering all the attachments of B_1 we encounter an attachment of another bridge B_i, then all B_i's attachments are between two consecutive attachments of B_1 which may also belong to B_i. We repeat the same process on B_i, etc. Since the number of bridges is finite, and those discarded will not "interfere" with the new ones, the process will yield the desired bridge.

This observation allows a proof by induction. First B_i is drawn, and since no other bridge uses any of the vertices of C between a_1 and a_t, we can take C' to be the circuit which describes the part of C from a_t to a_1, clockwise, and the boundary of B_i, from a_1 to a_t, to form a simple circuit C', whose inside is so far empty. The remaining bridges are also bridges of C' and, clearly, satisfy (1) and (2) with respect to C'.

Q.E.D.

Corollary 7.1: Let G be a nonseparable graph and C a simple circuit in G. G is planar if and only if the bridges $B_1, B_2, \ldots, B_k$ of G, with respect to C, satisfy the following conditions:

(1) For every $1 \leq i \leq k$, $C + B_i$ is planar.

(2) The set of bridges can be partitioned into two subsets, such that no two bridges in the same subset interlace.

Let us introduce coloring of graphs. Here we consider only 2-coloring of vertices. In later chapters we shall discuss more general colorings. A graph $G(V, E)$ is said to be *2-colorable* if V can be partitioned into V_1 and V_2 in such a way that there is no edge in G with two end vertices in $V_1(V_2)$. (Obviously, a 2-colorable graph is bipartite, and vice versa. It is customary to use the term "bipartite" if the partition is given, and "2-colorable", if one exists.) The following theorem is due to König [1].

Theorem 7.1: A graph G is 2-colorable if and only if it has no odd length circuits.

Proof: It is easy to see that if a graph has an odd length circuit then it is not 2-colorable. In order to prove the converse, we may assume that G is connected; for if each component is 2-colorable then the whole graph is 2-colorable.

Let v be any vertex. Let us perform BFS (Section 1.5) starting from v. There cannot be an edge $u - w$ in G, if u and w belong to the same layer; i.e. are the same distance from v. For if such an edge exists then we can display an odd length circuit as follows: Let $P_1(v, u)$ be a shortest path from v to u. $P_2(v, w)$ is defined similarly and is of the same length. Let x be the last vertex which is common to P_1 and P_2. The part of P_1 from x to u, and the part of P_2 from x to w are of equal length, and together with $u - w$ they form an odd length simple circuit.

Now, we can color all the vertices of even distance from v with one color and all the vertices of odd distance from v with a second color.

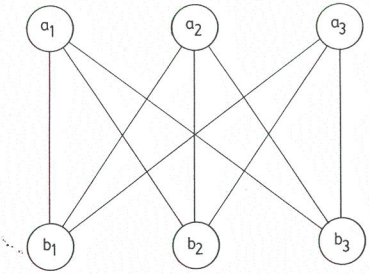
Q.E.D.

We can use the concept of 2-colorability to decide whether the bridges $B_1, B_2, \ldots, B_k$, with respect to a simple circuit C can be partitioned into two subsets, each pairwise non interlacing, as follows: Construct a graph G' whose vertices are the bridges $B_1, B_2, \ldots, B_k$, and two vertices are connected by an edge if and only if the corresponding bridges in G interlace. Now test whether the graph G' is 2-colorable, by giving one vertex color 1 and using some search technique, such as DFS or BFS to color alternate vertices with different colors out of $\{1, 2\}$. If no contradiction arises, a coloring is obtained and thus, a partition. If a contradiction occurs, there is no 2-coloring, and therefore, no partition of the bridges; in this case the graph is nonplanar, by Corollary 7.1.

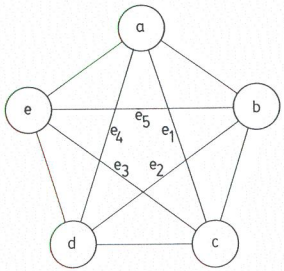
Let us now introduce the graphs of Kuratowski [2]: $K_{3,3}$ and K_5. They are shown in Fig. 7.2(a) and (b) respectively.

K_5 is a completely connected graph of 5 vertices, or a clique of 5 vertices. $K_{3,3}$ is a completely connected bipartite graph with 3 vertices on each side.

Lemma 7.3: Neither $K_{3,3}$ nor K_5 is planar



(a)



(b)

Figure 7.2

Proof: First, let us consider K_5, and its circuit C: $a - b - c - d - e - a$. Clearly, there are 5 bridges, all singular, corresponding to the edges e_1, e_2, e_3, e_4 and e_5. Let us construct G', as in the preceding discussion. It is shown in Fig. 7.3. For example, the bridge e_1 interlaces with e_5 and e_2, etc. Since G' contains an odd circuit, by Theorem 7.1 it is not 2-colorable, and the set of bridges of K_5 with respect to C is not partitionable. Thus, by Corollary 7.1, K_5 is not planar.

In the case of $K_{3,3}$, take C: $a_1 - b_1 - a_2 - b_2 - a_3 - b_3 - a_1$. The bridges $a_1 - b_2$, $a_2 - b_3$ and $a_3 - b_1$ form a triangle in the corresponding G'.

Q.E.D.

Before we take on Kuratowski's Theorem we need a few more definitions and a lemma.

Let $G(V, E)$ be a finite nonseparable plane graph with $|V| > 2$. A *face* of G is a maximal part of the plane such that for every two points x and y in it there is a continuous line from x and y which does not share with the realization of G any point. The contour of each face is a simple circuit of G; if any of these circuits is not simple then G is separable. Each of these circuits is called a *window*. One of the faces is of infinite area. It is called the *external* face, and its window is the external window. It is not hard to show that for every window W, there exists another plane realization, G', of the same graph, which has the same windows, but in G' W is external. First draw the graph on a sphere, maintaining the windows; this can be achieved by projecting

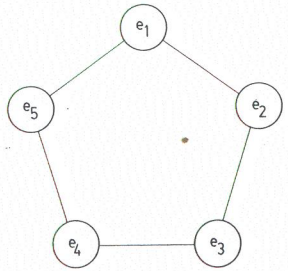


Figure 7.3

each point of the plane vertically up to the surface of a sphere whose center is in the plane and its intersecting circle with the plane encircles G. Next, place the sphere on a plane which is tangent to it, in such a way that a point in the face whose contour is W is the "north pole", i.e. furthest from the plane. Project each point P (other then the "north pole") of the sphere to the plane by a straight line which starts from the "north pole" and goes through P. The graph is now drawn in the plane and W is the external window.

Lemma 7.4: Let $G(V, E)$ be a 2-vertex connected graph with a separating pair a, b. Let $H_1, H_2, \ldots, H_m$ be the components with respect to $\{a, b\}$. G is planar if and only if for every $1 \leq i \leq m$ $H_i + (a \overset{e}{-} b)$ is planar, where $e \notin E$.

By $H_i + (a \overset{e}{-} b)$ we mean that a new edge connecting a and b is added to H_i.

Proof: In each H_i there is a path from a to b, or G is not 2-connected. Also $m > 1$. Thus, for each H_i we can find a path P from a to b in one of the other components. If G is planar, so is $H_i + P$, and therefore $H_i + (a \overset{e}{-} b)$ is planar. Now assume that each $H_i + (a \overset{e}{-} b)$ is planar. For each of these realization we can assume that e is on the external window. Thus, a planar realization of G exists, as demonstrated in Fig. 7.4 in the case of $m = 3$.

Q.E.D.

Two graphs are said to be *homeomorphic* if both can be obtained from the same graph by the insertion of new vertices of degree 2, in edges; i.e. an edge is replaced by a path whose intermediate vertices are all new.* Clearly, if two graphs are homeomorphic then either both are planar or both are not. We are now ready to state Kuratowski's Theorem [1].

Theorem 7.2: A graph G is nonplanar if and only if there is a subgraph of G which is homeomorphic to either $K_{3,3}$ or K_5.

Proof: If G has a subgraph H which is homeomorphic to either $K_{3,3}$ or K_5, then by Lemma 7.3, H is nonplanar, and therefore G is nonplanar. The converse is much harder to prove. We prove it by contradiction.

Let G be a graph which is nonplanar and which does not contain a

*Two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ are said to be **isomorphic** if there are 1-1 correspondences $f: V_1 \to V_2$ and $g: E_1 \to E_2$ such that for every edge $u \overset{e}{-} v$ in G_1 $f(u) \overset{g(e)}{-} f(v)$ in G_2. Clearly G_1 is planar if and only if G_2 is. Thus, we are not interested in the particular names of the vertices or edges, and distinguish between graphs only up to isomorphism.

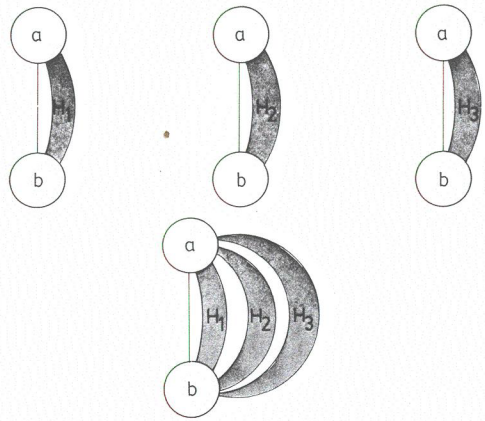


Figure 7.4

subgraph which is homeomorphic to one of Kuratowski's graphs; in addition let us assume that among such graphs G has the minimum number of edges.

First, let us show that the vertex connectivity of G is at least 3. Clearly, G is connected and nonseparable, or the number of its edges is not minimum. Assume it has a separating pair $\{a, b\}$, and let $H_1, H_2, \ldots, H_m$ be the components with respect to $\{a, b\}$, where $m > 1$. By Lemma 7.4, there exists an $1 \leq i \leq m$ for which $H_i + (a - b)$ is nonplanar. Clearly $H_i + (a - b)$ does not contain a subgraph which is homeomorphic to one of Kuratowski's graphs either. This contradicts the assumption that G has the minimum number of edges.

We now omit an edge $a_0 \overset{e_0}{-} b_0$ from G. The resulting graph, G_0, is planar. Since the connectivity of G is at least 3, G_0 is nonseparable. By Theorem 6.7, there is a simple circuit in G_0 which goes through a_0 and b_0. Let $\tilde{G}_0$ be a plane realization of G_0 and C be a simple circuit which goes through a_0 and b_0, such that C encircles the maximum number of faces of all such circuits in all the plane realizations of G_0. Note that the selection of $\tilde{G}_0$ and C is done simultaneously and not successively. Assuming u and v are vertices on C, let

$C[u, v]$ be the part of C going from u to v, clockwise. $C(u, v)$ is defined similarly, but the vertices u and v are excluded.

Consider now the external bridges of C in $\hat{G}_0$. If such a bridge, B, has two attachments either on $C[a_0, b_0]$ or $C[b_0, a_0]$ then C is not maximum. To see this, assume B has two attachments, a and b, in $C[a_0, b_0]$. There is a simple path $P(a, b)$ connecting a to b via edges and vertices of B, which is disjoint from C, and therefore is exterior to C. Form C' by adding to P the path $C[b, a]$. C' goes through a_0 and b_0, and the interior of C is either completely included in the interior of C' or in the exterior. In the first case C' has a larger interior than C, in $\hat{G}_0$. In the latter case we have to find another plane realization which is similar to G_0, but the exterior of C' is now the interior. In either case the maximality of the choice of $\hat{G}_0$ and C is contradictory.

By a similar argument we can also prove that neither a_0 nor b_0 can be an attachment of an external bridge. Thus, each bridge has one attachment in $C(a_0, b_0)$ and one in $C(b_0, a_0)$. Each of these bridges is singular (consists of a single edge); for otherwise its two attachments are a separating pair in G, contradicting its 3-connectivity. Finally, there is at least one such external singular bridge, or one could draw the edge e_0 outside C, to yield a planar realization of G. Similarly, there must be an internal bridge, B^*, which prevents the drawing of e_0 inside, and which cannot be transferred outside; i.e. B^* interlaces with an external singular bridge, say $a_1 \overset{e_1}{-} b_1$. The situation is schematically shown in Fig. 7.5. We divide the argument to two cases according to whether B^* has any attachment other than a_0, b_0, a_1, b_1.

Case 1: B^* has an attachment a_2 other than a_0, b_0, a_1, b_1. Without loss of generality we may assume that a_2 is on $C(a_1, a_0)$. Since B^* prevents the drawing of e_0, it must have an attachment on $C(a_0, b_0)$. Since B^* interlaces e_1, it must have an attachment on $C(b_1, a_1)$.

Case 1.1: G^* has an attachment b_2 on $C(b_1, b_0)$. In B^*, there is a path P connecting a_2 with b_2. The situation is shown in Fig. 7.6. The subgraph of G shown in Fig. 7.6 is homeomorphic to $K_{3,3}$, where a_1, a_0 and b_2 play the role of the upper vertices of Fig. 7.2(a), and a_2, b_1 *and* b_0 play the role of the lower vertices.

Case 1.2: B^* has no attachment on $C(b_1, b_0)$. Thus, B^* has one attachment b_2' on $C(a_0, b_1]$; i.e., it may be b_1 but not a_0. Also, B^* has an attachment b_2'' on $C[b_0, a_1)$. By Lemma 7.1, there exists a vertex v and three vertex disjoint paths in B^*: $P_1(v, a_2)$, $P_2(v, b_2')$ and $P_3(v, b_2'')$. The situation is shown in Fig. 7.7. If we erase from the subgraph of G, shown in Fig. 7.7 the path $C[b_1, b_0]$ and all its intermediate vertices, the resulting subgraph is

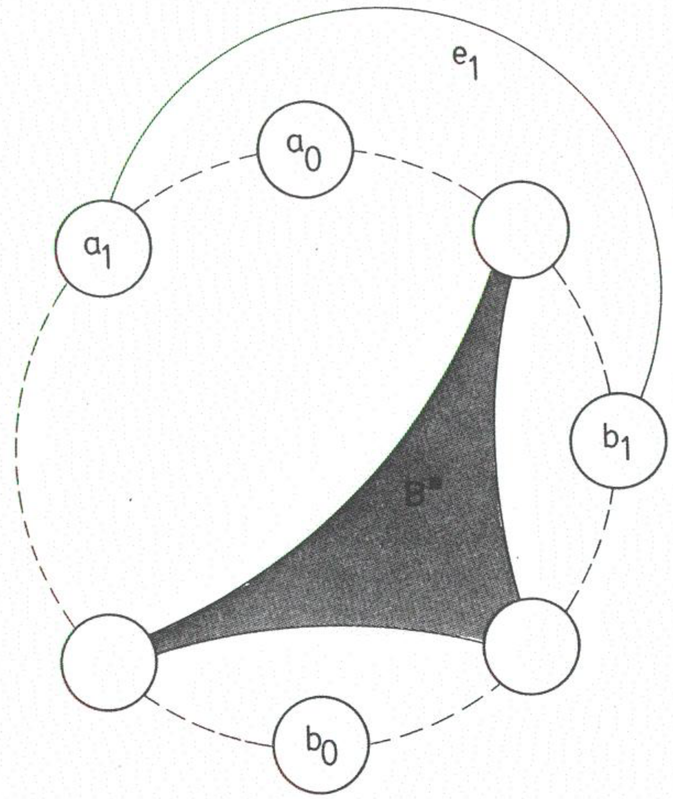


Figure 7.5

homeomorphic to $K_{3,3}$: Vertices a_2, b_2' and b_2'' play the role of the upper vertices, and a_0, a_1 and v, the lower vertices.

Case 2: B^* has no attachments other than a_0, b_0, a_1, b_1. In this case all four must be attachments; for if a_0 or b_0 are not, then B^* and e_1 do not interlace; if a_1 or b_1 are not, then B^* does not prevent the drawing of e_0.

Case 2.1 There is a vertex v, in B^*, from which there are four disjoint paths in B^*: $P_1(v, a_0)$, $P_2(v, b_0)$, $P_3(v, a_1)$ and $P_4(v, b_1)$. This case is shown in Fig. 7.8, and the shown subgraph is clearly homeomorphic to K_5.

Case 2.2: No vertex as in Case 2.1 exists. Let $P_0(a_0, b_0)$ and $P_1(a_1, b_1)$ be two simple paths in B^*. Let c_1 be the first vertex on P_1 which is common with P_0, and let c_2 be the last on P_1 which is common with P_0. We use only the first part, A, of P_1, connecting a_1 and c_1, and the last part, B, connecting c_2 with b_1. The pertaining subgraph of G is now shown in Fig. 7.9, and is

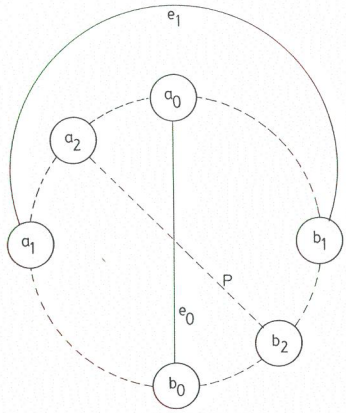


Figure 7.6

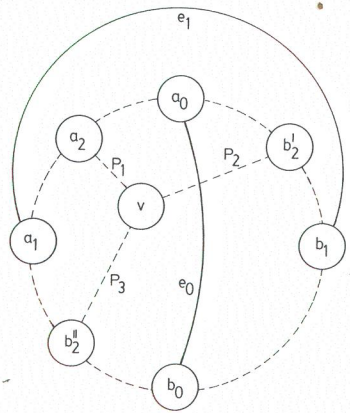


Figure 7.7

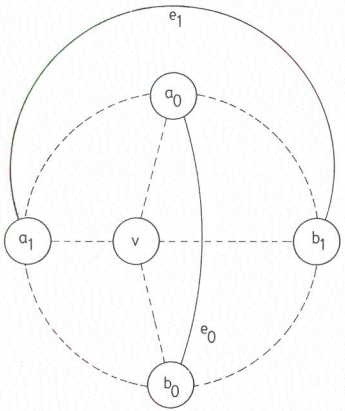


Figure 7.8

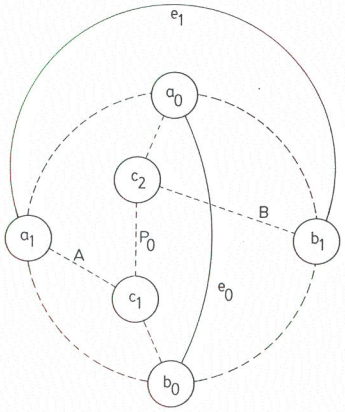


Figure 7.9

homeomorphic to $K_{3,3}$, after $C[a_0, b_1]$ and $C[b_0, a_1]$ and all their intermediate vertices are erased: Vertices a_0, b_1 and c_1 play the role of the upper vertices and b_0, a_1 and c_2, the lower. (If c_1 is closer to a_0 than c_2 then we erase $C[a_1, a_0]$ and $C[b_1, b_0]$, instead, and the upper vertices are a_0, a_1 and c_2.)

Q.E.D.

Kuratowski's theorem provides a necessary and sufficient condition for a graph to be planar. However, it does not yield an efficient algorithm for planarity testing. The obvious procedure, that of trying for all subsets of 5 vertices to see whether there are 10 vertex disjoint paths connecting all pairs, or for every pairs of 3 and 3 vertices whether there are 9 paths, suffers from two shortcomings. First there are $\binom{|V|}{5}$ choices of 5-sets and $\frac{1}{2}\binom{|V|}{3}\cdot\binom{|V|-3}{3}$ choices of 3 and 3 vertices; this alone is $O(|V|^6)$. But what is worse, we have no efficient way to look for the disjoint paths; this problem may, in fact, be exponential.

Fortunately, there are $O(|E|)$ tests, as we shall see in the next chapter, for testing whether a given graph is planar.

7.2 EQUIVALENCE

Let G_1 and $\hat{G}_2$ be two plane realizations of the graph G. We say that $\hat{G}_1$ and $\hat{G}_2$ are *equivalent* if every window of one of them is also a window in the other. G may be 2-connected and have nonequivalent plane realization; for example see Fig. 7.10.

Let us restrict our discussion to planar finite graphs, with no parallel edges and no self-loops, Our aim is to show that if the vertex connectivity of G, $c(G)$, is at least 3 then the plane realization of G is unique up to equivalence.

Lemma 7.5: A planar nonseparable graph G is 2-connected if there is a plane realization of it, $\hat{G}$, and one of its windows has more than one bridge.

Proof: If C is a window of $\hat{G}$ with more than one bridge, then all C's bridges are external. Therefore, no two bridges interlace. As in the first paragraph of the proof of Lemma 7.2, there exists a bridge B whose attachments can be ordered $a_1, a_2, \ldots, a_t$ and no attachments of any other bridge appear on $C(a_1, a_t)$. It is easy to see that $\{a_1, a_t\}$ is a separating pair; it separates the vertices of B and $C(a_1, a_t)$ from the set of vertices of all other bridges and $C(a_t, a_1)$, where neither set can be empty since G has no parallel edges.

Q.E.D.

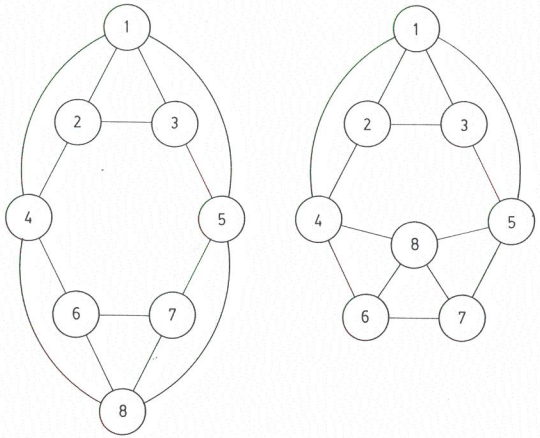


Figure 7.10

Theorem 7.3: If G is a plane graph with no parallel edges and no self-loops and if its vertex connectivity, $c(G)$, is at least 3, then every two plane realizations of G are equivalent.

Proof: Assume G has two plane realizations $\hat{G}_1$ and $\hat{G}_2$ which are not equivalent. Without loss of generality we may assume that there is a window C in $\hat{G}_1$ which is not a window in $\hat{G}_2$. Therefore, C has at least two bridges. By Lemma 7.5, and since C is a window in $\hat{G}_1$, G is 2-connected. A contradiction, since $c(G) \geq 3$.

Q.E.D.

7.3 EULER'S THEOREM

The following theorem is due to Euler.

Theorem 7.4: Let $G(V, E)$ be a non-empty connected plane graph. The number of faces, f, satisfies

$$|V| + f - |E| = 2. \tag{7.1}$$

Proof: By induction on $|E|$. If $|E| = 0$ then G consists of one vertex and there is one face, and 7.1 holds. Assume the theorem holds for all graphs with $m = |E|$. Let $G(V, E)$ be a connected plane graph with $m + 1$ edges. If G contains a circuit then we can remove one of its edges. The resulting plane graph is connected and has m edges, and therefore, by the inductive hypothesis, satisfies (7.1). Adding back the edge increases the number of faces by one and the number of edges by one, and thus (7.1) is maintained. If G contains no circuits, then it is a tree. By Lemma 2.1 it has at least two leaves. Removing a leaf and its incident edge yields a connected graph with one less edge and one less vertex which satisfies (7.1). Therefore, G satisfies (7.1) too.

Q.E.D.

The theorem implies that all connected plane graphs with $|V|$ vertices and $|E|$ edges have the same number of faces. There are many conclusions one can draw from the theorem. Some of them are the following:

Corollary 7.1: If $G(V, E)$ is a connected plane graph with no parallel edges, no self-loops and $|V| > 2$, then

$$|E| \leq 3|V| - 6. \tag{7.2}$$

Proof: Since there are no parallel edges, every window consists of at least three edges. Each edge appears on the windows of two faces, or twice on the window of one face. Thus, $3 \cdot f \leq 2 \cdot |E|$. By (7.1), $|E| = |V| + f - 2$. Thus, $|E| \leq |V| + \frac{2}{3}|E| - 2$, and (7.2) follows.

Q.E.D.

Corollary 7.2: Every connected plane graph with no parallel edges and no self-loops has at least one vertex whose degree is 5 or less.

Proof: Assume the contrary; i.e. the degree of every vertex is at least 6. Thus, $2 \cdot |E| \geq 6 \cdot |V|$; note that each edge is counted in each of its two end vertices. This contradicts (7.2).

Q.E.D.

7.4 DUALITY

Let $G(V, E)$ be a finite undirected and connected graph. A set $K \in E$ is called a *cutset* if it is a minimal separating set of edges; i.e. the removal of K from G interrupts its connectivity, but no proper subset of K does it. It is easy to see that a cutset separates G into two connected components: Consider first the removal of $K - \{e\}$, where $e \in K$. G remains connected. Now remove e. Clearly G breaks into two components.

The graph $G_2(V_2, E_2)$ is said to be the *dual* of a connected graph $G_1(V_1, E_1)$ if there is a $1 - 1$ correspondence $f: E_1 \to E_2$, such that a set of edges S forms a simple circuit in G_1 if and only if $f(S)$ (the corresponding set of edges in G_2) forms a cutset in G_2. Consider the graph G_1 shown in Fig. 7.11(a). G_2 shown in Fig. 7.11(b) is a dual of G_1, but so is G_3, shown in Fig. 7.11(c), as the reader can verify by considering all (six) simple circuits of G_1 and all cusets of G_2, or G_3.

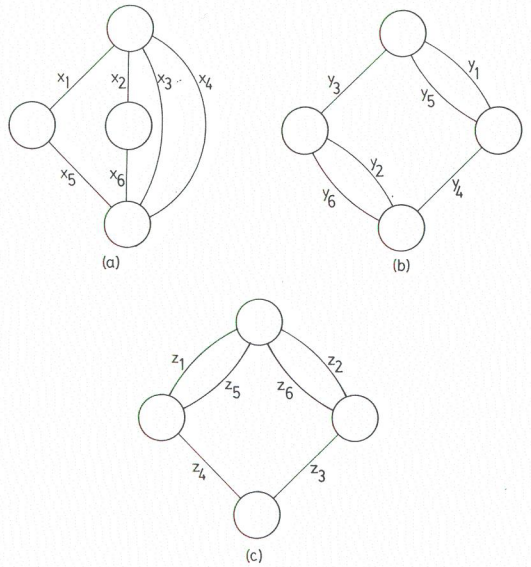


Figure 7.11

A *contraction* of an edge $x \overset{e}{-} y$ of a graph $G(V, E)$ is the following operation: Delete the edge e and merge x with y. The new contracted graph, G', has one less edge and one less vertex, if $x \neq y$. Clearly, if G is connected so is G'. A graph G' is a contraction of G if by repeated contractions we can construct G' from G.

Lemma 7.6: If a connected graph G_1 has a dual and G_1' is a connected subgraph of G_1 then G_1' has a dual.

Proof: We can get G_1' from G_1 by a sequence of two kinds of deletions:

(i) A deletion of an edge e of the present graph, which is not in G_1', and whose deletion does not interrupt the connectivity of the present graph.
(ii) A deletion of a leaf of the present graph, which is not a vertex of G_1', together with its incident edge.

We want to show that each of the resulting graphs, starting with G_1 and ending with G_1', has a dual.

Let G be one of these graphs, except the last, and its dual be G_d. First consider a deletion of type (i), of an edge e. Contract $f(e)$ in G_d, to get G_{dc}, If C is a simple circuit in $G - e$, then clearly it cannot use e, and therefore it is a circuit in G too. The set of edges of C is denoted by S. Thus, $f(S)$ is a cutset of G_d, and it does not include $f(e)$. Thus, the end vertices of $f(e)$ are in the same component of G_d with respect to $f(S)$. It follows that $f(S)$ is a cutset of G_{dc} too. If K is a cutset of G_{dc} then it is a cutset of G_d too. Thus, $f^{-1}(K)$ form a simple circuit C' in G. However, $f(e)$ is not in K, and therefore e is not in C'. Hence, C' is a simple circuit of $G - e$.

Next, consider a deletion of type (ii) of a leaf v and its incident edge e. Clearly, e, plays no role in a circuit. Thus, $f(e)$ cannot be a part of a cutset in G_d. Hence, $f(e)$ is a self-loop. The deletion of v and e from G, and the contraction of $f(e)$ in G_d (which effectively, only deletes $f(e)$ from G_d), does not change the sets of simple circuits in G and cutsets in G_d, and the correspondence is maintained.

Q.E.D.

Lemma 7.7: Let G be a connected graph and e_1, e_2 be two of its edges, neither of which is a self loop. If for every cutset either both edges are in it or both are not, then e_1 and e_2 are parallel edges.

Proof: If e_1 and e_2 are not parallel, then there exists a spanning tree which includes both. (Such a tree can be found by contracting both edges and finding a spanning tree of the contracted graph.) The edge e_1 is separating the tree into two connected components whose sets of vertices are S and $\bar{S}$. The set of edges between S and $\bar{S}$ in G in a cutset which includes e_1 and does not include e_2. A contradiction.

Q.E.D.

Lemma 7.8: Let G be a connected graph with a dual G_d and let f be the 1 − 1 correspondence of their edges. If $u \overset{e_1}{-} x_1 \overset{e_2}{-} x_2 \overset{e_3}{-} \cdots x_{l-1} \overset{e_l}{-} v$ is a simple path or circuit in G such that $x_1, x_2, \ldots, x_{l-1}$ are all of degree two, then $f(e_1), f(e_2), \ldots, f(e_l)$ are parallel edges in G_d.

Proof: Every circuit of G which contains one e_i, $1 \leq i \leq l$, contains all the rest. Thus, in G_d, if one edge, $f(e_i)$, is in a cutset then all the rest are. By Lemma 7.7, they are parallel edges.

Q.E.D.

Lemma 7.9: If a connected graph G_1 has a dual and G_2 is homeomorphic to G_1 then G_2 has a dual.

Proof: If an edge $x \overset{e}{-} y$ of G_1 is replaced in G_2 by a path $x \overset{e_1}{-} v_1 \overset{e_2}{-} v_2 - \cdots \overset{e_l}{-} y$, then in the dual $f(e)$ is replaced by parallel edges: $f(e_1), f(e_2), \ldots, f(e_l)$. If a path of G_1 is replaced in G_2 by an edge e, then the edges of the dual which correspond to the edges of the path are all parallel (by Lemma 7.8) and can be replaced by a single edge $f(e)$. It is easy to see that every circuit which uses an edge e, when it is replaced by a path, will use all the edges of the path instead; while in the dual, every cutset which uses $f(e)$ will use all the parallel edges which replace it. Thus, the correspondence of circuits and cutsets is maintained.

Q.E.D.

Theorem 7.5: A (connected) graph has a dual if and only if it is planar.

Proof: Assume $G_1(V_1, E_1)$ is a planar graph and $\tilde{G}_1$ is a plane realization of it. Choose a point p_i in each face F_i of $\tilde{G}_1$. Let V_2 be the set of these points. Since G_1 is connected, the boundary of every face is a circuit (not necessarily simple) which we call its window. Let W_i be the window of F_i, and assume it consists of l edges. We can find l lines, all starting in p_i, but no two share any other point, such that each of the lines ends on one of the l edges of W_i, one line per edge. If a separating edge* e appears on W_i, then clearly it appears

*An edge whose removal from G_1 interrupts its connectivity.

twice. In this case there will be two lines from p_i hitting e from both directions. These two lines can be made to end in the same point on e, thus creating a closed curve which goes through p_i and crosses e. If e is not a separating edge then it appears on two windows, say W_i and W_j. In this case we can make the line from p_i to e meet e at the same point as does the line from p_j to e, to form a line connecting p_i with p_j which crosses e. None of the set of lines, thus formed crosses another, and we have one per edge of $\hat{G}_1$. Now define $\hat{G}_2(V_2, E_2)$ as follows: The set of lines connecting the p_i's is a representation of E_2. The 1-1 correspondence $f\colon E_1 \to E_2$ is defined as follows: $f(e)$ is the edge of $\hat{G}_2$ which crosses e. Clearly, $\hat{G}_2$ is a plane graph which is a realization of a graph $G_2(V_2, E_2)$. It remains to show that there is a 1-1 correspondence of the simple circuits of G_1 to the cutsets of G_2. The construction described above is demonstrated in Fig. 7.12, where $\hat{G}_1$ is shown in solid lines, and $\hat{G}_2$ is shown in dashed lines.

Let C be a simple circuit of G_1. Clearly, in $\hat{G}_1$, C describes a simple closed curve in the plane. There must be at least one vertex of $\hat{G}_2$ inside this circuit, since at least one edge of $\hat{G}_2$ crosses the circuit, and it crosses the circuit exactly once. The same argument applies to the outside too. This implies that $f(S)$, where S is the set of edges of C, forms a separating set of G_2. Let us postpone the proof of the minimality of $f(S)$ for a little while.

Now, let K be a cutset of G_2. Let T and $\bar{T}$ be the sets of vertices of the two

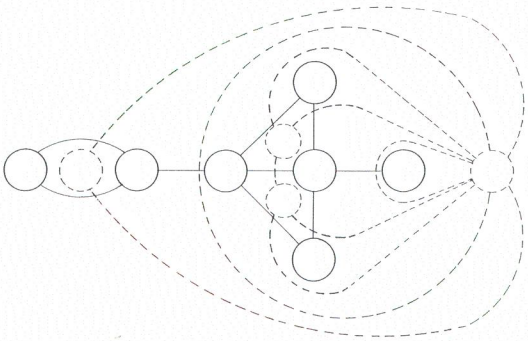


Figure 7.12

components of G_2 formed by the deletion of K. The set of faces of $\hat{G}_1$ which correspond to the vertices of T, forms a continuous region, but not the whole plane. The minimality of K implies that the boundary of this region is a circuit in G_1, whose edges correspond to K. Thus we have shown that every simple circuit of G_1 corresponds to a separating set of G_2, and every cutset of G_2 corresponds to a circuit of G_1.

Now let us handle the minimality. If S is the set of edges of a simple circuit C of G_1 and $f(S)$ is not a cutset of G_2, then there is a proper subset of $f(S)$ which is a cutset, say K. Therefore, $f^{-1}(K)$ is a circuit of G_1. However $f^{-1}(K) \subsetneq S$. A contradiction to the assumption that C is simple. The proof that if K is a cutset then $f^{-1}(K)$ is a simple circuit is similar.

This completes the proof that a connected planar graph has a dual. We turn to the proof that a nonplanar graph has no dual. Here we follow the proof of Parson [3].

First let us show that neither $K_{3,3}$ nor K_5 have a dual. In $K_{3,3}$ the shortest circuit is of length 4, and for every two edges there exists a simple circuit which uses one but does not use the other. Thus, in its dual no cutset consists of less than 4 edges and there are no parallel edges. Therefore, the degree of each vertex is at least 4 and there must be at least 5 vertices. The number of edges is, therefore, at least $10\,(5 \cdot 4/2)$, while $K_{3,3}$ has 9 edges. Thus, $K_{3,3}$ has no dual. In the case of K_5, it has 10 edges, 10 simple circuits of length 3 and no shorter circuits, and for every two edges there is a simple circuit which uses one but not the other. Thus, the dual must have 10 edges, 10 cutsets of three edges, no cutset of lesser size and therefore the degree of every vertex is at least three. Also, there are no parallel edges in the dual. If the dual has 5 vertices then it is K_5 itself (10 edges and no parallel edges), but K_5 has no cutsets of three edges. If the dual has 7 vertices or more, then it has at least 11 edges ($\lceil (7 \cdot 3/2 \rceil$). Thus, the dual must have 6 vertices. Since it has 10 custsets of 3 edges, there is one which separates S from $\bar{S}$ where neither consists of a single vertex. If $|S| = 2$ then it contains a vertex whose degree is 2. If $|S| = |\bar{S}| = 3$, then the maximum number of edges in the dual is 9. Thus, K_5 has no dual.

Now, if G is a nonplanar graph with a dual, then by Kuratowski's theorem (Theorem 7.2) it contains a subgraph G' which is homeomorphic to either $K_{3,3}$ or K_5. By Lemma 7.6, G' has a dual too. By Lemma 7.9, either $K_{3,3}$ or K_5 have a dual. A contradiction. Thus, no nonplanar graph has a dual.

Q.E.D.

Theorem 7.5 provides another necessary and sufficient condition for planarity, in addition to Kuratowski's theorem. However, neither has been shown to be useful for testing planarity.

There are many facts about duality which we have not discussed. Among them are the following:

(1) If G_d is a dual of G then G is a dual of G_d.
(2) A 3-connected planar graph has a unique dual.

The interested reader can find additional information and references in the books of Harary [4], Ore [5], and Wilson [6].

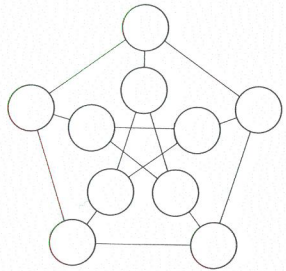
PROBLEMS

7.1 The purpose of this problem is to prove a variation of Kuratowski's theorem.

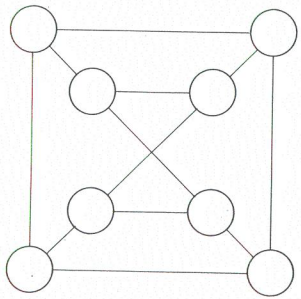
(a) Prove that if G is a connected graph and v_1, v_2, v_3 are three vertices then there exists a vertex v and three (vertex) disjoint paths $P_1(v, v_1)$, $P_2(v, v_2)$ and $P_3(v, v_3)$, one of which may be empty.
(b) Prove that if G is a connected graph and S is a set of four vertices then either there is a vertex v with four disjoint paths to the members of S or there are two vertices u and v such that two members of S are connected to u by paths, two to v, and u is connected to v; all five paths are vertex disjoint.
(c) Show that if a graph is contractible to $K_{3,3}$ then it has a subgraph homeomorphic to $K_{3,3}$.
(d) Show that if a graph is contractible to K_5 then either it has a subgraph which is homeomorphic to K_5 or it has a subgraph which is contractible to $K_{3,3}$.
(e) Prove the theorem: A graph is nonplanar if and only if it contains a subgraph which is contractible to $K_{3,3}$ or K_5.

7.2 Show a graph which is nonplanar but is not contractible to either $K_{3,3}$ or K_5. Does it contradict the result of Problem 7.1(e)?

7.3 Use Kuratowski's theorem to prove that the Petersen graph, shown below is nonplanar.



7.4 Is the graph shown below planar? Justify your answer.



7.5 A plane graph is called *triangular* if each of its windows is a triangle. Let N_i be the number of vertices whose degree is i.

(a) Prove that every plane graph with 3 or more vertices which has no self-loops and no parallel edges can be made triangular, by adding

new edges without creating any self-loops or parallel edges. (This process is called *triangulation.*)

(b) Let G be a triangular plane graph as above, with $|V| > 3$. Prove that

$$N_1 = N_2 = 0.$$

(c) Let G be a triangular plane graph as above. Prove that

$$12 = 3 \cdot N_3 + 2 \cdot N_4 + N_5 - N_7 - 2 \cdot N_8 - 3 \cdot N_9 - 4 \cdot N_{10} \cdots$$

(d) Prove that in a graph, as above, if there are no vertices of degree 3 or 4 then there are at least 12 vertices of degree 5.

(e) Prove that if G is a planar graph with no self-loops or parallel edges, and $|V| > 3$, then it has at least 4 vertices with degrees less than 6 and if $N_3 = N_4 = 0$ then $N_5 \geq 12$.

(f) Prove that if the vertex connectivity, $c(G)$, of a graph $G(V, E)$ is at least 5 then $|V| \geq 12$.

(g) Prove that if G is planar then $c(G) < 6$.

7.6 Prove that if $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ are homeomorphic then

$$|E_1| - |V_1| = |E_2| - |V_2|.$$

7.7 Show a triangular plane graph, without parallel edges, which is not hamiltonian.

Let G be a plane graph. The plane graph G^*, which is constructed by the procedure in the first part of the proof of Theorem 7.5, is called its *geometric dual.*

7.8 Prove that if G^* is the geometric dual of G then G is the geometric dual of G^*.

REFERENCES

[1] König, D., *Theorie der endlichen und unendlichen Graphen*. Leipzig, 1936. Reprinted Chelsea, 1950.

[2] Kuratowski, K., "Sur le Problème des Courbes Gauches en Topologie", Fund. Math., Vol. 15, 1930, pp. 217-283.

[3] Parson, T. D., "On Planar Graphs", Am. Math. Monthly, Vol. 78, No. 2, 1971, pp. 176-178.

[4] Harary, F., *Graph Theory*, Addison Wesley, 1969.

[5] Ore, O., *The Four Color Problem*, Academic Press, 1967.

[6] Wilson, R. J., *Intr. to Graph Theory*, Longman, 1972.

Chapter 8

TESTING GRAPH PLANARITY

8.1 Introduction

There are two known planarity testing algorithms which have been shown to be realizable in a way which achieves linear time ($O(|V|)$). The idea in both is to follow the decisions to be made during the planar construction of the graph, piece by piece, as to the relative location of the various pieces. The construction is not carried out explicitly because there are difficulties, such as crowding of elements into a relatively small portion of the area allocated, that, as yet, we do not know to avoid. Also, an explicit drawing of the graph is not necessary, as we shall see, to decide whether such a drawing is possible. We shall imagine that such a realization is being carried out, but will only decide where the various pieces are laid, relative to each other, and not of their exact shape. Such decisions may change later in order to make place for later additions of pieces. In both cases it was shown that the algorithm terminates within $O(|V|)$ steps, and if it fails to find a "realization" then none exists.

The first algorithm starts by finding a simple circuit and adding to it one simple path at a time. Each such new path connects two old vertices via new edges and vertices. (Whole pieces are sometimes flipped over, around some line). Thus, we call it the *path addition* algorithm. The basic ideas were suggested by various authors, such as Auslander and Parter [1] and Goldstein [2], but the algorithm in its present form, both from the graph theoretic point of view, and complexity point of view, is the contribution of Hopcroft and Tarjan [3]. They were first to show that planarity testing can be done in linear time.

The second algorithm adds in each step one vertex. Previously drawn edges incident to this vertex are connected to it, and new edges incident to it are drawn and their other endpoints are left unconnected. (Here too, sometimes whole pieces have to be flipped around or permuted). The algorithm is due to Lempel, Even and Cederbaum [4]. It consists of two parts. The first part was shown to be linarily realizable by Even and Tarjan [5];

the second part was shown to be linearily realizable by Leuker and Booth [6]. We call this algorithm the *vertex addition* algorithm.

Each of these algorithms can be divided into its graph theoretic part and its data structures and their manipulation. The algorithms are fairly complex and a complete description and proof would require a much more elaborate exposition. Thus, since this is a book on graphs and not on programming, I have chosen to describe in full the graph theoretic aspects of both algorithms, and only briefly describe the details of the data manipulation techniques. An attempt is made to convince the reader that the algorithms work, but in order to see the details which make it linear he will have to refer to the papers mentioned above.

Throughout this chapter, for reasons explained in Chapter 7, we shall assume that $G(V, E)$ is a finite undirected graph with no parallel edges and no self loops. Also, we shall assume that G is nonseparable. The first thing that we can do is check whether $|E| \leq 3 \cdot |V| - 6$. By Carollary 7.1, if this condition does not hold then G is nonplanar. Thus, we can restrict our algorithms to the cases where $|E| = O(|V|)$.

8.2 The Path Addition Algorithm of Hopcroft and Tarjan

The algorithm starts with a DFS of G. We assume that G is nonseparable. Thus, we drop from the DFS the steps for testing nonseparability. However we still shall need the lowpoint function, to be denoted now $L1(v)$. In addition we shall need the *second lowpoint* function, $L2(v)$, to be defined as follows. Let $S(v)$ be the set of values $k(u)$ of vertices u reachable from descendants of v by a single back edge. Clearly, $L1(v) = \text{Min}\,\{\{k(v)\} \cup S(v)\}$.

Define

$$L2(v) = \text{Min}\,\{\{k(v)\} \cup [S(v) - \{L1(v)\}]\}. \tag{8.1}$$

Let us now rewrite the DFS in order to compute these functions:

(1) Mark all the edges "unused". For every $v \in V$ let $k(v) \leftarrow 0$. Let $i \leftarrow 0$ and $v \leftarrow s$ (the vertex s is where we choose to start the DFS).
(2) $i \leftarrow i + 1$, $k(v) \leftarrow i$, $L1(v) \leftarrow i$, $L2(v) \leftarrow i$.
(3) If v has no unused incident edges go to Step (5).
(4) Choose an unused incident edge $v \overset{e}{-} u$. Mark e "used". If $k(u) \neq 0$ then do the following:

If $k(u) < L1(v)$ then $L2(v) \leftarrow L1(v)$,
$L1(v) \leftarrow k(u)$.
If $k(u) > L1(v)$ then $L2(v) \leftarrow \text{Min}\,\{L2(v), k(u)\}$.
Go to Step (3).
Otherwise ($k(u) = 0$), let $f(u) \leftarrow v$, $v \leftarrow u$ and go to Step (2).
(5) If $k(v) = 1$, Halt.
(6) ($k(v) > 1$; we backtrack).
If $L1(v) < L1(f(v))$ then $L2(f(v)) \leftarrow \text{Min}\,\{L2(v), L1(f(v))\}$,
$L1(f(v)) \leftarrow L1(v)$.
If $L1(v) = L1(f(v))$ then $L2(f(v)) \leftarrow \text{Min}\,\{L2(v), L2(f(v))\}$.
Otherwise ($L1(v) > L1(f(v))$), $L2(f(v)) \leftarrow \text{Min}\,\{L1(v), L2(f(v))\}$.
$v \leftarrow f(v)$.
Go to Step (3).

From now on, we refer to the vertices by their $k(v)$ number; i.e. we change the name of v to $k(v)$.

Let $A(v)$ be the *adjacency list* of v; i.e. the list of edges incident from v. We remind the reader that after the DFS each of the edges is directed; the tree edges are directed from low to high and the back edges are directed from high to low. Thus, each edge appears once in the adjacency lists. Now, we want to reorder the adjacency lists, but first, we must define an order on the edges. Let the value $\phi(e)$ of an edge $u \overset{e}{-} v$ be defined as follows:

$$\phi(e) = \begin{cases} 2 \cdot v & \text{if } u \overset{e}{-} v \text{ is a back edge.} \\ 2 \cdot L1(v) & \text{if } u \overset{e}{-} v \text{ is a tree edge and } L2(v) \geq u. \\ 2 \cdot L1(v) + 1 & \text{if } u \overset{e}{-} v \text{ is a tree edge and } L2(v) < u. \end{cases}$$

Next we order the edges in each adjacency list to be in nondecreasing order with respect to ϕ. [This can be done in $O(|V|)$ time as follows. First compute for each edge its ϕ value. Prepare $2 \cdot |V| + 1$ buckets, numbered $1, 2, \ldots, 2 \cdot |V| + 1$. Put each e into the $\phi(e)$ bucket. Now, empty the buckets in order, first bucket number 1, then 2 etc., putting the edges taken out into the proper new adjacency lists, in the order that they are taken out of the buckets.]

The new adjacency lists are now used, in a second run of a DFS algorithm, to generate the paths, one by one. In this second DFS, vertices are

not renumbered and there is no need to recompute $f(v)$, $L1(v)$ or $L2(v)$. The tree remains the same, although the vertices may not be visited in the same order. The paths finding algorithm is as follows:

(1) Mark all edges "unused" and let $v \leftarrow 1$.
(2) Start the circuit C; its first vertex is 1.
(3) Let the first edge on $A(v)$ be $v \overset{e}{\rightarrow} u$.
If $u \neq 1$ then add u to C, $v \leftarrow u$ and repeat Step (3). Otherwise, ($u = 1$), C is closed. Output C.
(4) If $v = 1$, halt.
(5) If $A(v)$ is used up then $v \leftarrow f(v)$ and go to Step (4).
(6) Start a path P with v as its first vertex.
(7) Let $v \overset{e}{\rightarrow} u$ be the first unused edge on $A(v)$.
If e is a back edge ($u < v$) terminate P with u, output P and go to (5).
(8) (e is a tree edge). Put u on P, $v \leftarrow u$ and go to Step (7).

Lemma 8.1: The paths finding algorithm finds first a circuit C which consists of a path from 1 (the root) to some vertex v, via tree edges, and a back edge from v to 1.

Proof: Let $1 \rightarrow u$ be the first edge of the tree to be traced (in the first application of Step (3)). We assume that G is nonseparable and $|V| > 2$. Thus, by Lemma 3.7, this edge is the only tree edge out of 1, and $u = 2$. Also, 2 has some descendants, other than itself. Clearly, $2 \rightarrow 3$ is a tree edge. By Lemma 3.5, $L1(3) < 2$, i.e. $L1(3) = 1$. Thus $L1(2) = 1$. The reordering of the adjacency lists assures that the first path to be chosen out of 1 will lead back to 1 as claimed.

Q.E.D.

Lemma 8.2: Each generated path P is simple and it contains exactly two vertices in common with previously generated paths; they are the first vertex, f, and the last l.

Proof: The edge scanning during the paths finding algorithm is in a DFS manner, in accord with the structure of the tree (but not necessarily in the same scanning order of vertices). Thus, a path starts from some (old) vertex f, goes along tree edges, via intermediate vertices which are all new, and ends with a back edge which leads to l. Since back edges always lead to ancestors, l is old. Also, by the reordering of the adjacency lists and the assumption that G is nonseparable l must be lower than f. Thus, the path is simple.

Q.E.D.

Let S_v denote the set of descendants of v, including v itself.

Lemma 8.3: Let f and l be the first and last vertices of a generated path P and $f \rightarrow v$ be its first edge.
(i) if $v \neq l$ then $L1(v) = l$.
(ii) l is the lowest vertex reachable from S_f via a back edge which has not been used in any path yet.

Proof: Let us partition $S_f - \{f\}$ into two parts, α and β, as follows. A descendant u belongs to α if and only if when the construction of P begins, we have already backtracked from u. Clearly, $f \in \beta$ and all the back edges out of α have been scanned already. Let u be the lowest vertex reachable via an unused back edge from β. Clearly, the first remaining (unused) edge of $A(f)$ is the beginning of a directed path to u, which is either an unused back edge from f to u or a path of tree edges, via vertices of β, followed by a single back edge to u. Thus, $u = l$, and the Lemma follows.

Q.E.D.

Lemma 8.4: Let P_1 and P_2 be two generated paths whose first and last vertices are f_1, l_1 and f_2, l_2, respectively. If P_1 is generated before P_2 and f_2 is a descendant of f_1 then $l_1 \leq l_2$.

Proof: The Lemma follows immediately from Lemma 8.3.

Q.E.D.

So far, we have not made any use of $L2(v)$. However, the following lemma relies on it.

Lemma 8.5: Let $P_1: f \overset{e_1}{\rightarrow} v_1 \rightarrow \cdots \rightarrow l$ and $P_2: f \overset{e_2}{\rightarrow} v_2 \rightarrow \cdots \rightarrow l$ be two generated paths, where P_1 is generated before P_2. If $v_1 \neq l$ and $L2(v_1) < f$ then $v_2 \neq l$ and $L2(v_2) < f$.

Proof: By the definition of $\phi(e)$, $\phi(e_1) = 2 \cdot l + 1$. If $v_2 = l$ or $L2(v_2) \geq f$ then $\phi(e_2) = 2 \cdot l$, and e_2 should appear in $A(f)$ before e_1. A contradiction.

Q.E.D.

Let C be $1 \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n \rightarrow 1$. Clearly $1 < v_1 < v_2 \cdots < v_n$. Consider now the bridges of G with respect to C.

Lemma 8.6: Let B be a non-singular bridge of G with respect to C, whose highest attachment is v_i. There exist an tree edge $v_i \overset{e}{\rightarrow} u$ which belongs to B and all other edges of B with endpoints on C are back edges.

Proof: The Lemma follows from the fact that the paths finding algorithm is a DFS. First C is found. We then backtrack from a vertex v_j only if all its descendants have been scanned. No internal part of B can be scanned before we backtrack into v_i. There must be a tree edge $v_i \rightarrow u$, where u belongs to B, for the following reasons. If all the edges of B, incident to v_i are back edges, they all must come from descendents or go to ancestors of v_i (see Lemma 3.4). An edge from v_i to one of its ancestor (which must be on C) is a singular bridge and is not part of B. An edge from a descendant w of v_i to v_i implies that w cannot be in B, for it has been scanned already, and we have observed that no internal part of B can be scanned before we backtrack into v_i. If any other edge $v_k - x$ of B is also a tree edge then, by the definition of a bridge, there is a path connecting u and x which is vertex disjoint from C. Along this path there is at least one edge which contradicts Lemma 3.4.

Q.E.D.

Corollary 8.1: Once a bridge B is entered, it is completely traced before it is left.

Proof: By Lemma 8.6, there is only one edge through which B is entered. Since eventually the whole graph is scanned, and no edge is scanned twice in the same direction, the corollary follows.

Q.E.D.

Assuming C and the bridges explored from vertices higher than v_i have already been explored and drawn in the plane. The following lemma provides a test for whether the next generated path could be drawn inside; the answer is negative even if the path itself can be placed inside, but it is already clear that the whole bridge to which it belongs cannot be placed there.

Lemma 8.7: Let $v_i \rightarrow u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_l(=v_j)$ be the first path, P, of the bridge B to be generated by the paths finding algorithm. P can be drawn inside (outside) C if there is no back edge $w \rightarrow v_k$ drawn inside (outside) for which $j < k < i$. If there is such an edge, then B cannot be drawn inside (outside).

Proof: The sufficiency is immediate. If there is no back edge, drawn inside C, which enters the path $v_j \rightarrow v_{j+1} \rightarrow \cdots v_i$ in one of its internal vertices, then there cannot be any inside edge incident to these vertices, since bridges

to be explored from vertices lower than v_i have not been scanned yet. Thus, P can be drawn inside if it is placed sufficiently close to C.

Now, assume there is a back edge $w \rightarrow v_k$, drawn inside, for which $j < k < i$. Let P' be the directed path from v_p to v_k whose last edge is the back edge $w \rightarrow v_k$. Clearly v_p is on C and $p \geq i$; P' is not necessarily generated in one piece by the path finding algorithm, if it is not the first path to be generated in the bridge B' to which it belongs.

Case 1: $p > i$. The bridges B and B' interlace by part (i) of the definition of interlacement. Thus, B cannot be drawn inside.

Case 2: $p = i$. Let P'' be the first path of B' to be generated, P'': $v_i \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow v_q$. By Lemma 8.4, $q \leq j$, since B' is explored before B.

Case 2.1: $q < j$. Since v_i and v_j are attachments of B, v_k and v_q are attachments of B' and $q < j < k < i$, the two bridges interlace. Thus, B cannot be drawn inside.

Case 2.2: $q = j$. P'' cannot consist of a single edge, for in this case it is a singular bridge and v_k is not one of its attachments. Also, $L2(x_1) \leq v_k$. Thus, $L2(x_1) < v_i$. By Lemma 8.5, $u_1 \neq v_j$ and $L2(u_1) < v_i$. This implies that B and B' interlace by either part (i) or part (ii) of the definition of interlacement, and B cannot be drawn inside.

Q.E.D.

The algorithm assumes that the first path of the new bridge B is drawn inside C. Now, we use the results of Corollary 7.1, Theorem 7.1 and the discussion which follows it, to decide whether the part of the graph explored so far is planar, assuming that $C + B$ is planar. By Lemma 8.7, we find which previous bridges interlace with B. The part of the graph explored so far is planar if and only if the set of its bridges can be partitioned into two sets such that no two bridges in the same set interlace. If the answer is negative, the algorithm halts declaring the graph nonplanar. If the answer is positive, we still have to check whether $C + B$ is planar.

Let the first path of B be P: $v_i \rightarrow u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow v_j$. We now have a circuit C' consisting of $C[v_j, v_i]$ and P. The rest of C is an outside path P', with respect to C', and it consists of $C[v_i, 1]$ and $C[1, v_j]$. The graph $B + C[v_j, v_i]$ may have bridges with respect to C', but none of them has all its attachments on $C[v_j, v_i]$, for such a bridge is also a bridge of G with respect to C, and is not a part of B.

Thus, no bridge of C', with attachments on $C(v_j, v_i)$ may be drawn outside C', since it interlaces with P'. We conclude that $C + B$ is planar if and only if $B + C[v_j, v_i]$ is planar and its bridges can be partitioned into an inside set and an outside set so that the outside set contains no bridge with attachments in $C(v_j, v_i)$. The planarity of $B + C[v_j, v_i]$ is tested by applying the algorithm recursively, using the established vertex numbering $L1$, $L2$, f functions and ordering of the adjacency lists. If $B + C[v_j, v_i]$ is found to be nonplanar, clearly G is nonplanar. If it is found to be planar, we check whether all its bridges with attachments in $C(v_j, v_i)$ can be placed inside. If so, $C + B$ is planar; if not, G is nonplanar.

Hopcroft and Tarjan devised a simple tool for deciding whether the set of bridges can be partitioned properly. It consists of three stacks Π_1, Π_2 and Π_3. Π_1 contains in nondecreasing order (the greatest entry on top) the vertices of $C(1, v_i)$ (where v_i is the last vertex of C into which we have backtraced), which are attachments of bridges placed inside C in the present partition of the bridges. A vertex may appear on Π_1 several times, if there are several back edges into it from inside bridges. Π_2 is similarly defined for the outside bridges. Both Π_1 and Π_2 are doubly linked lists, in order to enable switching over of complete sections from one of them to the other, investing, per section, time bounded by some constant. Π_3 consists of pairs of pointers to entries in Π_1 and Π_2. Its task will be described shortly.

Let S be a maximal set of bridges, explored up to now, such that a decision for any one of these bridges as to the side it is in, implies a decision for all the rest. (S corresponds to a connected component of the graph of bridges, as in the discussion following Theorem 7.1.) Let the set of entries in Π_1 and Π_2, which correspond to back edges from the bridges of S, be called a *block*.

Lemma 8.8: Let K be a block, whose highest element is v_h and lowest element is v_l. If v_p is an entry in Π_1 or Π_2 then v_p belongs to K if $v_l < v_p < v_h$.

Proof: We prove the lemma by induction on the order in which the bridges are explored. At first, both Π_1 and Π_2 are empty and the lemma is vacuously true. The lemma trivially holds after one bridge is explored.

Assume the lemma is true up to the exploration of the first path P of the new bridge B, where P: $v_i \rightarrow \ldots \rightarrow v_j$. If there is no vertex v_k on Π_1 or Π_2 such that $v_j < v_k < v_i$ then clearly the attachments of B (in $C(1, v_i)$) form a new block (assuming $C + B$ is planar) and the lemma holds. However, if there are vertices of Π_1 or Π_2 in between v_j and v_i then, by Lemma 8.7, the bridges, they are attachments of, all interlace with B. Thus, the old blocks which these attachments belong to, must now be merged into one new block with the attachments of B (in $C(1, v_i)$). Now, let v_l be the lowest vertex of the new block and v_h be the highest. Clearly, v_l was the lowest vertex of some old block whose highest vertex was v_h', and $v_h' > v_j$. Thus, by the inductive hypothesis, no attachment of another block could be in between v_l and v_h', and therefore cannot be in this region after the merger. Also, all the attachments in between v_j and v_h are in the new block since they are attachments of bridges which interlace with B. Thus, all the entries of Π_1 or Π_2 which are in between v_l and v_h belong to the new block. Q.E.D.

Corollary 8.2: The entries of one block appear consecutively on $\Pi_1(\Pi_2)$.

Thus, when we consider the first path P: $v_i \rightarrow \cdots \rightarrow v_j$ of the new bridge B, in order to decide whether it can be drawn inside, we check the top entries t_1 and t_2 of Π_1 and Π_2 respectively.

If $v_j \geq t_1$ and $v_j \geq t_2$ then no merger is necessary; the attachments of B (in $C(1, v_i)$) are entered as a block (if $C + B$ is found to be planar) on top of Π_1.

If $v_j < t_1$ and $v_j \geq t_2$ then we first join all the blocks for which their highest entry is higher than v_j. To this end there is no need to check Π_2, since $v_j \geq t_2$, but still several blocks may merge. Next, switch the sections of the new block, i.e. the section on Π_1 exchanges places with the section on Π_2. Finally, place the attachments of B in nondecreasing order, on top of Π_1; these entries join the new block.

If $v_i \geq t_1$ and $v_j < t_2$ then again we join all the blocks whose highest element is greater than v_j; only the sections on Π_2 need to be checked. The attachments of B join the same block and are placed on top of Π_1.

If $v_j < t_1$ and $v_j < t_2$ then all blocks whose highest element is greater than v_j are joined into one new block. As we join the blocks we examine them one by one. If the highest entry in the section on Π_1 is higher than v_j then we switch the sections. If it is still higher, then we halt and declare the graph nonplanar. If all these switches succeed, then the merger is completed by adding the attachments of B on top of Π_1.

In order to handle the sections switching, the examination of their tops and mergers efficiently, we use a third stack, Π_3. It consists of pairs of pointers, one pair (x, y) per block; x points to the lowest entry of the section in Π_1 and y to the lowest entry of the section in Π_2. (If the section is empty then the pointer's value is 0). Two adjacent blocks are joined by simply discarding the pair of the top one. When several blocks have to be joined together upon the drawing of the first path of a new bridge, only the pair

of the lowest of these blocks need remain, except when one of its entries is 0. In this case the lowest nonzero entry on the same side, of the pairs above it, if any such entry exists, takes its place.

When we enter a recursive step, a special "end of stack" marker E is placed on top of Π_2, and the three stacks are used as in the main algorithm. If the recursive step ends successfully, we first attempt to switch sections for each of the blocks with a nonempty section on Π_2, above the top most E. If we fail to expose E, then $C + B$ is nonplanar and we halt. Otherwise, all the blocks created during the recursion are joined to the one which includes v_j (the end vertex of the first path of B). The exposed E, on top of Π_2, is removed and we continue with the previous level of recursion. When we backtrack into a vertex v_i, all occurrences of v_i are removed from the top of Π_1 and Π_2, together with pairs of pointers of Π_3 which point to removed entries on both Π_1 and Π_2. (Technically, instead of pointing to an occurrence of v_i, we point to 0, and pairs (0, 0) are removed).

Theorem 8.1: The complexity of the path addition algorithm is $O(|V|)$.

Proof: As in the closing remarks of Section 8.1, we can assume $|E| = O(|V|)$. The DFS and the reordering of the adjacency lists have been shown to be $O(|V|)$. Each edge in the paths finding algorithm is used again, once in each direction. The total number of entries in the stacks Π_1 and Π_2 is bounded by the number of back edges ($|E| - |V| + 1$), and is therefore $O(|V|)$. After each section switching the number of blocks is reduced by one; thus the total work invested in section switchings is $O(|V|)$.

Q.E.D.

8.3 Computing an st-Numbering

In this section we shall define on st-numbering and describe a linear time algorithm to compute it. This numbering is necessary for the vertex addition algorithm, for testing planarity, of Lempel, Even and Cederbaum.

Given any edge $s - t$ of a nonseparable graph $G(V, E)$, a 1-1 function $g: V \to \{1, 2, \ldots, |V|\}$ is called an *st-numbering* if the following conditions are satisfied:

(1) $g(s) = 1$,
(2) $g(t) = |V|$ $(=n)$,
(3) for every $v \in V - \{s, t\}$ there are adjacent vertices u and w such that $g(u) < g(v) < g(w)$.

Lempel, Even and Cederbaum showed that for every nonseparable graph and every edge $s - t$, there exists an *st*-numbering. The algorithm described here, following the work of Even and Tarjan [5], achieves this goal in linear time.

The algorithm starts with a DFS whose first vertex is t and its first edge is $t - s$. (i.e., $k(t) = 1$ and $k(s) = 2$). This DFS computes for each vertex v, its DFS number, $k(v)$, its father, $f(v)$, its lowpoint $L(v)$ and distinguishes tree edges from back edges. This information is used in the paths finding algorithm to be described next, which is different from the one used in the path addition algorithm.

Initially, s, t and the edge connecting them are marked "old" and all the other edges and vertices are marked "new". The path finding algorithm starts from a given vertex v and finds a path from it. This path may be directed from v or into v.

(1) If there is a "new" back edge $v \overset{e}{\dashrightarrow} w$ (in this case $k(w) < k(v)$) then do the following:

    Mark e "old".
    The path is $v \overset{e}{\dashrightarrow} w$.
    Halt.

(2) If there is a "new" tree edge $v \overset{e}{\longrightarrow} w$ (in this case $k(w) > k(v)$) then do the following:

    Trace a path whose first edge is e and from there it follows a path which defined $L(w)$, i.e., it goes up the tree and ends with a back edge into a vertex u such that $k(u) = L(w)$. All vertices and edges on the path are marked "old". Halt.

(3) If there is a "new" back edge $w \overset{e}{\dashrightarrow} v$ (in this case $k(w) > k(v)$) then do the following:

    Start the path with e (going backwards on it) and continue backwards via tree edges until you encounter an "old" vertex. All vertices and edges on the path are marked "old". Halt.

(4) (All edges incident to v are "old"). The path produced is empty. Halt.

Lemma 8.9: If the path finding algorithm is always applied from an "old" vertex $v \neq t$ then all the ancestors of an "old" vertex are "old" too.

Proof: By induction on the number of applications of the path finding algorithm. Clearly, before the first application, the only ancestor of s is t

and it is old. Assuming the statement is true up to the present application, it is easy to see that if any of the four steps is applicable the statement continues to hold after its application.

Q.E.D.

Corollary 8.3: If G is nonseparable and under the condition of Lemma 8.9, each application of the path finding algorithm from an "old" vertex v produces a path, through "new" vertices and edges, to another "old" vertex, or in case all edges incident to v are "old", it returns the empty path.

Proof: The only case which requires a discussion is when case (2) of the path finding algorithm is applied. Since G is nonseparable, by Lemma 3.5, $L(w) < k(v)$. Thus, the path ends "below" v, in one of its ancestor. By Lemma 8.9, this ancestor is "old".

Q.E.D.

We are now ready to present the algorithm which produces an st-numbering. It uses a stack S which initially contains only t and s, s on top of t.

(1) $i \leftarrow 1$.
(2) Let v be the top vertex on S. Remove v from S. If $v = t$ then $g(t) \leftarrow i$ and halt.
(3) ($v \neq t$) Apply the path finding algorithm to v. If the path is empty then $g(v) \leftarrow i$, $i \leftarrow i + 1$ and go to Step (2).
(4) (The path is not empty) Let the path be $v - u_1 - u_2 - \cdots - u_l - w$. Put $u_l, u_{l-1}, \ldots, u_2, u_1, v$ on S in this order (v comes out on top) and go to Step (2).

Theorem 8.2: The algorithm above computes for every nonseparable graph $G(V, E)$ an st-numbering.

Proof: First we make a few observations about the algorithm:

(i) No vertex ever appears in two or more places on S at the same time.
(ii) Once a vertex v is placed on S, nothing under v receives a number until v does.
(iii) A vertex is permanently removed from S only after all its incident edges become "old".

Next, we want to show that each vertex v is placed on S before t is removed. Since t and s are placed on S initially, the statement needs to be proved for $v \neq s, t$ only. Since G is nonseparable, there exists a simple path from s to v which does not pass through t (see Theorem 6.7 part (6)). Let this path be $s = u_1 - u_2 - \cdots - u_l = v$. Let m be the first index such that u_m is not placed on S. Since u_{m-1} is placed on S, t can be removed only after u_{m-1} (fact (ii)), and u_{m-1} is removed only after all its incident edges are "old" (fact (iii)). Thus, u_m must be placed on S before t is removed.

It remains to be shown that the algorithm computes an st-numbering.

Since each vertex is placed on S, and eventually it is removed, each vertex v gets a number $g(v)$. Clearly, $g(s) = 1$, for it is the first to be removed. After each assignment i is incremented. Thus, $g(t) = |V|$. Every other vertex v is placed on S, for the first time, as an intermediate vertex on a path. Thus, there is an adjacent vertex stored below it, and an adjacent vertex stored above it. The one above it (by fact (ii)) gets a lower number and the one below it, a higher number.

Q.E.D.

It is easy to see that the whole algorithm is of time complexity $O(|E|)$: First, the DFS is $O(|E|)$. The total time spent on path finding is also $O(|E|)$ since no edge is used more than once. The total number of operations in the main algorithm is bounded also by $O(|E|)$ because the number of stack insertions is exactly $|E| + 1$.

8.4 The Vertex Addition Algorithm of Lempel, Even and Cederbaum

In this section we assume that $G(V, E)$ is a nonseparable graph whose vertices are st-numbered. From now on, we shall refer to the vertices by their st-number. Thus, $V = \{1, 2, \ldots, n\}$. Also, the edges are now directed from low to high.

A (graphical) *source* of a digraph is a vertex v such that $d_{\text{in}}(v) = 0$; a (graphical) *sink* is a vertex v such that $d_{\text{out}}(V) = 0$.* Clearly vertex 1 is a source of G and vertex n is a sink. Furthermore, due to the st-numbering, no other vertex is either a source or a sink.

Let $V_k = \{1, 2, \ldots, k\}$. $G_k(V_k, E_k)$ is the digraph induced by V_k, i.e., E_k consists of all the edges of G whose endpoints are both in V_k.

If G is planar, let $\hat{G}$ be a plane realization of G. It contains a plane realization of G_k. The following simple lemma reveals the reason for the st-numbering.

*Do not confuse with the source and sink of a network. The source of a network is not necessarily a (graphical) source, etc.

Lemma 8.10: If $\hat{G}_k$ is a plane realization of G_k contained in a plane digraph $\hat{G}$ then all the edges and vertices of $\hat{G} - \hat{G}_k$ are drawn in one face of $\hat{G}_k$.

Proof: Assume a face F of $\hat{G}_k$ contains vertices of $V - V_k$. Since all the vertices on F's window are lower than the vertices in F, the highest vertex in F must be a sink. Since G has only one sink, only one such face is possible.

Q.E.D.

Let B_k be the following digraph. G_k is a subgraph of B_k. In addition B_k contains all the edges of G which emanate from vertices of V_k and enter in G, vertices of $V - V_k$. These edges are called *virtual edges*, and the leaves they enter in B_k are called *virtual vertices*. These vertices are labeled as their counterparts in G, but they are kept separate; i.e., there may be several virtual vertices with the same label, each with exactly one entering edge. For example, consider the digraph shown in Fig. 8.1(a). B_3 of this digraph is shown in Fig. 8.1(b).

By Lemma 8.10, we can assume that if G is planar then there exist a plane realization of B_k in which all the virtual edges are drawn in the outside face. Furthermore, since $1 \prec n$ is always an edge in G, if $k < n$ then vertex 1 is on the outside window and one of the virtual edges is e. In this case, we can draw B_k in the following form: Vertex 1 is drawn at the bottom level. All the virtual vertices appear on one horizontal line. The remaining vertices of G_k are drawn in such a way that vertices with higher names are

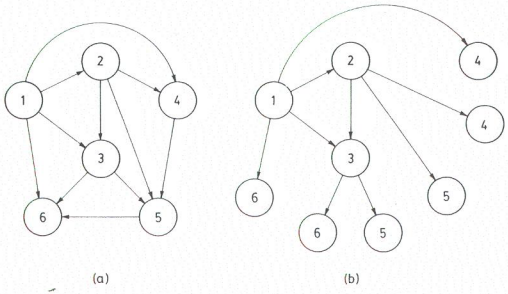


Figure 8.1

drawn higher. Such a realization is called a *bush form*. A bush form of B_3, of our example, is shown in Fig. 8.2.

In fact, Lemma 8.10 implies that if G is planar then there exists a bush form of B_k such that all the virtual vertices with labeled $k + 1$ appear next to each other on the horizontal line.

The algorithm proceeds by successively "drawing" $B_1, B_2, \ldots, B_{n-1}$ and G. If in the realization of B_k all the virtual vertices labeled $k + 1$ are next to each other, then it is easy to draw B_{k+1}: One joins all the virtual vertices labeled $k + 1$ into one vertex and "pulls" it down from the horizontal line. Now all the edges of G which emanate from $k + 1$ are added, and their other endpoints are labeled properly and placed in an arbitrary order on the horizontal line, in the space evacuated by the former virtual vertices labeled $k + 1$.

However, a difficulty arises. Indeed, the discussion up to now guarantees that if G is planar then there exists a sequence of bush forms, such that each one is "grown" from the previous one. But since we do not have a plane realization of G, we may put the virtual vertices, out of $k + 1$, in a "wrong" order. It is necessary to show that this does not matter; namely, by simple transformations it will be possible later to correct the "mistake".

Lemma 8.11: Assume v is a separation vertex of B_k. If $v > 1$ then exactly one component of B_k, with respect to v, contains vertices lower than v.

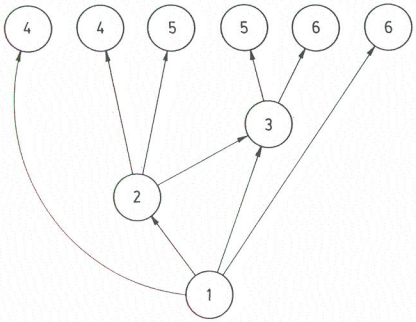


Figure 8.2

Note that here we ignore the direction of the edges, and the lemma is actually concerned with the undirected underlying graph of B_k.

Proof: The *st*-numbering implies that for every vertex u there exists a path from 1 to u such that all the vertices on the path are less than u. Thus, if $u < v$ then there is a path from 1 to u which does not pass through v. Therefore, 1 and u are in the same component.

Q.E.D.

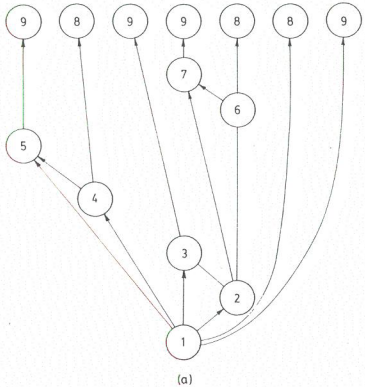
Lemma 8.11 implies that a separation vertex v of B_k is the lowest vertex in each of the components, except the one which contains 1 (in case $v > 1$). Each of these components is a sub-bush; i.e. it has the same structure as a bush form, except that its lowest vertex is v rather than 1. These sub-bushes can be permuted around v in any of the $p!$ permutations, in case the number of these sub-bushes is p. In addition, each of the sub-bushes can be flipped over. These transformations maintain the bush form. It is our purpose to show that if $\hat{B}_k^1$ and $\hat{B}_k^2$ are bush forms of a planar B_k then through a sequence of permutations and flippings, one can change $\hat{B}_k^1$ into a $\hat{B}_k^3$ such that the virtual vertices of B_k appear in $\hat{B}_k^2$ and $\hat{B}_k^3$ in the same order.

For efficiency reasons, to be become clear to those readers who will study the implementation through PQ-trees, we assume that when a sub-bush is flipped, smaller sub-bushes of other separation vertices of the component, are not flipped by this action. For example, consider the bush form shown in Fig. 8.3(a). The bush form of Fig. 8.3(b) is achieved by permuting about 1, Fig. 8.3(c) by flipping about 1 and Fig. 8.3(d) by flipping about 2.

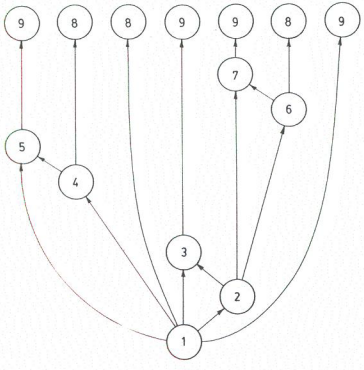
Lemma 8.12: Let H be a maximal nonseparable component of B_k and $y_1, y_2, \ldots, y_m$ be the vertices of H which are also endpoints of edges of $B_k - H$. In every bush form $\hat{B}_k$ all the y's are on the outside window of H and in the same order, except that the orientation may be reversed.

Proof: Since $\hat{B}_k$ is a bush form, all the y's are on the outside face of H. Assume there are two bush forms $\hat{B}_k^1$ and $\hat{B}_k^2$ in which the realizations of H are $\hat{H}^1$ and $\hat{H}^2$, respectively. If the y's do not appear in the same order on the outside windows of $\hat{H}^1$ and $\hat{H}^2$ then there are two y's, y_i and y_j which are next to each other in $\hat{H}^1$ but not in $\hat{H}^2$ (see Fig. 8.4). Therefore, in $\hat{H}^2$, there are two other y's, y_k and y_l which interpose between y_i and y_j on the two paths between them on the outside window of $\hat{H}^2$. However, from $\hat{H}^1$ we see that there are two paths, $P_1[y_i, y_j]$ and $P_2[y_k, y_l]$ which are completely disjoint. These two paths cannot exist simultaneously in $\hat{H}^2$. A contradiction.

Q.E.D.

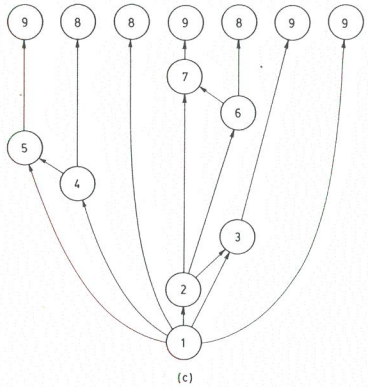


(a)

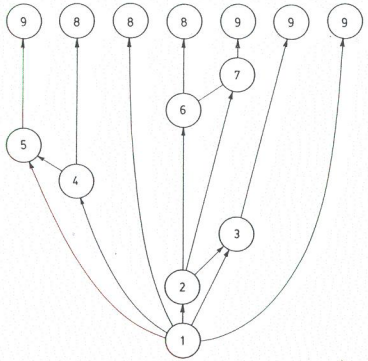


(b)

Figure 8.3 (a & b)



(c)



(d)

Figure 8.3 (c & d)

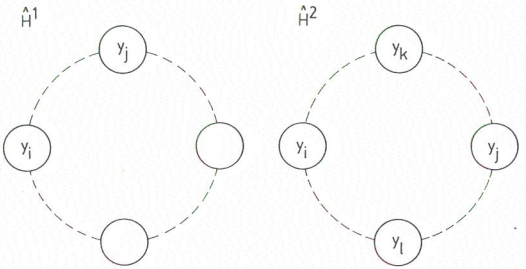


Figure 8.4

Theorem 8.3: If $\hat{B}_k{}^1$ and $\hat{B}_k{}^2$ are bush forms of the same B_k then there exists a sequence of permutations and flippings which transforms $\hat{B}_k{}^1$ into $\hat{B}_k{}^3$, such that in $\hat{B}_k{}^2$ and $\hat{B}_k{}^3$ the virtual vertices appear in the same order.

Proof: By induction on the size* of bush or sub-bush forms. Clearly, if each of the two (sub-)bushes consists of only one vertex and one virtual vertex, then the statement is trivial. Let v be the lowest vertex in the (sub-) bushes $\hat{B}^1$ and $\hat{B}^2$ of the same B. If v is a separation vertex, then the components of B appear as sub-bushes in $\hat{B}^1$ and $\hat{B}^2$. If they are not in the same order, by permuting them in $\hat{B}^1$, they can be put in the same order as in $\hat{B}^2$. By the inductive hypothesis, there is a sequence of permutations and flippings which will change each of the sub-bushs of $\hat{B}^1$ to have the order of its virtual vertices as in its counterpart in the $\hat{B}^2$, and therefore the theorem follows.

If v is not a separating vertex then let H be the maximal nonseparable component of B which contains v. In $\hat{B}^1(\hat{B}^2)$ there is a planar realization $\hat{H}^1(\hat{H}^2)$ of H. The vertices $y_1, y_2, \ldots, y_m$ of H, which are also endpoints of edges of $B - H$, by Lemma 8.12, must appear on the outside window of H in the same order, up to orientation. If the orientation of the y's in $\hat{H}^1$ is opposite to that of $\hat{H}^2$, flip the (sub-) bush $\hat{B}^1$ about v. Now, each of the y's is the lowest vertex of some sub-bush of B, and these sub-bushes appear in (the new) $\hat{B}^1$ and $\hat{B}^2$ in the same order. By the inductive hypothesis, each of these sub-bushes can be transformed by a sequence of permutations and

*The number of vertices.

flippings to have its virtual vertices in the same order as its counterpart in $\hat{B}^2$.

Q.E.D.

Corollary 8.4: If G is planar and $\hat{B}_k$ is a bush form of B_k then there exists a sequence of permutations and flippings which transforms $\hat{B}_k$ into a $\hat{B}_k'$ in which all the virtual vertices labeled $k+1$ appear together on the horizontal line.

It remains to be shown how one decides which permutation or flipping to apply, how to represent the necessary information, without actually drawing bush forms, and how to do it all efficiently. Lempel, Even and Cederbaum described a method which uses a representation of the pertinent information by proper expressions. However, a better representation was suggested by Booth and Leuker. They invented a data structure, called PQ-trees, through which the algorithm can be run in linear time. PQ-trees were used to solve other problems of interest; see [6].

I shall not describe PQ-trees in detail. The description in [6] is long (30 pages) although not hard to follow. It involves ideas of data structure manipulation, but almost no graph theory. The following is a brief and incomplete description.

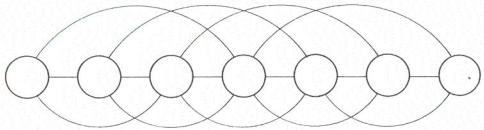
A PQ-tree is a directed ordered tree, with three types of vertices: P-vertices, Q-vertices and leaves. Each P-vertex, or Q-vertex, has at least one son. The sons of a P-vertex, which in our application represents a separating vertex v of B_k, may be permuted into any new order. Each of the sons, and its subtree, represents a sub-bush. A Q-vertex represents a maximal nonseparable component and its sons which represent the y's, may not be permuted, but their order can be reversed. The leaves represent the virtual vertices.

The attempt to gather all the leaves labeled $k+1$ into an unbroken run, is done from sons to fathers, starting with the leaves labeled $k+1$. Through a technique of template matching, vertices are modified while the $k+1$ labeled leaves are bunched together. Only the smallest subtree which contains all the $k+1$ labeled leaves is scanned. All these leaves, if successfully gathered, are merged into one P-vertex and its sons represent the virtual edges out of $k+1$. This procedure is repeated until $k+1=n$.

PROBLEMS

8.1 Demonstrate the path addition algorithm on the Peterson graph (see problem 7.3). Show the data for all the steps: The DFS for numbering the vertices, defining the tree and computing $L1$ and $L2$. The ϕ function on the edges. The sorting of the adjacency lists. Use the path finding algorithm in the new DFS, to decompose the graph into C and a sequence of paths. Use Π_1, Π_2, Π_3 and end of stack markers to carry out all the recursive steps up to planarity decision.

8.2 Repeat the path addition planarity test, as in Problem 8.1, for the graph given below.



8.3 Demonstrate the vertex addition planarity test on the Peterson graph. Show the steps for the DFS, the st-numbering and the sequence of bush forms.

8.4 Repeat the vertex addition planarity test for the graph of Problem 8.2.

8.5 Show that if a graph is nonplanar then a subgraph homeomorphic to one of the Kuratowski's graphs can be found in $O(|V|^2)$. (Hints: Only $O(|V|)$ edges need to be considered. Delete edges if their deletion does not make the graph planar. What is left?)

REFERENCES

[1] Auslander, L., and Parter, S. V., "On Imbedding Graphs in the Plane," J. Math. and Mech., Vol. 10, No. 3, May 1961, pp. 517-523.

[2] Goldstein, A. J., "An Efficient and Constructive Algorithm for Testing Whether a Graph Can be Embedded in a Plane," Graph and Combinatorics Conf., Contract No. NONR 1858-(21), Office of Naval Research Logistics Proj., Dept. of Math., Princeton Univ., May 16-18, 1963, 2 pp.

[3] Hopcroft, J., and Tarjan, R., "Efficient Planarity Testing," JACM, Vol. 21, No. 4, Oct. 1974, pp. 549-568.

[4] Lempel, A., Even, S., and Cederbaum, I., "An Algorithm for Planarity Testing of Graphs," *Theory of Graphs, International Symposium, Rome,* July, 1966. P. Rosenstiehl, Ed., Gordon and Breach, N.Y. 1967, pp. 215-232.

[5] Even, S., and Tarjan, R. E., "Computing an st-numbering," Th. Comp. Sci., Vol. 2, 1976, pp. 339-344.

[6] Booth, K. S., and Lueker, G. S., "Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-tree Algorithms," J. of Comp. and Sys. Sciences, Vol. 13, 1976, pp. 335-379.