CHAPTER 3
Planarity Testing

3.1 INTRODUCTION

The Problem of determining whether a given graph is planar, if there exists some geometric representation of \( G \) which can be drawn on a plane such that no two of its edges intersect, a graph that can be drawn on a plane without a crossover between its edges is called nonplanar.

A drawing of a geometric representation of a graph on any surface such that no edges intersect is called embedding. From the definition planar graphs we have that

A Geometric graph \( G \) is planar, if there exists a graph isometric to \( G \) that is embedded in a plane, An embedding of a plane graph \( G \) on a plane is called a plane representation of \( G \).

To test whether a Graph \( G \) is planar we follow the following Steps.

1. Apply algorithm to check for Connectedness. If Graph is disconnected, consider only one component at a time.
2. Remove all self-loops, and replace each set of parallel edges by a single edge.
3. Eliminate every vertex of degree two by merging the two edges incident on the vertex. Apply Steps 2 and 3 alternately and repeatedly, till the graph cannot be reduced any further.
4. Cut-vertices and Separability to partition the graph into its blocks (maximal nonseparable subgraphs).
5. Subject each block to reduction Steps 3 and 2 alternately till no further reduction is possible.
6. Each simplified block thus obtained, with $e$ edges and $n$ vertices infested for

\[ n \geq 5, \]
\[ e \geq 9, \]
\[ e \leq 3n-6. \]

If any of these inequalities is not satisfied, we move on to the next block. Every graph with $n < 5$ or $e < 9$ is planar and every simple graph with $e > 3n - 6$ is nonplanar.

One planarity testing algorithm due to Bruno[2], Steiglitz[11] and Weinberg [14] goes a Step further in Simplification. Each nonseparable graph is further broken down into its maximum 3-Connected subgraphs (called 3-connected components). The following theorem due to W.T. Tutte[12], is used.

Theorem: A Graph is planar iff all of its 3-Connected components are planar.
3.2 Preliminaries:

A Graph is said to be embeddable in the plane or planar, if it can be drawn in the plane so that its edges intersect only at their ends. Such a drawing of Planar Graph $G$ is called a Planar Embedding of $G$. A Planar embedding $\bar{G}$ of $G$ can itself be regarded as a graph isomorphic to $G$, the vertex set of $G$ is the set of points representing vertices of $G$, the edge set of $\bar{G}$ is the set of lines representing edges of $G$, and a vertex of $\bar{G}$ is incident with all the edges of $\bar{G}$ that contain it. We therefore sometimes refer to a planar embedding of a planar graph as a plane graph. Figure (3.1b) shows a planar embedding of the planar graph in figure (3.1a).

The study of Planar graphs are those which deal with Jordan Curves. A Jordan curve is a continuous non-self intersecting curve whose origin and terminus coincide. The union of the edges in a cycle of a plane graph constitutes a Jordan curve. This is the reason why properties of Jordan curves come into play in planar graph theory. We shall use a well-known theorem about Jordan curves and use it to demonstrate the nonplanarity of $K_5$.

![Planar Graph and Planar Embedding](image)
Let $J$ be a Jordan Curve in the Plane. Then the rest of the plane is partitioned into two disjoint open sets called the interior and exterior of $J$. We shall denote the interior and exterior of $J$, respectively, by $\text{int} \ J$ and $\text{ext} \ J$, and their closures by $\text{int} \ J$ and $\text{ext} \ J$. $\text{Int} \ J \cap \text{Ext} \ J = J$. The Jordan Curve theorem states that any line joining a point in $\text{int} \ J$ to a point in $\text{ext} \ J$ must meet $J$ in some point (see fig(3.2)).

Fig: (3.2)
**THEOREM**: $K_5$ IS NON PLANAR

**PROOF**: If possible let $G$ be a Plane Graph corresponding to $k$. Denote the vertices of $G$ by $v_1,v_2,v_3,v_4$ and $v_5$. Since $G$ is complete, any two of its vertices are joined by an edge. Now the cycle $C = v_1v_2v_3v_1$ is a Jordan curve in the plane, and the point $v_4$ must lie either in int $C$ or ext $C$.

![Figure](3.3)

Now let $v_4 \in \text{int } C$. Then the edges $v_4v_1$, $v_4v_2$ and $v_4v_3$ divide int $C$ into the three regions $\text{int } C_1$, $\text{int } C_2$ and $\text{int } C_3$, where $C_1 = v_1v_4v_2v_1$; $C_2 = v_2v_4v_3v_2$ and $C_3 = v_3v_4v_1v_3$ (see fig. (3.3)). Now $v_5$ must lie in one of the four regions $\text{ext } C_1$, $\text{int } C_2$ and $\text{int } C_3$. If $v_5 \in \text{ext } C$ then, since $v_4 \in \text{int } C$, it follows from the Jordan curve theorem that the edge $v_4v_5$ must meet $C$ in some point. But this contradicts the assumption that $G$ is a plane graph. The cases $v_5 \in \text{int } C_i$, $i=1, 2, 3$ can be disposed of in like manner.
Theorem: $K_{3,3}$ is non-planar.

Proof: Suppose $G$ is a graph of type $k_{3,3}$.
We assume that $G$ is Planar.

By Euler Formula $f = e - n + 2$

In $k_{3,3}$ number of vertices = 6
number of edges = 9

=> $f = 9 - 6 + 2 = 5$

=> Number of regions = 5

In $k_{3,3}$ there is no circuit which is less than four edges.

$D(vi) = 2e \geq 4f$

=> $e \geq 2f$

Here $e = 9, f = 5$

$9 \geq 10$

This is a Contradiction

=> $G$ is non-planar.

$K_{3,3}$ is non-planar.
From the definition of Planar graphs it follows that

**Lemma 1**: If \( G \) is non planar then every subdivision of \( G \) is non-planar

\[ 2 : \text{If } G \text{ is planar then every subgraph of } G \text{ is planar} \]

We now give the proof of

**Theorem**: A graph is planar iff it contains no subdivision of \( K_5 \) or \( K_{3,3} \).

**Proof**: since \( K_5 \) or \( K_{3,3} \) are non planar, it follows that for the above lemma that \( G \) cannot be planar.

We prove sufficiency by contradiction.

If possible, choose a non planar graph \( G \) that contains no subdivision of \( K_5 \) or \( K_{3,3} \) and has a few edges as possible. If \( G \) is simple and 3-connected. Then \( G \) must also be a minimal non planar graph.

Let \( uv \) be an edge of \( G \), and let \( H \) be a planar embedding of the planar graph \( G-uv \), since \( G \) is 3-connected, \( H \) is 2-connected and by corollary, \( u \) and \( v \) are contained together in a cycle of \( H \). Choose a cycle \( C \) of \( H \) that contains \( u \) and \( v \) and is such that the number of edges in \( \text{Int } C \) is as large as possible.

Since \( H \) is simple and 2-connected each bridge of \( C \) in \( H \) must have at least two vertices of attachment. Now all outer bridges of \( C \) must be 2-bridges that overlap \( uv \) because, if some outer bridge were a \( k \)-bridge for \( k \geq 3 \) or a 2-bridge that avoided \( uv \), then there would be a cycle \( C' \) containing \( u \) and \( v \) with more edges in its interior than \( C \), contradicting the choice of \( C \). These two cases are illustrated in figure(3.4) (with \( C \) indicated by heavy lines)

[Diagram of figure(3.4)]
In fact, all outer bridges of C in H must be single edges, for if a 2-bridge with vertices of attachment x and y has a third vertex, the set{x,y} would be a 2-vertex cut of G, contradicting the fact that G is 3-connected.

We know that inner (outer) bridges avoid one another and no two inner bridges overlap. Hence some inner bridge skew to uv must overlap some outer bridge. For otherwise all such bridges could be transferred (one by one), and then the edge uv could be drawn in Int C to obtain a planar embedding of G, which is not possible, since G is non-planar. Therefore, there is an inner bridge B that is both skew to uv and skew to some outer bridge xy. Now let us assume that B has a vertex of attachment different from u,v,x and y.

Then we can rename the vertices, so that B has a vertex of attachment vl in C(x,u) (see fig.(3.5)). If B has a vertex of attachment in C(y,v), B has a vertex of attachment v2 in C(y,v), then there is a (v1,v2) - path P in B that is internally-disjoint from C. But then (C ∪ P) + {uv,xy} is a subdivision of k3,3 in G, a contradiction (see fig(3.5)).

Fig. : (3.5)
If \( B \) has no vertex of attachment in \( C(y,v) \), then \( B \) is skew \( \not\sim uv \) and \( xy \). \( B \) must have vertices of attachment \( v_2 \) in \( C(u,y) \) and \( v_3 \) in \( C(v,x) \). Thus \( B \) has three vertices of attachment \( v_1,v_2 \) and \( v_3 \). There exist a vertex \( v_0 \) in \( V(B) \setminus V(C) \) and three paths \( p_1,p_2 \) and \( p_3 \) in \( B \) joining \( v_0 \) to \( v_1 \), \( v_2 \) and \( v_3 \) respectively, such that, for \( i \neq j \), \( p_i \) and \( p_j \) have only the vertex \( v_0 \) in common. But now \( (C \cup P_1 \cup P_2 \cup P_3) + \{uv, xy\} \) contains a subdivision of \( k_{3,3} \) a contradiction. This case is illustrated in fig. \( (3.6) \). The subdivision \( k_{3,3} \) is indicated by heavy lines.

\[
\text{Fig. (3.6)}
\]

Next \( B \) has no vertex of attachment other than \( u,v,x \) and \( y \) then \( B \) is skew to both \( uv \) and \( xy \). \( \not\sim \) that \( u,v,x \) and \( y \) must all be vertices of attachment of \( B \). \( \not\sim \) there exist a \((u,v)\) - path \( P \) and an \((x,y)\) - path \( Q \) in \( B \) such that \( (i) \) \( P \) and \( Q \) are internally disjoint from \( C \) and \( (ii) \) \( |V(P) \cap V(Q)| \geq 1 \).
If $|V(P) \cap V(Q)| = 1$

then we have that $(C \cup P \cup Q) + \{uv,xy\}$ is a subdivision of $K_5$ in $G$, again a contradiction (see fig.(3.7))

Fig.: (3.7)

Again we assume that $|V(P) \cap V(Q)| \geq 2$

Let $u'$ and $v'$ be the first and last vertices of $P$ on $Q$ and let $P_1$ and $P_2$ denote the $(u,u')$ - and $(v',v)$ sections of $P$. Then $(C \cup P_1 \cup P_2 \cup Q) + \{uv, xy\}$ contains a subdivision of $K_{3,3}$ in $G$, again a contradiction see fig(3.8) hence

A graph is planar iff it contains no subdivision of $K_5$ or $K_{3,3}$

Fig.: (3.8)
3.3 PROGRAM:

```c
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

struct point { int flag;
    int x1;
    int y1;
    int d1;
    int x2;
    int y2;
    int d2;
}

typedef struct point node;

node nlist[100];

int n,i,j;

main () {
    void c - par(void);
    void ele(void);
    int tflag, count, ecount;
    printf( "enter the number of edges \n ");
    scanf("%d", &d1);
    for (i=1; i<=n; i++)
    {
        printf("enter the %d th edge : \n",i);
        printf("enter the first coordinate : ");
        scanf("%d",&nlist[i].x1);
        scanf("%d",&nlist[i].y1);
        printf("enter the second coordinate");
        scanf("%d",&nlist[i].x2);
    }
}
```

scanf("%d", &nlist[i].y2);
nlist[i].flag = Q;
nlist[i].d1 = nlist[i].d2 = 1;
}

for (i=1; i<=n; i++)
{
    for (j=1; j<=n; j++)
    {
        if ( i != j )
        {
            if ((nlist[i].x1 = nlist[j].x1) && (nlist[i].y1 = nlist[j].y1))
                nlist[i].d1 ++;
            if ((nlist[i].x2 = nlist[j].x2) && (nlist[i].y2 = nlist[j].y2))
                nlist[i].d2 ++;
            if ((nlist[i].x1 = nlist[j].x2) && (nlist[i].y1 = nlist[j].y2))
                nlist[i].d1 ++;
            if ((nlist[i].x2 = nlist[j].x1) && (nlist[i].y2 = nlist[j].y1))
                nlist[i].d2 ++;
        }
    }
}

for (i=1; i<=n; i++)
    printf("%d %d", nlist[i].d1, nlist[i].d2);
for(i =1; i<=n; ++i)
{
    if ((nlist[i].x1 = nlist[i].x2) && (nlist[i].y1 = nlist[i].y2))
        nlist[i].flag =1;
}
count = 100;
tflag = Q;
while (count > 1) {
    c-par();
    ele();
    if (tflag == 1)
        ecount = count;
    count = Q;
    for (i=1; i<=n; i++)
        if (nlist[i].flag == Q)
            count ++;
    if (tflag == Q)
        ecount = count;
    else
        if (ecount == count)
            break;
        tflag = 1;
}
if (count == 1 || count == Q) {
    printf("\n planar ");
} else
    printf(" non planar");
return;
} /*--------------- end of main-------------------------*/

void c-par(void)
{
    int i,j;
    for (i=1; i<=n; i++)
for(j=1; j<=n; ++j)
{
    if(i != j) && (nlist[i].flag != 1) && (nlist[j].flag != 1)
    {
        if(nlist[i].x1 == nlist[j].x1) && (nlist[i].y1 == nlist[j].y1)
            nlist[i].flag = 1;
        if((nlist[i].x1 == nlist[j].x2) && (nlist[i].y1 == nlist[j].y2))
            nlist[i].flag = 1;
    }
}
}  /*----------------- end parallel ----------------------*/

/* for (i =1; i<*n; ++i)
 {
    printf(" %d",nlist[i].flag);
 } */

void ele(void)
{
    int i,j;
    for (i= 1; i<=n; i++)
    {
        for(j=1; j<=n; j++)
        {
            if (((i != j) && (nlist[i].d1 == 2) || (nlist[j].d2 == 2)))
            {
                if (nlist[j].flag == 1)
                    break;
            }
        }
    }
if((nlist[i].x1==nlist[j].x1) && (nlist[i].y1==nlist[j].y1))
{
    nlist[i].flag = 1;
    nlist[j].x2 = nlist[i].x2;
    nlist[j].y1 = nlist[i].y2;
}
if((nlist[i].x2 == nlist[j].x2) && (nlist[i].y2==nlist[j].y2))
{
    nlist[i].flag = 1;
    nlist[j].x2 = nlist[i].x1;
    nlist[j].y2 = nlist[i].y1;
}
if ((nlist[i].x1==nlist[j].x2) && (nlist[i].y1 == nlist[j].y2))
{
    nlist[i].flag = 1;
    nlist[j].x2 = nlist[i].x2;
    nlist[j].y2 = nlist[i].y2;
}
if ((nlist[i].x2==nlist[j].x1) && (nlist[i].y2 == nlist[j].y1))
{
    nlist[i].flag = 1;
    nlist[j].x1 = nlist[i].x2;
    nlist[j].y1 = nlist[i].y2;
}
} /*------------------ end of file ----------------*/
3.4 Result

Enter The No. Of Edges
3
Enter The 1th Edge :
Enter The First Coordinate : 2 2
Enter The Second Coordinate : 4 2
Enter The 2th Edge :
Enter The First Coordinate : 4 2
Enter The Second Coordinate : 3 3
Enter The 3th Edge :
Enter The First Coordinate : 3 3
Enter The Second Coordinate : 2 2
2  22   22  2

***PLANARITY***
Enter The No. Of Edges
3
Enter The 1th Edge:
Enter The First Coordinate: 1 1
Enter The Second Coordinate: 4 5
Enter The 2th Edge:
Enter The First Coordinate: 1 1
Enter The Second Coordinate: 6 8
Enter The 3th Edge:
Enter The First Coordinate: 1 1
Enter The Second Coordinate: 3 10
3 13 13 1

****PLANARITY****
Enter The No. Of Edges
5
Enter The 1th Edge:
Enter The First Coordinate: 2 8
Enter The Second Coordinate: 8 8
Enter The 2th Edge:
Enter The First Coordinate: 8 8
Enter The Second Coordinate: 8 4
Enter The 3th Edge:
Enter The First Coordinate: 8 4
Enter The Second Coordinate: 4 1
Enter The 4th Edge:
Enter The First Coordinate: 4 1
Enter The Second Coordinate: 2 4
Enter The 5th Edge:
Enter The First Coordinate: 2 4
Enter The Second Coordinate: 2 8
2 22 22 22 22 2

**** NON PLANARITY****