Introduction

- Graph Theory is an endless source of easily stated yet very hard problems.
- Since graphs are very widely used, it is very important to discover efficient algorithms for these problems.
- In this presentation we are going to explain efficient planarity testing alogorithm to determine whether a graph G can be embedded in a plane.

PLANARITY TESTING AND EMBEDDING
Research on Planarity Algorithms

1 The earliest characterization of planar graphs was given by Kuratowski . He proved that every non - planar graph contains a subgraphwhich upon removal of degree two vertices is isomorphic either to the complete graph on five vertices ot to a complete bipartite graph of six vertices . But his practical test of planarity require an amount of time proportinal to at least polynomial of degree six

2 Auslander and Parter came out with different kind of approach . First , a cycle is found in the graph . When the cycle is removed , the graph falls into several places . The algorithm is called recursively to embed each piece in the plane with the original cycle

Preliminary graph-theoretic concepts

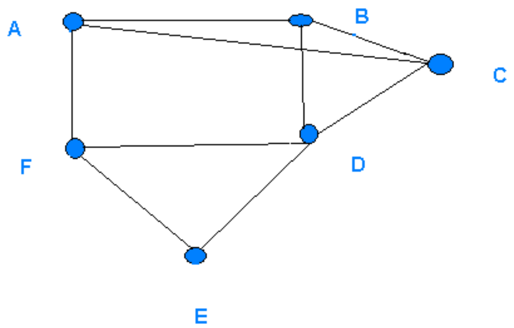
- *If each edge is an unordered pair of distinct vertices , then the graph is undirected*
- *If G is a directed graph , the undirected version of G is the undirected graph formed by converting each edge of G to an undirected edge and removing duplicate edges .*
- *A Graph G = (V1,E1) is a subgraph of a graph G1 = (V2,E2) if V1,E1 are subsets of V2,E2 respectively .*
- *If G contains three distinct vertices x,v,w such that w is a reachablefrom v but every path p:v≥contains x , then x is a cutnode or separation point of G .*
- *If G is connected and contains no separation points , then G is biconnected .*

Preliminary graph-theoretic concepts

- *If T is a tree which is a subgraph of a directed graph G and T contains all vertices of G , then T is a spanning tree of G .*
- **A Graph g is planar if and only if there exists a mapping of the vertices and edges of a graph into the plane such that (1) each vertex is mapped into a distinct point , (2) each edge (v,w) is mapped onto a simple curve , with the vertex v and w mapped onto the end-points of the curve , and (3) mappings of distict edges have only the mappings of their common endpoints in common**
- **A mapping of G which satisfies the conditions above is called a planar embedding of G**

Mathematical Testing for Planar Graphs

- *Euler's theorem relates the number of vertices and edges in planar graph*
- **E$\leq$ (3V-3)** *This condition cane be applied to test a graph for planarity*
- **Example** *In below figure , number of vertices(V) = 6 , number of edges(E) = 9 ,clearly 9 $\leq$ ((3*6)-3) is true*
- *The validity of the above equation can be checked through Mathematical Inducton*

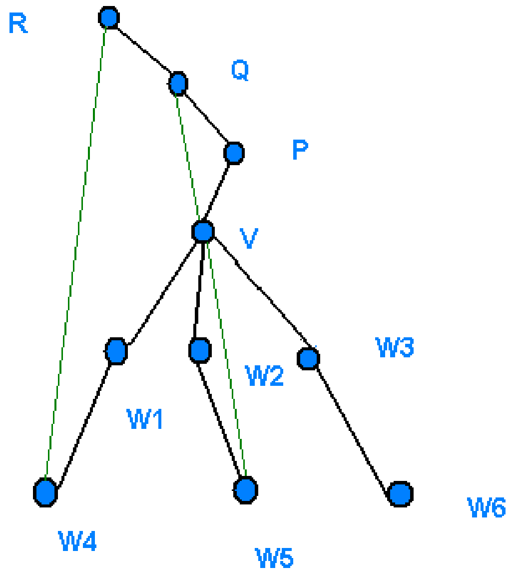


Graphical Representation

- *We use a list structure called adjacency structure to represent the graph . There is a set of adjacency lists A(v) for each , one for each vertex*
- *The list for vertex v contains each vertex w such that (v,w) is an edge of the graph . If G is undirected graph , each edge (v,w) will appear twice , v will appear in A(w) and v will appear in A(v)*
- **DFS** *A depth first search of G imposes a direction on each edge of G given by the direction in which the edge is traversed during the search . Search converts G into directed graph G1*

New terms

- *Tree edge - The edge (v,w) formed during the DFS . Notation :* $v \rightarrow w$
- *Fronds - The remaining edges of v leaving the edge (v,w) . Notation :* $v \dashrightarrow w$



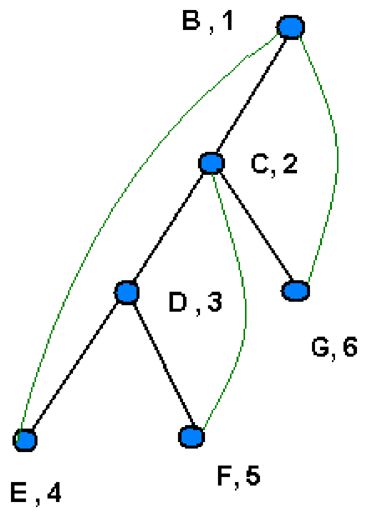
A Graph G is planar iff there exists a mapping of the vertices and edges of the graph into the plane such that

- each vertex is mapped into a distinct point.
- each edge (v,w) is mapped onto a simple curve, with the vertices v and w mapped onto the end-points of the curve, and
- mapping of distinct edges have only the mapping of their common endpoints in common.

A mapping of G which satisfies the conditions above is called a planar embedding of G.

New terms

- **LOWPT1(v)** *is the lowest vertex below v reachable by a frond from a descendant of v*
- **LOWPT2(v)** *is the second lowest vertex below vreachable by a frond from a descendant of v*

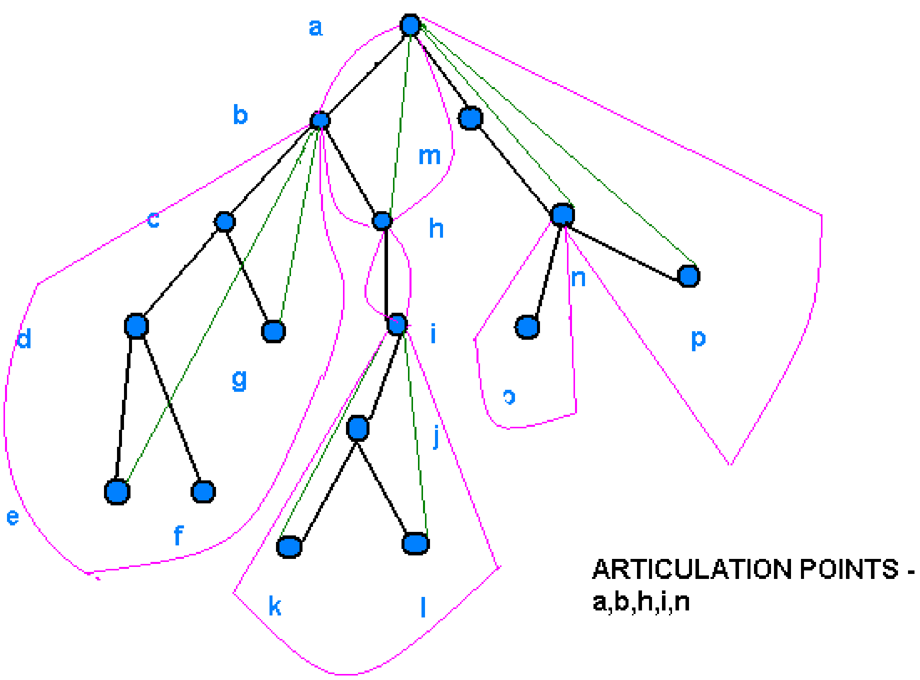


Vertex	L1	L2
B	1	1
C	1	2
D	1	2
E	1	4
F	2	5
G	1	6

L1 => LOWPT1(v)
L2 => LOWPT2(v)

New terms

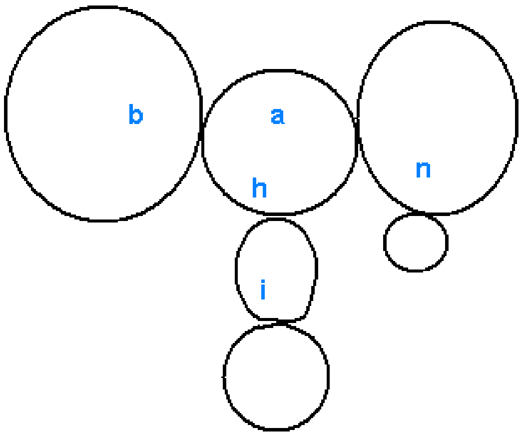
- **ARTICULATION POINTS -** *A vertex is called an articulation point if it disconnects it's descendants from rest of the graph*



- **BICONNECTED GRAPH -** *If there are at leat two vertex disjoint paths from every vertex to every other vertex . Similarly* **k-connected** *means if there are atleast k vertex disjoint paths between every two vertices*

New terms

- **BICONNECTED COMPONENTS** - *The components formed after dividing the graph around each articulation points*



BICONNECTED COMPONENTS

PATHFINDING

- Here the graph G is assumed to be biconnected graph which is explored using DFS to number the vertices and generate a plam tree P.
- If v is a vertex, then

$$S_v = \{w | \exists u \, (v \rightarrow *u \;\; and \;\; u- \rightarrow w)\}$$

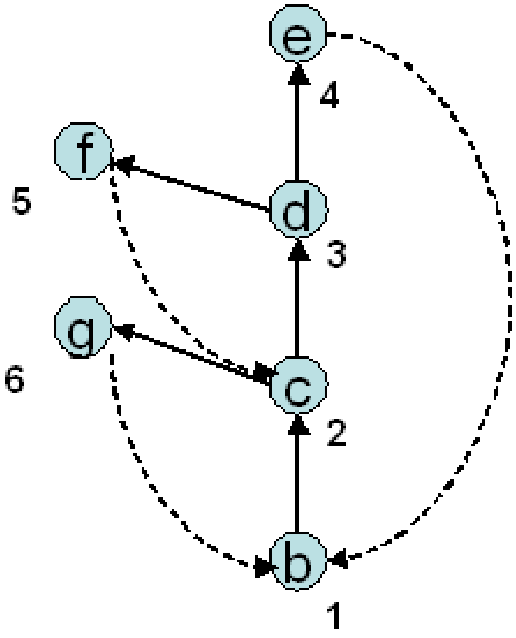
which implies S_v is the set of vertices reached by fronds from decendants of v.

LOW points

$$LOWPT1(v) = min(\{v\} \cup S_v)$$

$$LOWPT2(v) = min(\{v\} \cup (S_v - \{LOWPT1(v)\}))$$

- By convention, these values are equal to v if they are not defined.
- LOWPT1(v) $\neq$ LOWPT2(v) unless LOWPT1(v) = LOWPT2(v) = v.
- LOWPT values of v depends only on the LOWPT values of sons of v and on the fronds leaving v
- therefore, it is easy to calculate LOWPT values using DFS.



node(v)	LOWPT1(v)	LOWPT2(v)
b	1	1
c	1	2
d	1	2
e	1	4
f	2	5
g	1	6

- These Low Points are used in other graph algorithms also.
- LOWPT1 may be used to test the biconnectivity of Graph
- If G is a *biconnected* and

  $$v \rightarrow w,\ LOWPT1(w) < v\ unless\ v = 1$$

- In the above case if $v = 1$,

  $LOWPT1(w) = v = 1. Also, LOWPT1(1) = 1$

- To find paths, the adjecency lists of graph G are sorted according to LOWPT values and another Depth First Search is performed

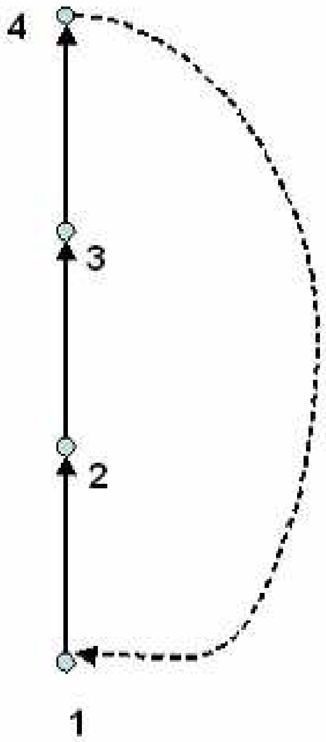
- Φ is a fuction defined on edges (v,w) of P.

$$\Phi((v,w)) = \begin{cases} 2w & \text{if } v- \to w, \\ 2LOWPT1(w) & \text{if } v \to w \ and \\ & \text{LOWPT2(w)} \geq \text{v}, \\ 2LOWPT1(w)+1 & \text{if } v \to w \ and \\ & \text{LOWPT2(w)} < \text{v}. \end{cases}$$

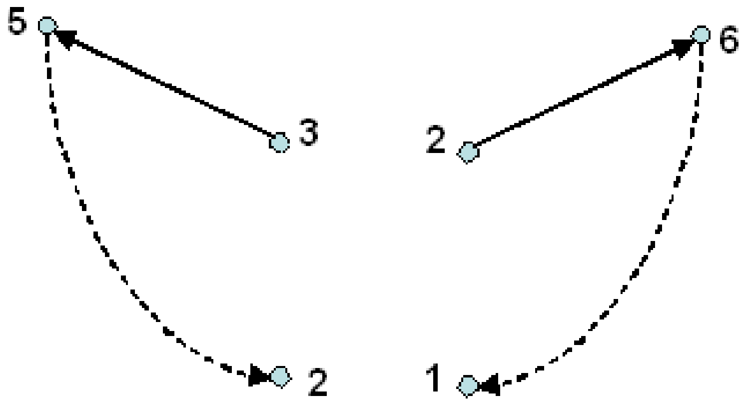
- caluculate Φ for each edge in P and order the adjacency lists according to increasing value of ϕ.

node	adjacency list
b	c,e,g
c	d,g,b,f
d	e,f,c
e	b,d
f	c,d
g	b,c

- The sorting of adjacency lista will give a set of properly ordered adjacency list representing P.
- To generate paths apply DFS to P using new adjacency lists.
- All we need to know is the starting and finishing vertex of each path.



first cycle 'c'



Paths Generated

HOPCROFT-TARJAN algorithm - *steps*

- Division of Graph into Biconnected Components .
- Finding cycle C and segments in PalmTree p
- Embedding of paths .

HOPCROFT-TARJAN'S ALGORITHM PLANARITY TESTING AND EMBEDDING

- Till now we have gone through key terms like :

  3 **Fronds**

  4 **Segments**

  5 **Biconnected Components**

  6 **Low points**

  7 **Articulation points**

  8 **Spanning Tree**

Applications of Embedding ::

- **Designing of Integrated Circuits**
- **Determination of** *Isomorphism* **of Chemical Structures**
- **Silicon Chips development**
- **Graph Drawing tools**

The importance of Embedding problem is obvious through the number of published algorithms !!

Defining *Planar Embedding* of a Graph

- **A drawing of a graph G is a function that: maps each vertex $v \epsilon$ V(G) to a point in the plane and it maps each edge uv to a distinct curve so that no two edges have a common point other than a common vertex else there will be a** *crossing.*

- **A graph is a planar if: it can be drawn without a crossing. We call such a drawing to be planar embedding of graph G.**

- *remember a graph is planar iff it can be embedded*

Terms used to explain embedding ::

- **Stack L and R :** *are used to keep entries for the fronds embedded on the left and right of the cycle.*

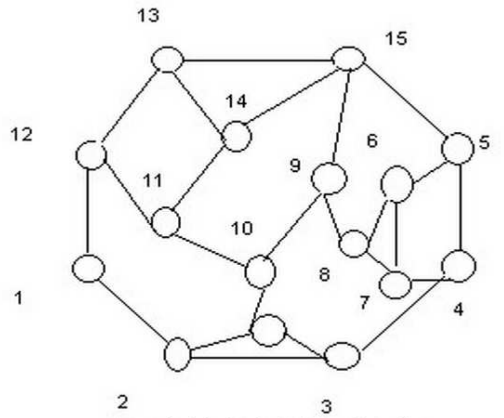
- **Stack B :** *Stack B contains information about the end of each block of entries.*

  Entries in Stack B are pair of vertices If (x,y) is an entry in Block B ,x denotes last entry in stack L while y denotes the last entry in stack R.

- *If x=0 then the block do not have an entry in L*
- *If y=0 then the block do not have an entry in R*

So, now it is obvious that intersection of block B **with** L **will result the adjacent on** L **while with** R **it gives the adjacent on stack** R.

- **Example :**



A graph G to be tested for planarity

- **The order of DFS is 1→2→3....→16**
- **The number of vertices are:16**
- **The number of edges are:23**
- **So,checking Euler's condition : 23 ≤ (3*16-3) =45** So,it holds.
- **Now next step is to compute** *adjacency list of graph*

- *Construct Weighted Adjacency List*
- *Finding Segments and Paths generated by Pathfinder Algorithm*
- *Go for Embedding Algorithm*

Outline of planarity Algorithm

- get rid of graphs with too many edges($E \leq 3V - 3$ for the graph to be planar)
- divide the graph into Biconnected components. (A graph is planar iff all its biconnected components are planar).
- apply DFS to each component to convert the graph into a plam tree and number the vertices.
- find the cycle in the graph and delete it, leaving a set of disconnected pieces.
- now the algorithm checks the planarity of each piece plus the original cycle(by applying itself recursively), and determine whether the embedding of pieces can be combined to give an embedding of the entire graph.

REFERENCES

- Planarity Notes.
- Introduction to algorithm , Udi Manber.
- Introduction to graph theory , Douglas B.West.
- JACM ,Volume 21.