INCIDENCE MATRIX

Representation of undirected graph

Consider a undirected graph G = (V, E) which has n vertices and m edges all labelled. The incidence matrix B = $\{bij\}$, is then $n \times m$ matrix,

where $b_{ij} = \begin{cases} 1 & \text{when edge } e_j \text{ is incident with vertex } v_i \\ 0 & \text{otherwise} \end{cases}$

We can make a number of observations about the incidence matrix B of G.

(i) Each column of B comprises exactly two unit entries.
(ii) A row with all 0 entries corresponds to an isolated vertex.
(iii) A row with a single unit entry corresponds to a pendent vertex.
(iv) The number of unit entries in row i of B is equal to the degree of the corresponding vertex v_i.
(v) The permutation of any two rows (any two columns) of B corresponds to a labelling of the vertices (edges) of G.
(vi) Two graphs are isomorphic if and only if their corresponding incidence matrices differ only by a permutation of rows and columns.
(vii) If G is connected with n vertices then the rank of B is $n - 1$.

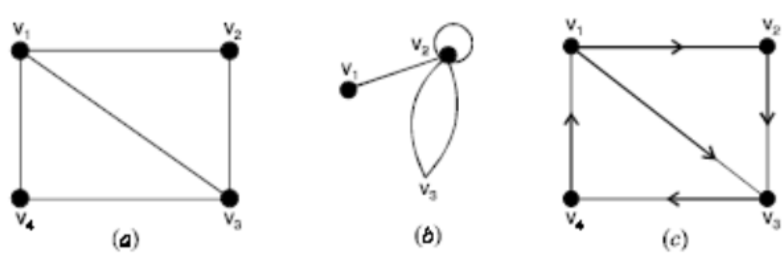
Incidence matrices can also be used to represent multiple edges and loops. Multiple edges are represented in the incidence matrix using columns with identical entries. Since these edges are incident with the same pair of vertices. Loops are represented using a column with exactly one entry equal to 1, corresponding to the vertex that is incident with this loop.

Representation of directed graph

The incidence matrix B = $\{b_{ij}\}$ of digraph D with n vertices and m edges is the $n \times m$ matrix in which

$$b_{ij} = \begin{cases} 1 & \text{if arc } j \text{ is directed away from vertex } v_i \\ -1 & \text{if arc } j \text{ is directed towards vertex } v_i \\ 0 & \text{otherwise} \end{cases}$$

Problem 14. *Use adjacency matrix to represent the graphs shown in Figure below*



Solution. We order the vertices in Figure (*a*) as v_1, v_2, v_3 and v_4.
Since there are four vertices, the adjacency matrix representing the graph will be a square matrix of order four. The required adjacency matrix A is

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

We order the vertices in Figure (*b*) as v_1, v_2 and v_3. The adjacency matrix representing the graph is given by

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 2 \\ 0 & 2 & 0 \end{bmatrix}$$

Taking the order of the vertices in Figure (*c*) as v_1, v_2, v_3 and v_4. The adjacency matrix representing the graph is given by

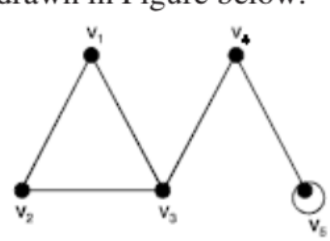
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Problem 15. *Draw the undirected graph represented by adjacency matrix A given by*

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Solution.

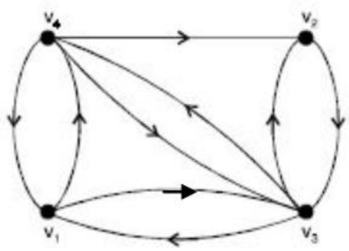
Since the given matrix is a square of order 5, the graph G has five vertices v_1, v_2, v_3, v_4 and v_5.Draw an edge from v_i to v_j where $a_{ij} = 1$.
The required graph is drawn in Figure below.



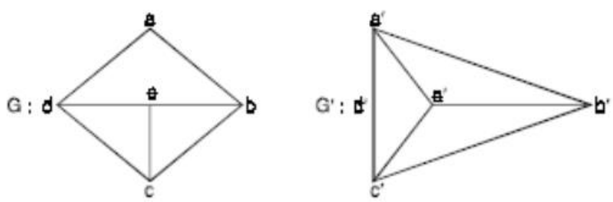
Problem 16. *Draw the digraph G corresponding to adjacency matrix*

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Solution. Since the given matrix is square matrix of order four, the graph G has 4 vertices v_1, v_2, v_3 and v_4 . Draw an edge from v_i to v_j where $a_{ij} = 1$.
The required graph is shown in Figure below.



Problem 17. *Show that the graphs G and G ' are isomorphic*



Solution. Consider the map f : G →G′ defined as $f(a) = d'$, $f(b) = a'$, $f(c) = b'$, $f(d) = c'$ and $f(e) = e'$
The adjacency matrix of G for the ordering a, b, c, d and e is

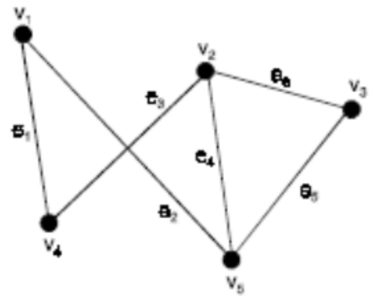
$$A(G) = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

The adjacency matrix of G′ for the ordering d', a', b', c' and e' is

$$A(G') = \begin{array}{c} \\ d' \\ a' \\ b' \\ c' \\ e' \end{array} \begin{array}{c} \begin{array}{ccccc} d' & a' & b' & c' & e' \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

i.e., $A(G) = A(G')$
Therefore G and G' are isomorphic.

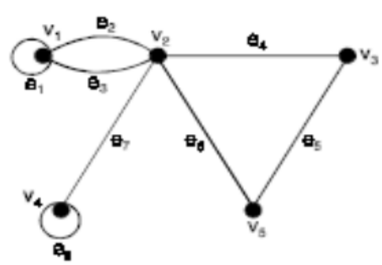
Problem 18. *Represent the graph shown in Figure below, with an incidence matrix.*



Solution. The incidence matrix is

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

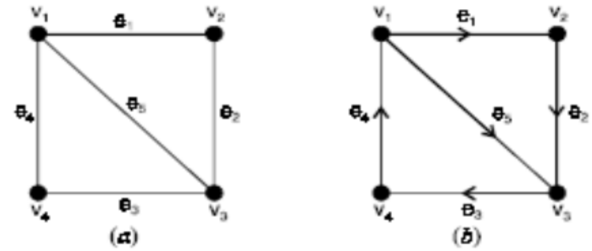
Problem 19. *Represent the Pseudo graph shown in Figure below, using an incidence matrix.*



Solution. The incidence matrix for this graph is

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Problem 20. *Find the incidence matrix to represent the graph shown in Figure below :*



Solution.

The incidence matrix of Figure (a) is obtained by entering for row v and column e is 1 if e is incident on v and 0 otherwise. The incidence matrix is

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array}\begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

The incidence matrix of the graph of Figure (b) is

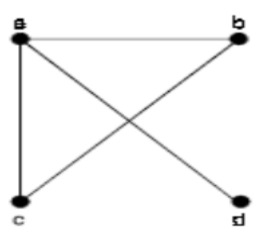
$$\begin{bmatrix} 1 & 0 & 0 & -1 & 1 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & -1 \\ 0 & 0 & -1 & 1 & 0 \end{bmatrix}$$

Problems for practice

1. *Draw the undirected graph G corresponding to adjacency matrix*

$$A = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 2 & 2 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

2. *Use an adjacency matrix to represent the graph shown in Figure below*



3. *Draw a graph with the adjacency matrix*

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

WALKS, PATHS AND CYCLES

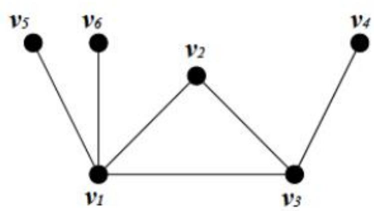
Definition

A walk in G is a sequence of vertices $v_0,\ v_1, \dots, v_k$ and a sequence of edges$(v_i,\ v_{1+1}) \in E\ (G)$. A walk is a path if all v_i are distinct. v_0 is the initial vertex and v_k is the terminal vertex. A zero length walk is just a single vertex v_0. If for such a path with $k\ \geq\ 2$, $(v_0,\ v_k)$is also an edge in G, then $v_0,\ v_1, \dots,\ v_k,\ v_0$ is a cycle. For multigraphs, we also consider loops and pairs of multiple edges to be cycles.

Definition

The length of a path, cycle or walk is the number of edges in it.

Example



$v_5v_1v_3v_4$ is a path of length 3

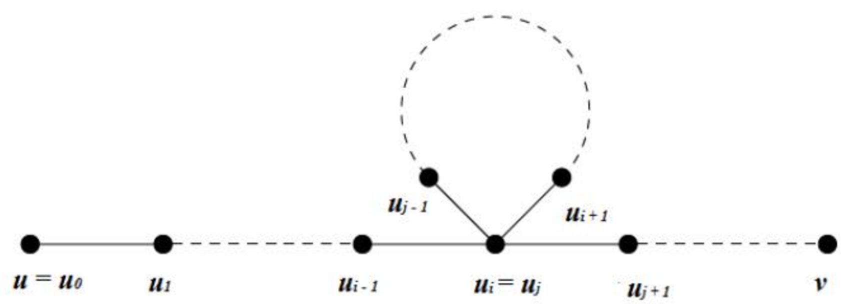
$v_1v_2v_3$ is a cycle of length 3

$v_5v_1v_2v_3v_1v_6$ is a walk of length 5

Proposition: Every walk from *u* to *v* in G contains a path between *u* and *v*.

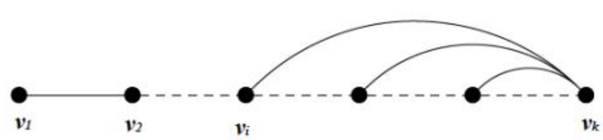
Proof.

By induction on the length *l* of the walk $u = u_0,\ u_1, \dots, v_l = v$.If $l\ =\ 1$ then our walk is also a path. Otherwise, if our walk is not a path there is $u_i = u_j$with $i\ <\ j$, then $u = u_0,\ u_1,\ u_i, u_{j+1}, v$ is also a walk from *u* to *v* which is shorter. We can use induction to conclude the proof.



Proposition: Every G with minimum degree $\delta \geq 2$ contains a path of length δ and a cycle of length at least $\delta + 1$.

Proof. Let $v_1,\ v_2, \dots, v_k$ be a longest path in G. Then all neighbors of v_k belong to $v_1, v_2, \dots, v_{k-1}$so $k - 1 \geq \delta$ and $k \geq \delta + 1$, and our path has at least δ edges. Let $i\ (1 \leq i \leq k)$be the minimum index such that $(v_i,\ v_k) \in E\ (G)$. Then the neighbors of v_k are among$v_1,\ v_2, \dots, v_{k-1}$, so $k -\ i \geq \delta$. Then $v_i,\ v_{i+1}, \dots, v_k$ is a cycle of length at least $\delta + 1$.



TREES

Definition:

A graph having no cycle is acyclic. A tree is a connected acyclic graph. A leaf (or pendant vertex) is a vertex of degree 1. A forest is an acyclic graph. A tree is a connected forest. A subforest is a subgraph of a forest. A connected subgraph of a tree is a subtree. A spanning tree of a connected graph is a subtree that includes all the vertices of that graph. The edges of a spanning tree are called branches.

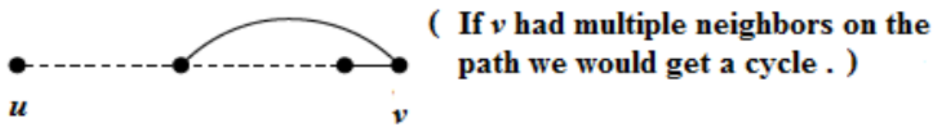
Example:

forest **tree**

Lemma: Every finite tree with at least two vertices has at least two leaves. Deleting a leaf from an n-vertex tree produces a tree with $n-1$ vertices.

Proof.

Every connected graph with at least two vertices has an edge. In an acyclic graph, the end points of a maximum path have only one neighbor on the path and therefore have degree 1.Hence the endpoints of a maximum path provide the two desired leaves.



Suppose v is a leaf of a tree G, and let $G' = G - v$. If $u, w \in V(G')$, then no u, w-path P in G can pass through the vertex v of degree 1, so P is also present in G'. Hence G' is connected. Since deleting a vertex cannot create a cycle, G' is also acyclic. We conclude that G' is a tree with $n-1$vertices.

Theorem: For an n-vertex simple graph G (with $n \geq 1$), the following are equivalent (and characterize the trees with n vertices).
(a) G is connected and has no cycles.
(b) G is connected and has $n-1$ edges.
(c) G has $n-1$ edges and no cycles.
(d) For every pair $u, v \in V(G)$, there is exactly one $u, v -$ path in G.

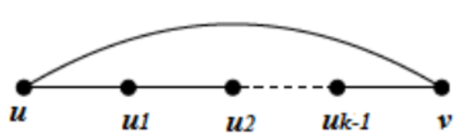
To prove this theorem we will need a small lemma.

Definition: An edge of a graph is a cut-edge if its deletion disconnects the graph.

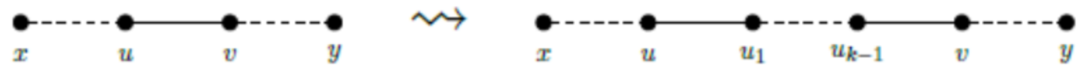
Lemma: An edge contained in a cycle is not a cut-edge.

Proof of the lemma:

Let (u, v) belong to a cycle.



Then any path x ... y in G which uses the edge (u, v) can be extended to a walk in $G - (u, v)$ asfollows:



Proof of Theorem:

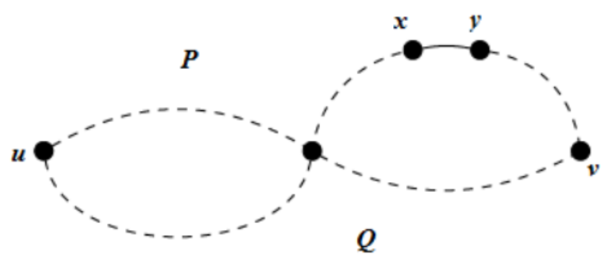
We first demonstrate the equivalence of (a), (b), (c) by proving that any two of {connected, acyclic, $n - 1$ edges} implies the third.

(a) ⇒ (b), (c): We use induction on n. For n = 1, an acyclic 1-vertex graph has no edge. For the induction step, suppose n > 1, and suppose the implication holds for graphs with fewer than n vertices. Given G, the Lemma provides a leaf v and states that $G' = G - v$ is acyclic and connected. Applying the induction hypothesis to G'yields $e(G') = n - 2$, and hence$e(G) = n - 1$.

(b) ⇒ (a), (c): Delete edges from cycles of G one by one until the resulting graph G' is acyclic. By Lemma, G is connected. By the paragraph above, G' has $n - 1$ edges. Since this equals$|E(G)|$, no edges were deleted, and G itself is acyclic.

(c) ⇒ (a), (b): Suppose G has k components with orders $n_1, . . . n_k$. Since G has no cycles, each component satisfies property (a), and by the first paragraph the ith component has $n_i - 1$ edges. Summing this over all components yields $e(G) = \sum(n_i - 1) = n - k$. We are given $e(G) = n - 1$, so $k = 1$, and G is connected.

(a) ⇒ (d): Since G is connected, G has at least one u, v −path for each pair $u, v \in V(G)$. SupposeG has distinct u, v −paths P and Q. Let $e = (x, y)$ be an edge in P but not in Q. The concatenation of P with the reverse of Q is a closed walk in which e appears exactly once. Hence, $(P \cup Q) - e$ is an x, y −walk not containing e. Thus we have a cycle with e and contradicts the hypothesis that G is acyclic. Hence G has exactly one u, v −path.



(d) ⇒ (a): If there is a u; v-path for every u; v $\mathcal{E}$ V (G), then G is connected. If G has a cycle C, then G has two paths between any pair of vertices on C.

Definition:

Given a connected graph G, a spanning tree T is a subgraph of G which is a tree and contains every vertex of G.

Corollary:

(a) Every connected graph on n vertices has at least $n - 1$ edges and contains a spanning tree;

(b) Every edge of a tree is a cut-edge;

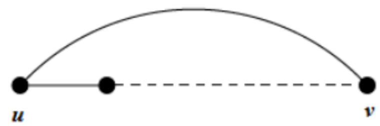
(c) Adding an edge to a tree creates exactly one cycle.

Proof.

(a) Delete edges from cycles of G one by one until the resulting graph G_0 is acyclic. By Lemma, G is connected. The resulting graph is acylic so it is a tree. Therefore G had at least $n - 1$edges and contains a spanning tree.

(b) Note that deleting an edge from a tree T on n vertices leaves $n - 2$ edges, so the graph is disconnected by (a).

(c) Let $u, v \in T$. There is a unique path in T between u and v, so adding an edge (u, v) closes this path to a unique cycle.



Theorem: A connected graph has at least one spanning tree.

Proof.

Consider the connected graph G with n vertices and m edges. If $m = n - 1$, then G is a tree. Since G is connected, $m \geq n - 1$. We still have to consider the case $m \geq n$, where there is a circuit in G. We remove an edge e from that circuit. G − e is now connected. We repeat until there are $n - 1$ edges. Then, we are left with a tree.

Theorem: If a tree is not trivial, then there are at least two pendant vertices.

Proof.

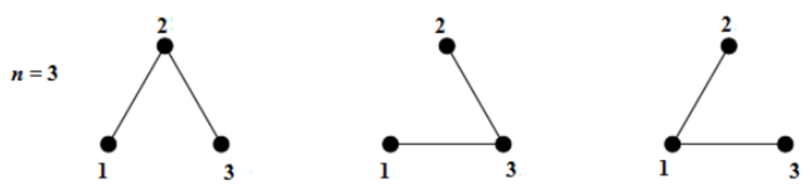
If a tree has n ≥2 vertices, then the sum of the degrees is 2(n − 1). If every vertex has a degree ≥ 2, then the sum will be $\geq 2n$. On the other hand, if all but one vertex have degree ≥ 2, then the sum would be $\geq 1 + 2(n - 1) = 2n - 1$. This is because a cut vertex of a tree is not a pendant vertex. A forest with k components is sometimes called a k-tree. (So a 1-tree is a tree.)

Theorem (Cayley's Formula). There are n^{n-2}trees with vertex set n.

Question: What is the number of spanning trees in a labeled complete graph on n vertices?

By Cayley's formula, it is n^{n-2}.

Example:



Theorem: If G is a tree, then the number of edges in G= $n - 1$.

Proof.

Let us denote the number of edges in G by m. By induction on n, when n = 1, G is isomorphic to K_1 and so the number of edges in G is $m = 0 = n - 1$. Suppose the theorem is true for all trees on fewer than v vertices and let G be a tree on $n \geq 2$ vertices. Let $(u, v) \in E(G)$, then $G - (u, v)$ contains no $u, v -$ path, since (u, v) is the unique $u, v -$ path in G. Thus $G - (u, v)$ is disconnected so $\omega(G - uv) = 2$.The components G_1 and G_2 of $G - (u, v)$, being acyclic are trees. Moreover, each has fewer than n vertices. Therefore by induction hypothesis, $E(G_i) = V(G_i) - 1$, for $i = 1, 2$. Thus$E(G) = E(G_i) + E(G_i) + 1 = V(G_1) + V(G_2) + 1 = V(G) - 1 = n - 1$.

CONNECTIVITY

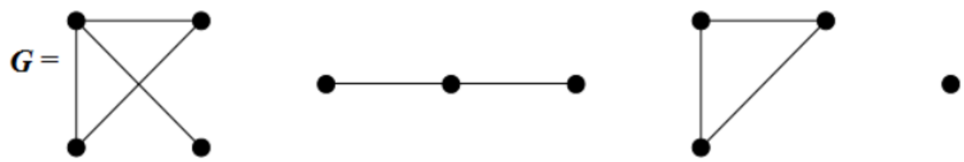
Definition:

A graph G is connected if, for all pairs $u, v \in V(G^{'})$, there is a path in G from u to v.

Note that it suffices for there to be a walk from u to v, by Proposition

Definition:

A (connected) component of G is a connected subgraph that is maximal by inclusion. We say G is connected if and only if it has one connected component. The graph G which is given below has 4 connected components.



Proposition: A graph with n vertices and m edges has at least $n - m$ connected components.

Proof.

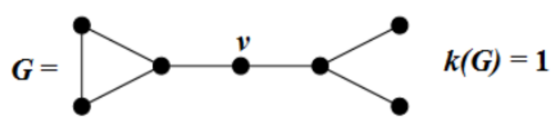
Start with the empty graph (which has n components), and add edges one-by-one. Note that adding an edge can decrease the number of components by at most 1.

Definition: (Vertex connectivity)

A vertex cut in a connected graph $G = (V, E)$ is a set $S \subseteq V$ such that $G \setminus S = G[V \setminus S]$ has more than one connected component. A cut vertex is a vertex v such that $\{v\}$is a cut.

Definition:

G is called k-connected if $|V(G)| > k$ and if $G \setminus X$ is connected for every set $X \subseteq V$with $|X| < k|$. In other words, no two vertices of G are separated by fewer than k other vertices. Every (non-empty) graph is 0-connected and the 1-connected graphs are precisely the non-trivial connected graphs. The greatest integer k such that G is $k -$connected is the connectivity $k(G)$of G. For example, if $G = K_n$, then $k(G) = n - 1$. In the below example, deleting v disconnects G, so v is a cut vertex.



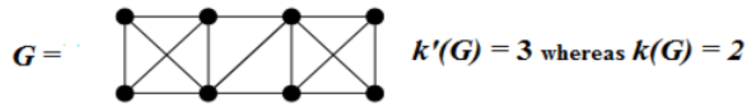
Proposition: For every graph G, $k(G) \leq \delta(G)$.

Proof.

Let $v \in V(G)$ be a vertex of minimum degree $d(v) = \delta(G)$. Then deleting $N(v)$ disconnects v from the rest of G.

Definition: (Edge connectivity)

A disconnecting set of edges is a set $S \subseteq E(G)$ such that $G \setminus F$has more than one component. Given $S, T \subseteq V(G)$ the notation [S, T] specifies the set of edges having one end point in S and the other in T. An edge cut is an edge set of the form [S, $\bar{S}$], where S is a non-empty proper subset of V (G). A graph is k-edge-connected if every disconnecting set has at least k edges. The edge-connectivity of G, written $k^{'}(G)$, is the minimum size of a disconnecting set. One edge disconnecting G is called a bridge. For example, if $G = K_n$, then $k^{'}(G) = n - 1$.



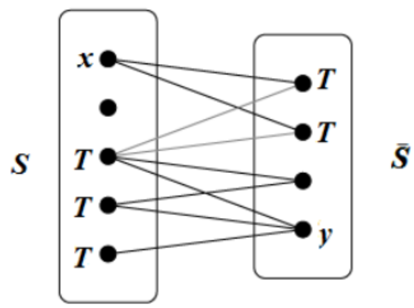
Remark: An edge cut is a disconnecting set but not the other way around. However, everyminimal disconnecting set is a cut.

Theorem:$k(G) \leq k^{'}(G) \leq \delta(G)$.

Proof.

The edges incident to a vertex v of minimum degree, form a disconnecting set, hence $k^{'}(G) \leq \delta(G)$. It remains to show that $k(G) \leq k^{'}(G)$. Suppose$|G| > 1$ and [S, S] is a minimum edge cut, having size $k^{'}(G)$.

If every vertex of S is adjacent to every vertex of $\bar{S}$ and $|G| = |V(G)| = n$, then $k^{'}(G) = |S||\bar{S}| = |S|(|G| - |S|)$. This expression is minimized at $|S| = 1$. By definition,$k(G) \leq |G| - 1$, so the inequality holds.



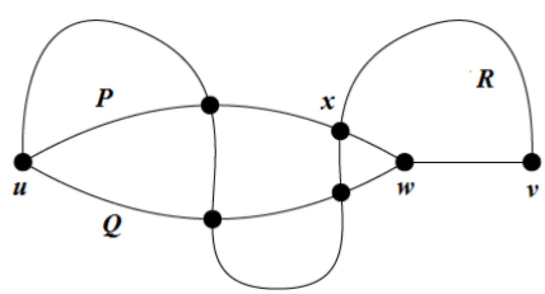
Hence we may assume there exists $x \in S$, $y \in \bar{S}$ with x not adjacent to y. Let T be the vertex set consisting of all neighbors of x in S and all vertices of $S \backslash x$ that have neighbours in S (illustrated below). Deleting T destroys all the edges in the cut [S, S] (but does not delete x or y), so T is a separating set. Now, by the definition of T we can injectively associate at least one edge of $[S,\bar{S}\ ]$ to each vertex in T, so .$k(G) \leq |T| \leq |[S,\bar{S}]| = k^{'}(G)$.

Definition: Two paths are internally disjoint if neither contains a non-endpoint vertex of the other. We denote the length of the shortest path from u to v (the distance from u to v) by d(u, v).

Theorem: (Whitney 1932).A graph G having at least three vertices is 2-connected if and only if each pair $u, v \in V(G)$ is connected by a pair of internally disjoint $u, v -$ paths in G.

Proof.

When G has internally disjoint $u, v -$paths, deletion of one vertex cannot separate u from v. Since this is given for every u, v, the condition is sufficient. For the converse, suppose that G is2-connected. We prove by induction on $d(u, v)$ that G has two internally disjoint u, v paths. When $d(u, v) = 1$, the graph $G \setminus (u, v)$ is connected, since $k^{'}(G) \geq k(G) = 2$. A $u, v -$ path in $G \setminus (u, v)$ is internally disjoint in G from the $u, v -$path consisting of the edge (u, v) itself.



For the induction step, we consider $d(u,v) = k > 1$ and assume that G has internally disjoint x, y −paths whenever $1 \leq d(x,y) \leq k$. Let w be the vertex before v on a shortest u, v −path. We have $d(u,w) = k - 1$, and hence by the induction hypothesis G has internally disjoint u, w − paths P andQ. Since $G \setminus w$ is connected, $G \setminus w$ contains a u, v −path R. If this path avoids P or Q, we are finished, but R may share internal vertices with both P and Q. Let x be the last vertex of R belonging to $P \cup Q$. Without loss of generality, we may assume, $x \in P$. We combine the u, x −subpath of P with the x, v −subpath of R to obtain a u, v −path internally disjoint from $Q \cup \{(w, v)\}$.

Corollary: G is 2-connected and$|V(G)| \geq 3$ if and only if every two vertices in G lie on a common cycle.

EULERIAN AND HAMILTONIAN PATHS

Definition: A trail is a walk with no repeated edges.

Definition: An Eulerian trail in a graph $G = (V, E)$ is a walk in G passing through every edge exactly once. If this walk is closed (starts and ends at the same vertex) it is called an Eulerian tour.

Theorem: A connected graph has an Eulerian tour if and only if each vertex has even degree.
In order to prove this theorem we use the following lemma.

Lemma: Every maximal trail in a graph where all the vertices have even degree is a closed trail.

Proof.

Let T be a maximal trail. If T is not closed, then T has an odd number of edges incident to the final vertex v. However, as v has even degree, there is an edge incident to v that is not in T. This edge can be used to extend T to a longer trail, contradicting the maximality of T.

Proof of Theorem

To see that the condition is necessary, suppose G has an Eulerian tour C. If a vertex v was visited k times in the tour C, then each visit used 2 edges incident to v (one in comingedge and one outgoing edge). Thus, $d(v) = 2k$, which is even.

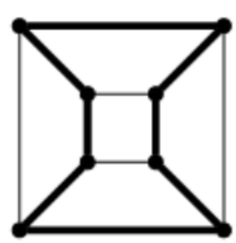
To see that the condition is sufficient, let G be a connected graph with even degrees. Let $T = e_1 e_2 \ldots e_l$ (where $e_i = (v_{i-1}, v_i)$)be a longest trail in G. Then, by Lemma, T is closed, that is, $v_0 = v_l$. If T does not include all the edges of G then, since G is connected, there is an edge outside of T such that $e = (u, v_i)$ for some vertex v_i in T. But then $T' = e e_{i+1} \ldots e_l e_1 e_2 \ldots e_i$ is a trail in G which is longer than T, contradicting the fact that T is a longest trail in G. Thus, we conclude that T includes all the edges of G and so it is an Eulerian tour.

HAMILTON PATHS AND CYCLES

Definition: A Hamilton path/cycle in a graph G is a path/cycle visiting every vertex of G exactly once. A graph G is called Hamiltonian if it contains a Hamilton cycle.

Hamilton cycles were introduced by Kirkman in 1985, and were named after Sir William Hamilton, who produced a puzzle whose goal was to find a Hamilton cycle in a specific graph.

Example: Hamilton cycle in the skeleton of the 3-dimensional cube.

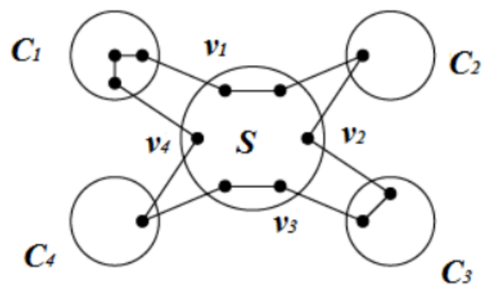


Proposition 5.3.

Theorem: If G is Hamiltonian then for any set $S \subseteq V(G)$ the graph $G \setminus S$has at most $|S|$connected components.

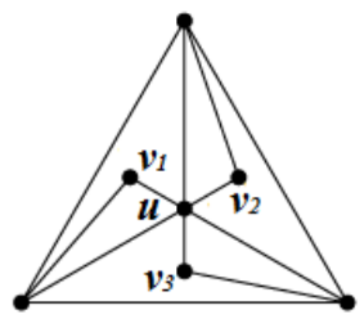
Proof.

Let $C_1, C_2, \ldots, C_k$ be the components of $G \setminus S$. Imagine that we are moving along a Hamilton cycle in some order, vertex-by-vertex (in the picture below, we are moving clockwise, starting from some vertex in C_1, say). We must visit each component of $G \setminus S$at least once, when we leaveC_i forthe first time, let v_i be the subsequent vertex visited (which must be in S). Each v_i must be distinct because a cycle cannot intersect itself. Hence, S must have at least as many vertices as the number of connected components of $G \setminus S$.



Example:

The condition in Proposition is not sufficient to ensure that a graph is Hamiltonian. The graph G above satisfies the condition of Proposition, but is not Hamiltonian. Indeed, one would need to include all the edges incident to the vertices v_1, v_2 and v_3 in a Hamiltoncycle of G, however, in that case the vertex u would have degree at least 3 in that Hamilton cycle, which is impossible. We also give some sufficient conditions for Hamiltonicity.



Theorem: (Dirac 1952). If G is a simple graph with $n \geq 3$ vertices and if $\delta(G) \geq \frac{n}{2}$, then G is Hamiltonian.

Proof.

The condition that $n \geq 3$ must be included since K_2 is not Hamiltonian but satisfies $\delta(G) = \frac{|K_2|}{2}$. If there is a non-Hamiltonian graph satisfying the hypotheses, then adding edges cannot reduce the minimum degree, so we may restrict our attention to maximal non-Hamiltonian graphs G with minimum degree at least $\frac{n}{2}$. By "maximal" we mean that for every pair (u, v) of non-adjacent vertices of G, the graph obtained from G by adding the edge $e = (u, v)$ is Hamiltonian.

The maximality of G implies that G has a Hamilton path, say from $u = v_1$, to $v = v_n$, because every Hamilton cycle in $G \cup \{e\}$ must contain the new edge e. We use most of this path $v_1, v_2, \ldots, v_n$with a small switch, to obtain a Hamilton cycle in G. If some neighbor of u immediately follows a neighbor of v on the path, say (u; vi+1) ∈ E(G) and (v; vi) ∈ E(G), then G has the Hamilton cycle$(u, v_{i+1}, v_{i+2}, \ldots, v_{n-1}, v, v_i, v_{i-1}, \ldots, v_2)$ shown below.

To prove that such a cycle exists, we show that there is a common index in the sets S and T defined by $S = \{i : (u, v_{i+1} \in E(G)\}$ and T= $\{i : (v, v_i \in E(G)\}$. Summing the sizes of these sets, yields$|S \cup T| + |S \cap T| = |S| + |T| = d(u) + d(v) \geq n$. Neither S nor T contains the index n. This implies that $|S \cup T| < n$, and hence $|S \cap T| \geq 1$, as required. This is a contradiction.

It can be observed that this argument uses only that $d(u) + d(v) \geq n$. Therefore, we can weaken the requirement of minimum degree $\frac{n}{2}$ to require only that $d(u) + d(v) \geq n$ whenever *u* is not adjacent to *v*.

Theorem: (Ore 1960). If G is a simple graph such that for every pair of non-adjacent vertices u, v of G we have $d(u) + d(v) \geq n$, then G is Hamiltonian.

BINARY TREES

Definition: A directed tree is a directed graph whose underlying graph is a tree.

Definition: A rooted tree is a tree with a designated vertex called the root. Each edge is implicitly directed away from the root.

Definition: In a rooted tree, the depth or level of a vertex v is its distance from the root, that is, the length of the unique path from the root to v. Thus, the root has depth 0.

Definition: The height of a rooted tree is the length of a longest path from the root (or the greatest depth in the tree).

Definition: If vertex v immediately precedes vertex w on the path from the root to w, then v is parent of w and w is child of v.

Definition: Vertices having the same parent are called siblings.

Definition: A vertex w is called a descendant of a vertex v (and v is called an ancestor of w), if v is on the unique path from the root to w.

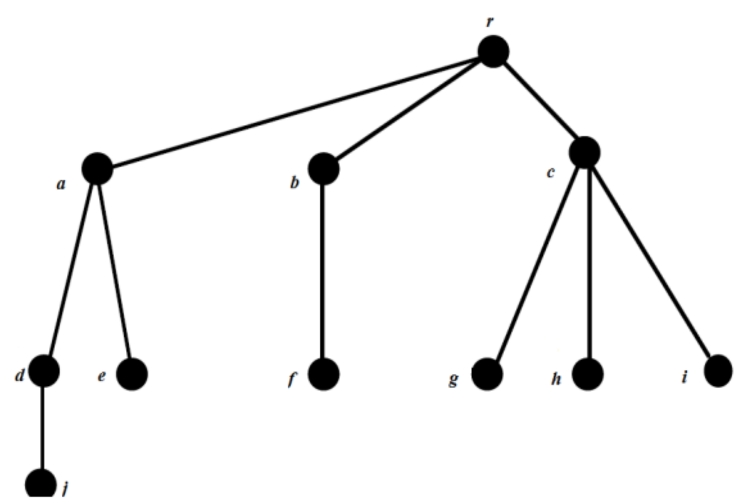
Definition: A leaf in a rooted tree is any vertex having no children.

Definition: An internal vertex in a rooted tree is any vertex that has at least one child. The root is internal, unless the tree is trivial (that is, a single vertex).

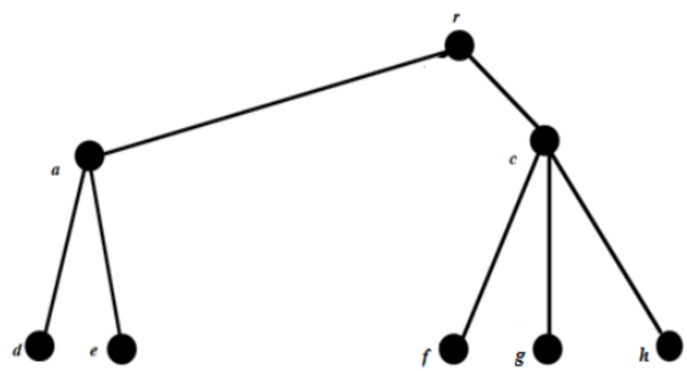
Example: The height of the tree given in the figure below is 3. Also r, a, b and d are the internal vertices; vertices e, f, g, h, i, and j are the leaves; vertices g, h and i are siblings; vertex a is an ancestor of j; and j is a descendant of a.

Definition : An ordered tree is a rooted tree in which the children of each vertex are assigned a fixed ordering.

Definition: In a standard plane drawing of an ordered tree, the root is at the top, the vertices at each level are horizontally aligned, and the left-to-right order of the vertices agrees with their prescribed order.



Definition: A binary tree is an ordered 2-ary tree in which each child is designated either a left-child or a right-child. A binary tree of height 2 is shown below.



Definition: The left (right) subtree of a vertex v in a binary tree is the binary subtree spanning the left (right)-child of v and all of its descendants.

Theorem: The complete binary tree of height h has $2^{h+1} - 1$ vertices.

Corollary: Every binary tree of height h has at most $2^{h+1} - 1$vertices.

Expression Trees

An expression tree is a special type of a binary tree that represents an algebraic expression in such a way that stores its structure and shows how the order of operations applies. This is a very important type of a tree in computer science. We're interested in a few different operators. We break these operators down into two categories:

- Binary Operators - operators that take two inputs
    - +
    - - (here, subtraction)
    - *
    - / (both integer and floating-point division)
    - % (modulus)
    - ^ or ** (exponentiation)

- Unary Operators - operators that take one input
    - - (here, negation)

Note that we don't mention parentheses. The expression tree's structure removes the need to talk about parentheses, as the structure encodes precedence.

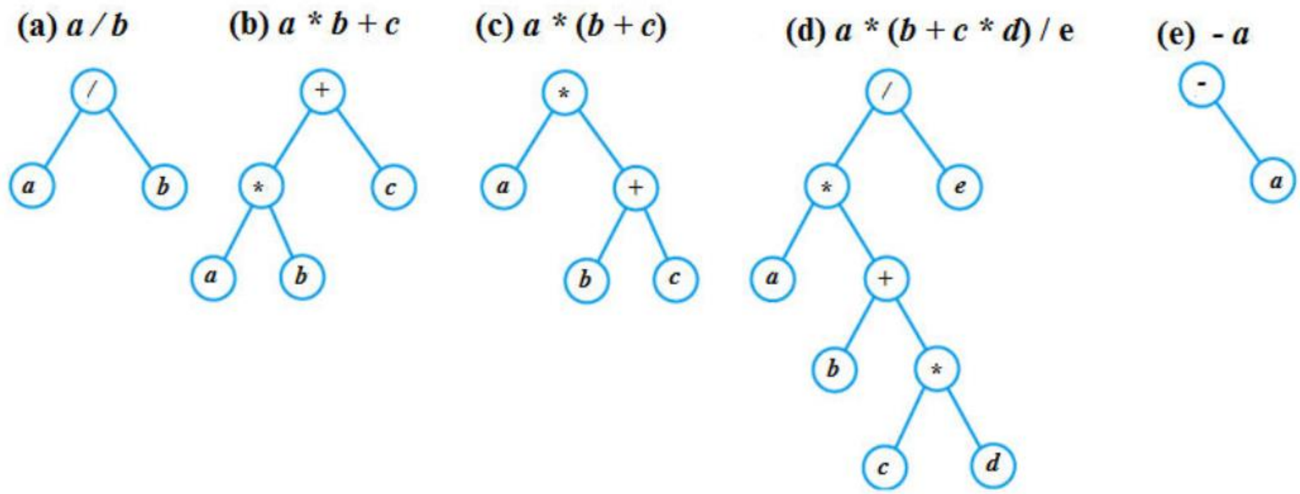
When we have a single expression based on a binary operator, we draw the expression tree as follows:

- The operator is the root of the tree.
- The operands are the children. Because some operations are *not* commutative, order does matter. The operand before the operator is the left child and the operand after the operator is the right child. Thus, we get a tree with a root and two children. For example see figure (a).

When we have a single expression based on a unary operator, we draw the expression tree as follows:

- The operator is the root of the tree.
- The operand is the child.

Thus, we get a tree with a root and one child. (It's really more of a linear structure than a tree, but it does fit the definition of a tree. We'll find that these kinds of trees are interesting when we join them together as part of more complicated expressions.) The Expression tree for –a is in figure (e). Note that we could treat negation as multiplication by -1 and eliminate the need for unary trees if we'd like to have all nodes in our tree having exactly 2 children (or no child). When we wish to work with more complicated expressions, we invoke the recursive nature of binary trees. When an operand is an expression rather than a single variable or constant, we simply put the expression tree for that expression in lieu of theoperand. Figures (b), (c) and (d) are examples of such expression trees.


(a) a / b
(b) a * b + c
(c) a * (b + c)
(d) a * (b + c * d) / e
(e) - a