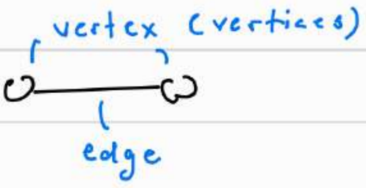
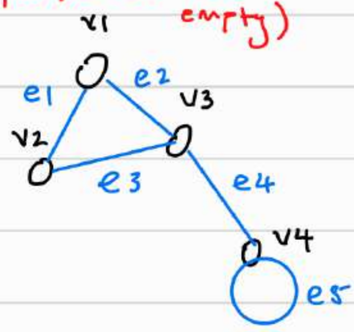
Graphs



- shows r/s b/w objects (structure)
- 2 finite sets of vertices $V(G)$ & edges $E(G)$
  - $V(G)$: nodes
  - $E(G)$: connections.

(minimum 1 vertex for graph, edge can be empty)

eg:



$V(G) = \{v_1, v_2, v_3, v_4\}$

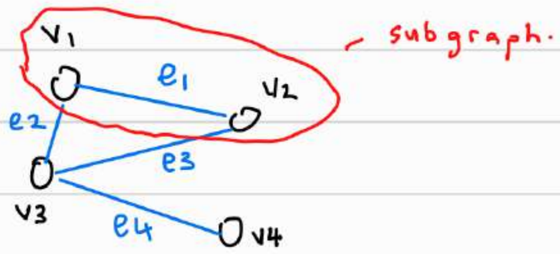
$E(G) = \{e_1, e_2, e_3, e_4, e_5\}$

* Each edge associated with at least 1 vertex

Subgraphs

- smaller graph, part of a larger graph.

eg:

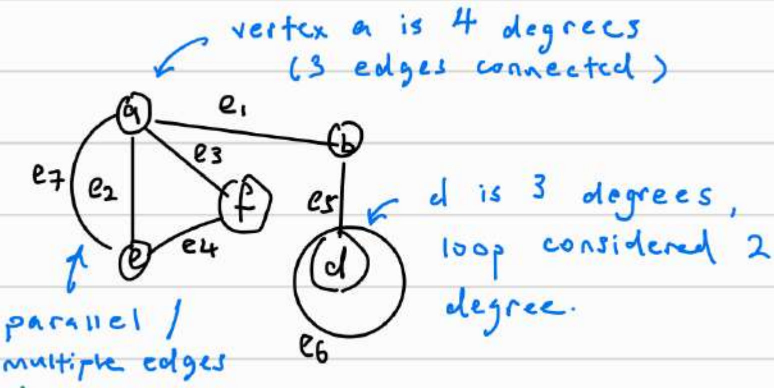


suppose subgraph = H, main graph = G.

- every vertex in H also in G
- every edge in H also in G
- every edge in H same endpoint as G

Degree (of vertex)

- How many edges connected to vertex
- loop edges considered 2 degrees
- if vertex by itself = 0. (isolated vertex)



Degree sequence. (1 for each vertex)

- Lay out degrees from smallest to largest.

eg: Deg. seq. of $G = (0, 2, 2, 3, 3, 4)$ ← degree for each vertex in G.
(G ← graph G)

Parallel edges

edges with same endpoint (vertices)

Overall / Total degrees = sum of degree of vertices.

Sum of Degrees of Vertices theorem (Handshake theorem)

Notation: $|V|$: sum of vertices. (order of graph). 4 vertices = 4th order

$|E|$: sum of edges. (size of graph). 5 edges, size = 5.

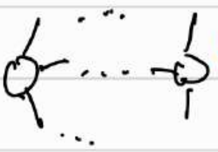
Theorem: Total degree of graph = $\boxed{2|E|}$ of graph. (twice sum of edges).

Corollary 1: Total degrees of graph is even

Corollary 2: In any graph, there are even numbers of vertices with odd degrees such that sum of degrees always even.

* if corollary violated, said graph will not be possible to construct.

eg:



← odd degree by itself but must have even numbers of odd degree vertices.

Maximum Degree ΔG : vertex with highest degree. (May be odd num)

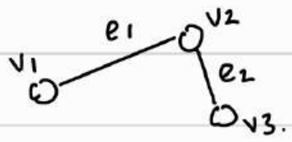
Minimum Degree δG : vertex with lowest degree (May be odd num).

Adjacency

- 2 vertices connected by edge
- 2 edges sharing vertex are adjacent.

Incidence

An edge is incident on its endpoint.



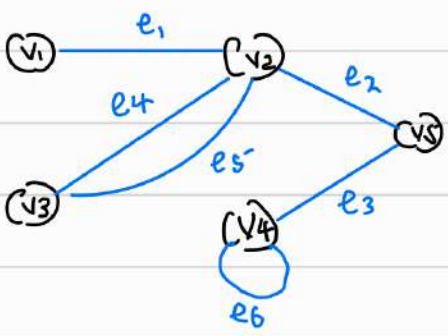
e_1 is incident on V_1 & V_2

e_2 is incident on V_2 & V_3

Adjacency Matrix

- A way to represent graphs, especially if graph is very large.

eg: if 5 vertices, create 5 x 5 matrix



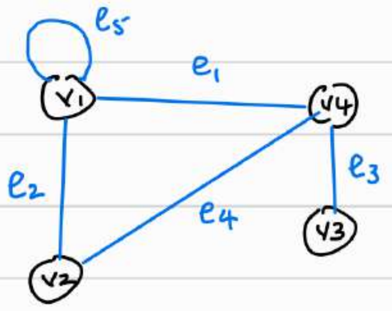
No. of edges joining V_1 & V_1

$$A = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

- Easy to input to computer for calculation.

Incidence matrix

- use both edges & vertices to create matrix



notation for incidence matrix

e_1 incident on V_1

loop are 2 degrees

$$M = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Isomorphism

- Graphs may look different in shape but if :
  1. same number of vertices
  2. same no. of edges
  3. edges have same endpoint

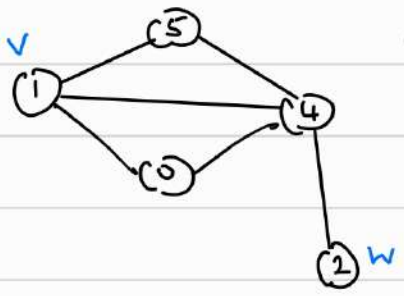
  } Graphs known as isomorphic (look different but essentially same).

* Label name of vertices & edges does not affect isomorphism.

Walk

- movement from 1 vertex, v to another vertex, w (can have repeated edge / vertex)

eg:



- multiple walks possible.

1, 0, 4, 2

1, 4, 2

1, 5, 4, 2

Length of walk : How many edges for given walk.

eg: $1 \to 4 \to 2$ = 2 edges

Open walk : start & end at <u>different</u> vertex

closed walk : start & end at <u>same</u> vertex

Trail

- A walk from v to w without a repeated edge.

Path

- A walk from v to w w/o repeated edge & vertex

Distance of Path

denoted $d(v, w)$: shortest path b/w 2 vertices

Circuit

- A trail that has at least 1 edge & start & end on same vertex

Eccentricity (of vertex)

- maximum distance a particular vertex can have (from another vertex)
- denoted $ecc(v)$, v is vertex.

Diameter (of graph)

- maximum eccentricity of graph, denoted $diam(G)$

Radius (of graph)

- minimum eccentricity of graph, denoted $rad(G)$

if $ecc(v) = diam(G)$, v = peripheral vertex

if $ecc(v) = rad(G)$, v = central vertex

Connectedness

connectedness of vertices:

- 2 vertices are connected if ∃ walk b/w them.

connectedness of graphs:

- All vertex in graph connected.

Disconnecting set : set of edges where, upon removal, causes graph to be disconnected

- can be multiple sets : $\{g\}, \{f, e, d\}, \{h, i\}$

Bridge: disconnecting set with size 1.

Edges connectivity $\lambda(G)$: Minimum no. of edges to delete to make graph disconnected.

Separating set : set of vertices where, upon removal, causes graph to be disconnected

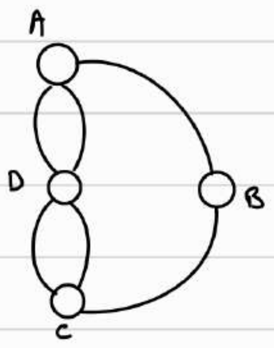
* if vertex deleted, any edge connected to vertex will also be deleted.

eg: $\{4\}, \{4, 6\}$

$K(G)$

vertex connectivity : minimum no. of vertex to delete to make graph disconnected

Euler trails & Circuits



is it possible to pass all vertex while only traversing each edge only once?

Euler trail : A trail that visits every edge exactly once.

1) if all but 2 vertices have even degrees, graph has an Euler trail. (i.e require 2 odd degree vertices).

2) Have to start & end at odd vertex

, every edge only once.

Euler Circuit: Euler trail that starts & ends on same vertex.

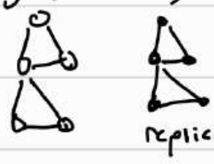
1) Every vertex must be even to have euler circuit.
- can have repeated vertices but not edge.

Eulerian graph: connected graph with all vertex even.

Fleury's Algorithm

- Algo to find Euler trail / circuits (esp. for big graphs)

1) create replica of existing graph.
2) choose any start vertex
3) traverse any avail edge from chosen vertex.
4) delete edge in replica graph.
   ↳ if deleting edge makes graph disconnected, choose another edge
5) repeat step 3 & 4 until Euler circuit created.



replica.

if want Euler trail, same steps but start & end on odd vertex

Hamiltonian Circuit

- visit every vertex exactly once.
- starts & stops on same vertex
- can have repeat edges but not vertices

Hamiltonian path

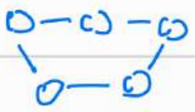
- visit every vertex exactly once
- start & stop on different vertex

No criteria to meet unlike Eulerian circuit / path.
all vertex (even degrees etc.)

Ore's theorem

- lays out a sufficient condition for a graph to be hamiltonian.
- if conditions met, graph will be hamiltonian.
- does not mean that if graph does not have these conditions then not hamiltonian.

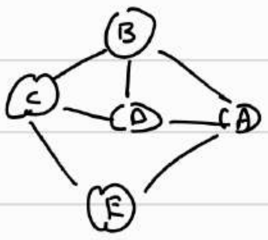
eg:



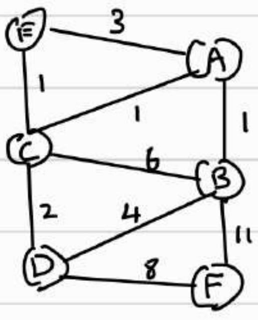
conditions:

1. simple graph: No more than 1 edge b/w 2 vertices, no reflex edge.
2. $n \geq 3$, n = no. of vertices
3. $\deg(v) + \deg(w) \geq n$, v & w non adjacent vertices.
   ↳ list down all non adjacent vertices

B, E : 5
A, C : 6
D, E : 5



Shortest Path



weighted graph: each edge have value.

- Dijkstra's algorithm.

1. label start & end vertex.
2. Assign start vertex = 0, all other vertices = ∞
3. Begin travel from start vertex, update ∞ values with new lowest weight value.
4. Repeat step 3 until all possible path traversed (note down vertex traversed when lower weight obtained)