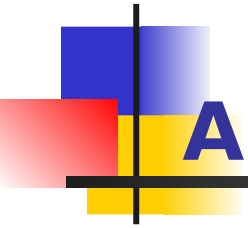


Introductory Class

Applied Graph Theory and Algorithms

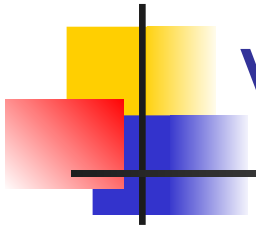
Course Code: CSC4066

Dr. Hasin A. Ahmed

Assistant Professor

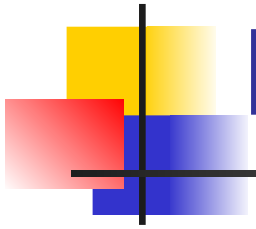
Department of Computer Science

Gauhati University



Varying Applications (examples)

- Computer networks
- Distinguish between two chemical compounds with the same molecular formula but different structures
- Solve shortest path problems between cities
- Scheduling exams and assign channels to television stations



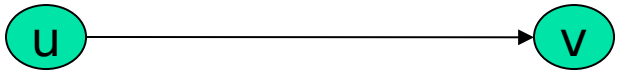
Definitions - Graph

A generalization of the simple concept of a set of dots, links, edges or arcs.

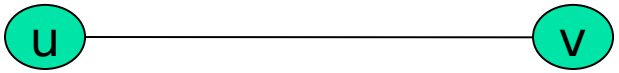
Representation: Graph G =(V, E) consists set of vertices denoted by V, or by V(G) and set of edges E, or E(G)

Definitions – Edge Type

Directed: Ordered pair of vertices. Represented as (u, v) directed from vertex u to v.

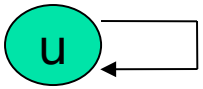


Undirected: Unordered pair of vertices. Represented as {u, v}. Disregards any sense of direction and treats both end vertices interchangeably.



Definitions – Edge Type

- **Loop:** A loop is an edge whose endpoints are equal i.e., an edge joining a vertex to it self is called a loop. Represented as {u, u} = {u}

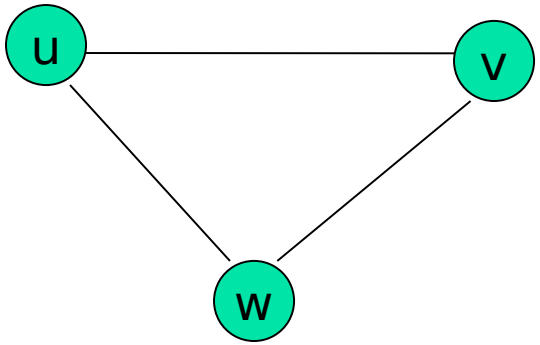


- **Multiple Edges:** Two or more edges joining the same pair of vertices.

Definitions – Graph Type

Simple (Undirected) Graph: consists of V, a nonempty set of vertices, and E, a set of unordered pairs of distinct elements of V called edges (undirected)

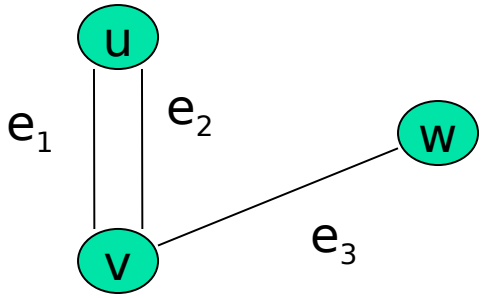
Representation Example: G(V, E), V = {u, v, w}, E = {{u, v}, {v, w}, {u, w}}



Definitions – Graph Type

Multigraph: G(V,E), consists of set of vertices V, set of Edges E and a function f from E to {{u, v}| u, v V, u ≠ v}. The edges e1 and e2 are called multiple or parallel edges if f (e1) = f (e2).

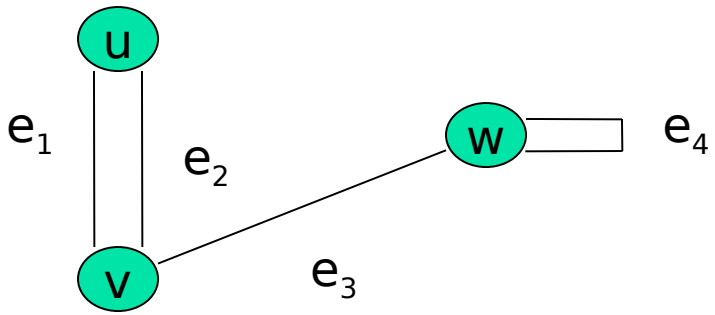
Representation Example: V = {u, v, w}, E = {e_1, e_2, e_3}



Definitions – Graph Type

Pseudograph: G(V,E), consists of set of vertices V, set of Edges E and a function F from E to {{u, v}| u, v Î V}. Loops allowed in such a graph.

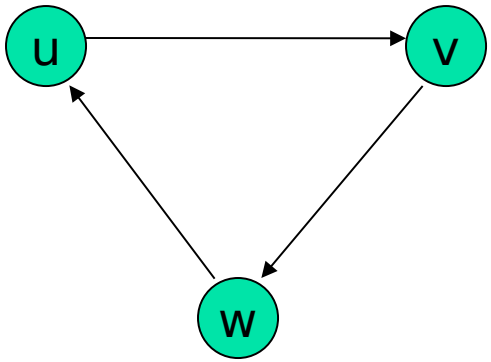
Representation Example: V = {u, v, w}, E = {e_1, e_2, e_3, e_4}



Definitions – Graph Type

Directed Graph: G(V, E), set of vertices V, and set of Edges E, that are ordered pair of elements of V (directed edges)

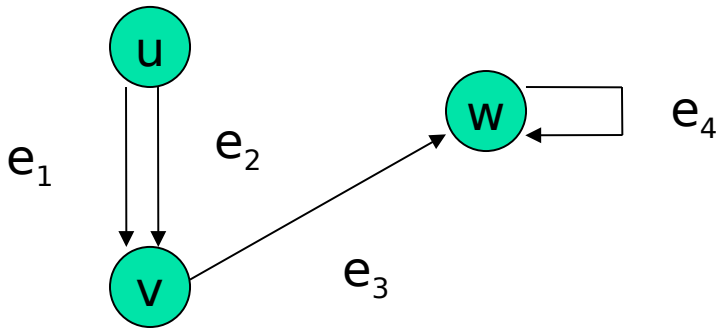
Representation Example: G(V, E), V = {u, v, w}, E = {(u, v), (v, w), (w, u)}



Definitions – Graph Type

Directed Multigraph: G(V,E), consists of set of vertices V, set of Edges E and a function f from E to {{u, v}| u, v V}. The edges e1 and e2 are multiple edges if f(e1) = f(e2)

Representation Example: V = {u, v, w}, E = {e_1, e_2, e_3, e_4}



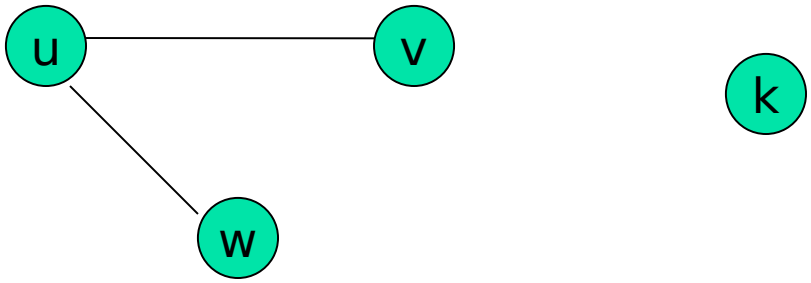
Definitions – Graph Type

Type	Edges	Multiple Edges Allowed ?	Loops Allowed ?
Simple Graph	undirected	No	No
Multigraph	undirected	Yes	No
Pseudograph	undirected	Yes	Yes
Directed Graph	directed	No	Yes
Directed Multigraph	directed	Yes	Yes

Terminology – Undirected graphs

- u and v are **adjacent** if {u, v} is an edge, e is called **incident** with u and v. u and v are called **endpoints** of {u, v}
- **Degree of Vertex (deg (v)):** the number of edges incident on a vertex. A loop contributes twice to the degree (why?).
- **Pendant Vertex:** deg (v) =1
- **Isolated Vertex:** deg (k) = 0

Representation Example: For V = {u, v, w} , E = { {u, w}, {u, v} }, deg (u) = 2, deg (v) = 1, deg (w) = 1, deg (k) = 0, w and v are pendant , k is isolated

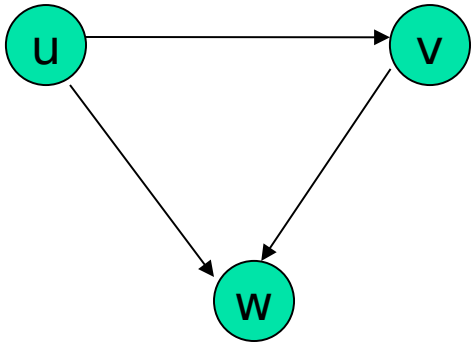


Terminology – Directed graphs

- For the edge (u, v), u is **adjacent to** v OR v is **adjacent from** u, u – **Initial vertex**, v – **Terminal vertex**
- **In-degree (deg^- (u)):** number of edges for which u is terminal vertex
- **Out-degree (deg^+ (u)):** number of edges for which u is initial vertex

Note: A loop contributes 1 to both in-degree and out-degree (why?)

Representation Example: For V = {u, v, w} , E = { (u, w), (v, w), (u, v) }, deg^- (u) = 0, deg^+ (u) = 2, deg^- (v) = 1,
deg^+ (v) = 1, and deg^- (w) = 2, deg^+ (u) = 0



Theorems: Undirected Graphs

Theorem 1

The Handshaking theorem:

$$2e = \sum_{v \in V} \deg(v)$$

(why?) Every edge connects 2 vertices

Theorems: Undirected Graphs

Theorem 2:

An undirected graph has even number of vertices with odd degree

Proof $V1$ is the set of even degree vertices and V2 refers to odd degree vertices

$$2e = \sum_{v \in V} \deg(v) = \sum_{u \in V_1} \deg(u) + \sum_{v \in V_2} \deg(v)$$

$\Rightarrow$ $\deg(v)$ is even for $v \in V_1$,

$\Rightarrow$ The first term in the right hand side of the last inequality is even.

$\Rightarrow$ The sum of the last two terms on the right hand side of the last inequality is even since sum is 2e.

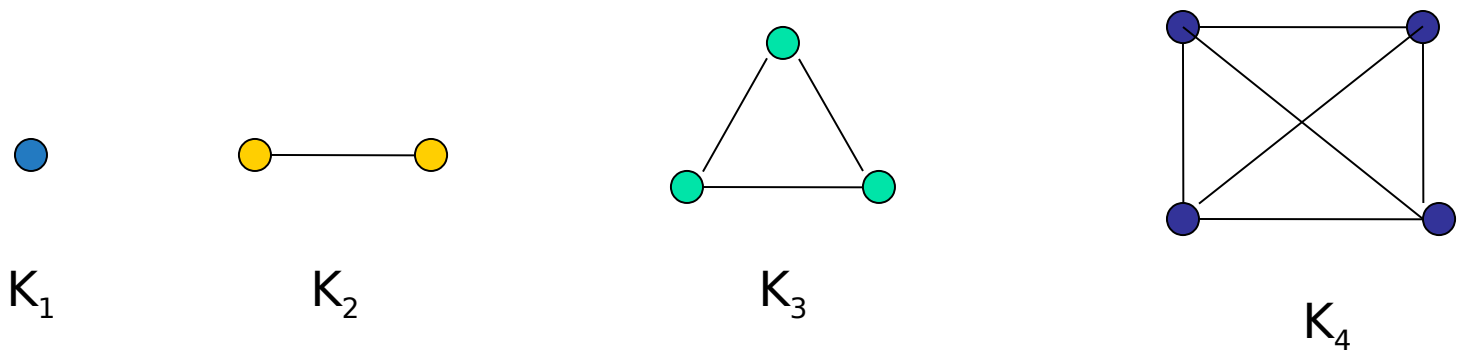
Hence second term is also even

$\Rightarrow$ second term $\sum_{v \in V_2} \deg(u) = even$

Simple graphs – special cases

- **Complete graph:** K_n, is the simple graph that contains exactly one edge between each pair of distinct vertices.

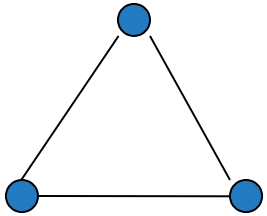
  Representation Example: K_1, K_2, K_3, K_4



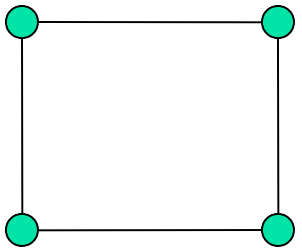
Simple graphs – special cases

- **Cycle:** C_n, $n \geq 3$ consists of n vertices v_1, v_2, v_3 … v_n and edges $\{v_1, v_2\}$, $\{v_2, v_3\}$, $\{v_3, v_4\}$ … $\{v_{n-1}, v_n\}$, $\{v_n, v_1\}$

  Representation Example: C_3, C_4



C_3



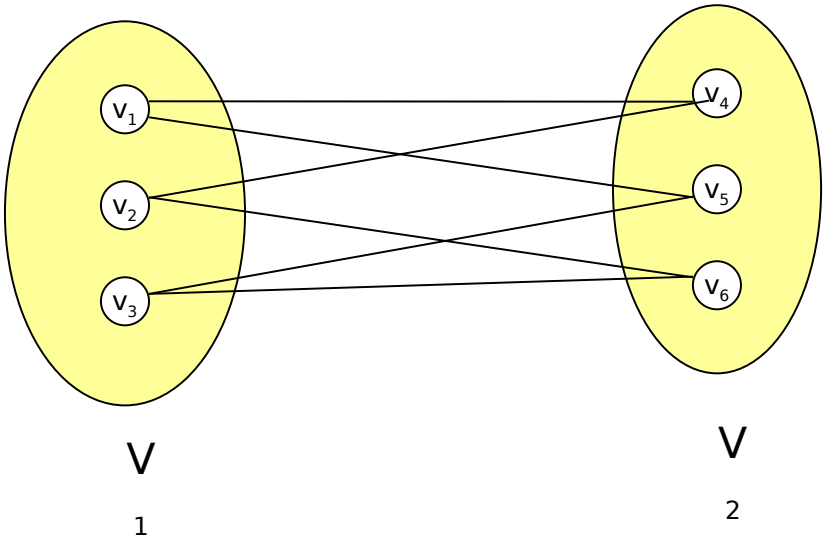
C_4

Bipartite graphs

- In a simple graph G, if V can be partitioned into two disjoint sets V_1 and V_2 such that every edge in the graph connects a vertex in V_1 and a vertex V_2 (so that no edge in G connects either two vertices in V_1 or two vertices in V_2)

  Application example: Representing Relations

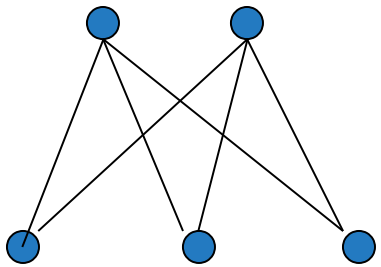
  Representation example: $V_1 = \{v_1, v_2, v_3\}$ and $V_2 = \{v_4, v_5, v_6\}$,



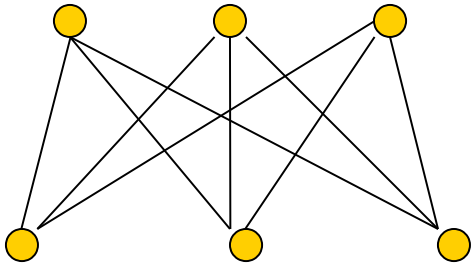
Complete Bipartite graphs

- $K_{m,n}$ is the graph that has its vertex set portioned into two subsets of m and n vertices, respectively There is an edge between two vertices if and only if one vertex is in the first subset and the other vertex is in the second subset.

  Representation example: $K_{2,3}$, $K_{3,3}$



$K_{2,3}$



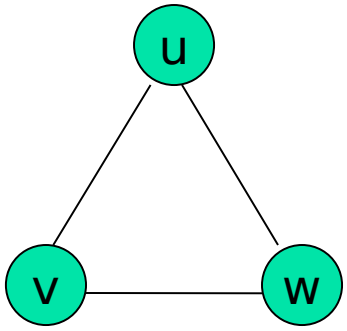
$K_{3,3}$

Subgraphs

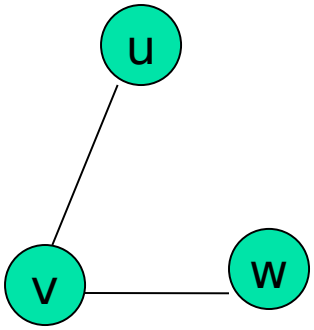
- A subgraph of a graph G = (V, E) is a graph H =(V', E') where V' is a subset of V and E' is a subset of E

  Application example: solving sub-problems within a graph

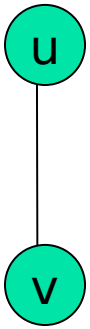
  Representation example: V = {u, v, w}, E = ({u, v}, {v, w}, {w, u}}, H_1 , H_2



G



H_1

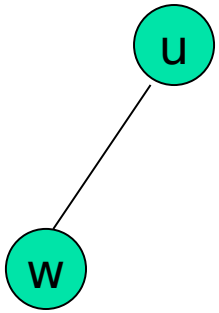


H_2

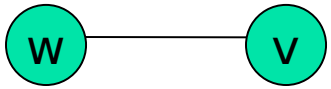
Subgraphs

- G = G1 U G2 wherein E = E1 U E2 and V = V1 U V2, G, G1 and G2 are simple graphs of G

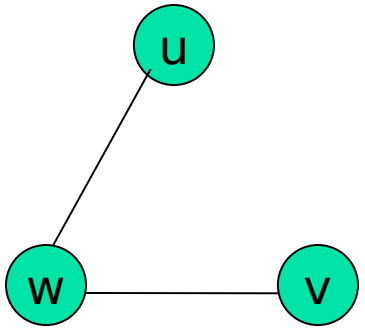
Representation example: V1 = {u, w}, E1 = {{u, w}}, V2 = {w, v}, E1 = {{w, v}}, V = {u, v ,w}, E = {{{u, w}, {{w, v}}



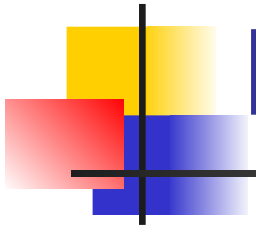
G1



G2



G



Representation

- **Incidence (Matrix):** Most useful when information about edges is more desirable than information about vertices.

- **Adjacency (Matrix/List):** Most useful when information about the vertices is more desirable than information about the edges. These two representations are also most popular since information about the vertices is often more desirable than edges in most applications

Representation- Incidence Matrix

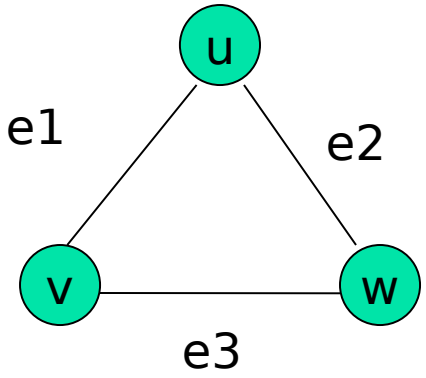
- G = (V, E) be an unditected graph. Suppose that $v_1, v_2, v_3, \ldots, v_n$ are the vertices and $e_1, e_2, \ldots, e_m$ are the edges of G. Then the incidence matrix with respect to this ordering of V and E is the nx m matrix $M = [m_{ij}]$, where

Can also be used to represent :
Multiple edges: by using columns with identical entries, since these edges are incident with the same pair of vertices
Loops: by using a column with exactly one entry equal to 1, corresponding to the vertex that is incident with the loop

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

Representation- Incidence Matrix

- Representation Example: G = (V, E)



	e_1	e_2	e_3
v	1	0	1
u	1	1	0
w	0	1	1

Representation- Adjacency Matrix

- There is an N x N matrix, where |V| = N , the Adjacenct Matrix (NxN) A = $[a_{ij}]$

For undirected graph

$$a_{ij} = \begin{cases} 1 \text{ if } \{v_i, v_j\} \text{ is an edge of } G \\ 0 \text{ otherwise} \end{cases}$$

- **For directed graph**

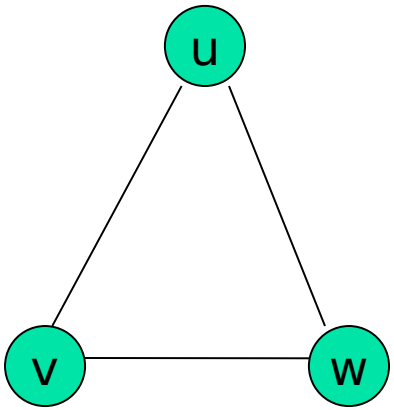
$$a_{ij} = \begin{cases} 1 \text{ if } (v_i, v_j) \text{ is an edge of } G \\ 0 \text{ otherwise} \end{cases}$$

- This makes it easier to find subgraphs, and to reverse graphs if needed.

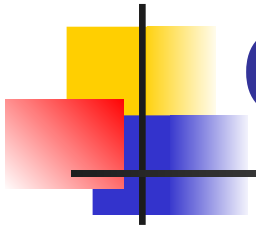
Representation- Adjacency List

Each node (vertex) has a list of which nodes (vertex) it is adjacent

Example: undirectd graph G (V, E)



node	Adjacency List
u	v , w
v	w, u
w	u , v



Graph - Isomorphism

- G1 = (V1, E2) and G2 = (V2, E2) are isomorphic if:
- There is a one-to-one and onto function f from V1 to V2 with the property that
  - a and b are adjacent in G1 if and only if f (a) and f (b) are adjacent in G2, for all a and b in V1.
- Function f is called isomorphism

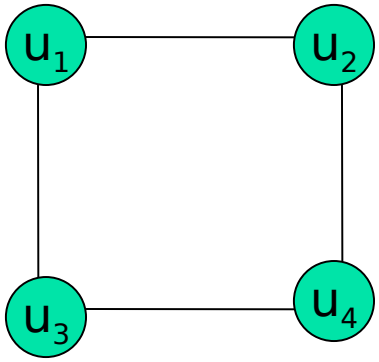
Application Example:

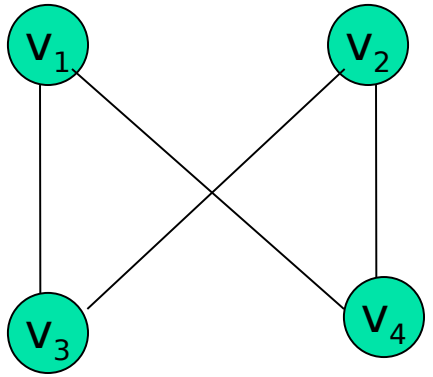
In chemistry, to find if two compounds have the same structure

Graph - Isomorphism

Representation example: G1 = (V1, E1) , G2 = (V2, E2)

$f(u_1) = v_1$, $f(u_2) = v_4$, $f(u_3) = v_3$, $f(u_4) = v_2$,

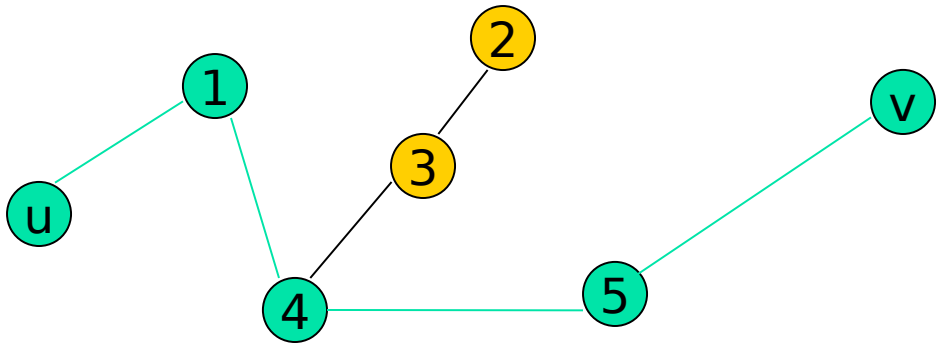




Connectivity – Path

A **Path** is a sequence of edges that begins at a vertex of a graph and travels along edges of the graph, always connecting pairs of adjacent vertices.

Representation example: G = (V, E), Path P represented, from u to v is {{u, 1}, {1, 4}, {4, 5}, {5, v}}

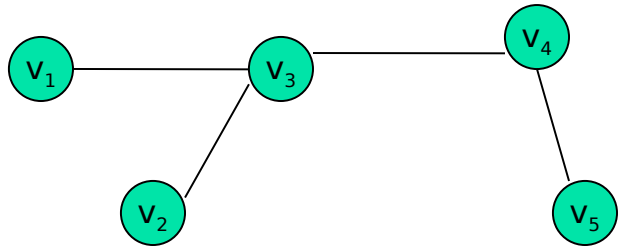


Connectivity – Connectedness

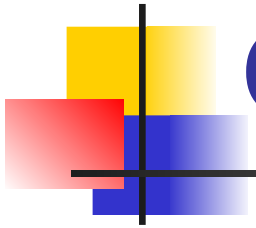
Undirected Graph

An undirected graph is connected if there exists is a simple path between every pair of vertices

Representation Example: G (V, E) is connected since for V = {v_1, v_2, v_3, v_4, v_5}, there exists a path between {v_i, v_j}, 1 ≤ i, j≤ 5



Connectivity – Connectedness



Directed Graph

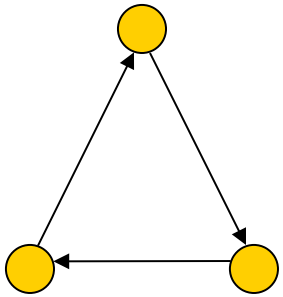
- A directed graph is **strongly connected** if there is a path from a to b and from b to a whenever a and b are vertices in the graph
- A directed graph is **weakly connected** if there is a (undirected) path between every two vertices in the underlying undirected path

A strongly connected Graph can be weakly connected but the vice-versa is not true (why?)

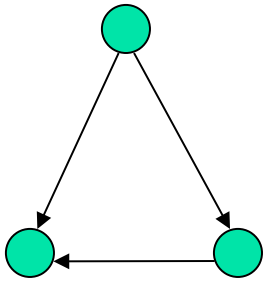
Connectivity – Connectedness

Directed Graph

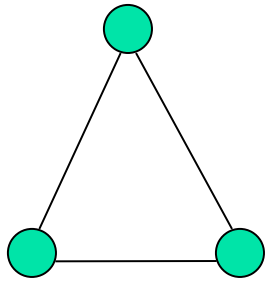
Representation example: G1 (Strong component), G2 (Weak Component), G3 is undirected graph representation of G2 or G1



G1



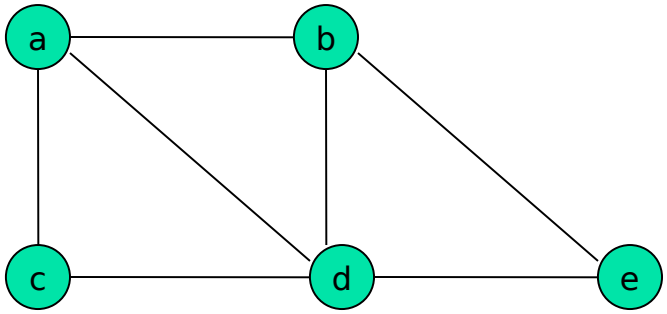
G2



G3

Euler - definitions

- An **Eulerian path** (**Eulerian trail**, **Euler walk**) in a graph is a path that uses each edge precisely once. If such a path exists, the graph is called **traversable**.
- An **Eulerian cycle** (**Eulerian circuit**, **Euler tour**) in a graph is a cycle that uses each edge precisely once. If such a cycle exists, the graph is called **Eulerian** (also **unicursal**).
- Representation example: G1 has Euler path a, c, d, e, b, d, a, b



Hamiltonian Graph

- **Hamiltonian path** (also called *traceable path*) is a path that visits each vertex exactly once.

- A **Hamiltonian cycle** (also called *Hamiltonian circuit*, *vertex tour* or *graph cycle*) is a cycle that visits each vertex exactly once (except for the starting vertex, which is visited once at the start and once again at the end).

- A graph that contains a Hamiltonian path is called a **traceable graph**. A graph that contains a Hamiltonian cycle is called a **Hamiltonian graph**. Any Hamiltonian cycle can be converted to a Hamiltonian path by removing one of its edges, but a Hamiltonian path can be extended to Hamiltonian cycle only if its endpoints are adjacent.

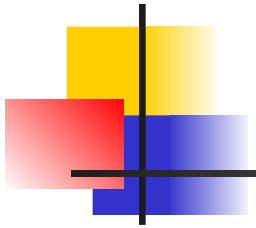
Shortest Path

- Generalize distance to weighted setting
- Digraph $G = (V,E)$ with weight function W: $E \rightarrow R$ (assigning real values to edges)
- Weight of path $p = v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k$ is

$$w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$$

- Shortest path = a path of the minimum weight
- Applications
  - static/dynamic network routing
  - robot motion planning
  - map/route generation in traffic

Shortest-Path Problems



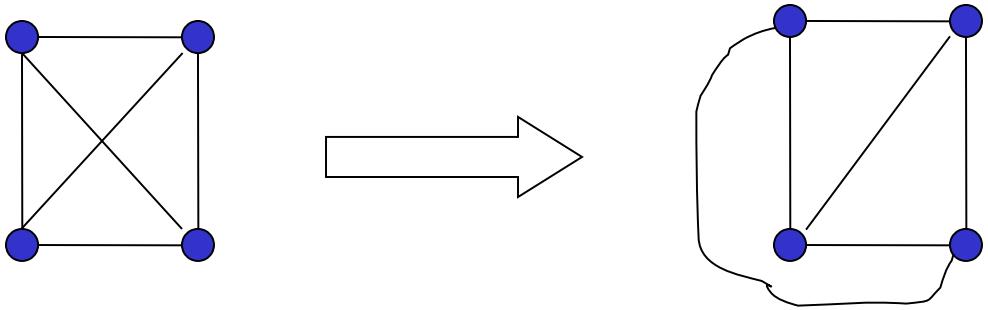
- Shortest-Path problems
  - **Single-source (single-destination).** Find a shortest path from a given source (vertex *s*) to each of the vertices. The topic of this lecture.
  - **Single-pair.** Given two vertices, find a shortest path between them. Solution to single-source problem solves this problem efficiently, too.
  - **All-pairs.** Find shortest-paths for every pair of vertices. Dynamic programming algorithm.
  - Unweighted shortest-paths – BFS.

Planar Graphs

- A graph (or multigraph) *G* is called *planar* if *G* can be drawn in the plane with its edges intersecting only at vertices of *G*, such a drawing of *G* is called an *embedding* of *G* in the plane.

Application Example: VLSI design (overlapping edges requires extra layers), Circuit design (cannot overlap wires on board)

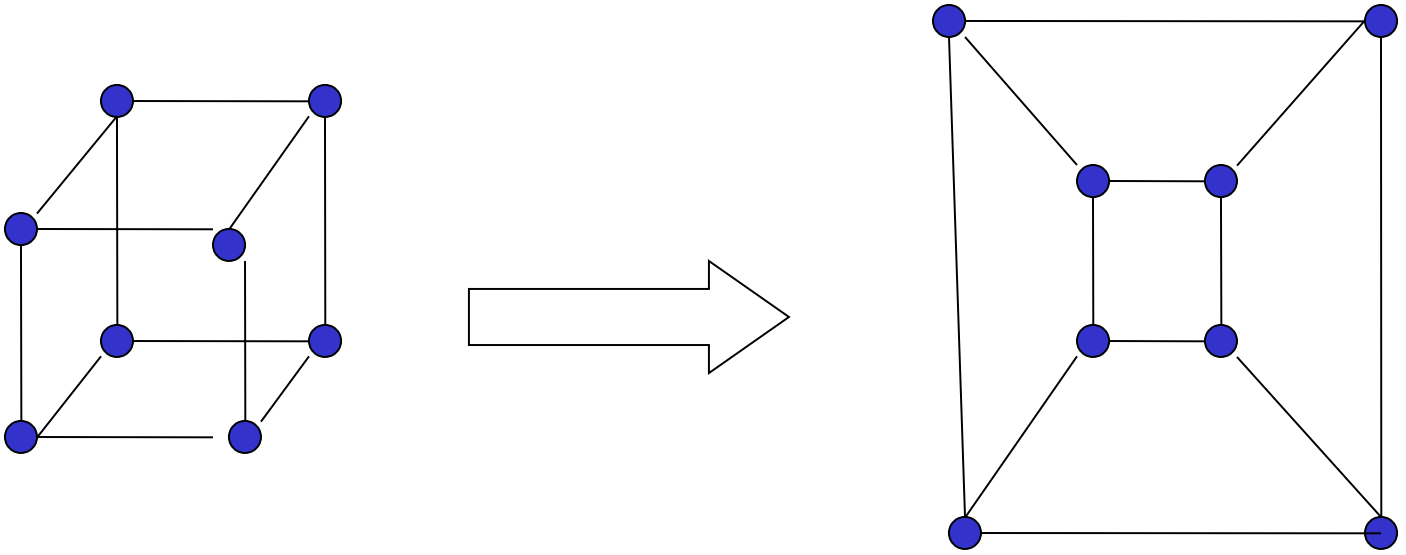
Representation examples: *K*1,*K*2,*K*3,*K*4 are planar, *Kn* for *n*>4 are non-planar



K_4

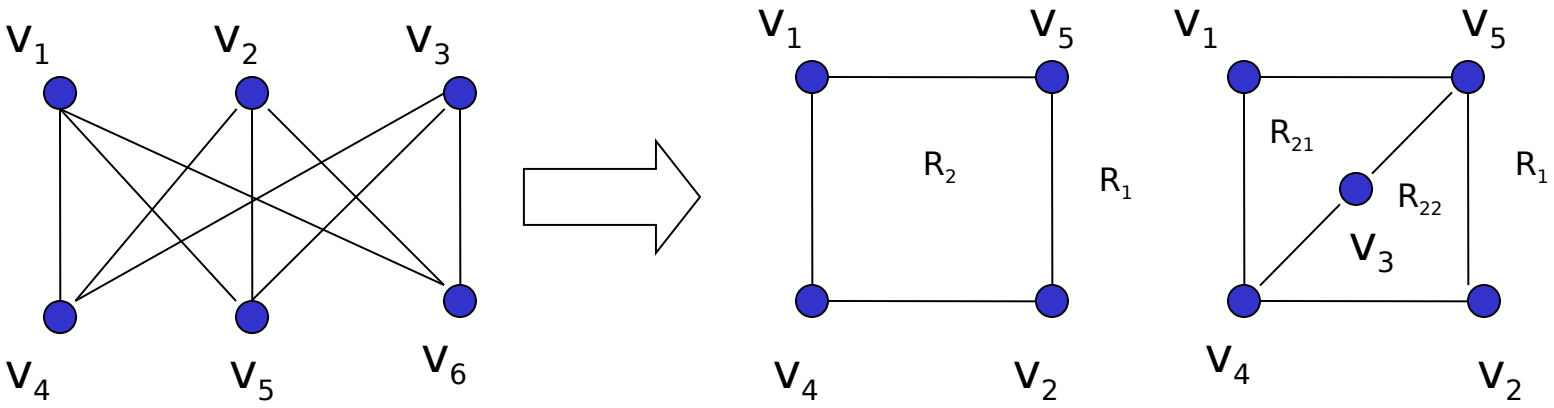
Planar Graphs

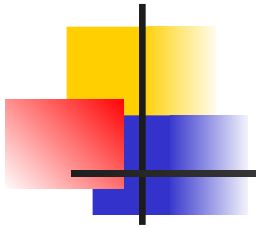
- Representation examples: Q_3



Planar Graphs

- Representation examples: $K_{3,3}$ is Nonplanar





Thank you