COMP1215 Foundations of Computer Science

A short introduction to graph theory

Andersen Ang

ECS, Uni. Southampton, UK

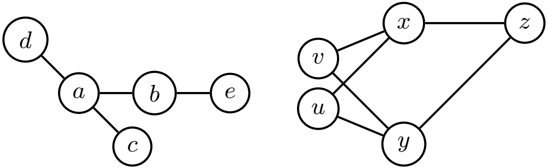
`andersen.ang@soton.ac.uk`

Homepage `angms.science`

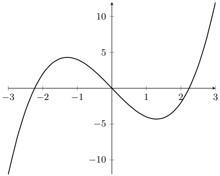
Version: November 25, 2023
First draft: May 24, 2023

The what-why-how of graph

- **What is graph**: "graph" in graph theory is not the about plotting $y = f(x)$



graphs in graph theory

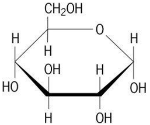


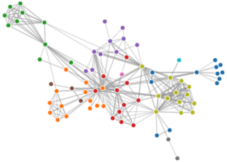
not the graph in graph theory

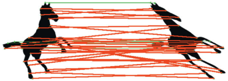
- **Why graph**: a useful modelling "language" about connectivity

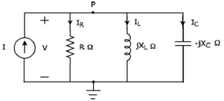
- **How to graph**: we use set, linear algebra and combinatorics to describe graph

Bahmanyar
Omar Khayyam
Saraf al-Dīn Muhammad al-Masʿūdī al-Marwazī
Kamal al-Din Yunus
Nasir al-Din al-Tusi
Shams al-Din al-Bukhari
Gregory Chioniadis
Manuel Bryennios
Theodore Metochites
Gregory Palamas
Nilos Kabasilas
Demetrios Kydones
Georgios Plethon Gemistos
Basilios Bessarion
Janus Lascaris
Marco Musuro
Giovanni Battista della Monte
Bassiano Landi
Theodor Zwinger
Petrus Ryff
Emmanuel Stupanus
Nikolaus Eglinger
Johann Bernoulli
Leonhard Euler
Joseph-Louis Lagrange
Siméon Poisson
Joseph Liouville
Eugène Charles Catalan
Charles Hermite
Henri Poincaré
Théophile De Donder
Théophile Lepage
Paul Pierre Gillis
Jacques Teghem
François Glineur
Nicolas Gillis
Andersen Ang

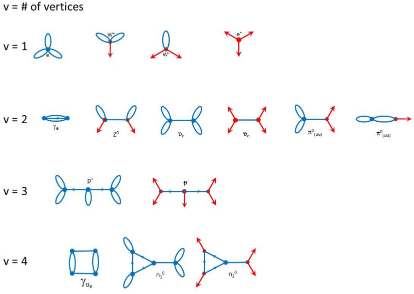


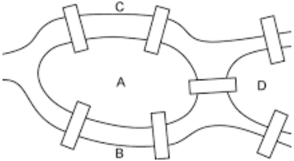


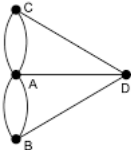


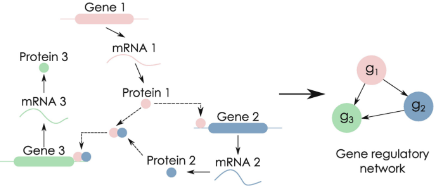


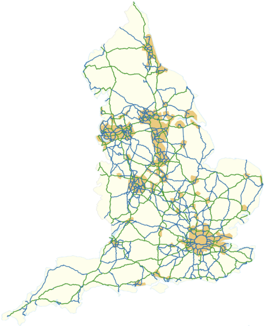
←my academic tree









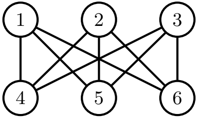


Everything is graph

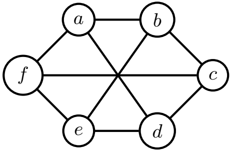
- Graph Theory study relation
- Everything has a relation
- Graph theory is everything

Graph isomorphism

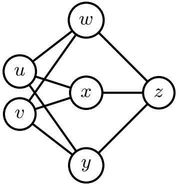
G_1



G_2



G_3



- These 3 graphs are the SAME: there is a function F that maps them

mapping between G_1 and G_2 $\begin{array}{c} F(a) = 6 \\ F(b) = 3 \\ F(c) = 5 \\ F(d) = 1 \\ F(e) = 4 \\ F(f) = 2 \end{array}$ mapping between G_1 and G_3 $\begin{array}{c} F(u) = 3 \\ F(v) = 1 \\ F(w) = 6 \\ F(x) = 5 \\ F(y) = 4 \\ F(z) = 2 \end{array}$

- Checking graph isomorphism is generally hard
  We actually don't even know how hard it is to check graph isomorphism

- The message: in graph theory $\begin{cases} \text{how we name the nodes doesn't matter} \\ \text{how we draw the lines doesn't matter} \\ \textbf{how the nodes connect matters} \end{cases}$

Pre-course information

- What is graph: connectivity structure
- The fancy name of graph is *1-dimensional CW complex in topology*
- Warning: graph theory is very hard
  - because it is one of the most difficult area in mathematics
  - because it is universal (can be used in everything)
    $\Longrightarrow$ it is important for computer science
  - because you probably have never experienced graph theory it before
- Study material: these lecture slides + $\underbrace{\text{workbook + reading books + watch online video yourself}}_{\text{self learning}}$
- Book
  - *Discrete Mathematics and Its Applications* by Kenneth Rosen — enough for this course
  - *A First Look at Graph Theory* by John Clark and Derek Allan Holton — first 41 pages
  - *Introduction to graph theory* by Douglas West
  - *Graph Theory* by Reinhard Diestel — free but not for first reading
  - *Schaum's Outline of Graph Theory* — for more practise problems
- Outcome: understand the very basics of graph

Prerequisite

- Set
- Matrix
- Combinatorics

Table of Contents

VVEEMAD

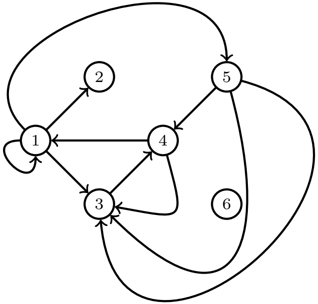
- **Definition** We denote a graph as $G(V, E)$, where V is a set of vertices, and E is a set of edges.

- V is the set of nodes. Here $V = \{1, 2, 3, 4, 5, 6\}$.
  $|V|$ is the number of node in V. I.e. the cardinality of V. Here $|V| = 6$.

- E is the set of edge connecting a pair of node $i, j \in V$.
  - $1 \rightarrow 1$ so we have an edge $(1, 1)$
  - $1 \rightarrow 2$ so we have an edge $(1, 2)$
  - $1 \rightarrow 3$ so we have an edge $(1, 3)$
  - $1 \rightarrow 5$ so we have an edge $(1, 5)$
  - $3 \rightarrow 4$ so we have an edge $(3, 4)$
  - $4 \rightarrow 1$ so we have an edge $(4, 1)$
  - $4 \rightarrow 3$ so we have an edge $(4, 3)$
  - $5 \rightarrow 3$ so we have an edge $(5, 3)_a$
  - $5 \rightarrow 3$ so we have an edge $(5, 3)_b$
  - $5 \rightarrow 4$ so we have an edge $(5, 4)$

$$E = \Big\{(1,1), (1,2), (1,3), (1,5), (3,4), (4,1), (4,3), (5,3)_a, (5,3)_b, (5,4)\Big\}$$

  $|E|$ is the cardinality of E, here $|E| = 10$.



- **Terminology**
  - Vertex is also called node, dots, points
  - Edge is also called arc, curve, line
  - Two edges sharing the same nodes are called *parallel*, e.g., $(5, 3)_a$ and $(5, 3)_b$.
  - $(1, 1)$ is a *self-loop*

VVEEMAD $E = \Big\{(1,1),(1,2),(1,3),(1,5),(3,4),(4,1),(4,3),(5,3)_a,(5,3)_b,(5,4)\Big\}$

- From E, we get incidence information M

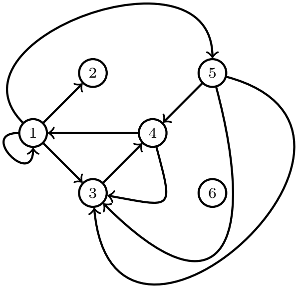
- M can be expressed as **incidence matrix** (preferred by math-people)

$$M \in \{0,1,2,\ldots,|V|\}^{|V|\times|V|}, \text{ where } [M]_{ij} = \begin{cases} 0 & \text{if } (i,j) \notin E \\ 1 & \text{if } (i,j) \in E \\ 2 & \text{if two } (i,j) \in E \\ \vdots & \end{cases}$$

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

`incidenList` =

1	$\rightarrow$	1	2	3	5
2					
3	$\rightarrow$	4			
4	$\rightarrow$	1	3		
5	$\rightarrow$	3	3	4	
6					



- M can be expressed as an **incidence list** (preferred by CS-people)

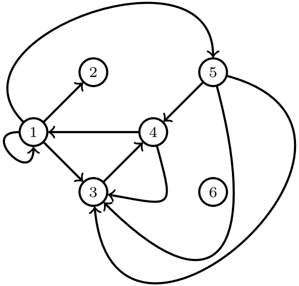
VVEEM**A**D $E = \Big\{(1,1),(1,2),(1,3),(1,5),(3,4),(4,1),(4,3),(5,3)_a,(5,3)_b,(5,4)\Big\}$

- From E we can get A, which stands for "adjacency".
- A can be expressed by an **adjacency matrix** (preferred by math-people)

$$A \in \{0,1,2,\ldots,|V|\}^{|V|\times|V|}, \text{ where } [A]_{ij} = \begin{cases} 0 & \text{if } (i,j) \text{ and } (j,i) \notin E \\ 1 & \text{if } (i,j) \text{ or } (j,i) \in E \\ 2 & \text{if two } (i,j) \text{ or } (j,i) \in E \\ \vdots & \end{cases}$$

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

space (storage) complexity: $\mathcal{O}(|V|^2)$
search (time) complexity: $\mathcal{O}(1)$



- A can be expressed by an **adjacency list** (preferred by CS-people)

`adjList =`

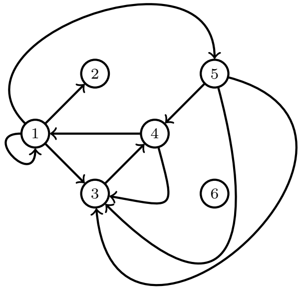
1	–	1	2	3	4	5
2	–	1				
3	–	1	4	4	5	5
4	–	1	3	3	5	
5	–	1	3	3	4	
6						

space (storage) complexity: $\mathcal{O}(|V|+|E|)$
search (time) complexity: $\mathcal{O}(\log |V|)$

VVEEMAD

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

- Indegree: number of edges come to the node
- Outdegree: number of edges leave from the node
- Degree: number of edges touching the node



$$\delta^- = D_{\mathsf{In}} = \begin{bmatrix} 2 &&&&& \\ & 1 &&&& \\ && 4 &&& \\ &&& 2 && \\ &&&& 1 & \\ &&&&& 0 \end{bmatrix}, \ \delta^+ = D_{\mathsf{Out}} = \begin{bmatrix} 4 &&&&& \\ & 0 &&&& \\ && 1 &&& \\ &&& 2 && \\ &&&& 3 & \\ &&&&& 0 \end{bmatrix}$$

$$D = D_{\mathsf{In}} + D_{\mathsf{Out}} = \begin{bmatrix} 6 &&&&& \\ & 1 &&&& \\ && 5 &&& \\ &&& 4 && \\ &&&& 4 & \\ &&&&& 0 \end{bmatrix}$$

- Degree-0 vertex is called **isolated**. E.g. vertex 6
- Degree-1 vertex is called **leaf**, very useful in **tree**. E.g. vertex 2.

First Theorem of Graph theory / Handshaking lemma

- **Theorem (Leonhard Euler, 1736)** For any graph G with $|E|$ edges and $|V| = n$ vertices

$$\text{The sum of degree of all nodes} = \sum_{i=1}^{|V|} \deg_i = 2|E|,$$

  where $\deg_i$ stands for degree of node i.

  Proof Each edge has two end vertices, thus contributes exactly 2 to the sum of the degrees.

- **Handshaking interpretation**
  In a party of n people, the total number of handshakes equals to 2 times the number of handshaked pairs.

- Pigeonhole principle.

All graph has even number of odd vertices

- **Definition** An vertex is called odd (even) if its degree is odd (even).

- **Corollary** In any graph, there is an even number of odd vertices.

  Proof

$$
\begin{aligned}
\sum_{i=1}^{|V|} \deg_i &= 2|E| && \text{Hanshaking lemma} \\
\iff \sum_{i \in \text{odd}} \deg_i + \sum_{i \in \text{even}} \deg_i &= 2|E| && \text{split vertices into odd and even group} \\
\iff \sum_{i \in \text{odd}} \deg_i &= 2|E| - \sum_{i \in \text{even}} \deg_i
\end{aligned}
$$

  Now $2|E| - \sum_{i \in \text{even}} \deg_i$ is an even number

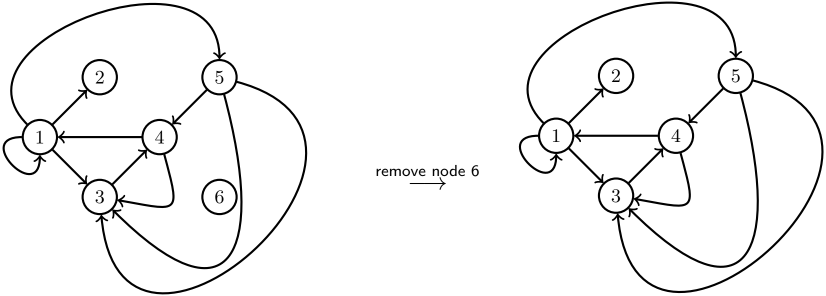
  - $2|E|$ is even
  - $\sum_{i \in \text{even}} \deg_i$ is even because all the node i here has even degree

  For $\sum_{i \in \text{odd}} \underbrace{\deg_i}_{\text{odd}}$ to be even, there must be even number of odd vertices. $\square$

We ignore isolated node(s)

- Vertex 6 is $\begin{cases} \textbf{isolated} \\ \text{has } \textbf{degree} = \mathbf{0} \\ \text{no edge connects to it} \end{cases}$

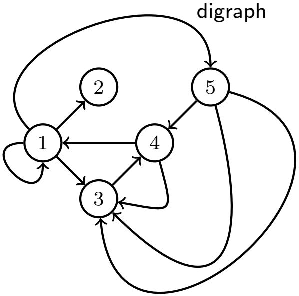
- Graph theory is about **interaction**, no edge $=$ no interaction, so we ignore isolated node

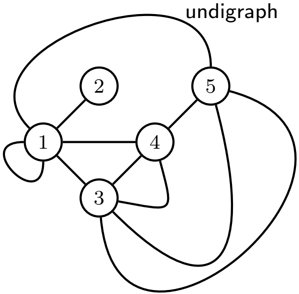


- Graphs that all nodes are isolated are **null graph**, denoted by N

Directed graph (digraph) and undirected graph (undigraph)

- Directed graph: have arrow $(1,2) \neq (2,1)$
- Undirected graph: no arrow $(1,2) = (2,1)$
- Undirected graph is not the same as bidirected graph (out of scope)





Simple graph, multigraph and pseudograph

	Pseudograph	Multigraph	Simple graph
have self loop	ok	no	no
have multiedge	ok	ok	no

- multiedge: edge connecting the same pair
- self loop: edge connecting the same node
- Multigraph can be converted to simple graph $\implies$ we focus on simple graph

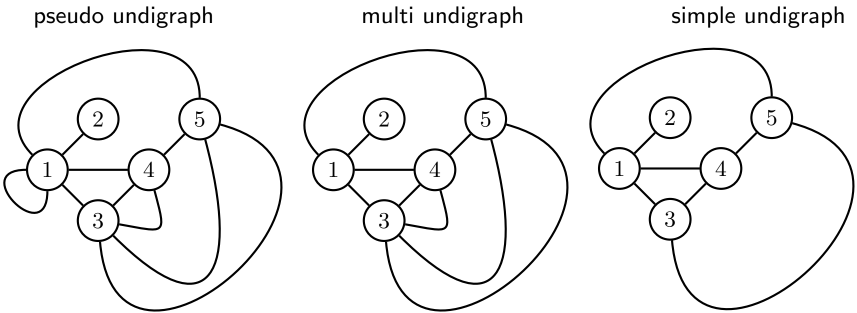
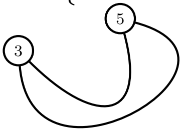


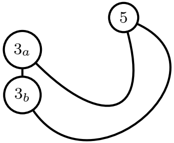
Table of Contents

Converting multigraph to simple graph

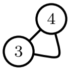
$$E = \Big\{(3,5)_a, (3,5)_b\Big\} \qquad \Longleftrightarrow \qquad E' = \Big\{(3_a,5), (3_b,5), (3_a,3_b)\Big\}$$



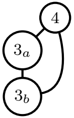
$\xrightarrow{3=\{3_a,3_b\}}$



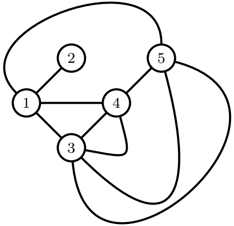
$$E = \Big\{(3,4), (4,3)\Big\} \overset{\text{equal in undigraph}}{=} \Big\{(3,4)_a, (3,4)_b\Big\} \qquad \Longleftrightarrow \qquad E' = \Big\{(3_a,5), (3_b,5), (3_a,3_b)\Big\}$$



$\xrightarrow{3=\{3_a,3_b\}}$



Rename 3_b as 6



$\xrightarrow{\text{simplification}}$

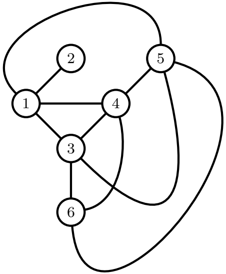


Table of Contents

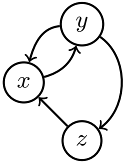
Draw a graph from M

- Consider a directed graph $G(V, E)$ with $M = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$.
- The draw the graph from the given information
  - From the number of rows in M, we know that there are 3 vertices, $|V| = 3$.
    Suppose we name the 3 nodes x, y, z.

$$V = \{x, y, z\}, \quad |V| = 3.$$

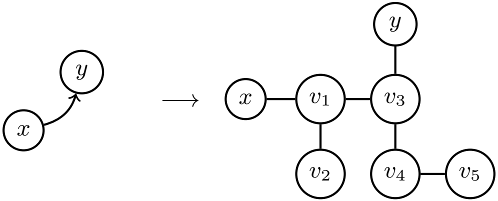
  - From M we know the arrows
  - $M_{1,2} = 1$ means $x \to y$
  - $M_{2,1} = 1$ means $y \to x$
  - $M_{2,3} = 1$ means $y \to z$
  - $M_{3,1} = 1$ means $z \to x$
  - 4 non-zeros in M means $|E| = 4$.

$$E = \{(x, y), (y, x), (y, z), (z, x)\}$$



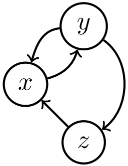
Converting digraph to undigraph

- Consider the edge $e(x, y)$
- Replace $e(x, y)$ by $(x, v_1), (v_1, v_2), (v_1, v_3), (v_3, v_4), (v_4, v_5), (v_3, y)$



- To go back from undigraph to digraph
  - Identify all leaf (degree-1 vertex) $\{x, v_2, v_5, y\}$
  - Find the leaf whose neighbour has degree 2 $\{v_5\}$ because $\deg(v_4) = 2$
  - The neighbour is v_4 and it has neighbour v_3
  - v_3 has a unique neighbour that
    - has degree 3, and
    - adjacent to a leaf

    (this neighbour of v_3 is v_1 and v_1 is adjacent to a leaf v_2)
  - The other neighbour of v_1 is x
  - The other neighbour of v_3 is y
  - Delete $v_1, v_2, ..., v_5$ and connect arrow from x to y



$\iff$

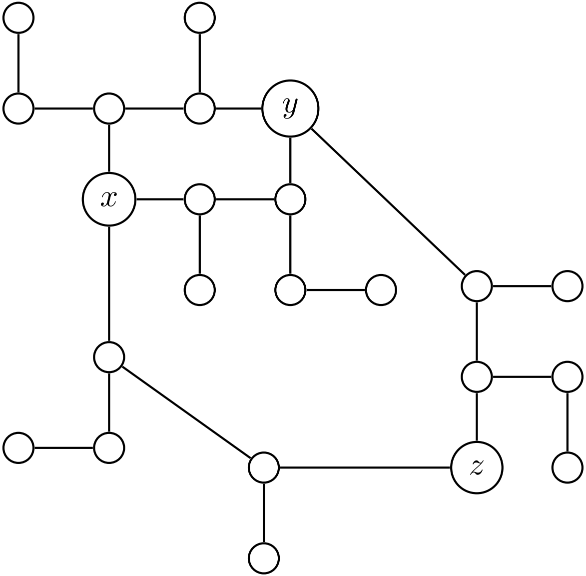
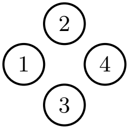


Table of Contents

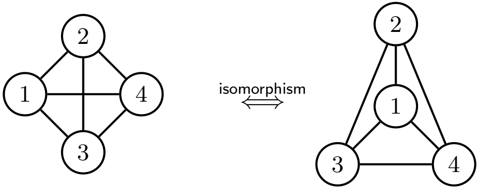
Null graph and complete digraph

- $V = \{1, 2, 3, 4, 5, 6\}$, $|V| = 6$
- $E = \varnothing$, $|E| = 0$
- $M = A = D = 0$
- Null graphs N_n are not interesting



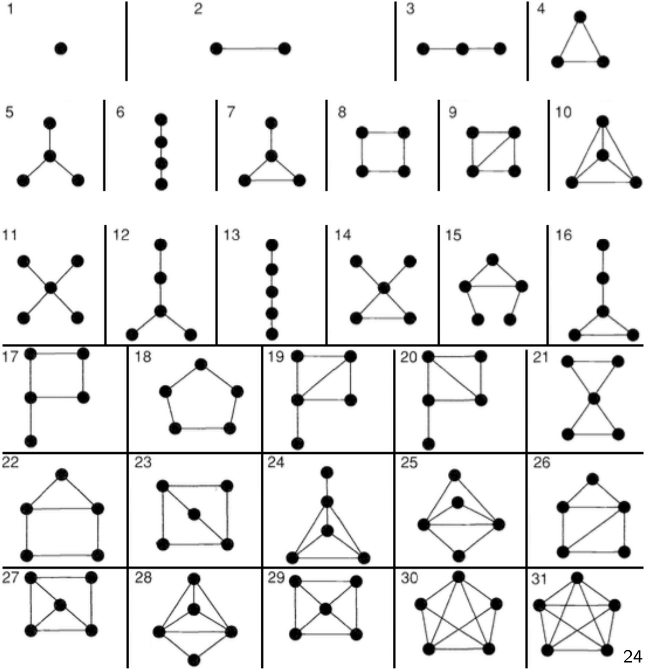
- $V = \{1, 2, 3, 4\}$, $|V| = 4$
- $E = \{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)\}$
- $|E| = 6 = \dfrac{|V|(|V| - 1)}{2} = \dbinom{|V|}{2}$

$$M = A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \; D = \begin{bmatrix} 3 & & & \\ & 3 & & \\ & & 3 & \\ & & & 3 \end{bmatrix}$$



- Complete graph of n vertices are denoted by K_n

Practice: write down the VVEEMAD



Summary

- $V, |V|, E, |E|, M, A, D$
- Direct, undirect, multiedge, self loop, simple
- Converting multigraph to simple graph
- Converting digraph to undigraph — not in exam

From now on we focus mainly on simple undigraph.

Table of Contents

Subgraph

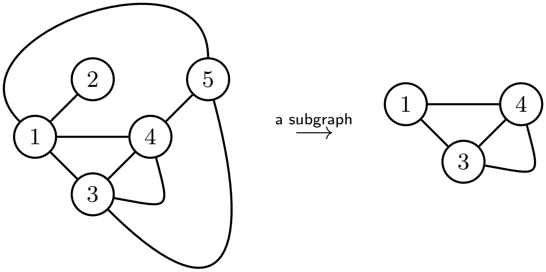
- Like subset in set, we can define subgraph of a graph

  A subgraph of a graph $G(V)$, is a graph $S(U, F)$ that $U \subseteq V, F \subseteq E$.

  Basically, S can be obtained from G by deleting edges and/or vertices.

- **Trivial fact**: every graph is a subgraph of itself.

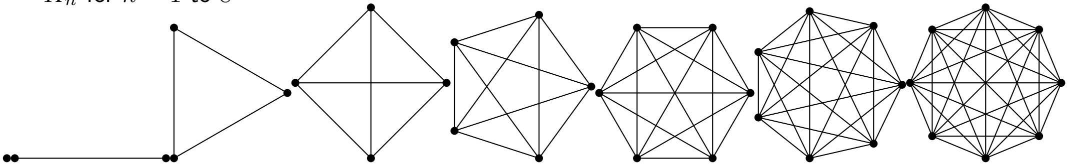
- **Example**



- Other set operations also carry over to graph
  - Intersection
  - Union
  - Complement

The two extreme simple graphs: null graph and complete graph

- Consider simple graph G with $|V| = n$.
- **Null graph** N_n is G with the smallest possible $|E|$ $\quad |E| = 0$
- **Complete graph** K_n is G with the largest possible $|E|$ $\quad |E| = \frac{|V|(|V|-1)}{2} = \binom{|V|}{2}$
  - A complete graph is a graph in which all pair of vertices is joined by an edge
  - K_n for $n = 1$ to 8



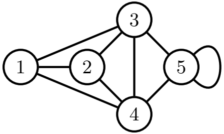
- All graphs with n nodes are between N_n and K_n, and we can define sparse and dense graph as
  - A graph is *sparse* if $|E| \ll \mathcal{O}\Big(\frac{|V|(|V|-1)}{2}\Big) = \mathcal{O}(|V|^2)$
  - A graph is *dense* if $|E| \approxeq \mathcal{O}(|V|^2)$
  - Recall Big-O notation $f(x) = \mathcal{O}(g(x))$ if $f(x) \leq Mg(x)$ for sufficiently large x

Walk, trail, path, cycle

- **Definition** A **walk** is a sequence $v_0 \to v_2 \to \dots v_m$ in a graph.
  - v_0: initial vertex/ source
  - v_m: final vertex/ sink
  - The number of edges in a walk, $m+1$, is called its length
- **Example** $1 \to 2 \to 3 \to 4 \to 5 \to 5 \to 4 \to 2$ is a length-7 walk.
  Walk can also be expressed using sequence of edge

$$W = \Big\{(1,2),(2,3),(3,4),(4,5),(5,5),(5,4),(4,2)\Big\}$$

  We say this walk *traverses* the edges in W



- **Definition** A **trail** is a walk if distinct edges.

$$\text{e.g. } \Big\{(1,2),(2,3),(3,4),(4,5),(5,5)\Big\}$$

  Another definition of trail: a walk that traverse each edge *at most once*.

	walk	trail	path
edge repeat	ok	no	no
vertex repeat	ok	ok	no

- **Definition** A **path** is a trail if distinct vertices except possibly source = sink

$$\text{e.g. } \Big\{(1,2),(2,3),(3,4),(4,5)\Big\}$$

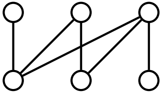
  Another definition of path: a trail that traverse each vertex *at most once*.
- **Definition** A **cycle** is a closed path (source = sink)

$$\text{e.g. } \Big\{(1,3),(3,4),(4,1)\Big\}$$

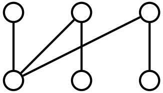
Other structure: circuit, clique, line, ... not in exam

- Circuit: a closed trial
  - Circuit vs cycle: disregard starting and ending nodes
    - Circuit allows repeated nodes
    - Cycle does not allow repeated nodes
  - Eulerian: a circuit consists of a closed path that visits every edge of a graph exactly once
  - Hamiltonian: a circuit that visits every node of a graph exactly once.
- Clique: a set C of G that all pair of distinct nodes in C are adjacent.
  - the subgraph induced by a clique is a complete graph
- Line graph 
- Petersen graph
- Boundary of graph
- Surface area of graph

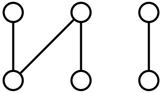
Connectivity, forest, tree and bipartite



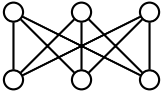
not forest not tree
(there is a cycle 4-2-5-3-4)
bipartite



tree
forest (1 tree)
bipartite

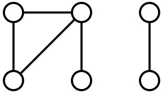


not tree
forest
bipartite



complete bipartite $K_{3,3}$

- A graph is **connected** if for any $i \in V, j \in V, i \neq j$, there is a path connecting i to j.
- **Definition** A **forest** is an acyclic graph. Acyclic = no cycle
- **Definition** A **tree** is an acyclic connected graph. Acyclic = no cycle
- **Definition** A **bipartite** is graph that vertices can be divided into two parts such that there is no edges within each part.
- **Theorems**
  - Every tree is bipartite
  - A graph is bipartite if and only if it has no subgraph that has an odd-length cycle.



not bipartite

- Application of bipartite: assignment problem, stable marriage problem (Not in exam)

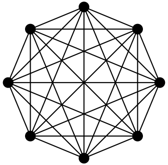
Summary

	repeated nodes	repeated edge	open/closed
Walk	Y	Y	Both
Trail	Y	N	O
Path	N	N	O
Cycle	N	N	C
Circuit	Y	N	C

Term	definition
Connected	there is a path for any (i, j)
Forest	acyclic graph
Tree	acyclic connected graph
Bipartite	vertices can be divised into two parts with no edges within each part

Table of Contents

How many k-cycles are there in a simple complete graph K_n?



Question: how many triangles are there in K_8?

Theorem There are $\frac{1}{2k}\frac{n!}{(n-k)!}$ length-k cycles in K_n.

Proof

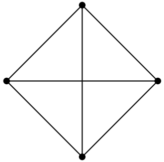
$\binom{n}{k}$ number of ways to choose k node among n vertices

$(k-1)!$ the number of orderings in the selected k-set

2 number of orientation of the cycle (clockwise and anticlockwise)

$\frac{\binom{n}{k}(k-1)!}{2}$ by product rule and division rule

$$\frac{\binom{n}{k}(k-1)!}{2} = \frac{\frac{n!}{(n-k)!k!}(k-1)!}{2} = \frac{1}{2k}\frac{n!}{(n-k)!}$$



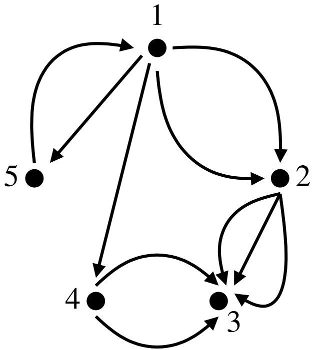
There are $\frac{1}{2\cdot 3}\frac{8!}{(8-3)!} = 56$ triangles in K_8. Similarly, there are $\frac{1}{2\cdot 3}\frac{4!}{(4-3)!} = 4$ triangles in K_4.

Table of Contents

Combinatorics, graph theory and matrix walk into a bar ...

- **Theorem** Given the adjacent matrix of a graph $G(V, E)$. The number of length-k walks starting from vertex i to vertex j is $(A^k)_{ij}$.
- **Proof by mathematical induction**
  - Base case: for $k = 1$, $A^k_{ij} = A_{ij}$ is the number of length-1 walk from i to j
  - Hypothesis case: assume the statement is true at case $k = n$.
    I.e., the number of length-k walks starting from vertex i to vertex j is $(A^k)_{ij}$.
  - Inductive step: consider the case $k = n + 1$.
    - Consider $A^{n+1} = A^n A$.
    - Now the number of length-$(n + 1)$ walks between i to j equals the number of length-n walks from i to v that is adjacent to j, which is the (i, j) entry of $A^n A = A^{n+1}$ the non-zero entries of the column of A corresponding to v are precisely the first neighbours of v.

Example: How walks from 5 to 3 that is length-3?



All the walks from 5 to 3 with length 3

1. $5 \to 1 \to_{top} 2 \to_{right} 3$
2. $5 \to 1 \to_{top} 2 \to_{middle} 3$
3. $5 \to 1 \to_{top} 2 \to_{left} 3$
4. $5 \to 1 \to_{bottom} 2 \to_{right} 3$
5. $5 \to 1 \to_{bottom} 2 \to_{middle} 3$
6. $5 \to 1 \to_{bottom} 2 \to_{left} 3$
7. $5 \to 1 \to 4 \to_{top} 3$
8. $5 \to 1 \to 4 \to_{bottom} 3$

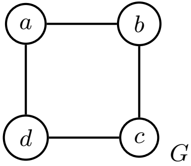
$$M = \begin{bmatrix} 0 & 2 & 0 & 1 & 1 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad M^2 = \begin{bmatrix} 1 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 1 & 1 \end{bmatrix} \qquad M^3 = \begin{bmatrix} 0 & 2 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & \color{red}{8} & 0 & 0 \end{bmatrix}$$

Table of Contents

Graph vertex coloring and chromatic polynomial

- **Proper coloring**: two adjacent nodes with different color

- **Chromatic polynomial**: the number of ways a graph can be properly colored
  - Notation: $\chi_G(t)$ is the chromatic polynomial of a graph G, here t is the number of color you can use
  - Line L_2, $\chi_{L_2}(t) = t(t-1) = t^2 - t$
  - Line L_3, $\chi_{L_3}(t) = t(t-1)^2 = t^3 - 2t^2 + t$
  - Line L_n, $\chi_{L_n}(t) = t(t-1)^{n-1}$
  - Triangle K_3, $\chi_{K_3}(t) = t(t-1)(t-2) = t^3 - 3t^2 + 2t$
  - Square $\chi_{\text{square}}(t) = t^4 - 4t^3 + 6t^2 - 3t = t(t-1)^2 + t(t-1)(t-2)^2$ WTF??



- For node a, you have t ways to color
- For nodes b and d, you have $t-1$ ways to color
  case 1. color of $b =$ color of d
  Then for c you have $t-1$ ways to color
  case 2. color of $b \neq$ color of d
  Then for c you have $t-2$ ways to color

$$\chi_{\text{square}}(t) = \underbrace{t}_{a}\underbrace{(t-1)}_{b}\cdot\underbrace{1}_{d}\cdot\underbrace{(t-1)}_{c} + \underbrace{t}_{a}\underbrace{(t-1)}_{b}\underbrace{(t-2)}_{d}\underbrace{(t-2)}_{c}$$

- **What's the big deal**: you can use a polynomial to represent a graph !!!!

Other topics

- Graph complement, graph disjoint, graph intersection, graph union
- Weighted graph, Graph cut, graph flow
- Eulerian Graph, Hamiltonian path, Petersen graph, Ramanujan graph
- Graph Laplacian $L = D - A$
- Graph theory + Linear algebra gives
  - Spectral graph theory — view graph as matrix, use eigendecomposition to study graph
  - Matroid — abstraction based on the notion of linear independence

Graph algorithms you will learn in the future

- Dijkstra's alg. — finding shortest path of weighted graph
- Bellman-Ford alg. — generalized Dijkstra
- Floyd-Warshall alg. — dynamic programming for finding shortest path of weighted graph
- Prim's alg. — node-based, finding min. spanning tree
- Kruskal's alg. — edge-based, finding min. spanning tree
- Ford-Fulkerson alg. — max-flow

Summary

- $V, |V|, E, |E|, M, A, D$
- Direct, undirect, multiedge, self loop, simple
- Converting multigraph to simple graph
- Walk, trail, path, circuit, cycle, forest, tree, bipartite
- Number of walks
- Coloring and chromatic polynomial