

# Topics on Computing and Mathematical Sciences I Graph Theory (1) Basic Concepts

Yoshio Okamoto

Tokyo Institute of Technology

April 9, 2008

## Lecture Style

- Language
  - Spoken: in English or Japanese
  - Slides: in English
  - Exercises: in English
  - Submission of exercise solutions: in English/Japanese (up to you)
- Feedback:
  - Submission of a piece of paper at the end of each lecture
  - There might be a survey at the term end

## Goal

This is a course on **mathematics** and/or **theory of computation**

### Goal of the course

- To get acquainted with **arguments in discrete mathematics** through graph theory (Graph theory just appears as one topic of discrete math)
- **Proofs** are important ingredients (From “ad-hoc” to “systematic”)

### Prerequisites

- Nothing in particular
- Other than a moderate familiarity with **basic math notation** and **mathematical proofs**; Need to know what is a proof
- And eagerness to learn

## Schedule

### Basic Topics

- 1 Definition of Graphs; Paths and Cycles
- 2 Cycles; Extremality
- 3 Trees; Matchings in Bipartite Graphs
- 4 Matchings and Factors
- 5 Connectivity
- 6 Coloring I
- 7 Coloring II

### Advanced Topics

- 8 Extremal Graph Theory I (Turán's theorem)
- 9 Extremal Graph Theory II (Erdős-Stone's theorem)
- 10 Extremal Graph Theory III (Szemerédi's regularity lemma)
- 11 Planarity
- 12 Minors
- 13 Ramsey Theory

Due to possible delay, the plan can be changed

## Administration

- Course Webpage:
  - <http://www.is.titech.ac.jp/~okamoto/lect/2008/gt/>
  - Reachable from the CompView website (<http://compview.titech.ac.jp/>)
  - Slides, exercises and references may be available there
- This is in the **Education Program for CompView**
- Lecturer: Yoshio Okamoto
  - Email: [okamoto at is.titech.ac.jp](mailto:okamoto@is.titech.ac.jp)
  - Office: W904 in West 8th Bldg.
  - Int. Phone: 3871
  - Office hours: by appointment, or you can try your luck any time
- Remark: **This graph theory course will not be given next year;**  
 Could be discrete geometry, data structures, enumerative combinatorics, extremal combinatorics, ..., I'm thinking

## Exercises

- Each lecture is accompanied with a set of exercises
- You do not have to submit your solutions to the exercises, but **submission is strongly encouraged**
- Please submit your solution at the end of the next lecture; I will give it back with comments
- Your solution itself doesn't give you any credit, but **solving exercises is the easiest way to learn, QUITE IMPORTANT**
- Recommended: **Work and discuss in groups!**
- Recommended: **Ask for hints!**
- If time permits, we may have some discussion on exercises in the class

## Evaluation

Basic: term-end exams (written or oral)

- Schedule tba (to be announced)
- **The materials from classes and exercises;**  
 Will heavily rely on the exercises
- Detail tba

Extra: resolution of an open problem

- Each lecture is accompanied with some open problems
- Resolution (or a progress) would give you extra points

## Algorithmic Remarks

**Caution!!**

Graph Algorithms are not a subject of this course

- However, algorithmic remarks are often provided
- Some open problems from the lectures may be algorithmic
- Slides on algorithms are highlighted in "red" color.

**Lesson**

Important to explicitly separate

- mathematical structures and
- computational structures
- (although they are highly interleaved)

## Another caution

## Caution!!

Modeling by Graphs is not a subject of this series of lectures

- And, no remark for modeling will be given
- Especially no real-world instance will be included
- You may try to model *your problems at hand* by utilizing graphs and/or discrete math; **It's up to you**

## References (1/2)

We do not follow any particular book but refer to good books in discrete math, combinatorics, and graph theory:

## Graph Theory Books

- R. Diestel. *Graph Theory*. 3rd ed. Springer, 2005.  
(Japanese translation of 2nd ed by S. Negami and K. Ohta should be available in Ookayama Coop Textbookstore)
- D.B. West. *Introduction to Graph Theory*. 2nd ed. Prentice Hall, 2001.
- J.A. Bondy and U.S.R. Murty. *Graph Theory*. Springer, 2008.

## References (2/2)

## Discrete Math and Combinatorics Books

- J. Matoušek and J. Nešetřil. *Invitation to Discrete Mathematics*. Oxford Univ Press, 1998.  
(Japanese translation by S. Negami and A. Nakamoto should be available in Ookayama Coop Textbookstore, **RECOMMENDED**)
- S. Jukna. *Extremal Combinatorics*. Springer, 2001.
- L. Lovász. *Combinatorial Problems and Exercises*. 2nd ed. Elsevier, 1993.
- R.L. Graham, M. Grötschel and L. Lovász, edn. *Handbook of Combinatorics*. Elsevier, 1995.

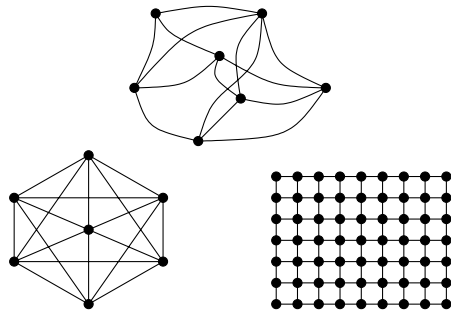
We also refer to several research papers

## Today's contents

- What is a graph, mathematically
- Graph isomorphisms
- Basic classes of graphs
- Degrees
- Paths and cycles
- Proof techniques

## What you might imagine about graphs

Pictures of “dots” and “lines”



Let us define a graph mathematically

## Definition: Graph

Definition: Graph

A **graph** is an ordered pair  $(V, E)$  of  $V$  and  $E$  satisfying the following

- $V$  is a set
- $E$  is a set of unordered pair on  $V$  (i.e.,  $E \subseteq \binom{V}{2}$ )

Example: Graph

$$V = \{1, 2, 3, 4, 5\}$$

$$E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{3, 4\}, \{4, 5\}\}$$

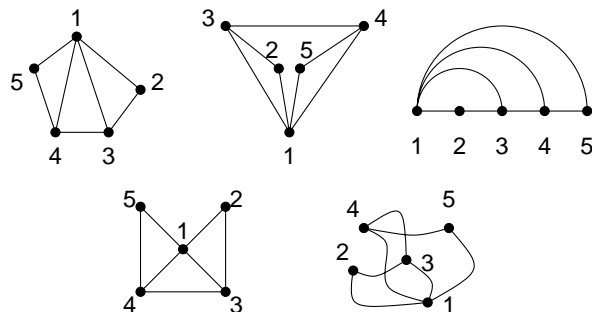
## Showing a graph by picture

A graph is frequently depicted by a picture

Example: Graph

$$V = \{1, 2, 3, 4, 5\}$$

$$E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{3, 4\}, \{4, 5\}\}$$



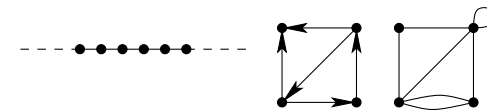
## Convention in the lectures

We assume the following unless stated otherwise

- $V$  is finite (then the graph is called **finite**)

If you've already been familiar with graphs, note that

- our definition excludes “directed” graphs
- our definition excludes graphs with loops or parallel edges



So, our graphs are always finite, undirected, and simple

### Vertices and edges

$G = (V, E)$  a graph

#### Definition: Vertex

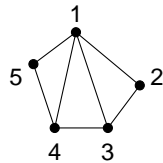
Each element of  $V$  is called a **vertex** of  $G$ ;  $V$  is the **vertex set** of  $G$

aliases of vertex: node, point, ...

#### Definition: Edge

Each element of  $E$  is called an **edge** of  $G$ ;  $E$  is the **edge set** of  $G$

aliases of edge: link, bond, ...



### Vertex and edge: Convention

$G$  a graph

#### Convention

The vertex set of  $G$  is often denoted by  $V(G)$

The edge set of  $G$  is often denoted by  $E(G)$

Example (to confuse you)

- For a graph  $G = (A, B)$  we have  $V(G) = A$  and  $E(G) = B$

### Numbers of vertices and edges

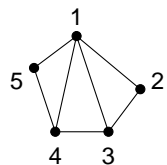
$G = (V, E)$  a graph

#### Notation: Number of vertices

$n(G) = |V|$  (often called the *order* of  $G$ )

#### Notation: Number of edges

$e(G) = |E|$  (often called the *size* of  $G$ )



$n(G) = 5, e(G) = 7$

$G$  is an  **$n$ -vertex graph** if  $n(G) = n$

### Endpoints, incidence, adjacency

$G = (V, E)$  a graph;  $u, v \in V$  vertices;  $e \in E$  an edge

#### Definition: Endpoint, incidence, adjacency

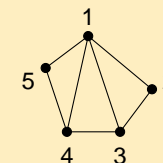
$v$  is an **endpoint** of  $e$  if  $v \in e$ ; Then  $v$  and  $e$  are **incident**;

$u, v$  are **adjacent** if  $\exists f \in E$  s.t.  $f = \{u, v\}$

#### Example

$V = \{1, 2, 3, 4, 5\}$

$E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{3, 4\}, \{4, 5\}\}$



1 is incident to  $\{1, 3\}$   
 $\{1, 3\}$  is incident to 3  
 $\{2, 3\}$  is not incident to 4  
 2 is adjacent to 3  
 3 is not adjacent to 5

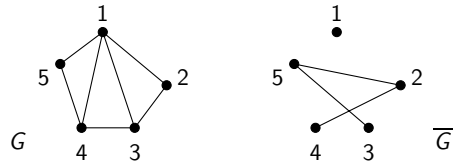
Operations on graphs: Complement

$G = (V, E)$  a graph

Definition: Complement

The **complement** of  $G$  is the graph denoted by  $\overline{G}$  defined as

- $V(\overline{G}) = V$
- $E(\overline{G}) = \binom{V}{2} \setminus E$



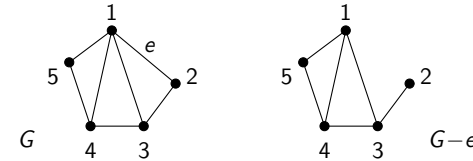
Operations on graphs: Deletion of an edge

$G = (V, E)$  a graph;  $e \in E$  an edge

Definition: Deletion

The **deletion** of  $e$  from  $G$  is the graph denoted by  $G-e$  defined as

- $V(G-e) = V$
- $E(G-e) = E \setminus \{e\}$



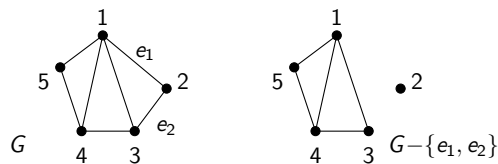
Operations on graphs: Deletion of an edge set

$G = (V, E)$  a graph;  $F \subseteq E$  an edge subset

Definition: Deletion

The **deletion** of  $F$  from  $G$  is the graph denoted by  $G-F$  defined as

- $V(G-F) = V$
- $E(G-F) = E \setminus F$



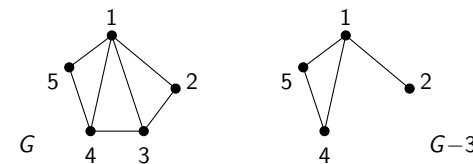
Operations on graphs: Deletion of a vertex

$G = (V, E)$  a graph;  $v \in V$  a vertex

Definition: Deletion

The **deletion** of  $v$  from  $G$  is the graph denoted by  $G-v$  defined as

- $V(G-v) = V \setminus \{v\}$
- $E(G-v) = E \setminus \{e \in E \mid v \in e\}$



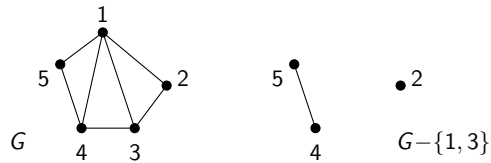
### Operations on graphs: Deletion of a vertex set

$G = (V, E)$  a graph;  $S \subseteq V$  a vertex subset

#### Definition: Deletion

The **deletion** of  $S$  from  $G$  is the graph denoted by  $G-S$  defined as

- $V(G-S) = V \setminus S$
- $E(G-S) = E \setminus \{e \in E \mid v \in e \text{ for some } v \in S\}$



### Today's contents

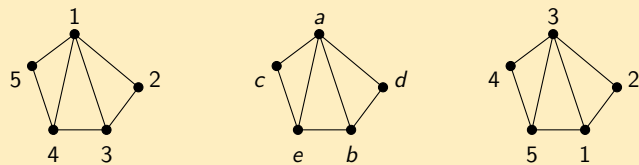
- What is a graph, mathematically
- Graph isomorphisms
- Basic classes of graphs
- Degrees
- Paths and cycles
- Proof techniques

### Graph isomorphism

#### Definition: Isomorphic graphs

Two graphs  $G = (V, E)$  and  $G' = (V', E')$  are **isomorphic** if  $\exists$  a bijection  $\varphi: V \rightarrow V'$  s.t.  $\{u, v\} \in E$  iff  $\{\varphi(u), \varphi(v)\} \in E'$ ; Such a bijection is an **isomorphism** from  $G$  to  $G'$

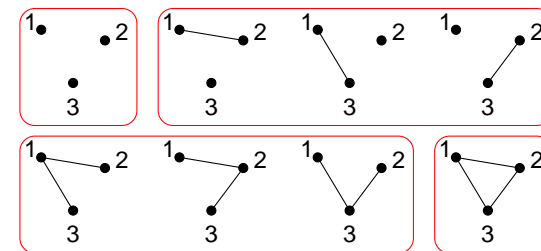
#### Example: isomorphic graphs



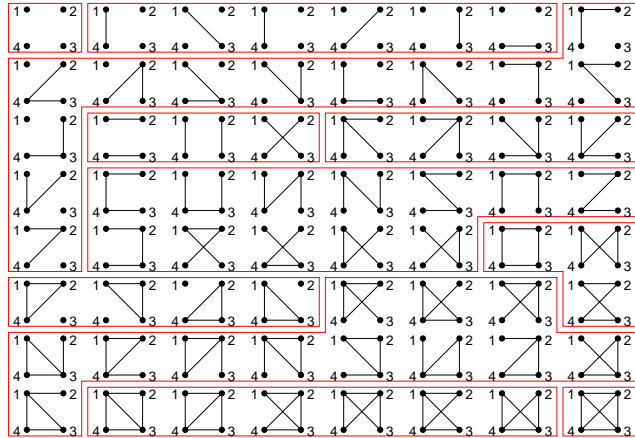
#### Notation: Isomorphic graphs

$G \simeq G'$  when  $G$  and  $G'$  are isomorphic

### The graphs on $\{1, 2, 3\}$



The graphs on {1, 2, 3, 4}



Isomorphism classes of graphs

Observation

The isomorphism relation is an equivalence on the graphs

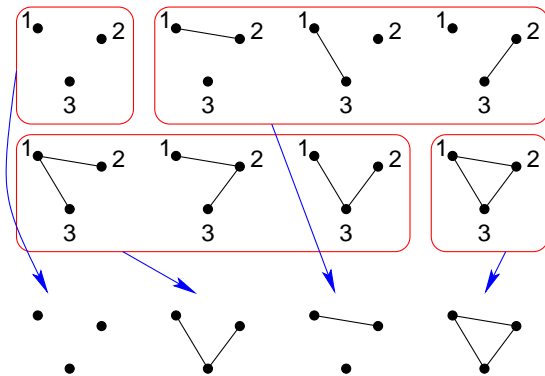
Therefore, we may introduce an equivalence class of graphs

Definition: Isomorphism classes

An **isomorphism class** of graphs is an equivalence class of graphs under the isomorphism relation

Isomorphism classes and unlabeled graphs

An isomorphism class can be drawn as an “unlabeled” graph



Deciding if two graphs are isomorphic

Problem (GRAPH ISOMORPHISM)

Input: Two graphs  $G$  and  $H$   
 Question: Is  $G$  isomorphic to  $H$ ?

Open Problem

We do not know this problem can be solved in poly-time or NP-hard

Current status

One of the central problem in algorithms and complexity

- Some evidences that the problem is not hard from complexity theory (many researchers)
- Best algorithm (theoretical):  $\exp(n^{2/3+\epsilon(1)})$  (Babai, Luks '83)
- Best software (practical): nauty (McKay '84–)  
<http://cs.anu.edu.au/~bdm/nauty/>

## Today's contents

- What is a graph, mathematically
- Graph isomorphisms
- Basic classes of graphs
- Degrees
- Paths and cycles
- Proof techniques

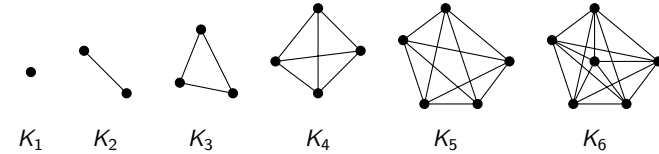
## Complete graphs

## Definition: Complete graph

A graph  $G$  is a **complete graph** if all pair of vertices are adjacent

## Notation: complete graph

$K_n$  a complete graph with  $n$  vertices



## Paths

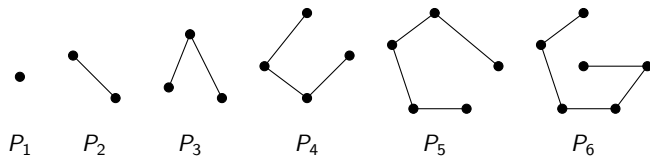
## Definition: Path

A graph  $G$  is a **path** if  $G$  is isomorphic to the following graph  $P$ :

$$V(P) = \{1, 2, \dots, n\}; E(P) = \{\{i, i+1\} \mid i \in \{1, \dots, n-1\}\}$$

## Notation: Path

$P_n$  a path with  $n$  vertices



The number of edges,  $n-1$ , is called the **length** of the path

## Cycles

## Definition: Cycle

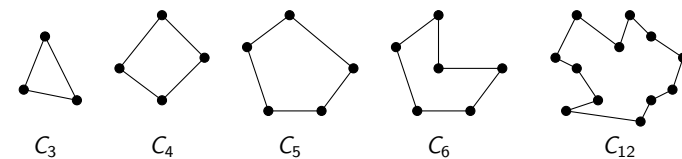
A graph  $G$  is a **cycle** if  $G$  is isomorphic to the following graph  $C$ :

$$V(C) = \{1, \dots, n\};$$

$$E(C) = \{\{i, i+1\} \mid i \in \{1, \dots, n-1\}\} \cup \{\{1, n\}\}$$

## Notation: Cycle

$C_n$  a cycle with  $n$  vertices



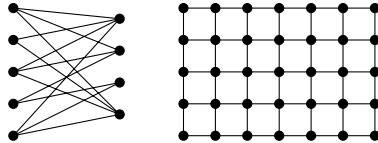
The number of edges,  $n$ , is called the **length** of the cycle

If the length is even (odd) then the cycle is called **even** (**odd**, resp.)

## Bipartite graphs

## Definition: Bipartite graph

A graph  $G = (V, E)$  is **bipartite** if  $V$  can be partitioned into two parts  $V_1, V_2$  s.t.  
 $\{u, v\} \in E \Rightarrow u \in V_1, v \in V_2$



$V_1, V_2$  are called **partite sets** of  $G$

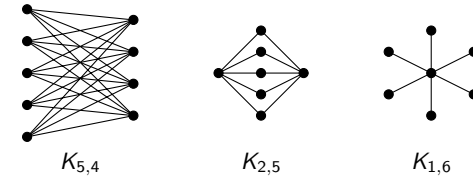
## Complete bipartite graphs

## Definition: Complete bipartite graph

A graph  $G = (V, E)$  is **complete bipartite** if  $V$  can be partitioned into two parts  $V_1, V_2$  s.t.  
 $\{u, v\} \in E \Leftrightarrow u \in V_1, v \in V_2$

## Notation: complete bipartite graph

$K_{n,m}$  a complete bipartite graph with partite sets of sizes  $n$  and  $m$



Remark:  $K_{n,m} \simeq K_{m,n}$  for any  $n, m$

## Bipartite testing

## Problem (BIPARTITENESS)

Input:  $G$  a graph

Question: Is  $G$  bipartite?

## Known fact

The problem above can be solved in  $O(n+m)$  time by any linear-time graph traversal algorithm

In algorithmic remarks,  $n = n(G)$  and  $m = e(G)$  always by convention

## Today's contents

- What is a graph, mathematically
- Graph isomorphisms
- Basic classes of graphs
- Degrees
- Paths and cycles
- Proof techniques

## Definition: Degree

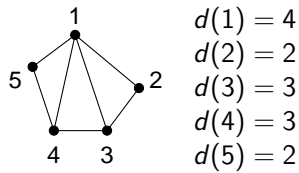
$G = (V, E)$  a graph;  $v \in V$  a vertex

## Definition: Degree

The **degree** of  $v$  in  $G$  is the number of edges incident to  $v$

## Notation: Degree

$d_G(v)$  the degree of  $v$  in  $G$ ; can be  $d(v)$  when  $G$  is clear



## Definition: Neighborhood

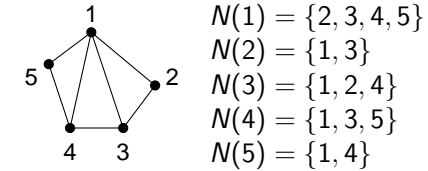
## Definition: Neighborhood

The **neighborhood** of  $v$  in  $G$  is the set of vertices adjacent to  $v$ ;

$$N_G(v) = \{u \in V \mid \{u, v\} \in E\};$$

Can be denoted by  $N(v)$  when  $G$  is clear;

Each vertex in  $N_G(v)$  is called a **neighbor** of  $v$  in  $G$



## Maximum and minimum degrees

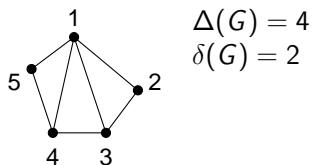
$G = (V, E)$  a graph

## Notation: Maximum degree, minimum degree

The **maximum degree** and the **minimum degree** of  $G$  are denoted by  $\Delta(G)$  and  $\delta(G)$  resp.

$$\Delta(G) = \max\{d_G(v) \mid v \in V\}$$

$$\delta(G) = \min\{d_G(v) \mid v \in V\}$$



## Regular graphs

$G = (V, E)$  a graph

## Definition: Regular graph

$G$  is **regular** if  $\Delta(G) = \delta(G)$ ;  **$k$ -regular** if  $\Delta(G) = \delta(G) = k$

## Example: Regular graphs

For which  $k$  are these graphs  $k$ -regular?

- Complete graphs  $K_n$
- Cycles  $C_n$
- Complete bipartite graphs  $K_{n,n}$
- Petersen graph

$$\begin{aligned}
 \bullet V &= \{X \mid X \subseteq \{1, 2, 3, 4, 5\}, |X| = 2\} = \binom{\{1, \dots, 5\}}{2} \\
 \bullet E &= \{\{X, Y\} \mid X \cap Y = \emptyset\}
 \end{aligned}$$

## First theorem: Handshaking lemma

$G = (V, E)$  a graph

## Theorem 1.1 (Handshaking Lemma)

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

## Lesson

When you're given a theorem (lemma, proposition, etc), try to find a (good) example to show "the theorem is not obviously false"

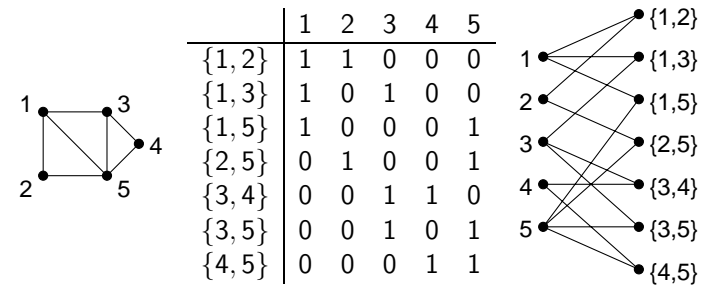
## Proof idea.

- Useful method "Double Counting" (or counting in two ways)
- Count the incident pairs  $(v, e) \in V \times E$  in two ways

□

## Double counting

A quite useful method in combinatorics and discrete math



## Consequences of the Handshaking Lemma

$G = (V, E)$  a graph

## Corollary 1.2 (The number of odd-degree vertices is even)

The number of odd-degree vertices in  $G$  is even

## Corollary 1.3 (Impossible regular graphs)

There exists **no** 133-regular graph with 827 vertices

## Corollary 1.4 (Average degree)

$$\delta(G) \leq \frac{2e(G)}{n(G)} \leq \Delta(G)$$

Corollary 1.5 (The number of edges in  $k$ -regular graphs)

$G$   $k$ -regular  $\implies e(G) = kn(G)/2$

## Today's contents

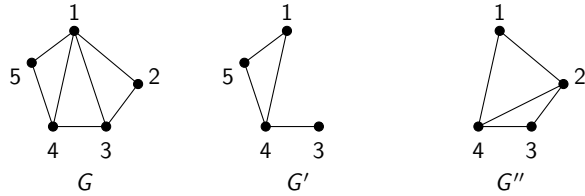
- What is a graph, mathematically
- Graph isomorphisms
- Basic classes of graphs
- Degrees
- Paths and cycles
- Proof techniques

Subgraphs

$G = (V, E), G' = (V', E')$  graphs

Definition: Subgraph

$G'$  is a **subgraph** of  $G$  if  $V' \subseteq V$  and  $E' \subseteq E$ ;  
Also we say  $G$  **contains**  $G'$ ; Denoted by  $G' \subseteq G$



$G$  contains  $G'$  but not  $G''$

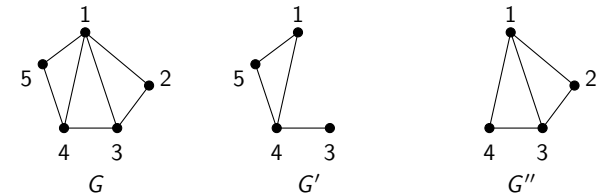
Convention: we also say  $G$  contains  $G'$  even if  $G$  contains a graph isomorphic to  $G'$  (For example,  $G$  above contains three  $K_3$ 's)

Induced subgraphs

$G = (V, E), G' = (V', E')$  graphs

Definition: Induced subgraph

$G'$  is an **induced subgraph** of  $G$  if  $G'$  is a subgraph of  $G$  and  $E' = E \cap \binom{V'}{2}$ ;  
 $G'$  is called the **subgraph induced by  $V'$** ; Denoted by  $G[V']$

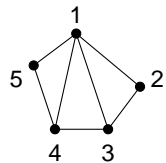


$G'$  is not an induced subgraph of  $G$ , but  $G''$  is

Paths and cycles as subgraphs

Convention

A path and cycle contained in a graph is often denoted by a list  $v_0, v_1, \dots, v_k$  of vertices



1, 3, 4 is a path  
2, 3, 4, 5 is a path  
1, 3, 4, 5, 1 is a cycle

Walks, trails in a graph

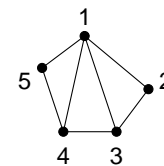
$G = (V, E)$  a graph

Definition: Walk, trail

A **walk** in  $G$  is a list  $v_0, v_1, \dots, v_k$  of vertices s.t.  $v_{i-1}$  and  $v_i$  are adjacent in  $G$  for all  $i \in \{1, \dots, k\}$ ;

A walk in  $G$  is called a **trail** in  $G$  if all of its edges are distinct;

Note: A path can be seen as a trail in which all vertices are distinct



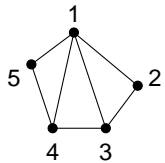
1, 3, 4, 5, 1, 4, 3, 2 is a walk, not a trail nor a path  
1, 3, 2, 1, 5 is a trail (hence a walk), not a path  
1, 3, 4 is a path (hence a trail, a walk)

Circuits in a graph

Definition: Circuit

A **circuit** in  $G$  is a walk in which  $v_0 = v_k$

Note: A cycle can be seen as a circuit in which all vertices are distinct



1, 3, 4, 5, 1, 4, 3, 2, 1 is a circuit, not a cycle

1, 3, 2, 3, 1 is a circuit, not a cycle

1, 3, 4, 5, 1 is a cycle, (hence a circuit)

$u, v$ -walk and  $u, v$ -path

$G = (V, E)$  a graph;  $u, v \in V$  vertices

Definition:  $u, v$ -walk and  $u, v$ -path

A walk  $v_0, v_1, \dots, v_k$  is called a  **$u, v$ -walk** if  $v_0 = u$  and  $v_k = v$ ;

A  **$u, v$ -path** is a  $u, v$ -walk that is a path

Proposition 1.6 (A walk contains a path)

A  $u, v$ -walk contains a  $u, v$ -path

Proof idea.

Useful method "Principle of Induction:" Induction on  $k$  □

Observation

The binary relation  $\sim$  on  $V$  defined as " $u \sim v$  iff there is a  $u, v$ -walk in  $G$ " is an equivalence relation on  $V$

Connected components

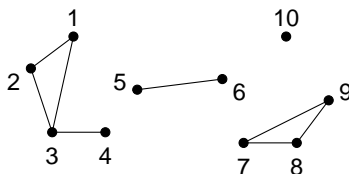
$G = (V, E)$  a graph

Definition: Connected component

A (**connected**) **component** of  $G$  is a subgraph of  $G$  induced by an equivalence class of  $\sim$

Definition: Isolated vertex

A vertex  $v \in V$  is **isolated** if it forms a connected component on its own; In other words, an isolated vertex is a vertex of degree zero



Connectedness

Definition: Connected graph

$G$  is **connected** if  $G$  itself is a connected component of  $G$ ;

Otherwise,  $G$  is **disconnected**

Caution

Do not use the phrase like

*A vertex  $u$  is connected to a vertex  $v$ .*

This is quite confusing. The phrase above could mean either " $u$  is adjacent to  $v$ " or "there is a  $u, v$ -walk," but it is almost impossible to distinguish them from the context. So, let's reserve the word "connected" for the connectedness of a graph.

## Less edges create more connected components

$G = (V, E)$  a graph

**Proposition 1.7** (The less edges, the more components)

$G$  has at least  $n(G) - e(G)$  connected components

**Proof idea.**

- Useful method “Principle of Induction”
- Let  $n(G)$  be fixed and change  $e(G) = 0, 1, \dots$

□

**Remark (or Lemma)**

Deleting an edge can increase the number of connected components at most by one

## Girth

**Definition: Girth**

The **girth**  $g(G)$  of a graph  $G$  is the length of a shortest cycle in  $G$ ;  
If  $G$  has no cycle then define  $g(G) = \infty$

**Proposition 1.8** (Girth of the Petersen graph)

The Petersen graph has girth five

**Proof idea.**

Let's do it without enumeration

- There is a cycle of length five!
- No cycle of length three: Look at an adjacent pair of vertices...
- No cycle of length four: Look at a non-adj. pair of vertices...

□

## Deciding connectedness

**Problem (#CONNECTED COMPONENTS)**

Input:  $G$  a graph

Output: the number of connected components of  $G$

**Known fact**

The problem above can be solved in  $O(n+m)$  time by any graph traversal algorithm

Consequently, we can decide whether a given graph is connected in  $O(n+m)$  time

## Computing the girth

**Problem (GIRTH)**

Input:  $G$  a graph

Output: the girth  $g(G)$  of  $G$

**Known fact** (Itai, Rodeh '78)

The problem above can be solved in  $O(nm)$  time or  $O(n^{2.376})$  time

**Open problem**

Find a faster algorithm for GIRTH

(Even we do not know how to decide if the girth is three or not faster)

## Today's contents

- What is a graph, mathematically
- Graph isomorphisms
- Basic classes of graphs
- Degrees
- Paths and cycles
- Proof techniques

## Basic and powerful proof techniques

Typical techniques in **discrete mathematics** (not only in graph theory)

- Double Counting  
(more examples to come in the lectures and exercises)
- Principle of Induction  
(more to come)
- Extremality  
(next lecture)

## Today's contents

- What is a graph, mathematically
- Graph isomorphisms
- Basic classes of graphs
- Degrees
- Paths and cycles
- Proof techniques
- **Open problems**

## Open problem: Hoffman-Singleton

## Open problem

Does there exist a 57-regular graph with girth 5 such that every non-adjacent pair of vertices has a common neighbor?

This is an important open problem from algebraic graph theory

## Known facts

If such a graph  $G$  exists,

- $n(G) = 3250, e(G) = 92625$
- $G$  cannot be vertex-transitive (Higman)
- the size of a maximum independent set  $\leq 400$
- ...

## Open Problem: Dean

## Conjecture (Dean)

$G$  a graph;  $k \geq 3$  an integer;

$\delta(G) \geq k \Rightarrow G$  contains a cycle of length  $0 \pmod{k}$

## Known facts

- True for  $k = 3$  (Chen, Saito '94)
- True for  $k = 4$  (Dean, Lesniak, Saito '93)

## Open Problem: Erdős-Gyárfás

## Conjecture (Erdős, Gyárfás '95)

$G$  a graph

$\delta(G) \geq 3 \Rightarrow G$  contains a cycle of length  $2^k$  for some  $k$

## Known facts

- True if  $G$  is planar and  $K_{1,3}$ -free (Daniel, Shauger '01)
- Counterexamples should have at least 17 vertices (Markström '04)

## Today's "piece of paper"

- **Write something!**  
This is a quick way to send a feedback to the lectures
- For example
  - whatever you got in today's lecture
  - whatever you expect in coming lectures
  - whatever you feel for this course
  - whatever you think about this special education program
  - whatever you saw this morning
  - whatever you like to share with me
  - ...
- **Replies to your comments would appear in the course webpage**
- You may put your name as well, but not necessarily  
(Your name doesn't appear on the course webpage)
- **Please hand in before you leave**