CS201 DISCRETE MATHEMATICS FOR COMPUTER SCIENCE

Dr. Qi Wang

Department of Computer Science and Engineering
Office: Room903, Nanshan iPark A7 Building
Email: wangqi@sustech.edu.cn

Course Information

- Instructor:

  Dr. QI WANG
  Department of Computer Science and Engineering
  Office: Room903, Nanshan iPark A7 Building
  Email: wangqi@sustech.edu.cn

- TA:

  They are mainly assignment markers.

- Course webpage:

  sakai.sustech.edu.cn → "CS201 1 fall2020"

Course Information

- Lectures:

  Liyuan Building 1

Tuesday Rm203		Friday Rm203
14:00 – 15:50	—	14:00 – 15:50
every week		every two weeks

Course Information

- Lectures:

  Liyuan Building 1

  Tuesday Rm203 — 14:00 – 15:50 — every week

  Friday Rm203 — 14:00 – 15:50 — every two weeks

- Office hours:

  Thu. 14:20-16:00 or send email for appointment

Course Information

- Lecture Notes (in progress)

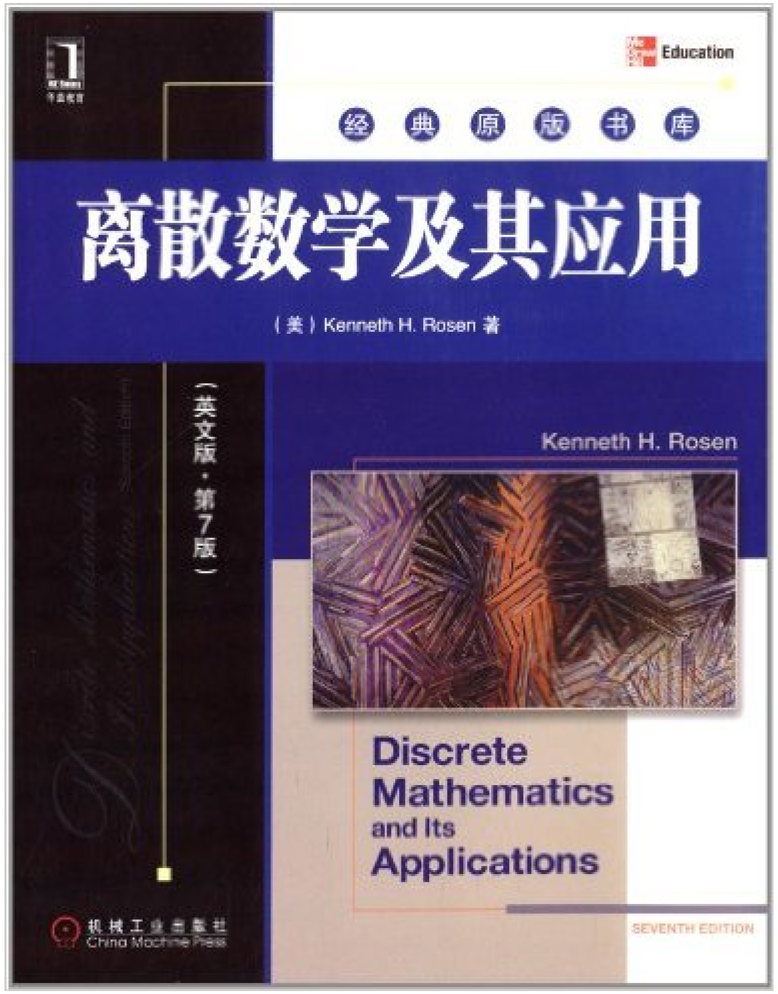
CS201 Discrete Math for Computer Science[1]

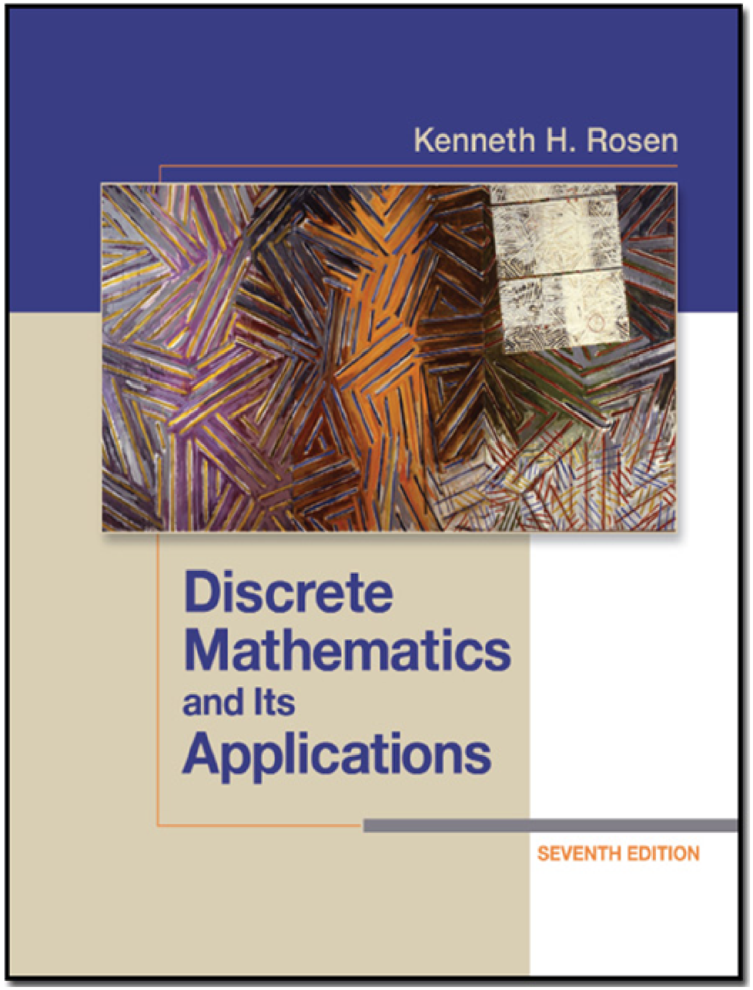
Qi Wang
Department of Computer Science and Engineering
Southern University of Science and Technology

Spring 2019

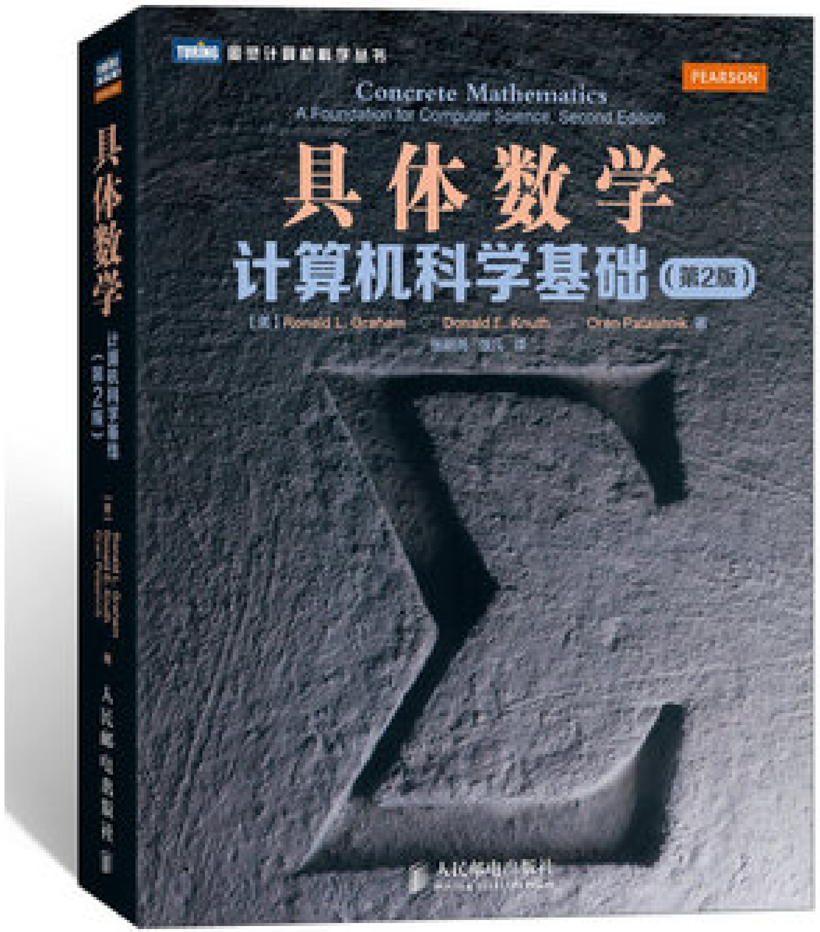
Course Information

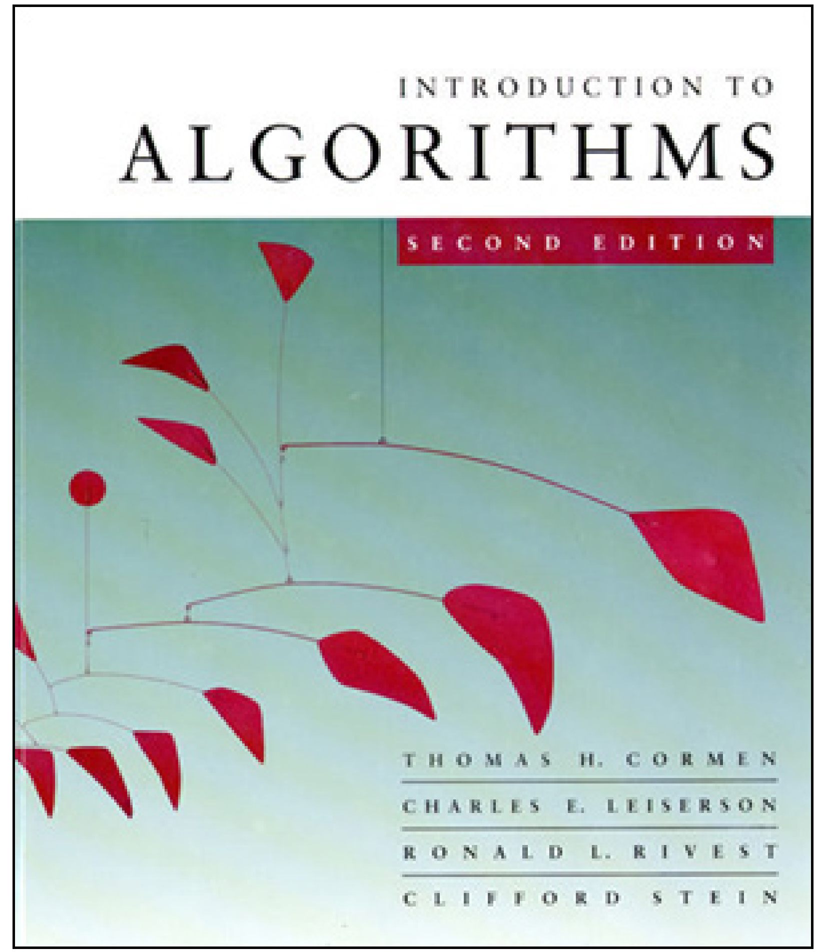
- Textbook:





- Reference Books:





Marking Scheme

- Quiz in class (10%)
- Homework assignments (20%)
  - assigned in class and posted on the course webpage
  - must be submitted online before due date
  - no extension policy with exceptions
- Midterm exam (30%)
  - To be announced
  - covers the first half of class material
- Final exam (40%)
  - covers the entire semester's material
- Project (5%, optional)

Important Messages to Students

- Main ideas will be covered in class but some details might be skipped. You are responsible for all materials in assigned sections of book, even if they are not taught in class.

- Homework assignments should be worked on and written up individually, though group discussions are allowed.

- Any unintellectual behavior and cheating on exams, homework assignments will be dealt with severely.

- If you get the main idea for a solution from someone else or a website, you must acknowledge that source in your submission.

Plagiarism Policy

* If an undergraduate assignment is found to be plagiarized, the first time the score of the assignment will be 0.

* The second time the score of the course will be 0.

As it may be difficult when two assignments are identical or nearly identical who actually wrote it, the policy will apply to BOTH students, unless one confesses having copied without the knowledge of the other.

What is OK and what is not OK?

- It's OK to work on an assignment with a friend, and think together about the program structure, share ideas and even the global logic. At the time of actually writing the code, you should write it alone.
- It's OK to use in an assignment a piece of code found on the web, as long as you indicate in a comment where it was found and don't claim it as your own work.
- It's OK to help friends debug their programs (you'll probably learn a lot yourself by doing so).
- It's OK to show your code to friends to explain the logic, as long as the friends write their code on their own later.

It's NOT OK to take the code of a friend, make a few cosmetic changes (comments, some variable names) and pass it as your own work.

What is OK and what is not OK?

计算机科学与工程系
Department of Computer Science and Engineering

Undergraduate Students Assignment Declaration Form

This is____________ (student ID: ____________, who has enrolled in __________ course, originated the Department of Computer Science and Engineering. I have read and understood the regulations on plagiarism in assignments and theses according to "Regulations on Academic Misconduct in Assignments for Undergraduate Students in the SUSTech Department of Computer Science and Engineering". I promise that I will follow these regulations during the study of this course.

Signature：

Date:

Important Messages to Students

- Please ask questions in class

If you're having trouble understanding something, then at least 50% of the class is also having trouble: they'll *be happy* if you ask for more explanation.

- The lecture slides are still in progress

If you find mistakes or just think that something's confusing, please email me. Your classmates and future students will *thank you*.

Discrete Mathematics

- What is discrete mathematics?

  ◇ *Discrete mathematics* is the study of *mathematical structures* that are fundamentally discrete rather than continuous.

  For example, integers, graphs, statements in logic, etc.

  ◇ Discrete mathematics therefore excludes topics in "continuous mathematics" such as *Calculus* and *Analysis*.

Discrete Mathematics

- What is discrete mathematics?

  ◇ *Discrete mathematics* is the study of *mathematical structures* that are fundamentally discrete rather than continuous.

  For example, integers, graphs, statements in logic, etc.

  ◇ Discrete mathematics therefore excludes topics in "continuous mathematics" such as *Calculus* and *Analysis*.

- Some applications

  ◇ *Computer science*: algorithms, programming languages, cryptography, artificial intelligence, ...

  ◇ *Electronic engineering*: digital communications, signal processing, information theory, coding theory, ...

Topics of This Course

- Propositional and Predicate Logic
- Mathematical Proofs
- Sets
- Functions
- Complexity of Algorithms
- Number Theory
- Groups, Rings, and Fields*
- Cryptography
- Mathematical Induction
- Recursion
- Counting
- Discrete Probability*
- Relations
- Graphs
- Trees
- Finite Fields*

Lecture Schedule (Tentatively)

01. Logic
02. Mathematical Proofs
03. Sets and Functions
04. Complexity of Algorithms
05. Number Theory
06. Groups, Rings and Fields*
07. Cryptography
08. Midterm Exam
09. Mathematical Induction
10. Recursion
11. Counting
12. Discrete Probability
13. Relations
14. Graphs I
15. Graphs II
16. Trees and Review Lecture

Learning Objectives

- Be able to read, understand, and construct mathematical arguments and proofs
- Understand the formulation of common problems in several areas of discrete mathematics, including *counting, graphs, number theory, cryptography, logic and proof, recursions, probability theory, finite fields*, etc.
- Learn a number of discrete mathematical tools
- Apply discrete mathematical tools to solve certain problems in computer science and electronic engineering

Acknowledgements

- Some of slides are based on lecture material used in the following institutions:

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

STANFORD UNIVERSITY

Massachusetts Institute of Technology

UNIVERSITY OF PITTSBURGH

Problem I

- **Q. 1** Prove that "For an integer n, if $3n + 2$ is odd, then n is odd".

Problem I

- **Q. 1** Prove that "For an integer n, if $3n+2$ is odd, then n is odd".

  Proof I (*direct proof*)

  Note that $3n+3$ is even, and so is $n+1$. It then follows that n is odd.

Problem I

- **Q. 1** Prove that "For an integer n, if $3n+2$ is odd, then n is odd".

Proof II (*proof by contrapositive*)

It is *equivalent* to prove that "If n is even , then $3n+2$ is even." – Why?

w.l.o.g. suppose that $n = 2k$ for some integer k, then

$$3n+2 = 6k+2,$$

which is even.

Problem I

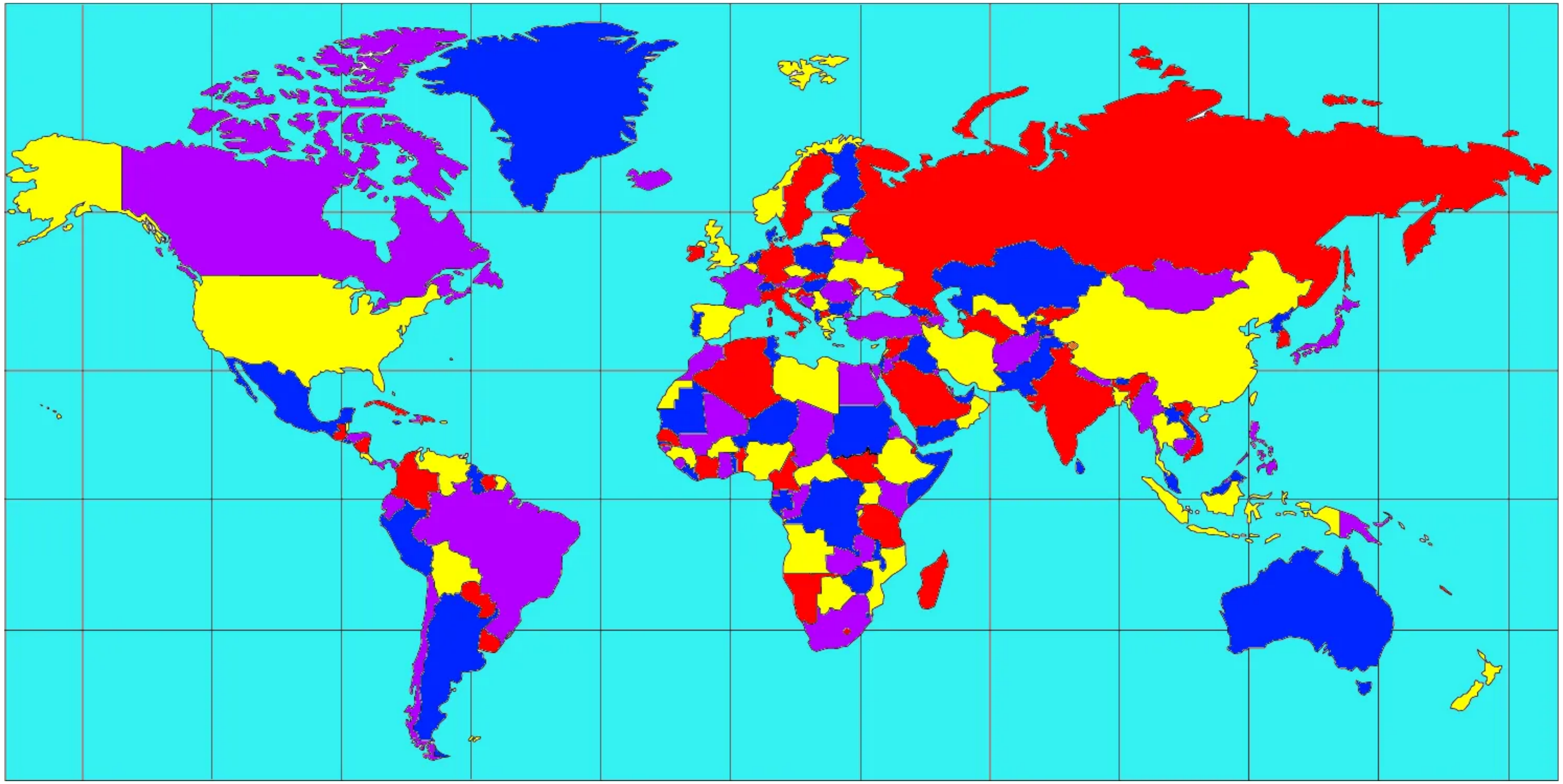
- **Q. 1** Prove that "For an integer n, if $3n+2$ is odd, then n is odd".

  Proof III (*proof by contradiction*)

  Assume to the contrary that there exists an integer n such that $3n+2$ is odd and n is even. Since n is even, so is $3n$. Then 2 must be odd, leading to a contradiction.

Problem II

- **Four-color theorem** Given any separation of a plane into contiguous regions, producing a figure called a *map*, no more than four colors are required to color the regions of the map so that no two adjacent regions have the same color.

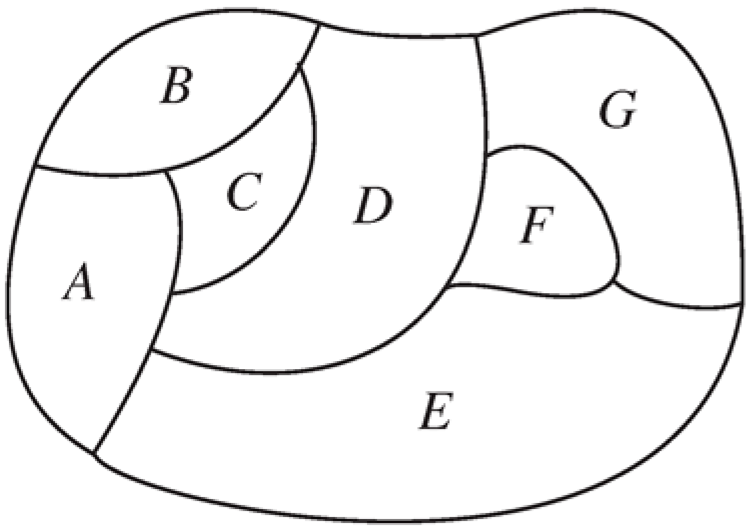


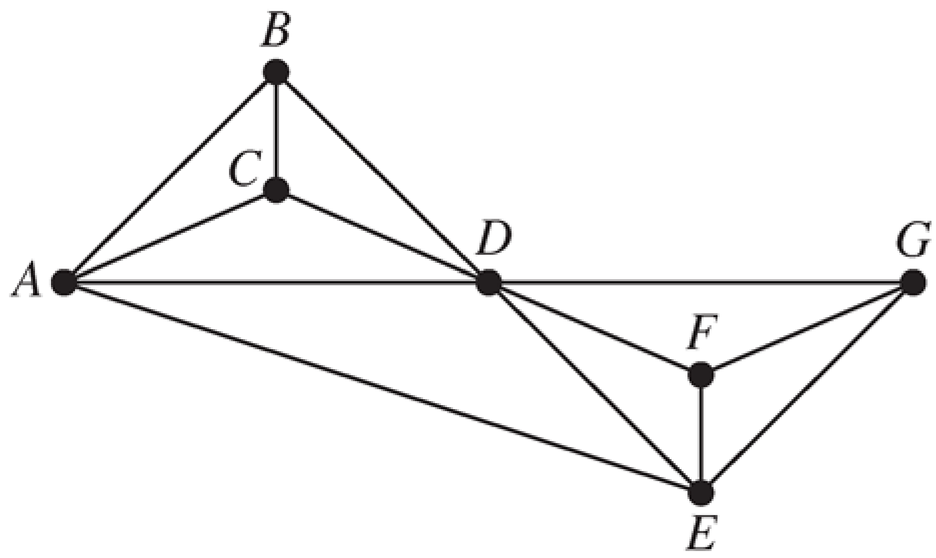
Problem II

- **Four-color theorem** (for planar graphs)
  - ◇ first proposed by Francis Guthrie in 1852
  - ◇ his brother Frederick Guthrie told Augustus De Morgan
  - ◇ De Morgan wrote to William Hamilton
  - ◇ Alfred Kempe proved it incorrectly in 1879
  - ◇ Percy Heawood found an error in 1890 and proved the *five-color theorem*
  - ◇ Finally, Kenneth Appel and Wolfgang Haken proved it with case by case analysis by computer in 1976 (*the first computer-aided proof*)
  - ◇ Kempe's incorrect proof serves as a basis

Problem II

- **Four-color theorem**





Problem II

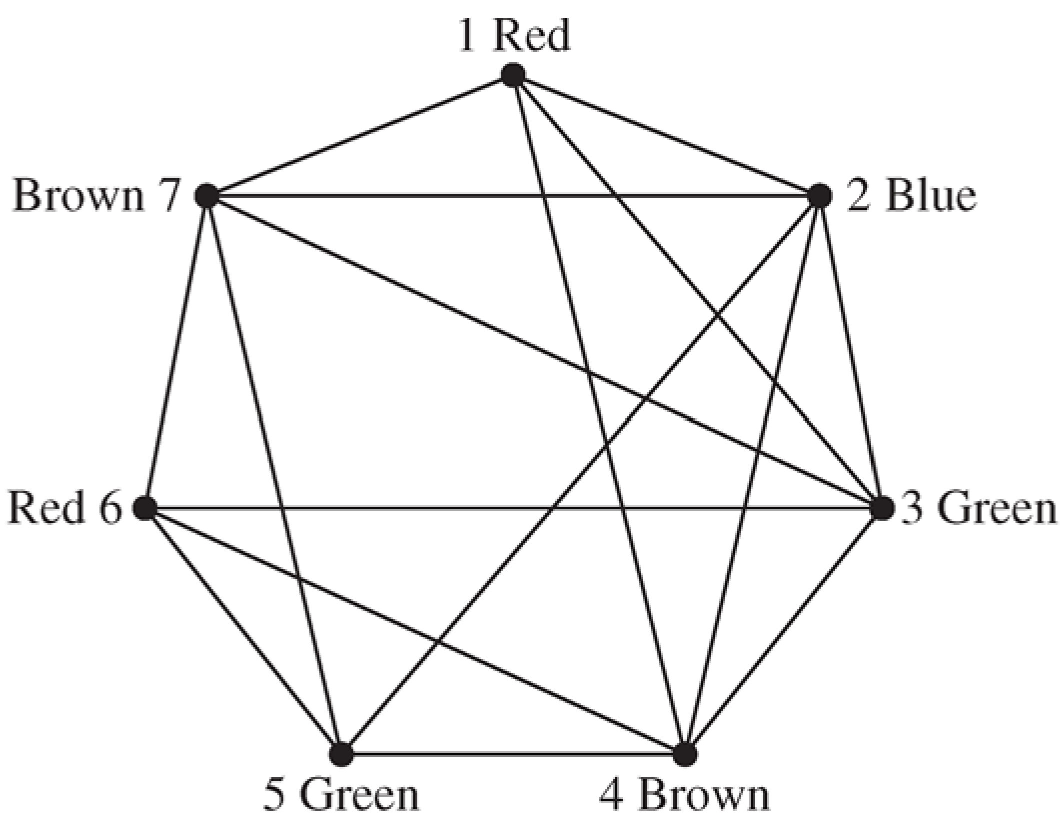
- Applications of *graph colorings*
  - ◇ Scheduling Final Exams

    Vertices represent courses, and there is an edge between two vertices if there is a common student in the courses.

Problem II

- Applications of *graph colorings*
  - ◇ Scheduling Final Exams

    Vertices represent courses, and there is an edge between two vertices if there is a common student in the courses.



Time Period	Courses
I	1, 6
II	2
III	3, 5
IV	4, 7

Problem II

- Applications of *graph colorings*
  - ◇ Scheduling Final Exams

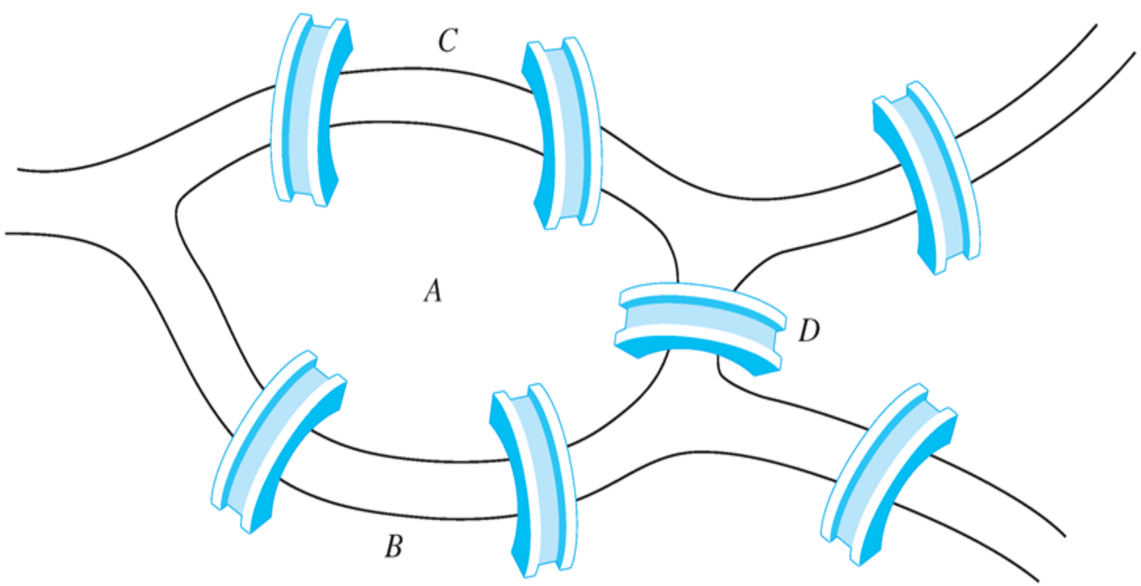
    Vertices represent courses, and there is an edge between two vertices if there is a common student in the courses.

  - ◇ Graph coloring is *computationally hard*!!!

Problem III

- **Königsberg seven-bridge problem**

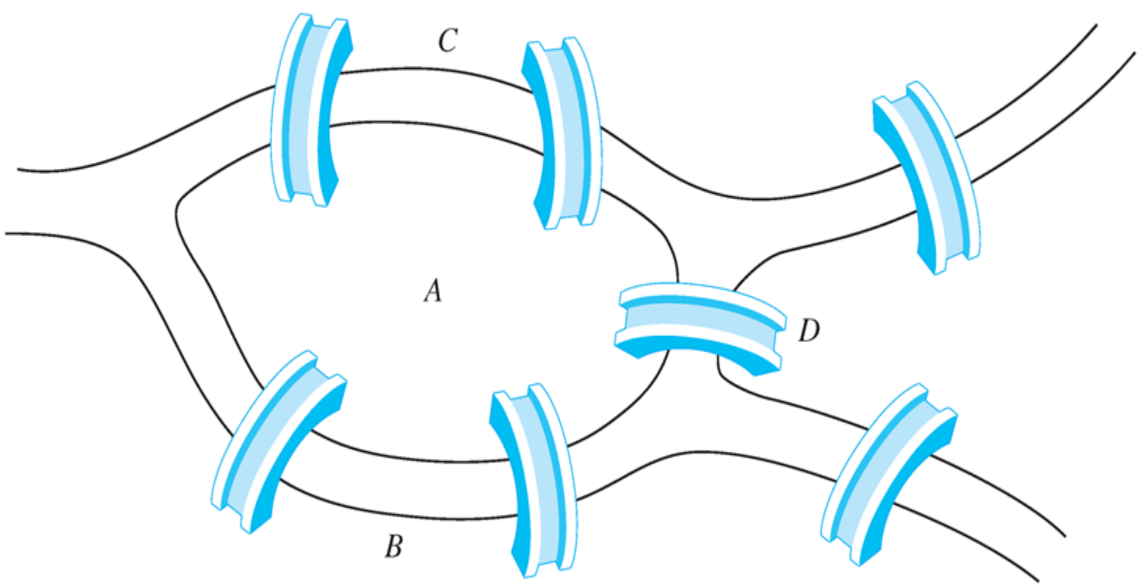
  People wondered whether it was possible to start at some location in the town, travel across all the bridges once without crossing any bridge twice, and return to the starting point.

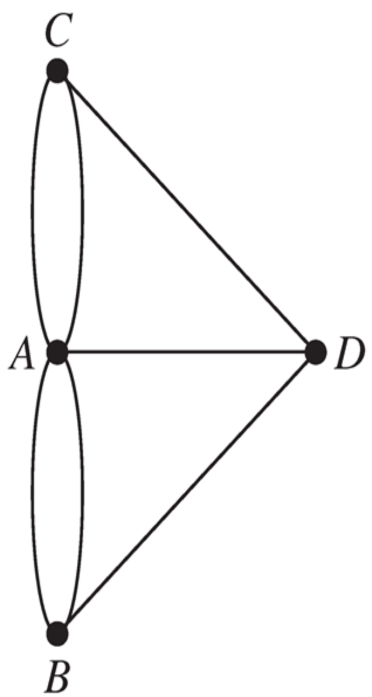


Problem III

- **Königsberg seven-bridge problem**

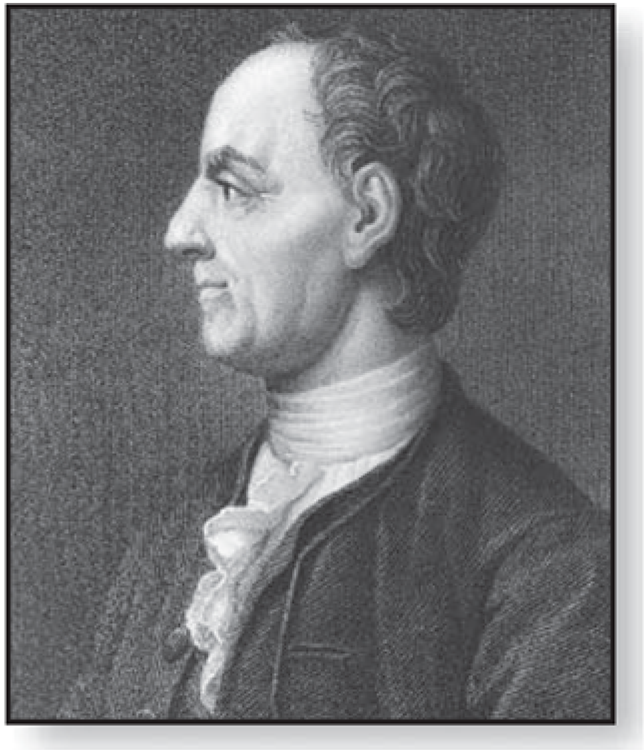
  People wondered whether it was possible to start at some location in the town, travel across all the bridges once without crossing any bridge twice, and return to the starting point.



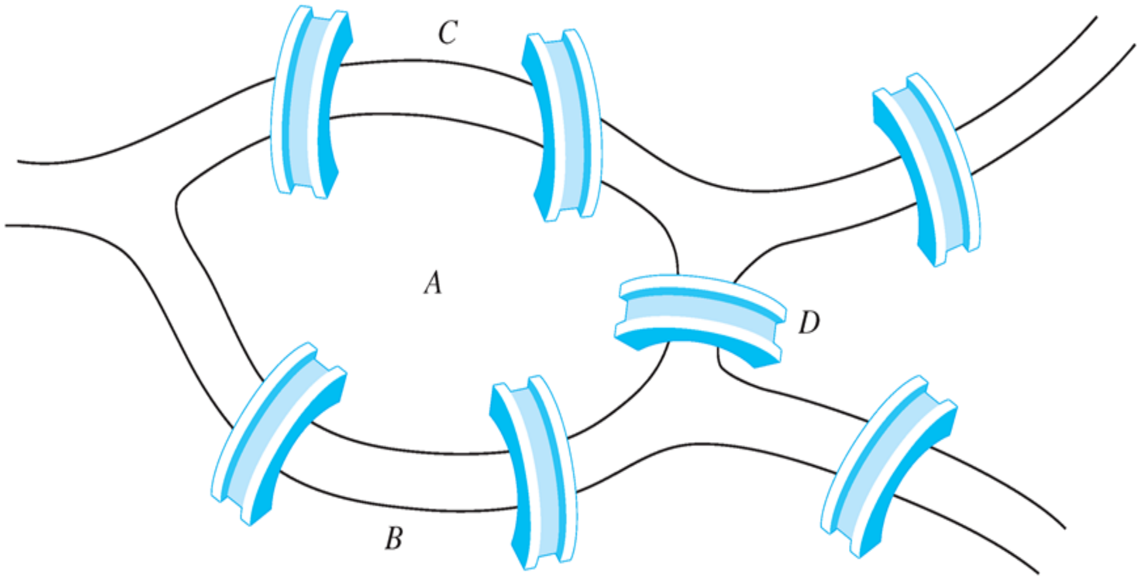


Problem III

- **Königsberg seven-bridge problem**



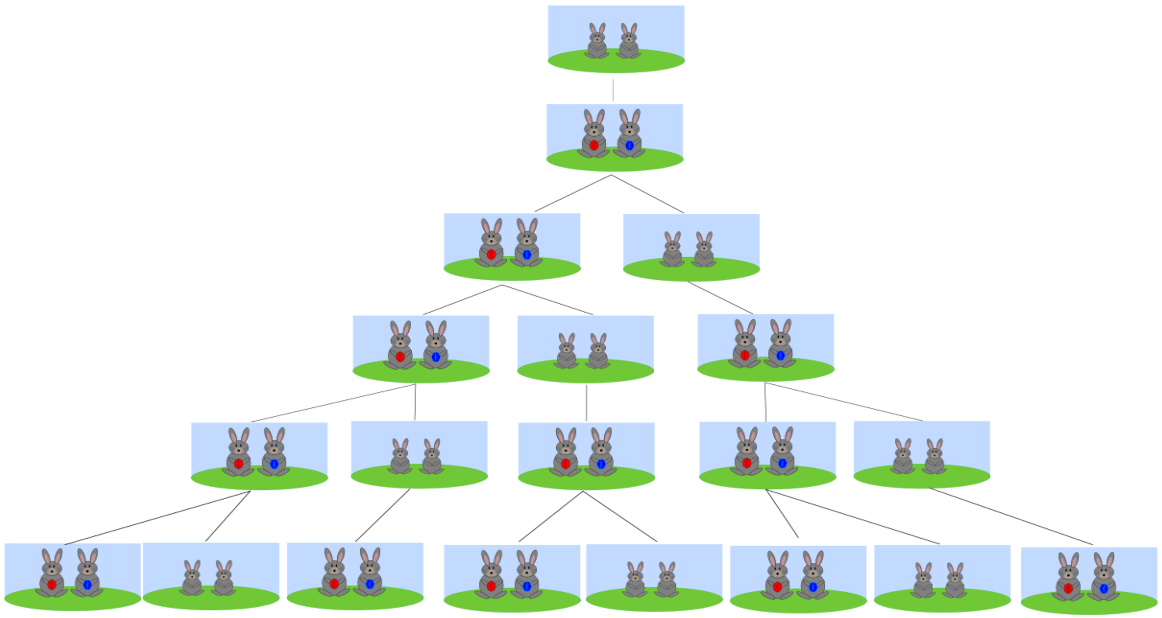
Leonhard Euler

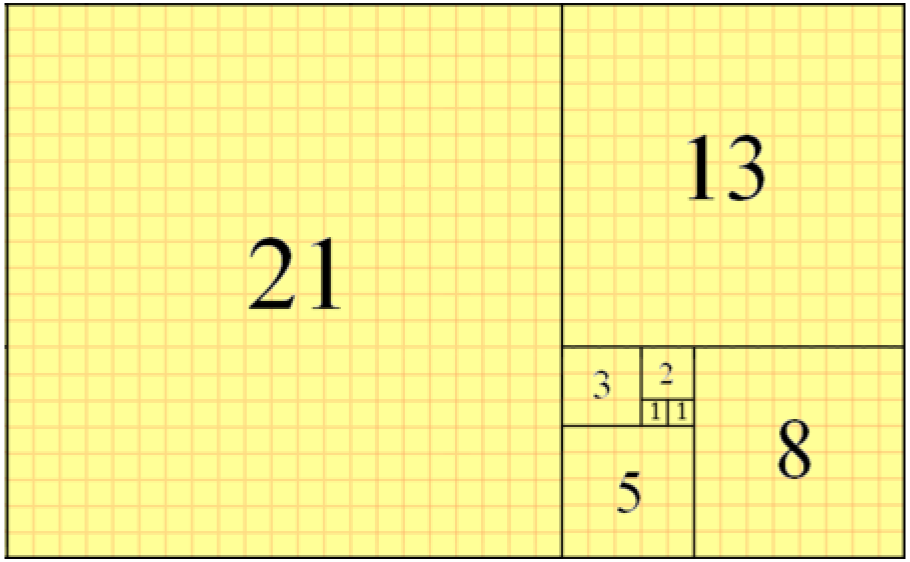


Theorem A connected multigraph with at least two vertices has an *Euler circuit* if and only if each of its vertices has **even** degree.

Problem IV

- **Fibonacci number**





Problem IV

- **Fibonacci number**

  $F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$

Problem IV

- **Fibonacci number**

  $F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$

  ◇ What is the closed-form expression of F_n?

Problem IV

- **Fibonacci number**

  $F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$

  ◇ What is the closed-form expression of F_n?

  Consider $x^n = x^{n-1} + x^{n-2}$, with $x \neq 0$. There are two different roots

  $$\phi = \frac{1+\sqrt{5}}{2}, \quad \psi = \frac{1-\sqrt{5}}{2}$$

  Then F_n can be the form of $a\phi^n + b\psi^n$. By $F_0 = 0$ and $F_1 = 1$, we have $a + b = 0$ and $\phi a + \psi b = 1$, leading to $a = \frac{1}{\sqrt{5}}$, $b = -a$. Therefore,

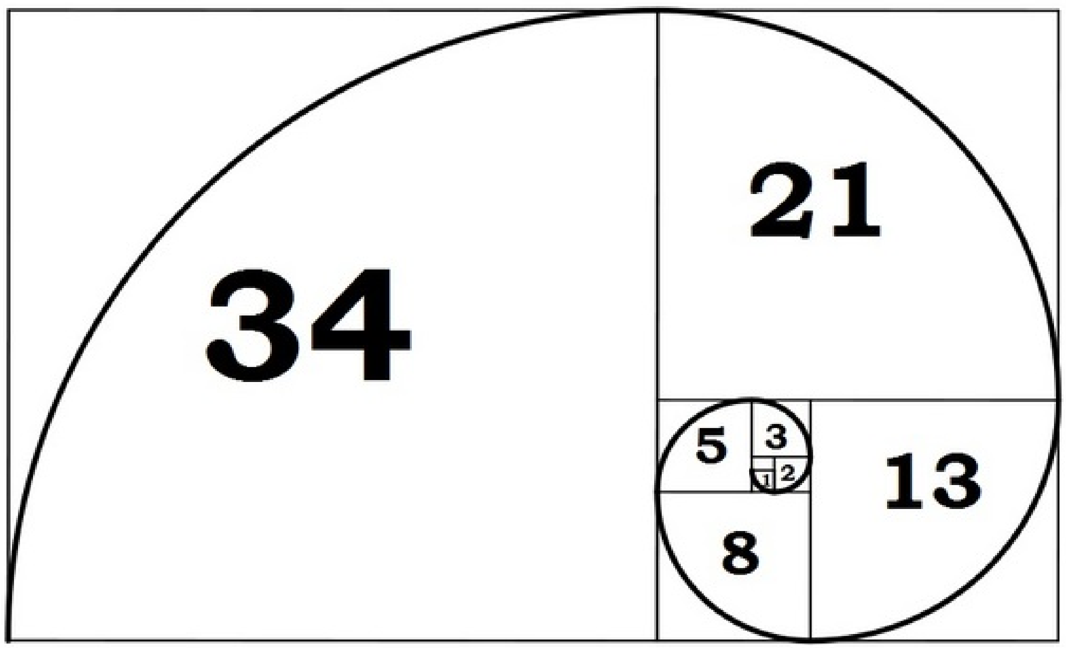
  $$F_n = \frac{\phi^n - \psi^n}{\sqrt{5}}$$

Problem IV

- **Fibonacci number**

  $F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$

◇ What is t



Consider x^n … wo different roots

Then F_n ca … and $F_1 = 1$, we have $a +$ … $\frac{1}{\sqrt{5}}$, $b = -a$. Therefore,

$$F_n = \frac{\phi^n - \psi^n}{\sqrt{5}}$$

Let's start!

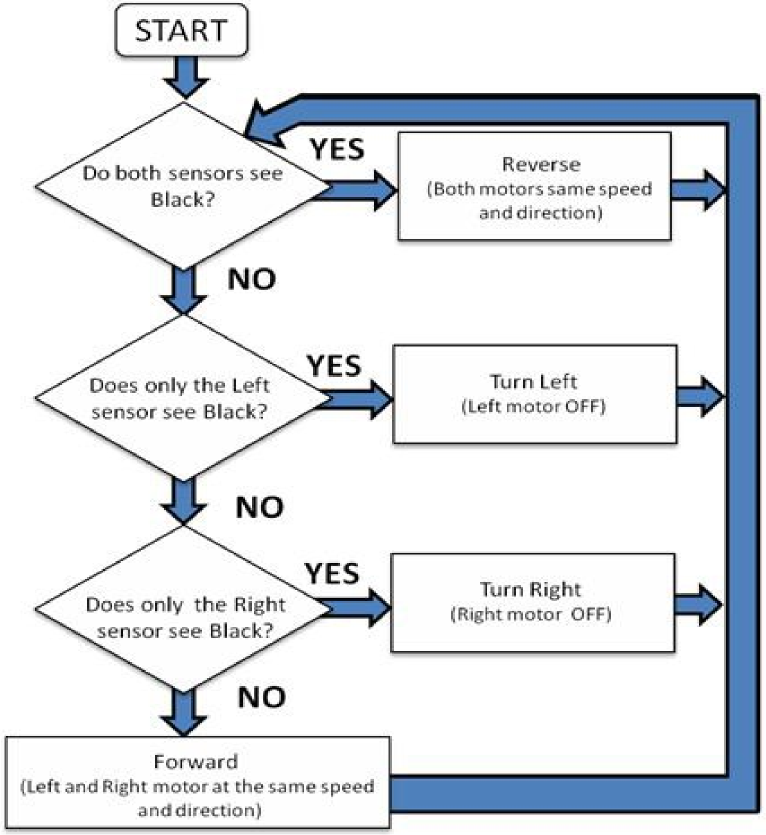
"All you need is a cool head, a large sheet of paper, and a fairly decent handwriting."

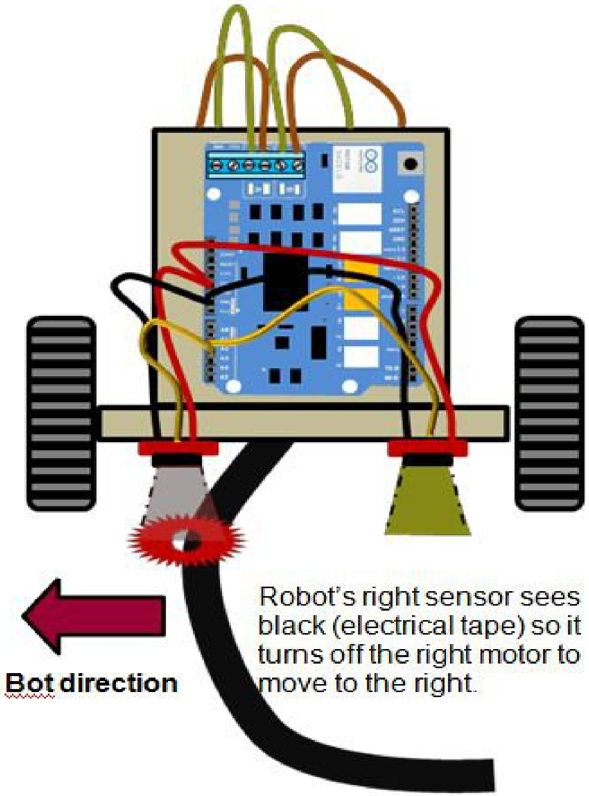
Logic

- Logic is the basis of all mathematical reasoning.
  - *syntax* of statements
  - the *meaning* of statements
  - the rules of logical *inference*

Logic

- Logic is the basis of all mathematical reasoning.
  - ◇ *syntax* of statements
  - ◇ the *meaning* of statements
  - ◇ the rules of logical *inference*





Propositional Logic

- A *proposition* is a declarative statement that is either true or false.

Propositional Logic

- A *proposition* is a declarative statement that is either true or false.

  Examples:

  ◇ SUSTech is located in Shenzhen. (T)

  ◇ $1 + 1 = 2$ (T)

  ◇ $2 + 2 = 3$ (F)

Propositional Logic

- A *proposition* is a declarative statement that is either true or false.

  Examples:

  - ◇ SUSTech is located in Shenzhen. (T)
  - ◇ $1 + 1 = 2$ (T)
  - ◇ $2 + 2 = 3$ (F)

"declarative" + "either true or false"

Propositional Logic

- A *proposition* is a declarative statement that is either true or false.

  Examples:

  - ◇ SUSTech is located in Shenzhen. (T)
  - ◇ $1 + 1 = 2$ (T)
  - ◇ $2 + 2 = 3$ (F)

  "declarative" + "either true or false"

  - ◇ No parking.
  - ◇ How old are you?
  - ◇ $x + 2 = 5$
  - ◇ She is very talented.

Propositions

- What about the following?
  - ◇ There are infinitely many twin prime numbers.
  - ◇ This is an unsettled conjecture (a.k.a. twin-prime conjecture).
  - ◇ There are other life forms on other planets in the universe.

Propositions

- What about the following?
  - ◇ There are infinitely many twin prime numbers.
  - ◇ This is an unsettled conjecture (a.k.a. twin-prime conjecture).
  - ◇ There are other life forms on other planets in the universe.

- Usually, a proposition is condition-based.
  - ◇ If you would donate 1 billion, you will become the president.

Compound Propositions

- More complex propositions can be built from elementary statments using logical connectives.

Compound Propositions

- More complex propositions can be built from elementary statments using logical connectives.

Example:

◇ Proposition A: It rains outside.

◇ Proposition B: We will watch a movie.

◇ A new proposition: If it rains outside then we will watch a movie.

Compound Propositions

- More complex propositions can be built from elementary statments using logical connectives.

- Logical connectives:
  - *Negation*
  - *Conjunction*
  - *Disjunction*
  - *Exclusive or*
  - *Implication*
  - *Biconditional*

Negation

- Let p be a proposition. The statment "It is not the case that p." is called the *negation of p*, denoted by $\neg p$, and read as "not p".

Negation

- Let p be a proposition. The statment "It is not the case that p." is called the *negation of p*, denoted by $\neg p$, and read as "not p".

  Examples:

  - p – SUSTech is located in Shenzhen. (T)
  - $\neg p$ – SUSTech is not located in Shenzhen.
    It is not the case that SUSTech is located in Shenzhen.

Negation

- Let p be a proposition. The statment "It is not the case that p." is called the *negation of p*, denoted by $\neg p$, and read as "not p".

  Examples:

  - ◇ p – SUSTech is located in Shenzhen. (T)
  - ◇ $\neg p$ – SUSTech is not located in Shenzhen.
    It is not the case that SUSTech is located in Shenzhen.
  - ◇ $5 + 2 \neq 8$
  - ◇ 10 is not a prime number.
  - ◇ It is not the case that classes begin at 8:00am.

Negation

- A *truth table* displays the relationships between truth values (T or F) of different propositions.

p	$\neg p$
T	F
F	T

Negation

- A *truth table* displays the relationships between truth values (T or F) of different propositions.

p	$\neg p$
T	F
F	T

Rows: contains all possible values of elementary propositions.

Conjunction (and)

- Let p and q be propositions. The *conjunction* of p and q, denoted by $p \wedge q$, is true when both p and q are true and is false otherwise.

Conjunction (and)

- Let p and q be propositions. The *conjunction* of p and q, denoted by $p \wedge q$, is true when both p and q are true and is false otherwise.

  Examples:

  ◇ p – SUSTech is located in Shenzhen.
  ◇ q – $5 + 2 = 8$

  ◇ $p \wedge q$ – SUSTech is located in Shenzhen and $5 + 2 = 8$

Conjunction (and)

- Let p and q be propositions. The *conjunction* of p and q, denoted by $p \wedge q$, is true when both p and q are true and is false otherwise.

  Examples:

  ◇ p – SUSTech is located in Shenzhen.
  ◇ q – $5 + 2 = 8$

  ◇ $p \wedge q$ – SUSTech is located in Shenzhen and $5 + 2 = 8$

  ◇ There are infinitely many twin prime numbers and $2 + 2 = 3$

  ◇ 2 is a prime number and 9 is a prime power.

Disjunction (or)

- Let p and q be propositions. The *disjunction* of p and q, denoted by $p \vee q$, is false when both p and q are false and is true otherwise.

Disjunction (or)

- Let p and q be propositions. The *disjunction* of p and q, denoted by $p \vee q$, is false when both p and q are false and is true otherwise.

  Examples:

  ◇ p – SUSTech is located in Shenzhen.

  ◇ q – $5 + 2 = 8$

  ◇ $p \vee q$ – SUSTech is located in Shenzhen or $5 + 2 = 8$

Disjunction (or)

- Let p and q be propositions. The *disjunction* of p and q, denoted by $p \vee q$, is false when both p and q are false and is true otherwise.

  Examples:

  ◇ p – SUSTech is located in Shenzhen.
  ◇ q – $5 + 2 = 8$

  ◇ $p \vee q$ – SUSTech is located in Shenzhen or $5 + 2 = 8$

  ◇ There are infinitely many twin prime numbers or $2 + 2 = 3$

  ◇ 2 is a prime number or 9 is a prime power.

Truth Tables

- Conjunction and disjunction

p	q	$p \land q$	$p \lor q$
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

Rows: all possible values of elementary propositions (2^n).

Exclusive or

- Let p and q be propositions. The *exclusive or* of p and q, denoted by $p \oplus q$, is true when exactly one of p and q is true and is false otherwise.

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

Conditional Statement (implication)

- Let p and q be propositions. The *conditional statment* (a.k.a. *implication*) $p \to q$, is the proposition "if p, then q", is false when p is true and q is false, and true otherwise. In $p \to q$, p is called the *hypothesis* and q is called the *conclusion*.

Conditional Statement (implication)

- Let p and q be propositions. The *conditional statment* (a.k.a. *implication*) $p \to q$, is the proposition "if p, then q", is false when p is true and q is false, and true otherwise. In $p \to q$, p is called the *hypothesis* and q is called the *conclusion*.

p	q	$p \to q$
T	T	T
T	F	F
F	T	T
F	F	T

Implication

- $p \rightarrow q$ is read in a variety of equivalent ways:
  - ◇ if p then q
  - ◇ p implies q
  - ◇ p is sufficient for q
  - ◇ q is necessary for p
  - ◇ q follows from p
  - ◇ q unless $\neg p$
  - ◇ p only if q

Implication

- $p \to q$ is read in a variety of equivalent ways:
  - ◇ if p then q
  - ◇ p implies q
  - ◇ p is sufficient for q
  - ◇ q is necessary for p
  - ◇ q follows from p
  - ◇ q unless $\neg p$
  - ◇ p only if q

Example:

◇ If you get 100 on the final, then you will get an A.

p q

Implication

- The *converse* of $p \to q$ is $q \to p$. 反命题
  The *contrapositive* of $p \to q$ is $\neg q \to \neg p$. 逆否命题
  The *inverse* of $p \to q$ is $\neg p \to \neg q$. 否命题

Implication

- The *converse* of $p \to q$ is $q \to p$.
  The *contrapositive* of $p \to q$ is $\neg q \to \neg p$.
  The *inverse* of $p \to q$ is $\neg p \to \neg q$.

  Examples:

  ◇ If you get 100 on the final, then you will get an A. ($p \to q$)

  ◇ If you get an A, then you get 100 on the final. ($q \to p$)
  ◇ If you don't get an A, then you don't get 100 on the final. ($\neg q \to \neg p$)
  ◇ If you don't get 100 on the final, then you don't get an A. ($\neg p \to \neg q$)

Implication

- The *converse* of $p \to q$ is $q \to p$.
  The *contrapositive* of $p \to q$ is $\neg q \to \neg p$.
  The *inverse* of $p \to q$ is $\neg p \to \neg q$.

Examples:

◇ If you get 100 on the final, then you will get an A. ($p \to q$)

◇ If you get an A, then you get 100 on the final. ($q \to p$)
◇ If you don't get an A, then you don't get 100 on the final. ($\neg q \to \neg p$)
◇ If you don't get 100 on the final, then you don't get an A. ($\neg p \to \neg q$)

$\neg q \to \neg p$ is *equivalent* to $p \to q$

Biconditional

- Let p and q be propositions. The *biconditional statment* (a.k.a. *bi-implications*) $p \leftrightarrow q$, is the proposition "p if and only if q", is true when p and q have the same truth values, and false otherwise.

Biconditional

- Let p and q be propositions. The *biconditional statment* (a.k.a. *bi-implications*) $p \leftrightarrow q$, is the proposition "p if and only if q", is true when p and q have the same truth values, and false otherwise.
  - p is necessary and sufficient for q
  - if p then q, and conversely
  - p iff q

Biconditional

- Let p and q be propositions. The *biconditional statment* (a.k.a. *bi-implications*) $p \leftrightarrow q$, is the proposition "p if and only if q", is true when p and q have the same truth values, and false otherwise.
  - ◇ p is necessary and sufficient for q
  - ◇ if p then q, and conversely
  - ◇ p iff q

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Compound Propositions

- A *proposition* is a declarative statement that is either true or false. More complex propositions can be built from elementary statements using logical connectives.

- Logical connectives:
  - $\neg p$ (*Negation*)
  - $p \wedge q$ (*Conjunction*)
  - $p \vee q$ (*Disjunction*)
  - $p \oplus q$ (*Exclusive or*)
  - $p \rightarrow q$ (*Implication*)
  - $p \leftrightarrow q$ (*Biconditional*)

Compound Propositions

- p: 2 is a prime (T)
  q: 6 is a prime (F)

Compound Propositions

- p: 2 is a prime (T)
  q: 6 is a prime (F)

- Determine the truth value of the following.

$\neg p$ F

$p \wedge q$ F

$p \wedge \neg q$ T

$p \vee q$ T

$p \oplus q$ T

$p \rightarrow q$ F

$q \rightarrow p$ T

Compound Propositions

- p: 2 is a prime (T)
  q: 6 is a prime (F)
- Determine the truth value of the following.

$\neg p$	F
$p \wedge q$	F
$p \wedge \neg q$	T
$p \vee q$	T
$p \oplus q$	T
$p \rightarrow q$	F
$q \rightarrow p$	T

Constructing the Truth Table

- Construct a truth table for $p \vee q \to \neg r$

Constructing the Truth Table

- Construct a truth table for $p \vee q \to \neg r$

p	q	r	$\neg r$	$p \vee q$	$p \vee q \to \neg r$
T	T	T	F	T	F
T	T	F	T	T	T
T	F	T	F	T	F
T	F	F	T	T	T
F	T	T	F	T	F
F	T	F	T	T	T
F	F	T	F	F	T
F	F	F	T	F	T

Computer Representation of True and False

- A *bit* is sufficient to represent two possible values: 0 (false) or 1 (true)
- A varaiable that takes on values 0 and 1 is called a *Boolean variable*.
- A *bit string* is a sequence of zero or more bits. The *length* of this string is the number of bits in the string.

Computer Representation of True and False

- A *bit* is sufficient to represent two possible values: 0 (false) or 1 (true)

- A varaiable that takes on values 0 and 1 is called a *Boolean variable*.

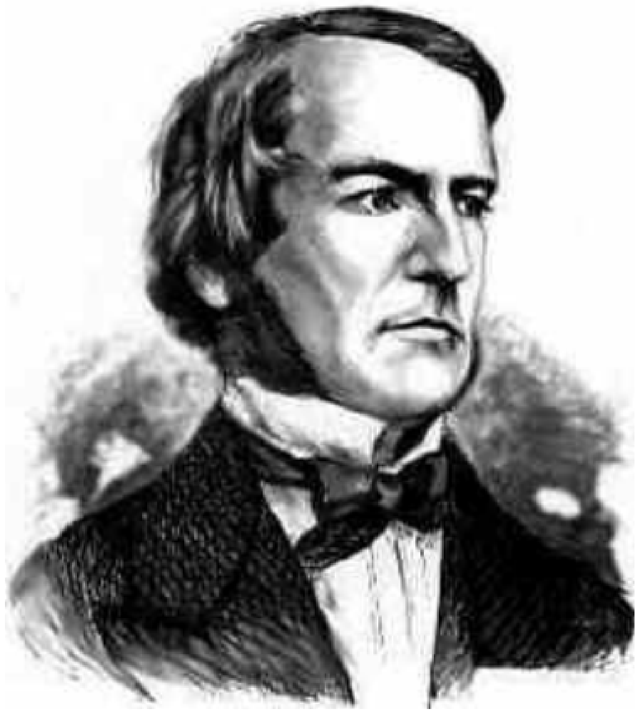
- A *bit string* is a sequence of zero or more bits. The *length* of this string is the number of bits in the string.

- *bitwise operations*: replace "T" and "F" with 1 and 0.

$$\begin{array}{r} 1011\ 0011 \\ \vee\ \underline{0110\ 1010} \\ 1111\ 1011 \end{array}$$

George Boole

- British mathematician (b. 1815, d. 1864)
  - ◇ The inventor of Boolean algebra
  - ◇ Truth tables are an example of B.A.



◇ Although Booles's work was not originally perceived as particularly interesting, even by other mathematicians, he is now seen as one of the founders of the field of Computer Science.

Next Lecture

- applications of propositional logic, predicate logic, ...