COMP 2012
Discrete Mathematics

Fall 2020 Dr. Ken YIU

Lecture: 11:30–13:30 @ *online*
every Wednesday

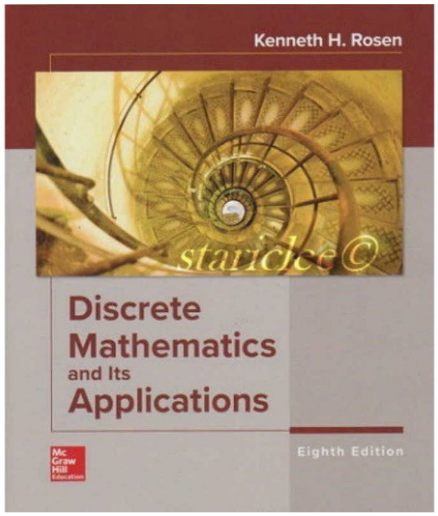
General Information

	Instructor	Teaching assistants
Name & Office	Dr. Ken YIU ☞ PQ 712	Mr. Zhe LI Mr. Jiaping CAO
Email	csmlyiu@comp.polyu.edu.hk	richie.li@connect.polyu.hk jiaping.cao@connect.polyu.hk
Website	*http://www.comp.polyu.edu.hk/~csmlyiu/*	

- Office hours
  - Please make appointment by email / phone

Readings

- Textbook
  - Rosen, K.H.,
    Discrete Mathematics and Its Applications,
    8th Edition, McGraw Hill, 2019.



You may either buy the latest version from bookstore
or borrow previous version(s) from library

Online Platforms

- Learn@PolyU (https://learn.polyu.edu.hk/)
  - For hosting content
    - E.g., lecture slides, tutorials, solutions
  - For submitting your answers
    - E.g., assignments, quizzes
  - Announcements
    - Email notifications will be sent to your PolyU email account (……@connect.polyu.hk)
    - Please check your email account regularly!
- Microsoft Teams
  - For conducting online lectures & online tutorials

Why do we study this course?

- Several areas in computer science originate from discrete mathematics
  - E.g., data structures, algorithms, cryptography, computation theory
- Learn the following abilities
  - Mathematical reasoning
  - Analyze a problem by using discrete math.
  - Solve a problem by using discrete math.
- Who can benefit from this course?
  - Software engineers, data scientists, computer scientists, ……

Tentative Schedule

No.	Topic
1	Introduction
2	Logic and Proofs
3	Basic Structures
4	Algorithm
5	Induction and Recursion [*Quiz 1*]
6	Counting
7	Graphs I

No.	Topic
8	Graphs II
9	Graphs III [*Quiz 2*]
10	Trees I
11	Trees II
12	Boolean Algebra and Circuits
13	Revision [*Quiz 3*]

Teaching Methods

- Lectures (2 hours)
  - Understand the main concepts
- Tutorials (1 hour)
  - Hands-on practice
- Assignments
  - Apply your skills
  - Solve problems

Benjamin Franklin's quote

"*Tell* me and I forget;

Teach me and I may remember;

Involve me and I learn."

Assessment

- Continuous Assessment: 60%
  - 3 Assignments: 36%
  - 3 Quizzes: 24%

- Final Exam*: 40%
  - Final take-home assignment

Late Policy

- You are expected to submit your solutions before deadline
  - A grace period (10 minutes) is given to accommodate network issues, i.e., no discount if your submission is within 10 minutes late

- Late submissions of quizzes & final take-home assignment won't be accepted
  - Beyond the grace period: 0 marks

- Late submissions of assignments:
  - < 24 hours late: deduct your score by 10%
  - Between 24 and 48 hours late: deduct your score by 20%
  - > 48 hours late: 0 marks

Plagiarism

- It is fine for you to discuss with classmates on known content (e.g., lectures, tutorial solutions)
- You must write up assignments & quizzes in your own words
  - Don't discuss with others about your answers!
- What will happen in a plagiarism case?
  - Both students (the one who copied & the one who provided solution) will get 0 marks
  - Serious cases will be reported to the university

Let’s start now!

Lecture 1
Introduction

Ken Yiu @ 2020

Our Roadmap

- What is discrete mathematics?
- Some history about discrete maths.
- Symbols & tools
- Logic & proofs
- Algorithms & problem solving
- From problems to knowledge

Discrete vs. Continuous

Topics in this course

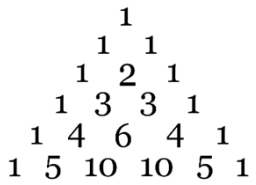
Set

$\{1, 3, 6\}$

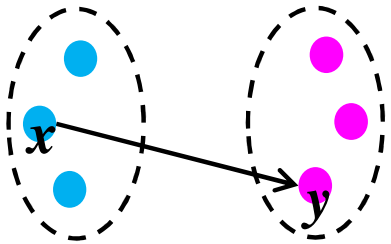
Logic & proofs

$\forall\, x \in \mathbb{N}\ (\ A(x) \rightarrow B(x)\ )$

Counting



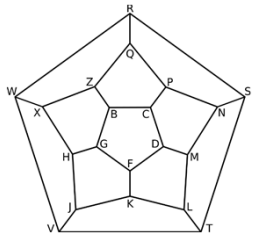
Function



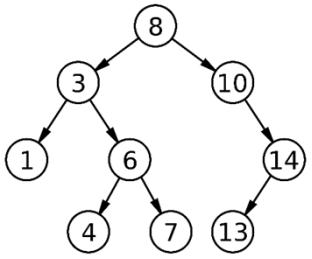
Algorithm

Induction

Graph



Tree



Boolean algebra

Beyond our scope

Discrete probability

Number theory

Computation theory

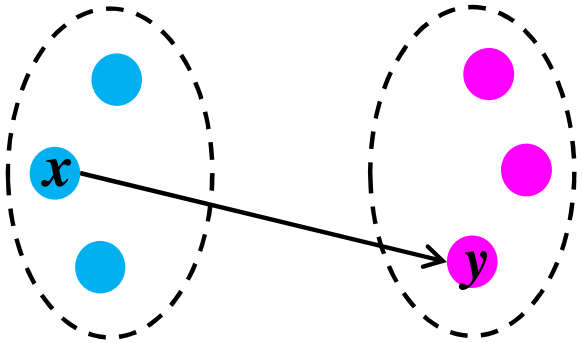
Complexity theory

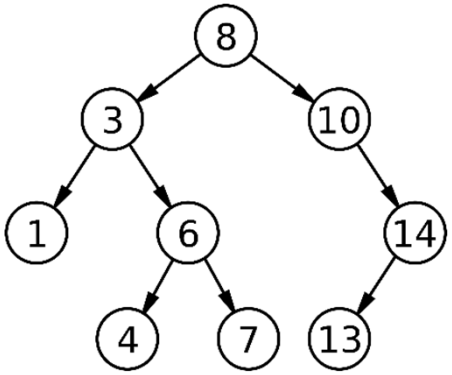
What is discrete mathematics?

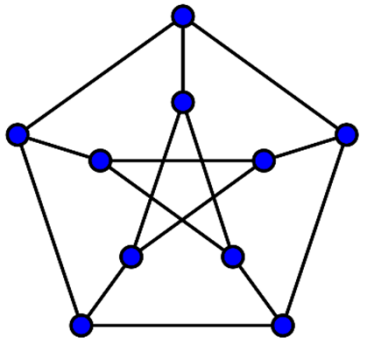
- Mathematics about integers
  (and concepts that can be built from integers)

- Examples of concepts:
  - Integer
  - Set, Sequence
  - Function
  - Graph, Tree
  - Statement, logic

$\{1, 3, 6\}$



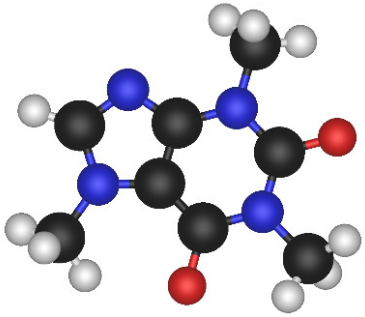


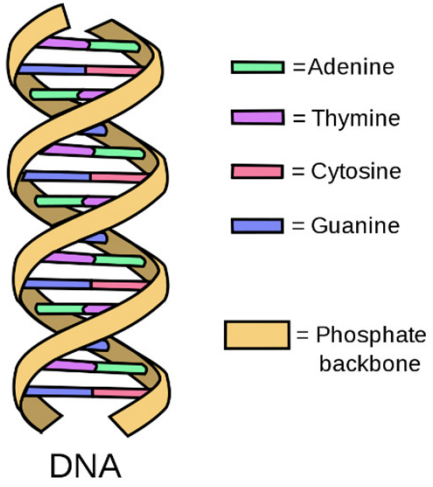


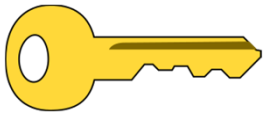
$$\forall\, x \in \mathbb{N}\ (\, A(x) \rightarrow B(x)\,)$$

Applications

- Networking
- Chemistry
- Engineering
- Linguistics
- Biology
- Internet
- ……







Types of questions in discrete math.

- 1. **Calculate** the result
- 2. **Test** whether object *X* satisfies requirement *Y*
- 3. **Construct** an object *X* so that it satisfies condition *Y*
- 4. **Prove** a statement
- 5. **Run** an algorithm (on graph/tree) and show the running steps (and the result)
- 6. **Analyze** an algorithm (correctness, running time)

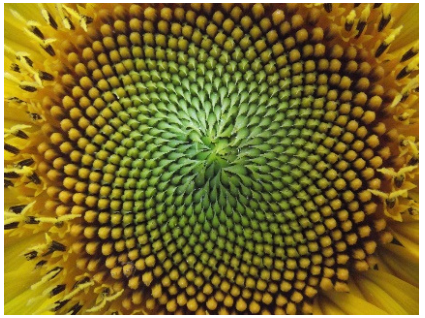
Our Roadmap

- What is discrete mathematics?

- Some history about discrete maths.
- Symbols & tools
- Logic & proofs
- Algorithms & problem solving
- From problems to knowledge

Some history about discrete maths.

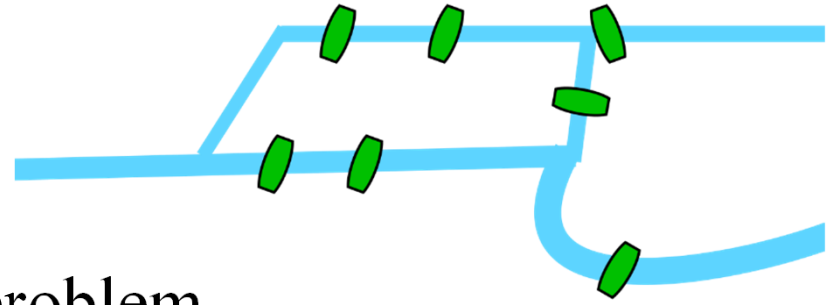
- Counting / Combinatorics
  - Pigeonhole principle
    - Dirichlet [1834]
  - Fibonacci sequence
    - Fibonacci [1202]
  - Pascal triangle
    - Blaise Pascal [1665]



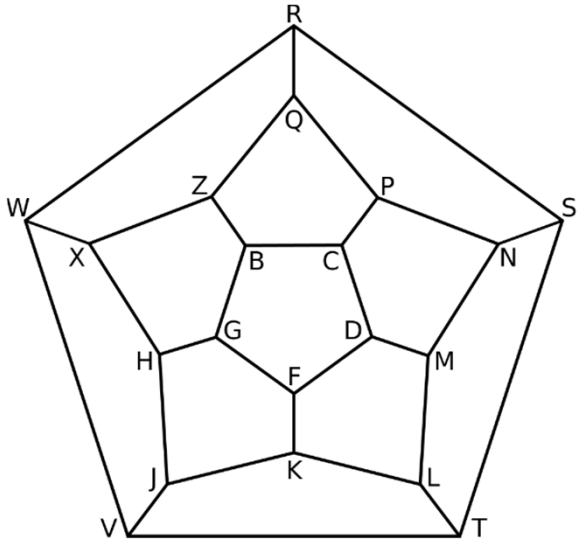
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1

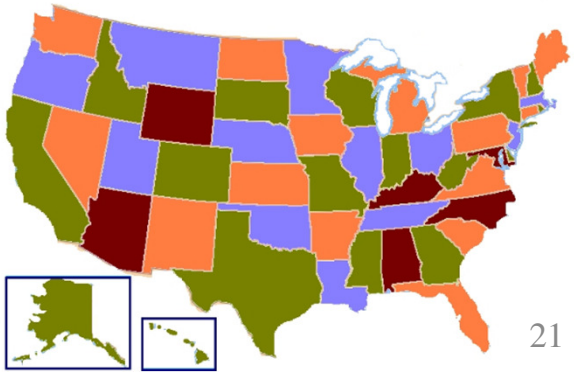
Some history about discrete maths.

- Graph Theory



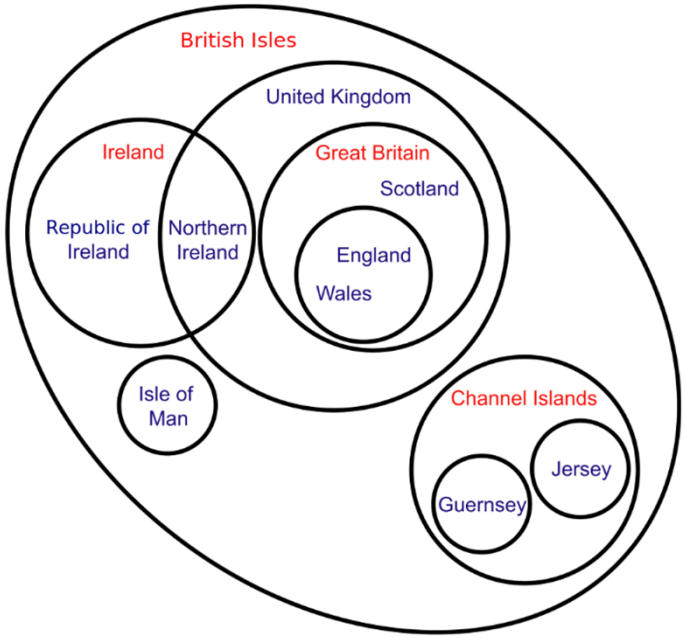
  - The Seven Bridges of Königsberg problem
    - Leonhard Euler [1736]
  - Hamilton's puzzle
    - William Rowan Hamilton [1857]
  - Four color conjecture / theorem
    - Conjected by Francis Guthrie [1852]
    - Proved by Kenneth Appel and Wolfgang Haken [1976]

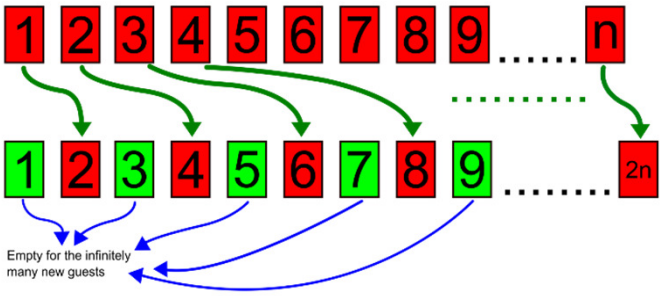




Some history about discrete maths.

- Set theory and mathematical logic
  - Proof techniques
    - Proof by contradiction [300 BC]
    - Induction [1665]
  - Boolean algebra
    - George Boole [1847]
  - Language for (mathematical) logic
    - Gottlob Frege [1879]
  - Venn diagram
    - John Venn [1880]
  - Hilbert's paradox of the Grand Hotel
    - David Hilbert [1924]





Our Roadmap

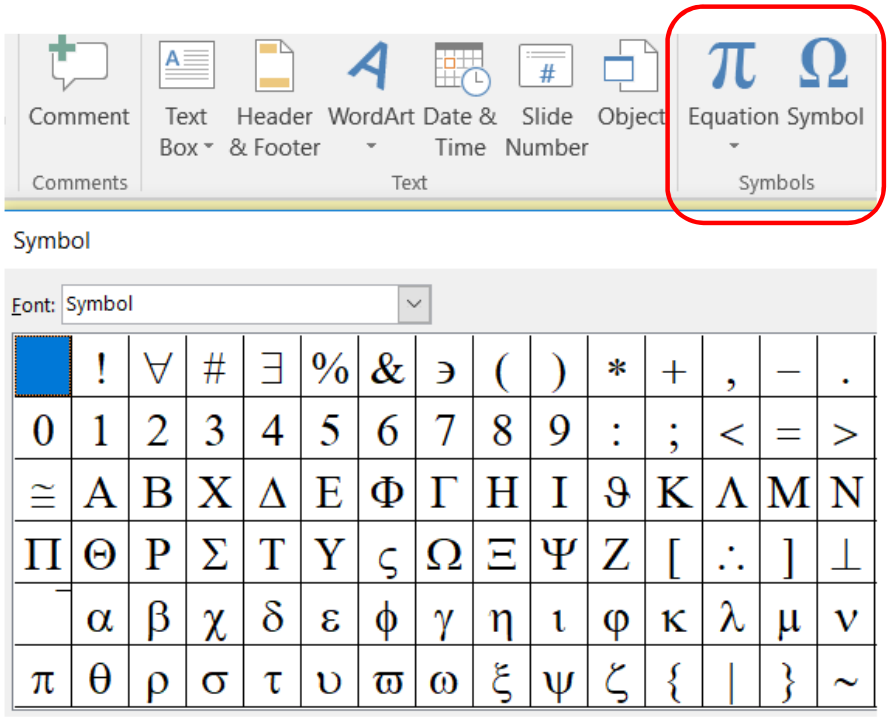
- What is discrete mathematics?
- Some history about discrete maths.
- Symbols & tools
- Logic & proofs
- Algorithms & problem solving
- From problems to knowledge

Commonly used in discrete mathematics

- Variables
  - a, b, c
- Superscripts
  - x^3, y^3
  - Often used to represent powers
- Subscripts
  - a_1, a_2, a_i, $x_{i,j}$
  - Often used to represent an array/matrix of variables
- Symbols
  - $\rightarrow$, $\leftrightarrow$, $\forall$, $\exists$, $\in$, $\cap$, ∞, $\sum$, $\prod$, $\mathbb{N}$, $\mathbb{Z}$
  - We will find out their meanings throughout this course

How to type equations or symbols?

- Microsoft Word / Powerpoint
  - Graphical user interface
  - You may click “Equation” or “Symbol”



- LaTeX
  - A markup language for writing a document
  - We can express an equation by text
    - E.g., $f(x)=x^2$
  - https://www.latex-tutorial.com/tutorials/amsmath/

Our Roadmap

- What is discrete mathematics?
- Some history about discrete maths.
- Symbols & tools
- Logic & proofs
- Algorithms & problem solving
- From problems to knowledge

Logic & proofs

- We may express the properties of some problems/methods as statements, e.g.,
  - (1) Every instance of problem *A* satisfies *B*.
  - (2) Every subpath $P_{i,j}$ of a shortest path $P_{a,b}$ is also a shortest path (from *i* to *j*).
  - (3) Algorithm *X* always returns the correct output for every instance of problem *A*.

- An argument like "*I believe … is true*" is not convincing!

- A *proof* (with logical arguments) is used to show that a given statement is always true

Logic & proofs

- Are these two statements equivalent? Why?
  - (A) If the #8 storm signal is issued today, then it is cloudy today.
  - (B) If it is not cloudy today, then the #8 storm signal is not issued today.

- How to prove or disprove the following statements?
  - (1) Let n be an integer. If $5n+4$ is odd, then n is odd.
  - (2) There exist non-zero integers x,y,z such that $x^2+y^2=z^2$
  - (3) Given that 8 people have birthday in the same week. There exists some day x such that at least two people have birthday on the same day.

Examples of proofs

- *Example*: show that the sum of two odd integers a, b is even
  - **Proof:**
  - Since a, b are even, there exist integers x, y such that $a=2x+1$ and $b=2y+1$
  - $a+b = 2x+1+2y+1$
    $= 2(x+y+1)$
  - Thus, their sum is even

- *Example*: Given that 8 people have birthday in the same week.
  There exists some day x so that at least two people have birthday on the same day.
  - **Proof:**
  - Use proof by contradiction, assume that each day has at most 1 person birthday
  - The total number of people $\leq 1*7=7$
  - This contradicts with the given number of people (8)!

Axiomatic Approach to Mathematics

- Example: define natural numbers and derive their properties
  - https://en.wikipedia.org/wiki/Peano_axioms
- Axiom
  - A statement assumed to be true (i.e., accepted without question)
  - E.g., 0 is a natural number
  - E.g., if x is a natural number, then its successor $S(x)$ is a natural number
- Theorem, Lemma, Corollary, Law
  - A statement proven to be true (by using axioms or proven theorems)
  - E.g., for any two natural numbers x, y, $x + y = y + x$
- Conjecture
  - A statement not yet proven to be true
  - E.g., for any even integer x greater than 5,
    there exist two prime numbers y, z such that $x = y + z$
    - We can verify it for limited cases (6, 8, 10, …, 1000000)
    - But how about all other cases?

Our Roadmap

- What is discrete mathematics?
- Some history about discrete maths.
- Symbols & tools
- Logic & proofs

- Algorithms & problem solving
- From problems to knowledge

In this course, we will also learn some existing **algorithms** to **solve** some real world **problems**

Sort cards

Find a fast path

Procedure of Sorting Cards

- *Problem*: Sort poker cards on hand
- Input:
  - A list of cards
- Procedure:
  - 1. **Put** all cards on the left hand
  - 2. **Pick** the smallest card from table
  - 3. **Move** that card to the right hand
  - 4. Repeat Steps 2-3 until the left hand is empty

Problem Solving

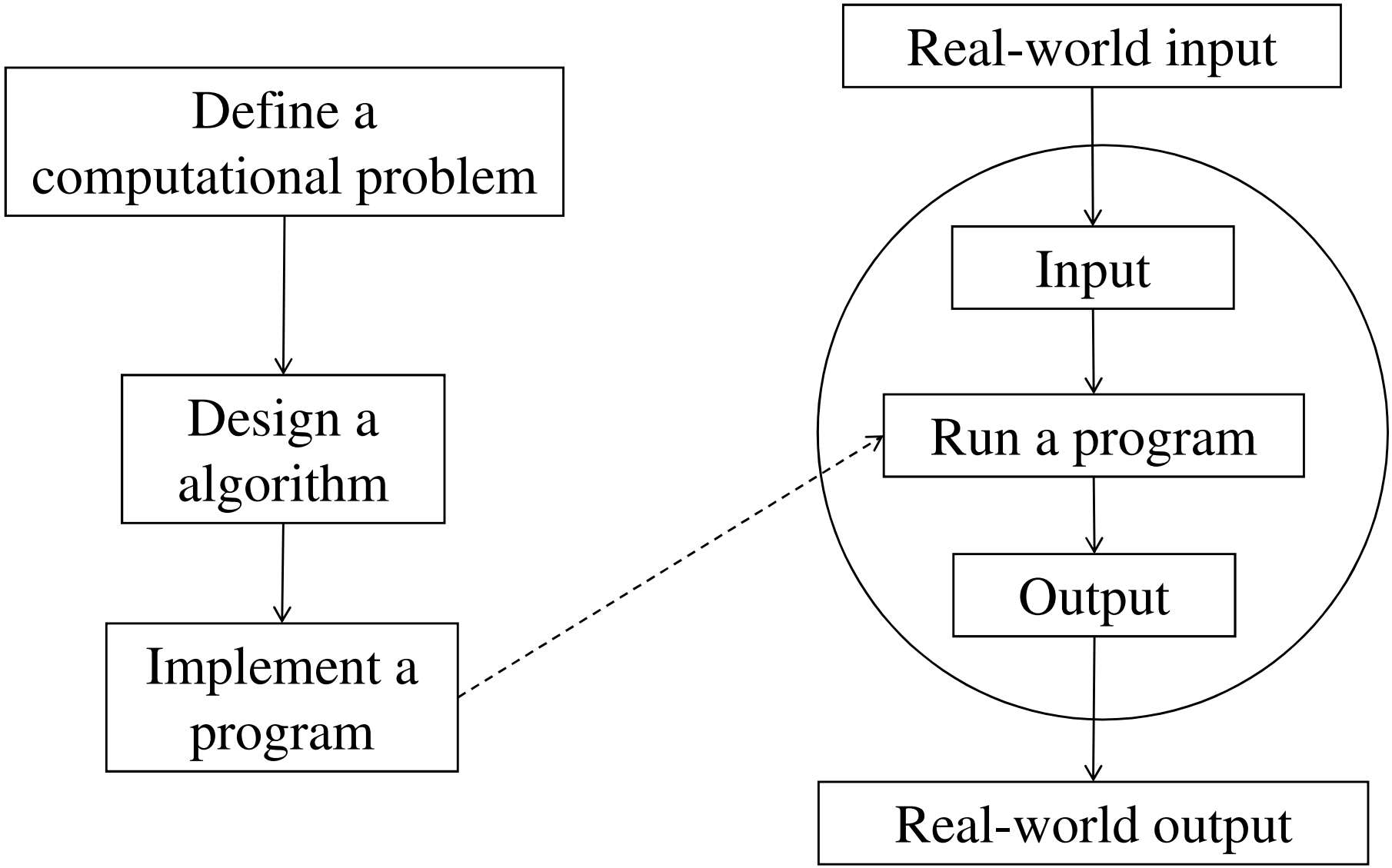
- Example: a sorting problem
  - Sort a set of cards by rank
  - Sort the student list by score
- How does a human solve a problem?
  - *Uses* brain, hands, tools
- How does a computer solve a problem?
  - *Uses* CPU, memory
  - *Primitive operations*: compare two integers, move an integer to memory cell X, etc

Algorithms

- *Algorithm*: a well defined sequence of steps for solving a computational problem
  - It produces the *correct output*
  - It uses *primitive* steps/defined operations
  - It finishes in *finite time*
- Algorithm is a template
  - It is more readable than a program
  - You can implement it in any language (Java, C++, etc.)

What are the *input*, *output*, and *steps* of a sorting algorithm?

Algorithms for Problem Solving



Algorithm vs. Program

- Algorithm is the *template* of program
  - It is *independent* of the programming language
  - It is more *readable* than a program

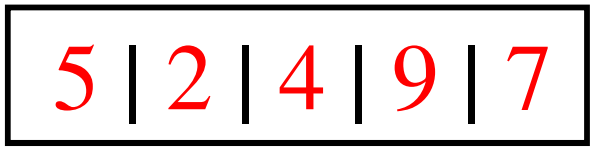
Algorithm

```
Selection-Sort ( Array A, Integer n )
1. for integer i ← 1 to n–1
2.       k ← i
3.       for integer j ← i+1 to n
4.             if   A[k] > A[j]  then
5.                   k ← j
6.       swap A[i] and A[k]
```

Java program

```
void Selection-Sort ( int[] A )
      int i, j, k, temp ;
      int n = A.length;
      for ( i=0 ; i<n–1; i++ ) {
            k = i ;
            for ( j=i+1 ; j<n; j++ )
                  if  ( A[k] > A[j] )
                        k = j ;
            int temp = A[i] ;
            A[i] = A[k] ;
            A[k] = temp ;
}
```

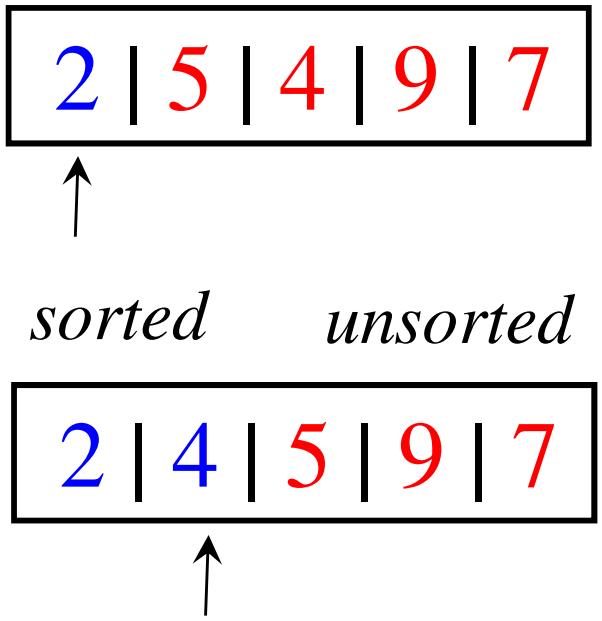
Algorithms



- Example: selection sort algorithm
  - Input: an **array** A of n numbers
  - Output : an **array** A of n numbers in the ascending order

  - Selection-Sort (A, n)

```
1. for integer i ← 1 to n–1
2.         k ← i
3.         for integer j ← i+1 to n
4.                 if   A[k] > A[j]  then
5.                         k ← j
6.          swap A[i] and A[k]
```



Algorithms: Running Steps

- Try to run an algorithm manually, e.g., draw running steps / figures
  - These *running steps* are useful for understanding the algorithm

Selection-Sort (Array A, Integer n)

1. for integer $i \leftarrow 1$ to $n-1$
2. $\quad k \leftarrow i$
3. $\quad$ for integer $j \leftarrow i+1$ to n
4. $\quad\quad$ if $A[k] > A[j]$ then
5. $\quad\quad\quad k \leftarrow j$
6. $\quad$ swap $A[i]$ and $A[k]$

input	5 | 2 | 4 | 9 | 7
$i = 1$	2 | 5 | 4 | 9 | 7
$i = 2$	2 | 4 | 5 | 9 | 7
$i = 3$	2 | 4 | 5 | 9 | 7
$i = 4$	2 | 4 | 5 | 7 | 9
$i = 5$	2 | 4 | 5 | 7 | 9

Algorithms: Analysis

- Is this sorting algorithm always correct? Why?

- Estimate the running time of this algorithm as a function of the input size n

Selection-Sort (Array A, Integer n)

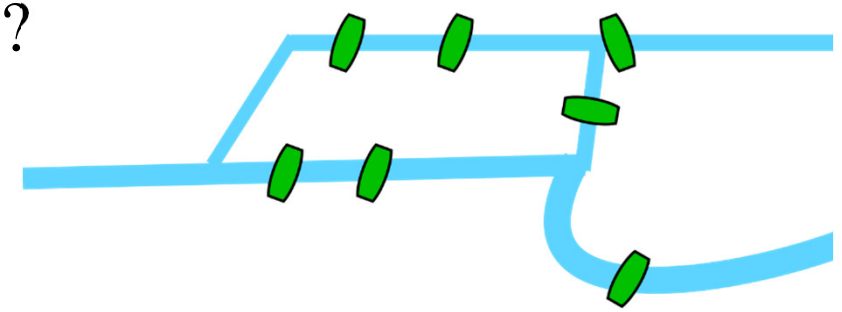
```
1. for integer i ← 1 to n–1
2.         k ← i
3.         for integer j ← i+1 to n
4.                 if   A[k] > A[j]  then
5.                         k ← j
6.         swap A[i] and A[k]
```

Our Roadmap

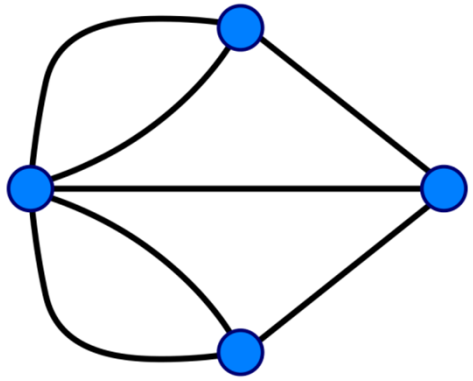
- What is discrete mathematics?
- Some history about discrete maths.
- Symbols & tools
- Logic & proofs
- Algorithms & problem solving
- From problems to knowledge

From Problems to Knowledge

- The Seven Bridges of Königsberg problem
  - Can you walk through this town such that
    (i) each bridge is crossed only once, and
    (ii) all parts of the town are visited ?

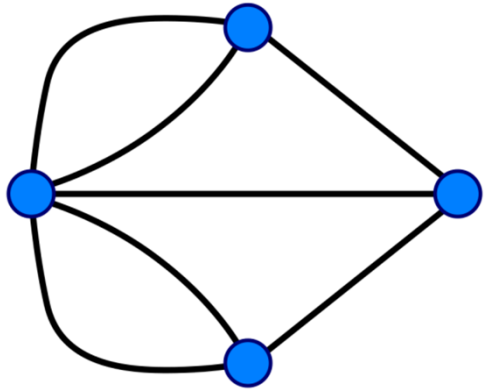


- Solution to this problem
  - 1. Use a graph to represent relevant features
    - Each edge represents a bridge
    - Each vertex represents a part of the town
    - Ignore irrelevant features (e.g., length of a bridge, the area of a region of the town



  - 2. Try all possible ways
  - Limitation: time consuming!

From Problems to Knowledge



- How to deal with a large graph?
- How to solve this problem **quickly**?
  - Check whether the graph is connected and it has exactly **zero or two nodes** of **odd degree**
    - The degree of a node is the number of edges touching it
  - Why is this condition correct?
- The first theorem in graph theory!

From Problems to Knowledge

Problem	*Knowledge*
How to model computation?	Turing machine
How to estimate the running time of an algorithm?	Time complexity analysis
How to classify problems based on their difficulties?	Complexity classes
Can we decide whether a program will terminate in finite time?	The existence of "undecidable problems"
……	……

Summary

- Overview of some concepts in discrete mathematics

- *Next lecture*: logic and proofs