❖ MATH_0480(Applied Discrete Mathematics)[1)]

Instructor : Jeungphill Hanne

❖ Agenda for today

1. SCUPI 2024 Fall Academic Calendar

- Academic Calendar : Midterms & Final etc.
- My Schedule : Office hours etc.

2. Course Introduction

Come from "Number theory" in Mathematics

- Course information
  - Subject, Text book, Lecture Hour, Office hour, Course website, etc.
- Course Objective & Scope, Course Learning Key Points
- Course Grading & Tentative Course Schedule

3. A bit Closer look on the **Discrete Mathematics**

- Briefly addressed by the key topics

1) : Recent curricular recommendations from The Institute for Electrical and Electronic Engineers Computer Society (IEEE-CS) and the Association for Computing Machinery (ACM) include discrete mathematics as the largest portion of "core knowledge" for computer science students and state that students should take at least a one-semester course in the subject as part of their first-year studies, with a two-semester course preferred when possible. (From the textbook)

1. SCUPI 2024 Fall Academic Calendar

- **Academic Calendar : Two Midterms & Final etc.**

SCUPI Academic Calendar for 2024-2025 Fall

	Aug.	Sep.				Oct.				Nov.					Dec.				Jan.					Feb.			
Monday	26	2	9	16	23	30	7	14	21	28	4	11	18	25	2	9	16	23	30	6	13	20	27	3	10	17	24
Tuesday	27	3	10	17	24	1	8	15	22	29	5	12	19	26	3	10	17	24	31	7	14	21	28	4	11	18	25
Wednesday	28	4	11	18	25	2	9	16	23	30	6	13	20	27	4	11	18	25	1	8	15	22	29	5	12	19	26
Thursday	29	5	12	19	26	3	10	17	24	31	7	14	21	28	5	12	19	26	2	9	16	23	30	6	13	20	27
Friday	30	6	13	20	27	4	11	18	25	1	8	15	22	29	6	13	20	27	3	10	17	24	31	7	14	21	28
Saturday	31	7	14	21	28	5	12	19	26	2	9	16	23	30	7	14	21	28	4	11	18	25	1	8	15	22	1
Sunday	1	8	15	22	29	6	13	20	27	3	10	17	24	1	8	15	22	29	5	12	19	26	2	9	16	23	2
SCU Week	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**
SCU Term	2024 Fall Teaching Weeks																			Final Weeks		Winter Recess					

1st Midterm

2nd Midterm

Final

This schedule is preliminary!!

1. SCUPI 2024 Fall Academic Calendar

- **My Schedule : Office hours etc.**

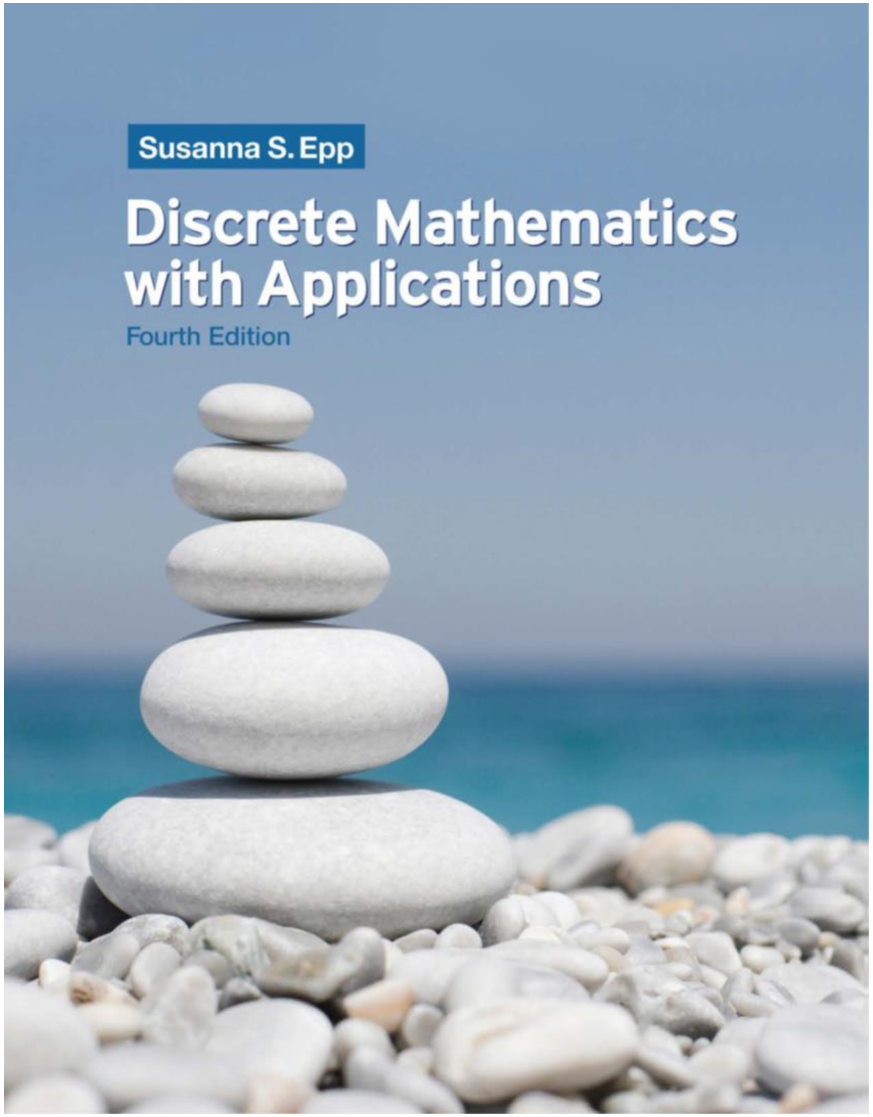
2024-2025 Fall Semester Course Schedule					
Class time	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**
08:15-09:00				Physics 1 05 Teach Bulg 1-A603	
09:10-09:55				Physics 1 05 Teach Bulg 1-A603	
10:15-11:00		Physics 1 05 Teach Bulg 1-A603		Office Hour Physics 1 05	
11:10-11:55		Physics 1 05 Teach Bulg 1-A603		Office Hour Applied Discrete Math	
Lunch Break					
13:50-14:35	Applied Discrete Math 3-106		Office Hour Applied Discrete Math		
14:45-15:30	Applied Discrete Math 3-106		Office Hour Physics 1 05		
15:40-16:25	Applied Discrete Math 3-106		Office Hour Physics 2 01		
16:45-17:30	Physics 2 01 3-101		Physics 2 01 3-101		
17:40-18:25	Physics 2 01 3-101		Physics 2 01 3-101		

*But, you can come to my office anytime when I am in my office ^^*

2. Course Introduction

• Course information

- **Applied Discrete Mathematics(MATH0480)**
- **Text Book**
  - **- Discrete Mathematics with Applications by Susanna S. Epp, 4th edition**
    : ISBN-13: 978-0-495-39132-6, ISBN-10: 0-495-39132-8
  - **- Hand outs (for Discrete Optimization)**
- **Lecture**
  - **-** Instructor : Jeungphill Hanne, PhD
    jeungphill.hanne@scupi.cn
  - Time : Please refer to my schedule
  - Office Hour: Wed(13:50-14:35), & Thr(11:10 -11:55)
  - Office : 412@New Building, Jiang'an South
- **TA :** Hanven Liu
  - Office Hrs : To be announced.
- **Course Format**
  - **-** Lecture, and Active Participation (i.e. Quiz, Question, Answers, etc.)
- **Course Grading**
  - **-** Two Midterms, Final, Homework, Quiz, and Attitude (ex. Attendance, Focus, Class Engagement, Punctuality for HW, etc.)



2. Course Introduction

• Course Scope & Objective

- Objective : To introduce the important discrete structures that appear in both pure and applied math as well as computer science, computer engineering, computer security and information systems, and thereby be able to offer the mathematical foundations on "Computer programming" while giving an glimpse on how it is originated from "Computer architecture" . In addition, this course is an excellent preparation for classes in Combinatorics, Graph Theory, Algebra and Number Theory.

- Topics or Scope :

- Logics of the Statements
- Basic Number theory & Mathematical proof
- Sequence & Mathematical Induction
- Sets, Functions & Growth of Functions
- Courting & Discrete Probability
- Graphs, Trees & Discrete geometry
- Analysis of Algorithm Efficiency
- Introduction on Discrete Optimization(Algorithms& Complexity, Network flows, Traveling Salesperson Problem, Revisited(Minimum spanning trees & Shortest path), the Knapsack problem etc.)

Come from "Number theory" in Mathematics

• Course Grading :

- Grading : HW+ Quiz (15~20%), Midterm I (25%), Midterm II (25%), Final (25%) and Attitude(5~10% : Attendance, Focus, Class Engagement (i.e. work on "practice problems"), Punctuality for HW, etc.)

→ Less than 60% attendance might be failed for the course!

Can be flexible!

• Tentative Course Schedule

Week	MATH_0408(ADM)	Topics	Assignment
Week 1 (9/2-9/8)	Introduction & Chap 1	**Syllabus, Overview & Mathematical Languages**	
Week 2 (9/9-9/15)	Chap 2	**The Logic of Compound Statements**	HW2
Week 3 (9/16-9/22)	Chap 2 & Chap3	**The Logic of Quantified Statements**	
Week 4 (9/23-9/29)	Chap 3 & Chap 4	**Elementary Number Theory and Methods of Proof**	HW3
Week 5 (9/30-10/6)	Chap 4		HW4
Week 6 (10/7-10/13)	Chap 5 & **Mid Term 1**	**Sequences, Mathematical Induction, and Recursion**	
Week 7 (10/14-10/20)	Chap 5		HW5
Week 8 (10/21-10/27)	Chap 6	**Set Theory**	HW6
Week 9 (10/28-11/3)	Chap 7	**Functions**	HW7
Week 10 (11/4-11/10)	Chap 8	**Relations**	HW8
Week 11 (11/11-11/17)	Chap 8 & Chap 9	**Counting and Probability**	
Week 12 (11/18-11/24)	Review & **Mid Term 2**		
Week 13 (11/25-12/1)	Chap 9		HW9
Week 14 (12/2-12/8)	Chap 10	**Graphs and Trees**	
Week 15 (12/9-12/15)	Chap 10 & Chap 11	**Analysis of Algorithm Efficiency**	HW10
Week 16 (12/16-12/22)	Chap 11		HW11
Week 17 (12/23-12/29)	Chap 12	**Regular Expressions and Finite-State Automata**	HW12
Week 18 (12/30-1/5)	Handouts	**Discrete Optimization I**	
Week 19 (1/6-1/12)	Handouts	**Discrete Optimization II**	HW13
Week 20 (1/13-1/20)	**Final**		

Can be flexible!

3. A bit Closer look on the Discrete Mathematics ← by several topics

• Discrete Mathematical Structures

→ Abstract structures describing, categorizing, and revealing the underlying relationships among discrete mathematical objects, which can be mathematically better understood by the subjects such as " Set theory" , " Logic & Boolean algebras", "Functions", Relations", "Graphs and Trees", etc.

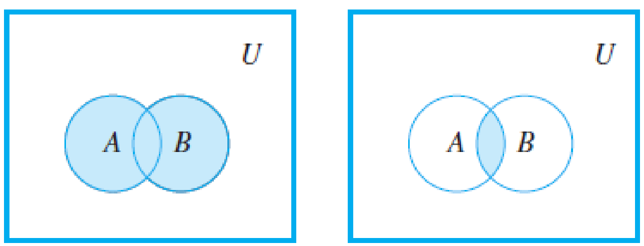
< "Set" examples w/ Venn diagram >

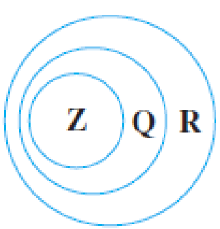
$A = \{m \in \mathbf{Z} \mid m = 2a \text{ for some integer } a\}$
$B = \{n \in \mathbf{Z} \mid n = 2b - 2 \text{ for some integer } b\}$

Chap 6

$A = \{m \in \mathbf{Z} \mid m = 6r + 12 \text{ for some } r \in \mathbf{Z}\}$
$B = \{n \in \mathbf{Z} \mid n = 3s \text{ for some } s \in \mathbf{Z}\}.$

"Abstract group" for … "be categorized"..





Z : Integers
Q :
R : Real numbers

< Logic & Boolean expression>

• **Logic**

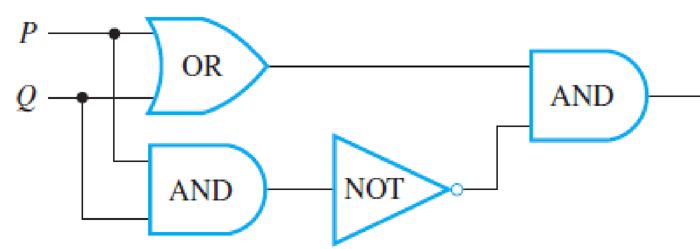
Chap 2&3

		conclusion		hypothesis	
p	q	$\sim p$	$\sim q$	$p \vee \sim q$	$p \vee \sim q \to \sim p$
T	T	F	F	T	F
T	F	F	T	T	F
F	T	T	F	F	T
F	F	T	T	T	T

"Truth", or "False" for …, "Various types of Statements"

One Foundation for "Mathematical proof" in "Algorithm"

• **Boolean expression**



3. A bit Closer look on the Discrete Mathematics

← by several topics

Chap 7 < "Functions" >

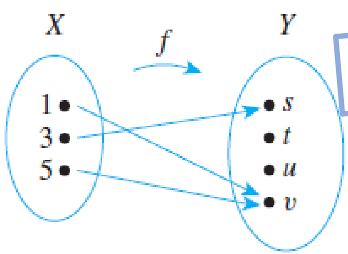
"Set" to "Set" relation!

< "Relations" > Chap 8

$f: \mathbf{R} \to \mathbf{R}$ is the function defined by the rule

$$\boxed{f(x) = 4x - 1 \quad \text{for all real numbers } x.}$$

Let $X = \{1, 3, 5\}$ and $Y = \{s, t, u, v\}$. Define $f: X \to Y$ by the following arrow diagram.



A Function by an Arrow Diagram

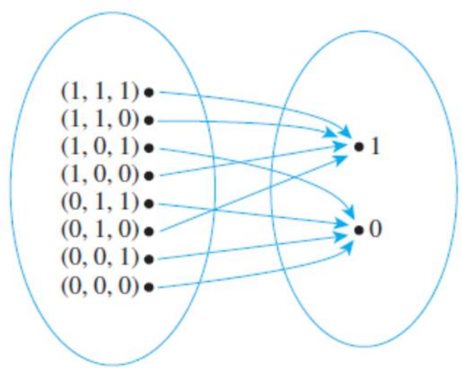
"Set" is a basic core for "Relation"

? Difference b/w "Functions" &"Relations"

Input			Output
P	Q	R	S
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	0
0	0	0	0

(a)

Two Representations of a Boolean Function



(b)

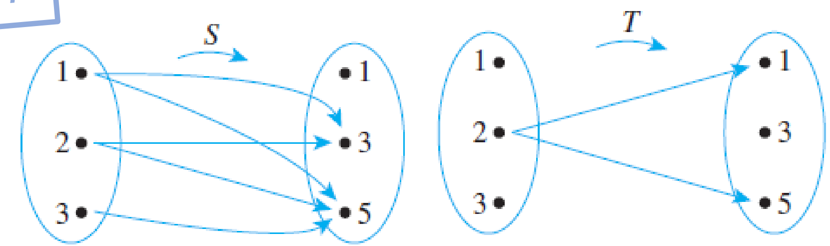
Define a relation L from $\mathbf{R}$ to $\mathbf{R}$ as follows:

For all real numbers x and y,

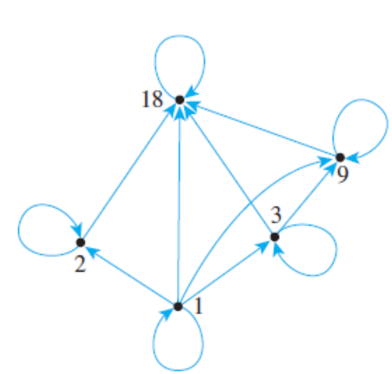
$$x \; L \; y \Leftrightarrow x < y.$$

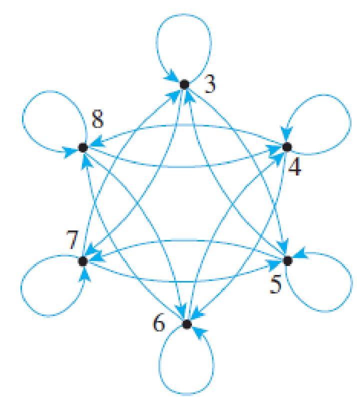
a. Is $57 \; L \; 53$? b. Is $(-17) \; L \; (-14)$?

Solution a. No, $57 > 53$ b. Yes, $-17 < -14$



Arrow Diagrams of Relations





Hasse Diagrams *Directed Graph of a Relation*

3. A bit Closer look on the Discrete Mathematics

← by several topics

< "Graphs" & "Trees" >

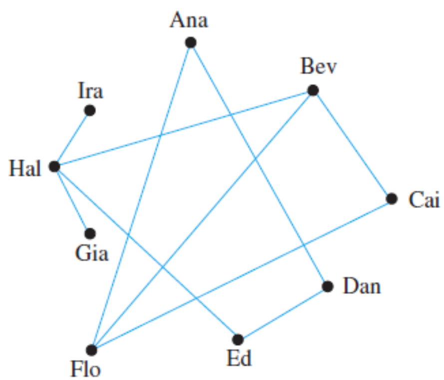
"Pictorial representations" for Set to Set relations

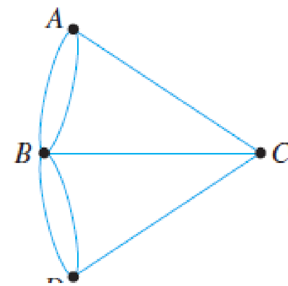
"Graphs" seen in "Relation diagrams" (Chap 8), and "Trees" for a "Possibility tree" In (Chap9)

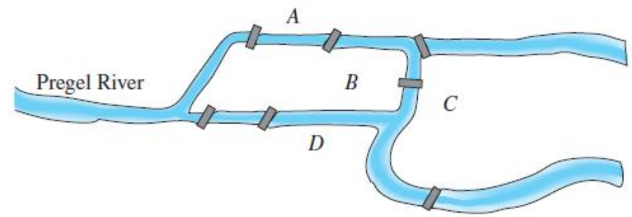
Chap 10 → further investigated with their mathematics & properties and be applied

- **Graphs**

Name	Past Partners
Ana	Dan, Flo
Bev	Cai, Flo, Hal
Cai	Bev, Flo
Dan	Ana, Ed
Ed	Dan, Hal
Flo	Cai, Bev, Ana
Gia	Hal
Hal	Gia, Ed, Bev, Ira
Ira	Hal

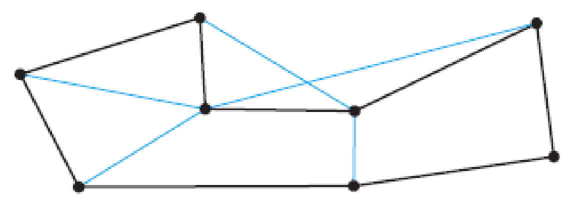






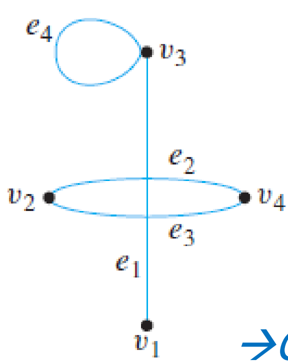
Graph Version of Königsberg Map

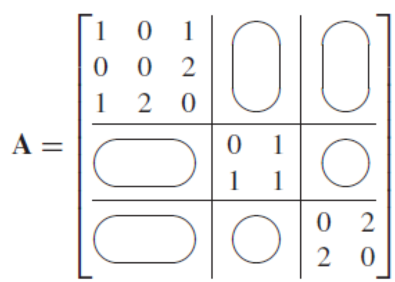
Hamiltonian Circuits



→ *Like "Pictorial connections"*

Edge	Endpoints
e_1	$\{v_1, v_3\}$
e_2	$\{v_2, v_4\}$
e_3	$\{v_2, v_4\}$
e_4	$\{v_3\}$

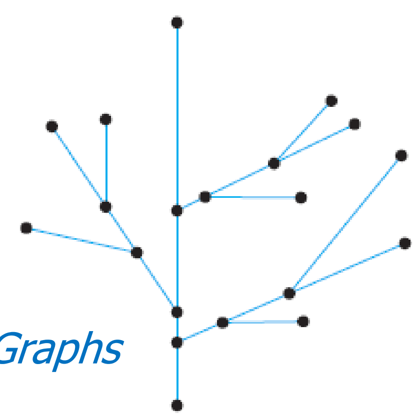




→Can be connected to "Matrix representation"

- **Trees**

→Circuit-free,
& connected Graphs



3. A bit Closer look on the **Discrete Mathematics**

← by several topics

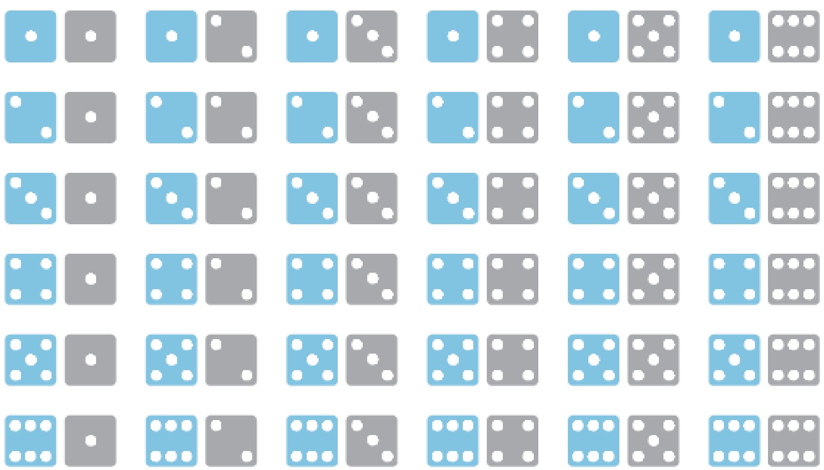
Like "*Application*" for "Set", "Relations" & "Functions"

- **Combinatory and Discrete probability**

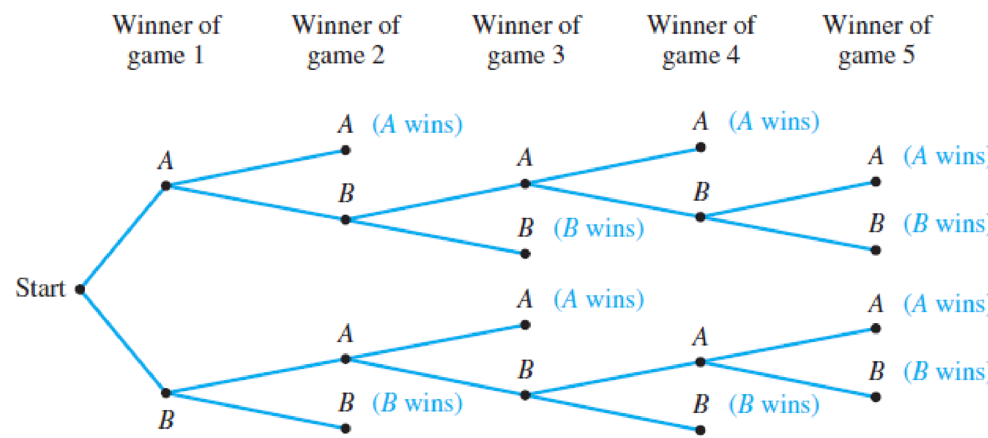
→ Combinatorics is the mathematics of counting and arranging objects, and probability is the study of laws concerning the measurement of random or chance events. Discrete probability focuses on situations involving discrete sets of objects, such as finding the likelihood of obtaining a certain number of heads when an unbiased coin is tossed a certain number of times. Skills from them is used in almost every discipline where mathematics is applied, from economics to biology, to computer science, to chemistry and physics, to business management" (from the text book)

Chap 9

- **Throwing the two dices**



- **Possibility Trees**



The Outcomes of a Tournamen

→Subsequent events can be drawn by a "Tree diagrams"

Equally Likely Probability Formula

If S is a finite sample space in which all outcomes are equally likely and E is an event in S, then the **probability of** E, denoted $P(E)$, is

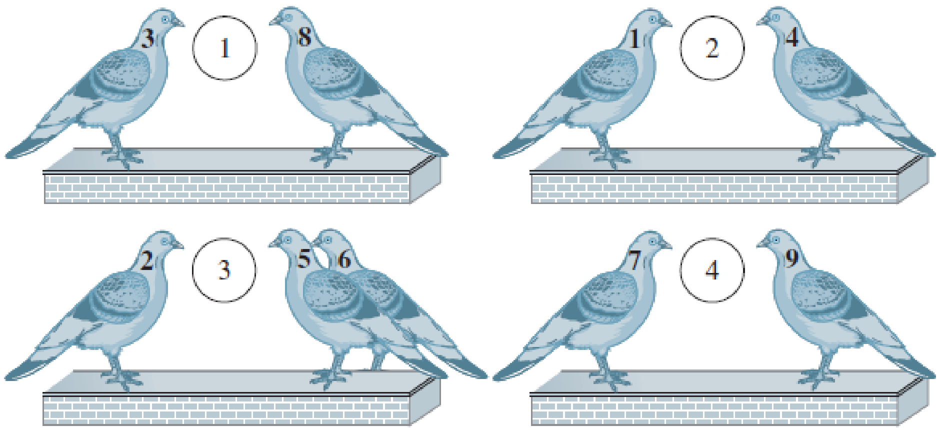
$$P(E) = \frac{\text{the number of outcomes in } E}{\text{the total number of outcomes in } S}.$$

3. A bit Closer look on the Discrete Mathematics

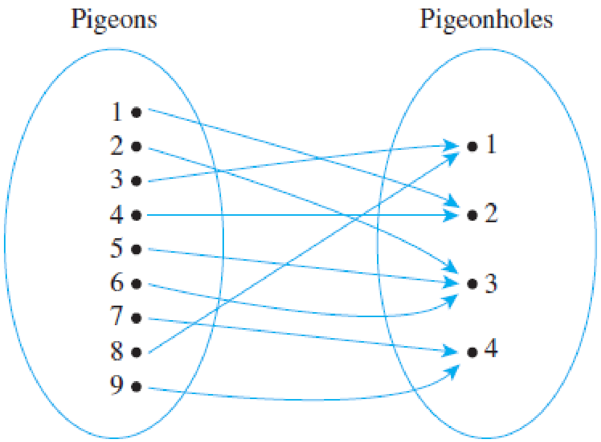
← by several topics

Like "Application" for "Set", "Relations" & "Functions"

- **Pigeonhole case or principle**



(a)



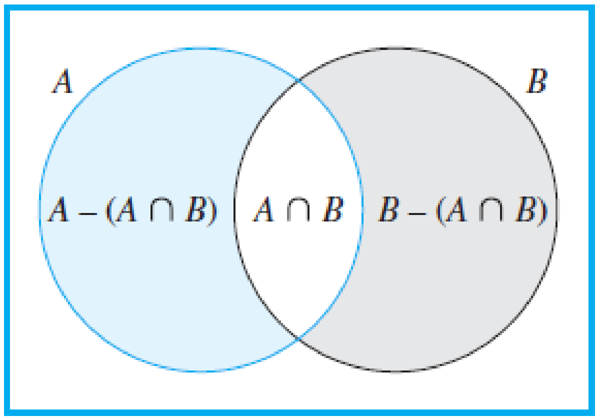
(b)

Represented by "Set" & "Functions"

Probability of a General Union of Two Events

If S is any sample space and A and B are any events in S, then

$$P(A \cup B) = P(A) + P(B) - P(A \cap B).$$



Can be expressed by "Set" property & diagram

3. A bit Closer look on Discrete Mathematics
← by several subjects

Chap 4&5

- **Mathematical Reasoning** & **Mathematical Induction**

→ An exciting development of recent years has been the increased appreciation for the power and beauty of **"recursive thinking."** To think recursively means to address a problem by assuming that similar problems of a smaller nature have already been solved and figuring out how to put those solutions together to solve the larger problem. Such thinking is widely used in the analysis of algorithms, where recurrence relations that result from recursive thinking often give rise to formulas that are verified by **mathematical induction** (from the text book)

Another Foundation for "Mathematical proof" in "Algorithm"

- ***"Algorithms" for computing → sequential structures, or sometimes "recursive"***

- **Sequence**

$$1, \quad -\frac{1}{4}, \quad \frac{1}{9}, \quad -\frac{1}{16}, \ldots$$

$$a_k = \frac{(-1)^k}{(k+1)} \quad \text{for all integers } k \geq 0.$$

- **Recursive relation**

(1) $m_k = 2m_{k-1} + 1$

(2) $m_1 = 1$

Principle of Mathematical Induction

Let $P(n)$ be a property that is defined for integers n, and let a be a fixed integer.

Suppose the following two statements are true:

1. $P(a)$ is true.
2. For all integers $k \geq a$, if $P(k)$ is true then $P(k+1)$ is true.

Then the statement

for all integers $n \geq a$, $P(n)$

is true.

- ***Sequences in Computer Programming***

An important data type in computer programming consists of finite sequences. In computer programming contexts, these are usually referred to as *one-dimensional arrays*. For example, consider a program that analyzes the wages paid to a sample of 50 workers. Such a program might compute the average wage and the difference between each individual wage and the average. This would require that each wage be stored in memory for retrieval later in the calculation. To avoid the use of entirely separate variable names for all of the 50 wages, each is written as a term of a one-dimensional array:

W[1], W[2], W[3], . . . , W[50].

Note that the subscript labels are written inside square brackets. The reason is that until relatively recently, it was impossible to type actual dropped subscripts on most computer keyboards.

- ***Recursively Defined Sets***

I. BASE: A statement that certain objects belong to the set.

II. RECURSION: A collection of rules indicating how to form new set objects from those already known to be in the set.

III. RESTRICTION: A statement that no objects belong to the set other than those coming from I and II.

- ***Application: Correctness of Algorithms***

3. A bit Closer look on the Applied Discrete Mathematics
← by several topics

Chap 4, 5 & Chap 11

- **Algorithms** & **Their Analysis**

→ To solve a problem on a computer, it is necessary to find **an algorithm** or **step-by-step sequence** of instructions for the computer to follow. Designing an algorithm requires an understanding of the mathematics underlying the problem to be solved. Determining whether or not an algorithm is correct requires a sophisticated use of mathematical induction. Calculating the amount of time or memory space the algorithm will need in order to compare it to other algorithms that produce the same output requires knowledge of combinatorics, recurrence relations, functions, and O-, -, and –notations (from the text book)

- Ex) *The Euclidean Algorithm*

the greatest common divisor of two integers.

• Definition

Let a and b be integers that are not both zero. The **greatest common divisor** of a and b, denoted $\mathbf{gcd}(a, b)$, is that integer d with the following properties:

1. d is a common divisor of both a and b. In other words,
$$d \mid a \quad \text{and} \quad d \mid b.$$
2. For all integers c, if c is a common divisor of both a and b, then c is less than or equal to d. In other words,
$$\text{for all integers } c, \text{ if } c \mid a \text{ and } c \mid b, \text{ then } c \le d.$$

Lemma 4.8.1

If r is a positive integer, then $\gcd(r, 0) = r$.

Lemma 4.8.2

If a and b are any integers not both zero, and if q and r are any integers such that
$$a = bq + r,$$
then
$$\gcd(a, b) = \gcd(b, r).$$

Algorithm 4.8.2 Euclidean Algorithm

[Given two integers A and B with $A > B \ge 0$, this algorithm computes $\gcd(A, B)$. It is based on two facts:

1. $\gcd(a, b) = \gcd(b, r)$ *if a, b, q, and r are integers with $a = b \cdot q + r$ and $0 \le r < b$.*
2. $\gcd(a, 0) = a$.*]*

Input: A, B *[integers with $A > B \ge 0$]*

Algorithm Body:

$a := A, b := B, r := B$

[If $b \ne 0$, compute $a \bmod b$, the remainder of the integer division of a by b, and set r equal to this value. Then repeat the process using b in place of a and r in place of b.]

while $(b \ne 0)$

$r := a \bmod b$

[The value of $a \bmod b$ can be obtained by calling the division algorithm.]

$a := b$

$b := r$

end while

*[After execution of the **while** loop, $\gcd(A, B) = a$.]*

$\gcd := a$

Output: gcd *[a positive integer]*

3. A bit Closer look on the **Applied Discrete Mathematics**

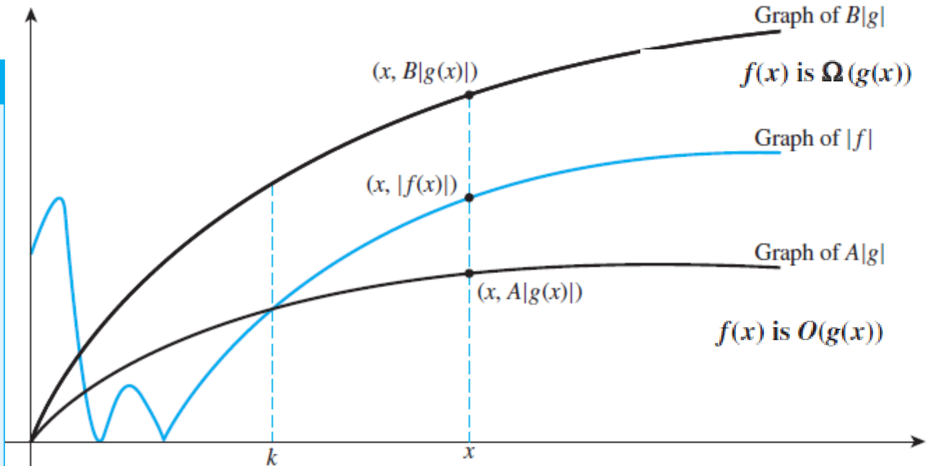
← by several topics

Chap 4, 5 & Chap 11

- **Algorithms** & **Their Analysis**

The **analytic geometry** of Descartes provides the foundation on the important subjects for an analysis of "Algorithm efficiency": **Θ, Ω, O, notations** (from the text book)

- **Θ, Ω, O, notations**



- **Ex)** *Time Efficiency of an Algorithm*

• Definition

Let A be an algorithm.

1. Suppose the number of elementary operations performed when A is executed for an input of size n depends on n alone and not on the nature of the input data; say it equals $f(n)$. If $f(n)$ is $\Theta(g(n))$, we say that **A is $\Theta(g(n))$** or **A is of order $g(n)$.**
2. Suppose the number of elementary operations performed when A is executed for an input of size n depends on the nature of the input data as well as on n.
   a. Let $b(n)$ be the *minimum* number of elementary operations required to execute A for all possible input sets of size n. If $b(n)$ is $\Theta(g(n))$, we say that **in the best case, A is $\Theta(g(n))$** or **A has a best-case order of $g(n)$.**
   b. Let $w(n)$ be the *maximum* number of elementary operations required to execute A for all possible input sets of size n. If $w(n)$ is $\Theta(g(n))$, we say that **in the worst case, A is $\Theta(g(n))$** or **A has a worst-case order of $g(n)$.**

Table 11.3.1 Time Comparisons of Some Algorithm Orders

Approximate Time to Execute $f(n)$ Operations Assuming One Operation per Nanosecond*				
$f(n)$	$n = 10$	$n = 1{,}000$	$n = 100{,}000$	$n = 10{,}000{,}000$
$\log_2 n$	3.3×10^{-9} sec	10^{-8} sec	1.7×10^{-8} sec	2.3×10^{-8} sec
n	10^{-8} sec	10^{-6} sec	0.0001 sec	0.01 sec
$n \log_2 n$	3.3×10^{-8} sec	10^{-5} sec	0.0017 sec	0.23 sec
n^2	10^{-7} sec	0.001 sec	10 sec	27.8 min
n^3	10^{-6} sec	1 sec	11.6 days	31,688 yr
2^n	10^{-6} sec	3.4×10^{284} yr	3.1×10^{30086} yr	2.9×10^{3010283} yr

*one nanosecond $= 10^{-9}$ second

• Definition

Let f and g be real-valued functions defined on the same set of nonnegative real numbers. Then

1. **f is of order at least g**, written **$f(x)$ is $\Omega(g(x))$**, if, and only if, there exist a positive real number A and a nonnegative real number a such that
$$A|g(x)| \le |f(x)| \quad \text{for all real numbers } x > a.$$
2. **f is of order at most g**, written **$f(x)$ is $O(g(x))$**, if, and only if, there exist a positive real number B and a nonnegative real number b such that
$$|f(x)| \le B|g(x)| \quad \text{for all real numbers } x > b.$$
3. **f is of order g**, written **$f(x)$ is $\Theta(g(x))$**, if, and only if, there exist a positive real number A, B, and a nonnegative real number k such that
$$A|g(x)| \le |f(x)| \le B|g(x)| \quad \text{for all real numbers } x > k.$$

3. A bit Closer look on the Applied Discrete Mathematics
← by several topics

"Hand outs"

- **Algorithms Complexity** & **Discrete Optimization**

→ For the complex algorithms, Discrete optimization is an "approach" to find the best solution out of finite number of possibilities in a computationally efficient way. And here will show the several examples for how some algorithms are optimized (modified from a reference [2)])

- Can be studied , or practiced through the examples (problems encountered), as follows.

- **Minimum spanning trees**
- **The Shortest Path problem**
- **Traveling Salesperson Problem**
- **Network flows** (**Maximum flows, Min Cost flows, etc.)**
- **Optimal Matchings**
- **The Knapsack problem**
- **Integer Programming**
- **NP and NP-complete problem**
- **Matroid**
-
-
-

"Discrete Optimization examples"

2) : Discrete Optimization, Spring 2020 , Thomas Rothvoss, University of Washington

Any question so far?

? Why are we taking “this course” for “what” ?

And let’s move on Chap 1 !

“Speaking Mathematically”

❖ Chapter 1 Speaking Mathematically

1–1 Variables

1–2 The Languages of Sets

1–3 The Languages of Relations and Functions

"Briefly introduced in the text book", and we did last time

→ Will be studied in more detail later

& No HW this time!

• 1–1 Variables

1) Three Basic Mathematical Statements

- Statement : "Expressed for something, some situation, …..

• **A universal statement** says that a certain property is true for all elements in a set. (For example: *All positive numbers are greater than zero.*)

∀

→ Stated by "all", "every", any"… : notated by „∀"

• **A conditional statement** says that if one thing is true then some other thing also has to be true. (For example: *If 378 is divisible by 18, then 378 is divisible by 6.*)

P Q

→ Stated with "If P, (then) Q", : notated by $P \rightarrow Q$

• **Existential statement**

Given a property that may or may not be true, an **existential statement** says that there is at least one thing for which the property is true. (For example: *There is a prime number that is even.*)

∃

→ Stated with "There is…", "There exist…", "some", "has", …. : notated by „∃"

In each of 8–13, fill in the blanks to rewrite the given statement.

9. For all equations E, if E is quadratic then E has at most two real solutions.
   a. All quadratic equations ____.
   b. Every quadratic equation ____.
   c. If an equation is quadratic, then it ____.
   d. If E ____, then E ____.
   e. For all quadratic equations E, ____.

• 1–1 Variables

2) Mixed from the Basic Mathematical Statements

∀ P → Q

• *Universal Conditional Statements* For all animals a, if a is a dog, then a is a mammal.

If an animal is a dog, then the animal is a mammal.

If a is a dog, then a is a mammal.

"P → Q" : Implicit

For all dogs a, a is a mammal.

All dogs are mammals.

∀ ∃

• *Universal Existential Statements* Every real number has an additive inverse.

For all real numbers r, there is a real number s such that s is an additive inverse for r.

For all real numbers r, there is an additive inverse for r.

All real numbers have additive inverses.

∃ ∀

• *Existential Universal Statements*

There is a positive integer that is less than or equal to every positive integer

Some positive integer is less than or equal to every positive integer.

There is a positive integer m that is less than or equal to every positive integer.

There is a positive integer m such that every positive integer is greater than or equal to m.

There is a positive integer m with the property that for all positive integers n, $m \leq n$.

• ∃ P → Q

? "Existential Conditional Statements"

2) Mixed from the Basic Mathematical Statements

Example 1.1.2 Rewriting a Universal Conditional Statement

Fill in the blanks to rewrite the following statement:

For all real numbers x, if x is nonzero then x^2 is positive.

a. If a real number is nonzero, then its square ____.

b. For all nonzero real numbers x, ____.

c. If x ____, then ____.

d. The square of any nonzero real number is ____.

e. All nonzero real numbers have ____.

Example 1.1.3 Rewriting a Universal Existential Statement

Fill in the blanks to rewrite the following statement: Every pot has a lid.

a. All pots ____.

b. For all pots P, there is ____.

c. For all pots P, there is a lid L such that ____.

• 1–1 Variables

2) Mixed from the Basic Mathematical Statements

Example 1.1.4 Rewriting an Existential Universal Statement

Fill in the blanks to rewrite the following statement in three different ways:

There is a person in my class who is at least as old as every person in my class.

a. Some ____ is at least as old as ____.

b. There is a person p in my class such that p is ____.

c. There is a person p in my class with the property that for every person q in my class, p is ____.

• $\exists$ $\forall$ $P \rightarrow Q$

All mixed w/ "Existential Universal Conditional Statements"

- Ex) Definition of a mathematical "limit" of a sequence "a_n" : $\lim_{n\to\infty} a_n = L$

if $a_1, a_2, a_3, \ldots$ is a sequence of real numbers, the limit of a_n as n approaches infinity is L

$\forall$ $\exists$

for all $\boxed{\text{positive}}$ $\boxed{\text{real numbers}}$ ε, **there is** an $\boxed{\text{integer}}$ N such that **for all** integers n, **if** $n > N$ **then** $-\varepsilon < a_n - L < \varepsilon$.

$\forall$

: "maybe Difficult, but a very solid definition"

So, need to "Think" & "Understand"

"More used by the mathematical notations"

$\boxed{}$ **: Also needed to be defined "mathematically"**

Any question?

So, in the next time
Start with Chap 2,
“The Logic of Compound Statements”