Computer Science 601.231
Automata and Computation Theory

Xin Li

lixints@cs.jhu.edu

Johns Hopkins University

Background: computation

- Computation is closely related to mathematics.
- Mathematicians have been trying to find informal "algorithms" for centuries, from ancient Greece.
- Euclidean algorithm: Given two positive integers (a, b), find their largest common divisor.

Euclid
(325–265 BC)

$$q_1 = \left\lfloor \frac{a}{b} \right\rfloor \qquad a = b\,q_1 + r_1 \qquad r_1 = a - b\,q_1$$

$$q_2 = \left\lfloor \frac{b}{r_1} \right\rfloor \qquad b = q_2\,r_1 + r_2 \qquad r_2 = b - q_2\,r_1$$

$$q_3 = \left\lfloor \frac{r_1}{r_2} \right\rfloor \qquad r_1 = q_3\,r_2 + r_3 \qquad r_3 = r_1 - q_3\,r_2$$

$$q_4 = \left\lfloor \frac{r_2}{r_3} \right\rfloor \qquad r_2 = q_4\,r_3 + r_4 \qquad r_4 = r_2 - q_4\,r_3$$

$$q_n = \left\lfloor \frac{r_{n-2}}{r_{n-1}} \right\rfloor \qquad r_{n-2} = q_n\,r_{n-1} + r_n \qquad r_n = r_{n-2} - q_n\,r_{n-1}$$

$$q_{n+1} = \left\lfloor \frac{r_{n-1}}{r_n} \right\rfloor \qquad r_{n-1} = q_{n+1}\,r_n + 0 \qquad r_n = r_{n-1} / q_{n+1}$$

Sieve of Eratosthenes

- Find all prime numbers up to a given number.
- Iteratively remove the multiples of each prime, starting with the first prime number 2.

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

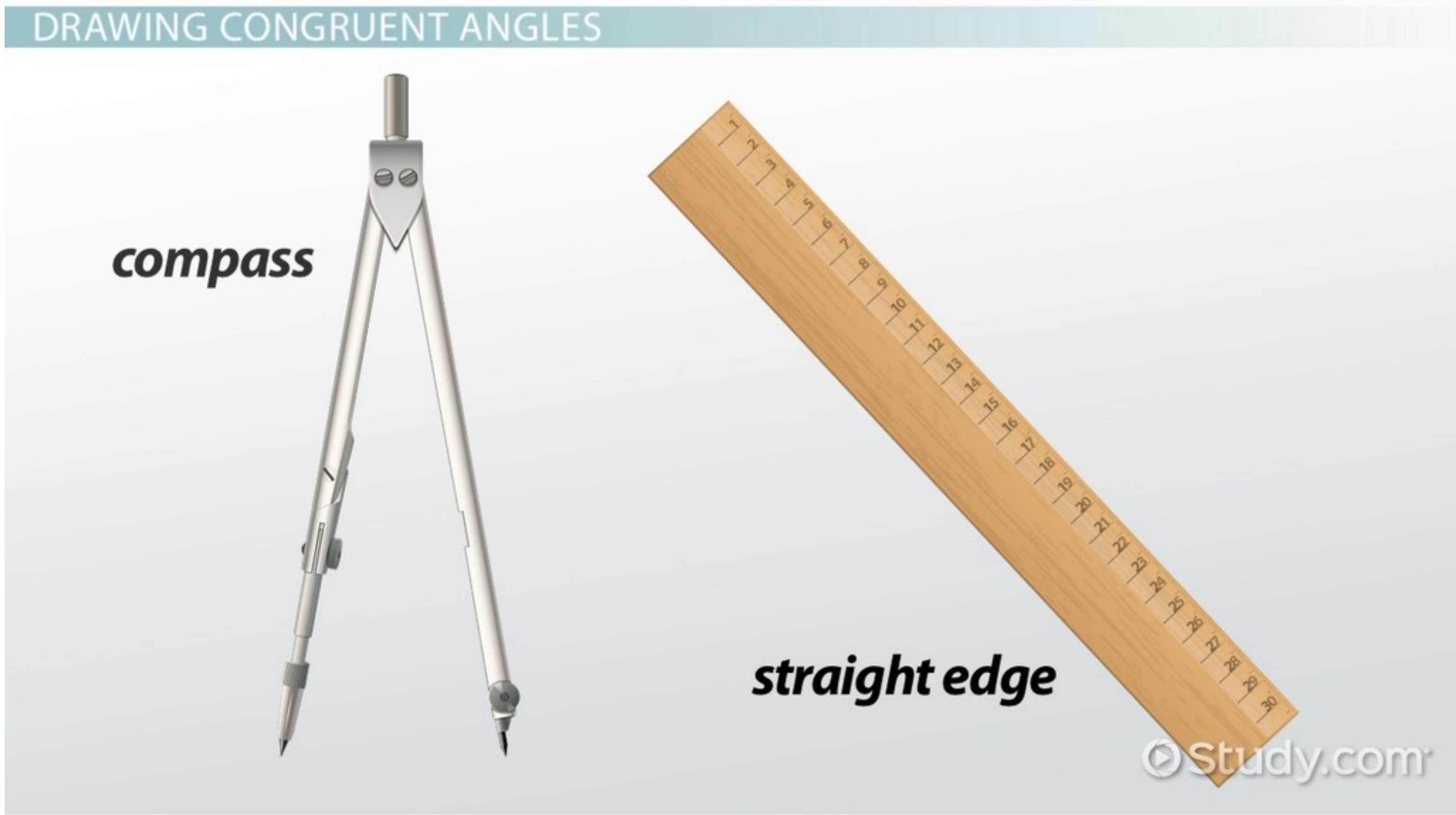
2 3 X 5 X 7 X 9 X 11 X 13 X 15 X 17 X

2 3 X 5 X 7 X X X 11 X 13 X X X 17 X

Many variants and improvements by Euler, Sundaram, Atkin…

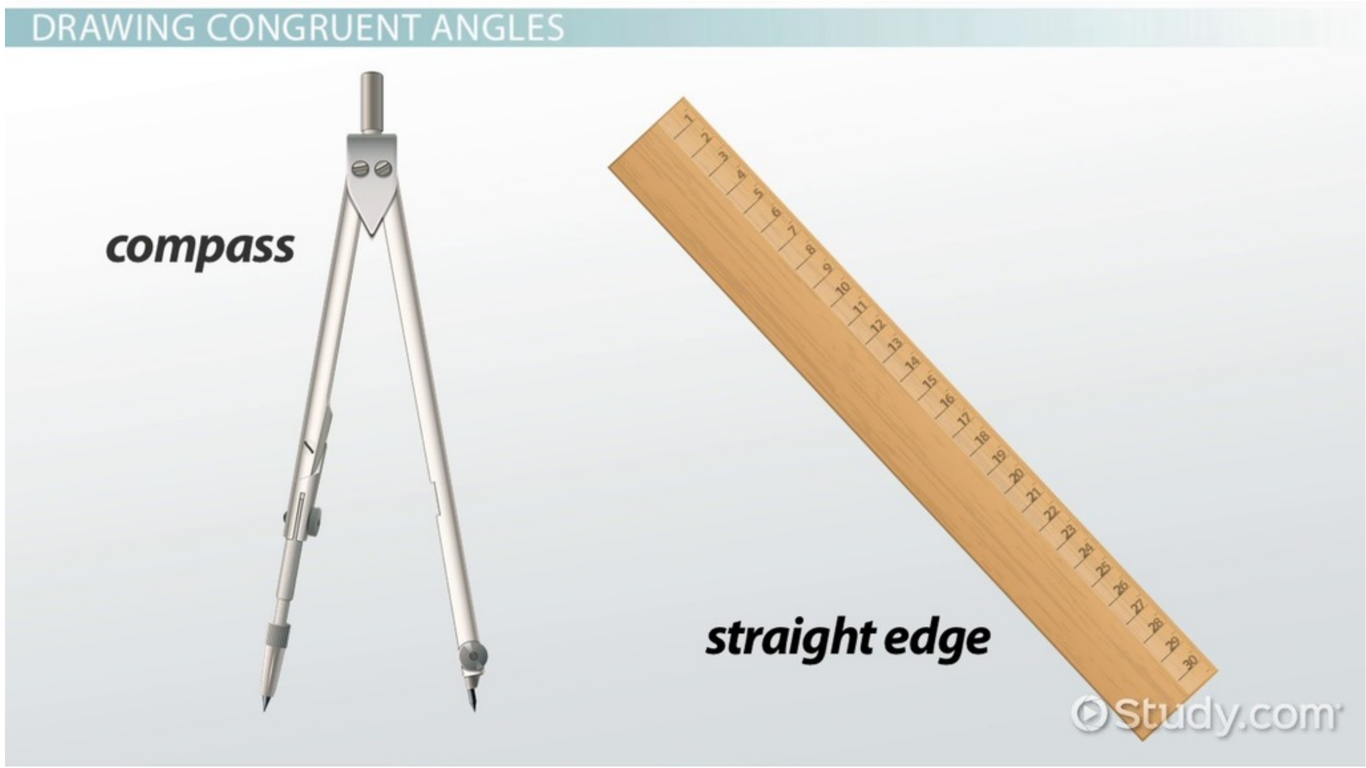
Success and failure

- *Ancient Greeks: geometric constructions using only a straightedge without markings and a compass*
- *Bisect any angle ✓*
- *Construct an equilateral triangle ✓*
- *Trisect any angle?*
- *Doubling of a cube?*
- *Square the circle?*



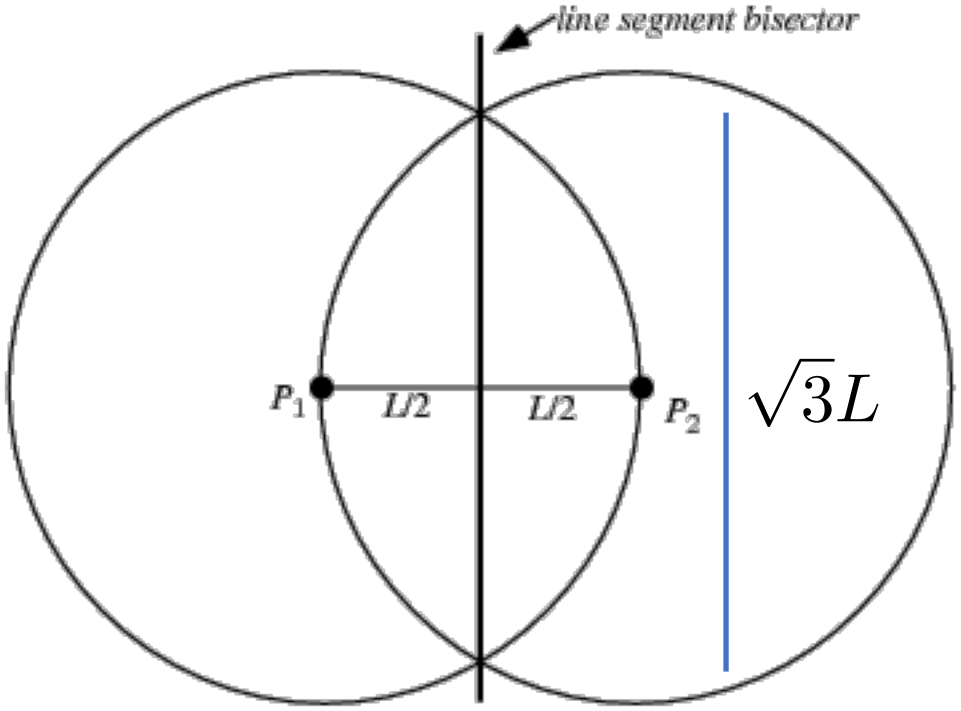
Geometric constructions using only straightedge and compass

- *Solved in 19th century after revolution in abstract algebra.*
- *Bisect any angle ✓*
- *Construct an equilateral triangle ✓*
- *Trisect any angle ✗*
- *Doubling of a cube ✗*
- *Square the circle ✗*



Geometric constructions using only straightedge and compass

- Any such construction can be viewed as computing a number



Line: linear equation

Circle: quadratic equation

Crucial: any such number c has the property

$$[Q(c) : Q] = 2^k$$

The minimum degree of a polynomial over Q that has c as a root is a power of 2.

Geometric constructions using only straightedge and compass

- *Bisect any angle : $\cos(2x)=2\cos^2 x - 1$.*
- *Construct an equilateral triangle : height = $\sqrt{3}/2$.*
- *Trisect any angle* ✗ *: $\cos(3x)=4\cos^3 x - 3\cos x$.*
- *Doubling of a cube* ✗ *: the edge length becomes $\sqrt[3]{2}$.*
- *Square the circle* ✗*: the edge length becomes $\sqrt{\pi}$ which is transcendental over Q (not a root of any polynomial with finite degree).*

Success and failure

- *The previous questions can be viewed as a limited form of computation:*
- *Computing using only straightedge (linear equation) and compass (quadratic equation).*
- *The more limited the computation, the easier to prove impossibility results.*
- *Let us see a more general form of computation.*

Success and failure

- *Finding explicit formulas for the roots of one variable equations.*
- *Linear equations: $ax+b=c$ where a is not 0, $x=(c-b)/a$.*
- *Quadratic equations: $ax^2+bx+c=0$ where a is not 0.*

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

- *Cubic equations, formula found by Ferro and Tartaglia in the 16th century.*
- *Quartic equations, formula found by Ferrari and Cardano in the 16th century.*
- *Can this go on for larger degrees, e.g., degree 5 equations?*

Explicit formulas for roots

- *Major open problem from the 16th century, not solved until the 19th century.*
- *Surprisingly, the answer is NO in general!*
- *The permutation group S_5 is not solvable in general.*
- *Abel-Ruffini impossibility theorem (1824), Galois theory (1846) further provides a characterization of polynomial equations solvable by explicit formulas.*
- *Hermite' 1873: e is transcendental; Lindemann' 1882: π is transcendental (over Q).*

Success and failure

- *The previous questions can be viewed as a limited form of computation:*
- *Computing using only finite degree equations, and finding the roots using only arithmetic operations and radicals.*
- *Let us see a more general form of computation.*

Success and failure

- *Hilbert's 10th problem (posed at ICM 1900)*

- *Given a Diophantine Equation (a polynomial equation with integer coefficients and a finite number of variables), find a process to decide if it has an integer solution.*

$$3x^2-2xy-y^2z-7=0: x=1, y=2, z=-2$$

What Hilbert is actually asking: an algorithm to solve this problem.

Success and failure

- *Hilbert's 10th problem*
- *Has such an algorithm been found?*
- *No! MRDP theorem (1970): no such algorithm exists!*
- *In modern terminology: Hilbert's 10th problem is* ***undecidable.***

Modern use of computation

Computing the orbits of astronomical objects

Total solar eclipse 8/21/2017

Super blood moon 1/20/2019

Can predict accurately to minutes or seconds

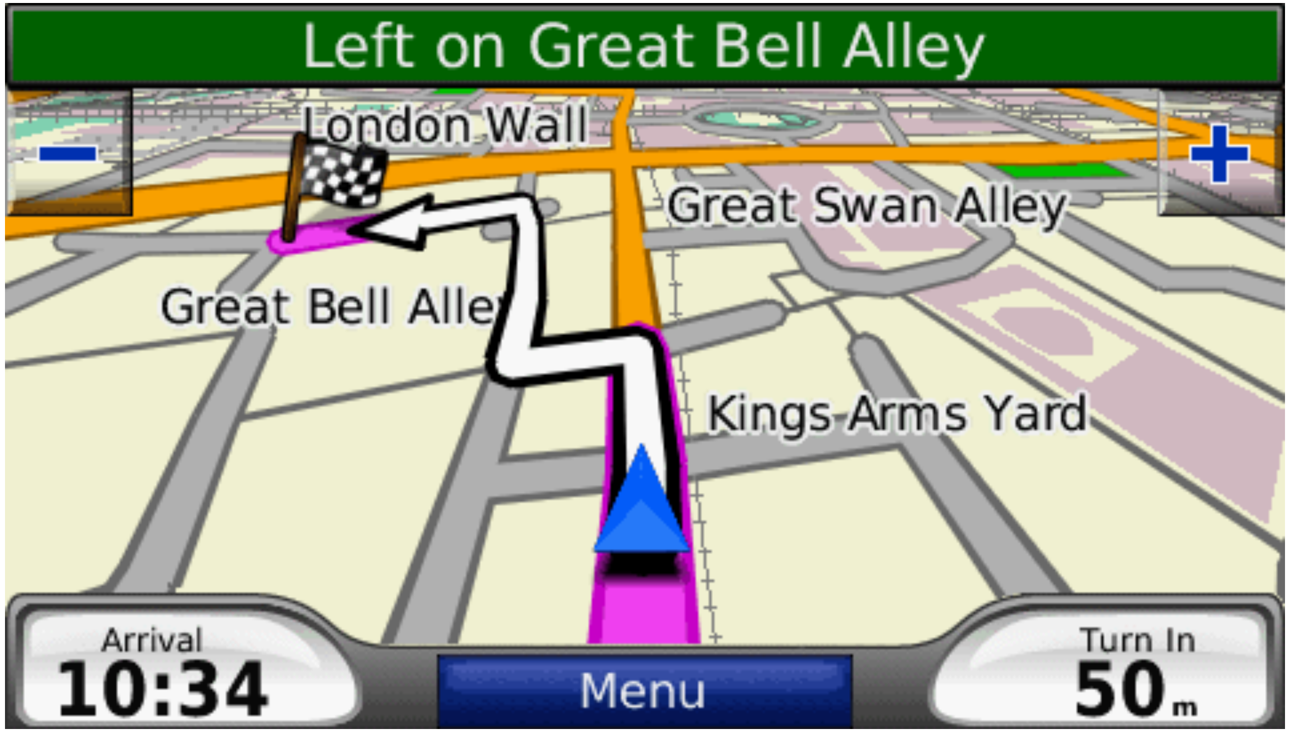
Modern use of computation

Weather forecast

Can predict pretty accurately for the following week

Modern use of computation

GPS routing



Can quickly pick the best route and estimate arrival time

Modern use of computation

Computer graphics in video games

Can create super natural and cool video effects

Modern use of computation

Artificial Intelligence

Can beat human in certain situations

The topic of this course

What is the general theory of computation behind all these applications?

Specific goals

- *Computational models.*
- *Abstract and mathematical models of computation.*
- *By abstraction we can study the common powers and limitations of ALL computation.*

Specific goals

- *Computability.*
- *What can be computed and what cannot be computed?*
- *The true limits of computers/computation.*

Specific goals

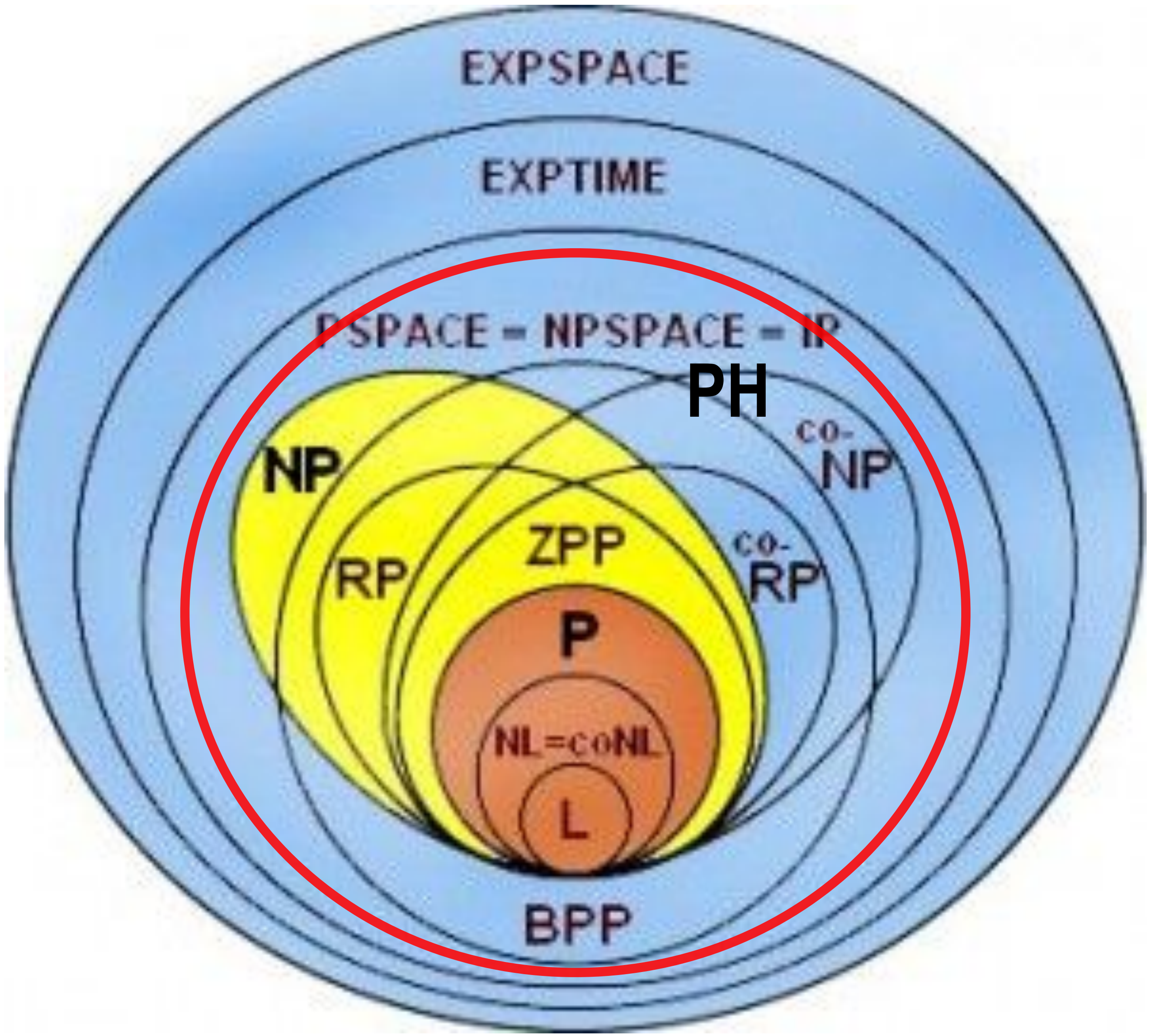
- *Complexity theory.*
- *For problems that can be computed, what is the amount of resources (e.g., time, space etc.) needed?*
- *A more refined and complete understanding of computation.*

Specific goals

- *Complexity classes.*
- *A complexity class is a set of problems that can be solved given a certain amount of resources.*
- *We will see several different complexity classes in this class.*
- *A major question: the relations between different complexity classes.*

P vs NP Problem: one of the 7 (now 6) unsolved millennium problems which can earn you $1M.

If you give a positive answer to this one, you can “solve” all the others! So you get $6M.



Complexity Zoo

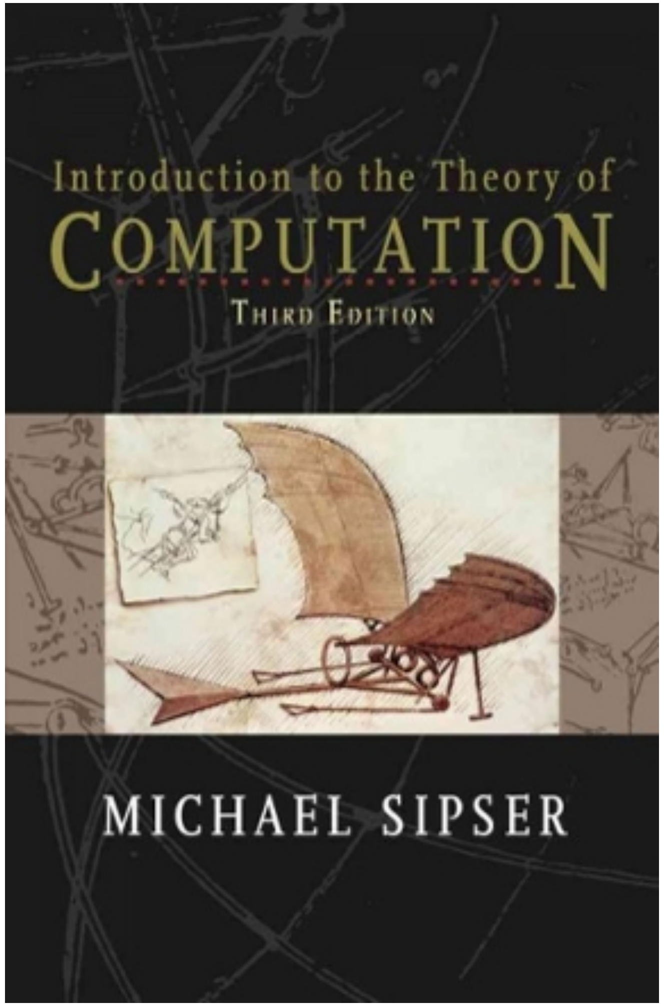
We will see some of the classes in this picture.

- *What this class is about: concepts, models, proofs…*

- *What this class is not about: programming and coding….*

Textbook

Required



Second edition is also OK

Discrete Math Review

- *Sets*
- *Functions and Relations*
- *Graphs*
- *Strings and languages*
- *Proofs*

Sets

- *A set is a group of objects (called elements), treated as one unit.*
- *Example: S={3, 4, 5, 6}. Then $3 \in S$, $4 \in S$, but $10 \notin S$. S is* ***finite****.*
- *Description of sets: 1. list all elements; 2. describe a common property of all elements.*
- *Example: S'={x | x is a multiple of 2}={2, 4, 6, 8, …}. S' is* ***infinite****.*

Relations of Sets

- *Subset: A is a subset of B, denoted as* $A \subseteq B$.
- *Proper subset: A is a proper subset of B, denoted as* $A \subset B$.
- *The empty set* $\Phi=\{\}$ *is a subset of any other set.*

Operations on Sets

- *Union:* $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.

- *Intersection:* $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$.

- *Complement:* $\bar{A} = \{x \mid x \notin A\}$.

- *Set difference:* $B \setminus A = \{x \mid x \in B \text{ and } x \notin A\}$.

Cartesian Product of Sets

- *Cartesian Product :* $A \times B = \{ (x, y) \mid x \in A \text{ and } y \in B\}$*={all ordered 2-tuples from A, B}.*

- *Example:* $A = \{1, 2\}$, $B = \{c, d\}$. *Then* $A \times B = \{(1, c), (1, d), (2, c), (2, d)\}$. $A \times B \neq B \times A$.

- $A \times A = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$. $A \times A \times A = \{(1, 1, 1), (1, 1, 2), (1, 2, 1), (1, 2, 2) \ldots\}$ *={all ordered 3-tuples of A}.*

- *Define* $A^k = A \times A \times \ldots \times A$ *for k times={all ordered k-tuples of A}.*

Functions

- *A function f: A -> B is a mapping from a set A to another set B.*
- *A: domain, B: range. f(x)=y, x: input, y: output.*
- *The same input always produces the same output.*
- *Representation: by enumeration of all input output pairs, or in closed form, e.g., f(x)=2x+1.*

Functions

- *A function f: A -> B is a mapping from a set A to another set B.*
- *A: domain, B: range. f(x)=y, x: input, y: output.*
- *The same input always produces the same output.*
- *Representation: by enumeration of all input output pairs, or in closed form, e.g., f(x)=2x+1.*

Functions

- *Functions are important because they can be used to model the input output behavior of computers.*
- *Surjective functions: every element in the range has at least one pre-image in the domain.*
- *Injective functions: every element in the range has at most one pre-image in the domain.*
- *Bijective functions: both injective and surjective. Implies that the domain and range have the same cardinality.*

Relations

- Relations are generalizations of functions.
- Any set S of k-tuples in A^k defines a k-ary relation on A.
- Special case: 2-ary / binary relation.
- Example: < and = are both binary relations on N (the set of all natural numbers). 1<2, 3=3.

Relations

- *If R is a binary relation that corresponds to the set $S \subseteq A \times A$, then a R b iff $(a, b) \in S$.*

- *R is reflexive if for any $x \in A$, x R x.*

- *R is symmetric if for any $x, y \in A$, x R y => y R x.*

- *R is transitive if for any $x, y, z \in A$, x R y and y R z => x R z.*

Relations

- Examples: < and =.

- Which of the 3 properties does < have?

- Which of the 3 properties does = have?

- A binary relation with all 3 properties is called a equivalence relation.

Equivalence Relation

- *Another example:* $x \equiv_5 y$ *iff* $(x-y)$ *is a multiple of 5.*

- *This is an equivalence relation on N.*

- *The set can be partitioned into equivalent classes based on an equivalence relation.*

- *An equivalent class consists of all elements that mutually satisfy the relation.*

Graphs

- *A graph G=(V, E) consists of a set V of vertices, and a set E of edges.*
- *Edges can be directed or undirected.*
- *A path in a graph is a sequence of vertices connected by edges.*
- *A graph is connected if every pair of vertices has a path between them.*
- *An undirected graph is a tree if it’s connected and has no cycle.*

Directed Graphs

- *A directed path in a directed graph is a path in which all edges point to the same direction.*
- *A directed graph is strongly connected if there is a directed path connecting every ordered pair of vertices.*
- *The degree of a vertex is the number of its neighbors.*
- *Representation of a graph: adjacency matrix.*

Strings

- *Given an alphabet Σ, a string over Σ is a finite sequence of symbols from Σ.*
- *Example: Σ={0, 1}, binary strings. Σ={a, b, c… z}, words.*
- *If w is a string, then |w| denotes the length (number of symbols) of w.*
- *The string of length 0 is called the empty string (ε).*

Operations on Strings

- *Reverse of a string* $w=w_1 w_2 \ldots w_n$ *is* $w^R = w_n \ldots w_2 w_1$.
- *Concatenation of two strings x and y is x y.*
- *Concatenation of k copies of a string x:* $x^k = x\, x \ldots x$ *for k times.*

Languages

- *A language is a set of finite strings over some alphabet Σ.*

- *Example: the English language is a language over Σ={a, b, …, z, , . ? ! ….}*

- *Example:* Σ^k *=(the k times Cartesian product of Σ) = {all strings of length k}.*

- *Example:* $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \ldots = \cup_{k\geq 0} \Sigma^k$ *= {all strings of finite length}.*

Boolean Functions

- *A Boolean function is a function f: {0, 1}* -> {0, 1}, i.e., the input is a finite length binary string, the output is 0 or 1.*

- *Boolean functions are of special importance in computer science.*

- *Every finite object can be represented as a binary string.*

- *Every function with finite domain and range can be represented as Boolean functions —-> A computer’s input/output.*

Languages and Boolean Functions

- *Any Boolean function corresponds to a language over {0, 1}, and vice versa.*
- *Let f: $\{0, 1\}^*$ -> {0, 1}.*
- *Consider the language L =$\{x \mid x \in \{0, 1\}^*$ s.t. $f(x)=1\}$.*
- *There is a bijection between the Boolean functions f and the languages L. Say f computes the characteristic function of L.*

Mathematical Proofs

- *A mathematical proof is a convincing logical argument that a statement is true.*

- *Structure of a proof:*

Something given/known to be true.

↓

A sequence of steps, each one implied by the previous one in a formal sense.

↓

Conclusion

There is no fixed strategy to come up with proofs.

Some Ways of Proofs

- *Proof by construction.*
- *Proof by contradiction.*
- *Proof by induction.*

Proof by Construction

- *Example: for any real numbers a and b s.t. $a \neq 0$, there is a real number r s.t. $ar+b=0$.*

- *Proof: in order for $ar+b=0$, it suffices to have $ar=-b$.*

- *Since $a \neq 0$, we can take $r = -b/a$.*

- *Thus we have found such an r by construction.*

Proof by Contradiction

- *Example: for any two integers a, b, we have* $a^2-4b \neq 6$.
- *Proof: cannot enumerate all possible (a, b) since there are infinitely many.*
- *Suppose there exist integers a, b, s.t.* $a^2-4b = 6$.
- $a^2= 4b+6$ *is even => a must be even. So* $a=2c$ *for some integer c.*

Proof by Contradiction

- $a^2 = 4b+6 \Rightarrow a^2 = 4c^2 = 4b+6.$
- *So* $6=4(c^2-b) \Rightarrow 3=2(c^2-b).$
- *L.H.S. is odd, R.H.S. is even, a contradiction.*

Proof by Induction

- *Establish the statement for the base case, e.g., $n=1$.*
- *Establish the inductive step, e.g., assume statement holds for $n=k$, prove the statement holds for $n=k+1$.*
- *Strong induction: assume statement holds for all $n \leq k$, prove the statement holds for $n=k+1$*
- *Important: must cover all base cases.*

Proving Two Sets are Equal

- *How do you prove two sets A and B are s.t. A=B?*
- *Must prove two directions*
- *1. Prove $A \subseteq B$.*
- *2. Prove $B \subseteq A$.*