THEORY OF COMPUTATION

Pre-req: Familiarity with discrete mathematical structures and mathematical proofs.

Text: [Sipser] Introduction to the Theory of Computation.

What is the course about?

① What is computation? — Automata Theory

② What can / cannot be computed? — Computability Theory

③ How efficiently (e.g. fast)? — Complexity Theory

① Automata Theory (Define precise mathematical model for a "computer").

- Finite Automata
  - Regular Languages
  - Eg. Text search (UNIX Grep command)
- Push Down Automata
  - Context Free Languages
  - E.g. Parser
- Turing Machine
  - Decidable languages
  - Eg. Primality testing.

Note As we'll see, a "language" is a formal way of describing what a "computational problem" is.

Finite Automata Push Down Auto. Turing m/c

are computational models with increasing "power". They are able to "solve" larger and larger class of "computational problems" (= "languages"). These classes are, respectively,

Regular languages Contex Free lang. Decidable Lang.

Note As we'll see, Turing m/c captures our contemporary notion of a "real computer".

(2) Computability Theory

It turns out that there are languages that are "undecidable", i.e. cannot be "(definitively) solved" (even) by a Turing m/c.

Example 1 Given a C-program, when executed, does it (eventually) halt?

Example 2 Given a polynomial equation with integer coefficients, does it have an integer solution? [Hilbert's 10th Problem].

E.g. $2x^2 - yz + wx^4 = 5$. YES ($x=1$, $y=z=0$, $w=3$).

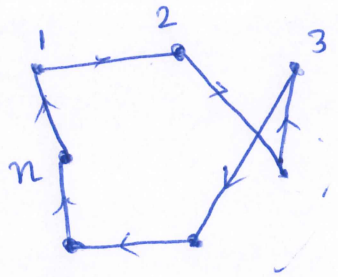
$x^2 + y^2 = 7$. NO.

In both examples, there is no Turing machine ("algorithm") that on every input instance carries out computation (for arbitrarily long "time") and (eventually) outputs the correct YES/NO answer!

③ Complexity Theory

Once the Turing m/c model is defined, one can define time as well as memory space required to solve problems.

- Certain problems appear to be inherently hard. E.g. Travelling Sales Person (TSP):



Given n cities and all pairwise distances, what is the shortest tour of all the cities?

This (and many other) problem is "NP hard". Researchers do not believe that there is an algorithm that runs in time <u>polynomial</u> in n (say n^{100}). Of course, there is $n!$ time algorithm that tries out all possible tours.

- Factoring integers seems hard.

Factoring and some other seemingly hard problems are the basis of (supposedly) secure cryptography. So hardness is useful too!

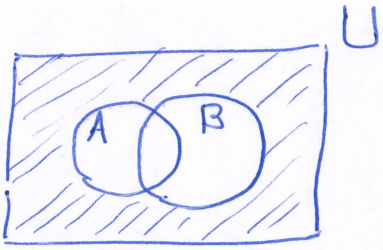
Warm-Up of Mathematical Proofs

Direct Proofs | Proof by Induction | Proof by Contradiction | Proof by Cases | .....

Direct Proofs

Theorem For sets $A, B \subseteq U$, (U: universe)

$$\overline{A \cup B} = \bar{A} \cap \bar{B}.$$



Proof We'll show that for every element $x \in U$,

$$x \in \overline{A \cup B} \iff x \in \bar{A} \cap \bar{B}.$$

(if and only if)

Indeed,

$$x \in \overline{A \cup B}$$

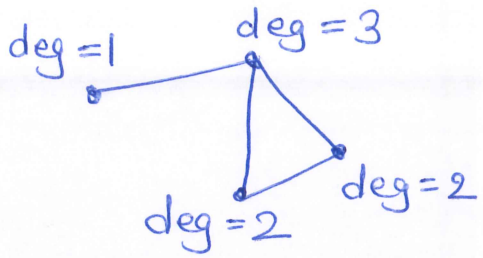
$$\iff x \notin A \cup B$$

$$\iff x \notin A \text{ and } x \notin B$$

$$\iff x \in \bar{A} \text{ and } x \in \bar{B}$$

$$\iff x \in \bar{A} \cap \bar{B}.$$

Theorem For a graph G (undirected, no self-loops), the sum of degrees of all its vertices is even.



$1+3+2+2 = 8$

Proof In fact for a graph $G(V, E)$, (V: vertex set, E: edge set) if $d_1, d_2, \ldots, d_n$ are degrees, $|V| = n$, then

$$\sum_{i=1}^{n} d_i = 2|E| \quad \text{(2 times \#edges)}.$$

This is because if one counted all the edges incident on all the vertices, each edge (i, j) will be counted exactly twice, once from perspective of vertex i and once for j. ▨

Proofs by Induction

Theorem For every integer $n \geq 1$,

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}.$$

Proof Base case $n=1$. $1 = \frac{1 \cdot 2}{2}$. True.

Inductive step: Assume that the hypothesis holds for certain $n \geq 1$. I.e.

$$1+2+\cdots+n = \frac{n(n+1)}{2} \qquad \text{——(*)}$$

We'll show that it holds for $n+1$; I.e.

$$1+2+\cdots+n+(n+1) = \frac{(n+1)(n+2)}{2}.$$

Indeed,

$$\underbrace{1+2+\cdots+n}_{\downarrow \frac{n(n+1)}{2}} + (n+1)$$

By inductive hypothesis (*)

$$= (n+1)\cdot\left(\frac{n}{2}+1\right)$$

$$= \frac{(n+1)(n+2)}{2}$$

as required.

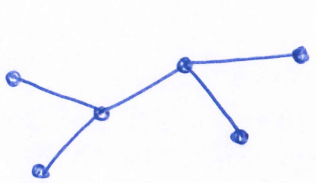
Exercise $1^2+2^2+\cdots+n^2 = \frac{n(n+1)(2n+1)}{6}$

$$1^3+2^3+\cdots+n^3 = \frac{n^2(n+1)^2}{4} = (1+2+\cdots+n)^2.$$

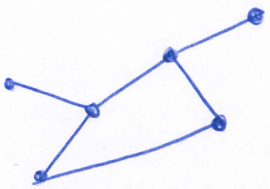
!!!

Theorem Every tree on n vertices has exactly $n-1$ edges.

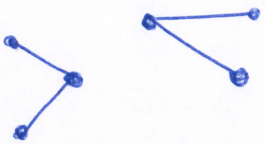
Note A tree is a connected graph with no cycles.



Tree



(cycle)



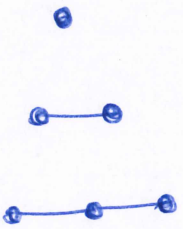
(disconnected)

Not a tree

Proof Base case $n=1$.

$n=2$

$n=3$

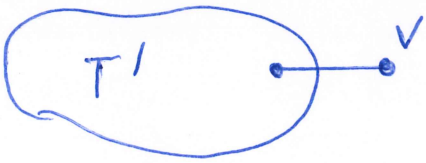


} Tree.

Inductive step. Assume that every tree on n vertices has $n-1$ edges.

Now consider a tree T on $n+1$ vertices. We'll show that T must have a vertex v of degree 1, so it looks as

T =



where T' is a tree on n vertices. We

note that since T is connected with no cycles, so is T' (and hence T' is a tree). Then by inductive hypothesis, T' has $n-1$ edges and hence T has n edges, completing the proof.

It only remains to show that T has a degree 1 vertex. Start at any vertex v_1 and take a walk

$$v_1 - v_2 - \cdots v_i - v_{i+1} \cdots v_\ell$$

where for every i, $1 \leq i \leq \ell - 1$, one moves from v_i to v_{i+1}, provided $v_{i+1} \neq v_{i-1}$. Since T has no cycles, this walk will not have a repeated vertex. Since the graph is finite, the walk must terminate, at say vertex v_ℓ. It follows that v_ℓ has degree 1.

Proofs by Contradiction

Theorem $\sqrt{2}$ is irrational.

Proof Suppose on the contrary that $\sqrt{2}$ is rational. If so, we can write $\sqrt{2} = \frac{p}{q}$ where p, q are positive integers with no common factor.

Now, $2 = \frac{p^2}{q^2}$ $\therefore p^2 = 2q^2$

$\therefore 2 \mid p$, $p = 2r$

$\therefore 4r^2 = 2q^2$

$\therefore q^2 = 2r^2$

$\therefore 2 \mid q$.

But now p, q have a common factor 2, a contradiction. ▨

Theorem There are infinitely many primes.

Proof Suppose on the contrary that there are only finitely many primes $P_1, P_2, \ldots, P_k$.

Consider integer $N = 1 + P_1 P_2 \cdots P_k = 1 + \prod_{i=1}^{k} P_i$.

Let p be any prime factor of N. Since $p \mid N$ but $P_i \nmid N$ (why?), $p \neq P_i \ \forall 1 \leq i \leq k$.

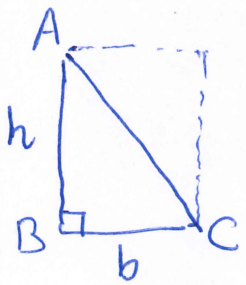
This is a contradiction since p is a "new" prime not in the set $\{P_1, \ldots, P_k\}$. ▨

Proof by Cases

Theorem Area of a triangle $= \frac{1}{2} \times \underbrace{\text{height}}_{h} \times \underbrace{\text{base}}_{b}$.

Proof

Case 1 Right angled Δ.



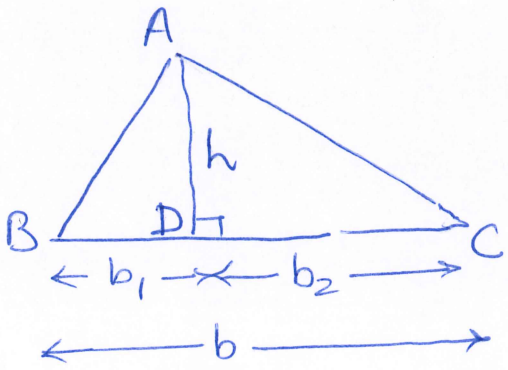
$\text{Area}(\Delta ABC)$

$= \frac{1}{2} \text{Area(Rectangle)}$

$= \frac{1}{2} \cdot h \cdot b$

Using this ↖

Case 2 Acute angled Δ.



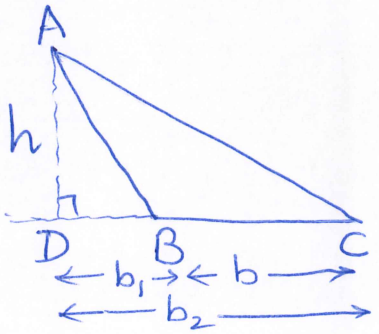
$\text{Area}(\Delta ABC)$

$= \text{Area}(\Delta ABD) + \text{Area}(\Delta ACD)$

$= \frac{1}{2} h b_1 + \frac{1}{2} h b_2$

$= \frac{1}{2} h (b_1 + b_2) = \frac{1}{2} hb.$

Case 3. Obtuse angled Δ.



$\text{Area}(\Delta ABC)$

$= \frac{1}{2} b_2 h - \frac{1}{2} b_1 h$

$= \frac{1}{2} hb.$

▨

Strings and Languages.

↑ Input

↑ Computational Problem

Def Alphabet is a finite set of symbols.
Denoted typically by Σ, Γ.

Ex. $\Sigma = \{0,1\}$. $\Sigma = \{0,1,\ldots,9\}$.

$\Gamma = \{a,b,\ldots,z\}$ $\Gamma = \{a\ldots z, A\ldots Z, \#, \$, *, \ldots, @\}$

Def A string over alphabet Σ is a finite sequence of symbols from Σ.
Denoted typically by x, y, z, u, v.

$|x|$ = Length of string x = Number of symbols in x.

One can write $x = x_1 x_2 \ldots x_n$, $|x| = n$, where $x_i \in \Sigma$ are its symbols, $1 \leq i \leq n$.

Def ε denotes empty string with no symbols and length zero.

Ex. Strings over $\Sigma = \{0,1\}$.

$\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots$

Def Σ^* = Set of all finite length strings over Σ.

Ex. $\Sigma = \{0,1\}$.

$\Sigma^* = \{\underbrace{\varepsilon}, \underbrace{0, 1}, \underbrace{00, \ldots, 11}, \underbrace{000, \ldots, 111}, \ldots\}$.

length 2 length 3 ...

Def A language L over alphabet Σ is a subset of Σ^*. $L \subseteq \Sigma^*$.

Ex.

$\Sigma = \{0,1\}$. L_{even} = Set of all even length strings.

$\Sigma = \{0,1,\ldots,9\}$. L_{primes} = Set of all strings that are prime integers (decimal)

$\Gamma = \{a, \ldots, z\}$. $L_{English}$ = Set of all valid English words.

$\Gamma = \{a \ldots z, A \ldots Z, \#, \ldots, @\}$. $L_{programs}$ = Set of all strings that are syntactically correct C-programs.

Language Recognition Problems

Def For a language $L \subseteq \Sigma^*$, we can associate a computational problem P_L as (referred as lang. recognition problem)

"Given string $x \in \Sigma^*$, is $x \in L$?"

Note
- x is thought of as input.
- P_L is a decision problem, i.e. it seeks just YES/NO answer.

Note The recognition problems for languages defined:

$P_{L_{even}}$ ≡ Is given string of even length?

$P_{L_{primes}}$ ≡ Is given integer a prime?

$P_{L_{English}}$ ≡ Is given word a valid English word? (Dictionary Lookup)

$P_{L_{programs}}$ ≡ Is given code a syntactically correct C-program? (compiler!)

Note - L_{even}, L_{primes}, $L_{programs}$ are infinite.

- $L_{English}$ is finite. Finite languages are not very interesting.

Punchline!

① Language recognition problem is a precise formalization of "computational problem" (more specifically, decision problems).

② "All" decision problems can be recast (formulated) as language recognition problems via encodings.

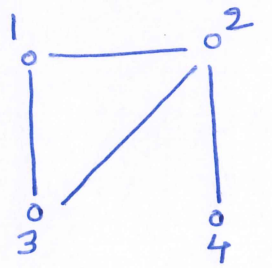
③ Decision problems capture the essence of computation (so we'll restrict ourselves to decision / lang. recognition problems.)

Encodings

"Discrete / combinatorial objects can be encoded as strings".

Ex. Graphs can be encoded as strings over alphabet (say) $\Sigma = \{0, 1, \#\}$.

Let $\langle G \rangle$ denote encoding of graph G.

G 

	1	2	3	4
1	0	1	1	0
2	1	0	1	1
3	1	1	0	0
4	0	1	0	0

Adjacency matrix

$\langle G \rangle \longrightarrow$ 0110 # 1011 # 1100 # 0100

$L_{Graphs} = \{\langle G \rangle \mid G \text{ is a graph}\} \subseteq \{0, 1, \#\}^*$.

$L_{Conn} = \{\langle G \rangle \mid G \text{ is a connected graph}\}$.

$L_{Bipartite} = \{\langle G \rangle \mid$ " bipartite " $\}$.

$L_{Hamil} = \{\langle G \rangle \mid$ " Hamiltonian " $\}$.

And many many more!

Note The corresponding language recognition problems capture graph theoretic problems of deciding whether a graph is connected, bipartite, Hamiltonian, ..., in fact "any" (decision) problem one thinks of!

To emphasize

"Given G, is it connected"

$\equiv$ equivalent

"Given $x \in \{0,1,\#\}^*$, is $x \in L_{conn}$".

Observation Similarly, one can encode Sets, functions, integers, ... etc, polynomials (with say integer coefficients), C-programs (which are already strings) as strings and formulate decision problems about them! In particular, one can frame problems about C-programs!