Agenda

We've done

- Greedy Method
- Divide and Conquer
- Dynamic Programming
- Network Flows & Applications

Now

- $\mathbf{NP}$-completeness

Up to this point

- Most problems we have seen can be solved in "polynomial" time
  - All Pairs Shortest Paths in $O(|V|^3)$
  - Single Source Shortest Paths in $O(|V| \lg |V| + |E|)$
  - Minimum Spanning Trees in $O(|V| \lg |V|)$
  - Sorting in $O(n \lg n)$
  - ...
- Actually, no problem we have seen required more than $O(n^5)$

A Natural Question

Can all "natural" problems be solved in polynomial time?

There are Many Harder Problems

- VERTEX COVER: given a graph G, find a minimum size vertex cover
- 0-1 KNAPSACK: A robber found n items in a store, the ith item is worth v_i dollars and weighs w_i pounds ($v_i, w_i \in \mathbb{Z}$), he can only carry W pounds. Which items should he take?
- TRAVELING SALESMAN (TSP): find the shortest route for a salesman to visit each of the n given cities once, and return to the starting city.
- ... and about 10,000 more natural problems

No-one has ever come up with a poly-time solution to any of these problems!

Dealing with "Hard" Problems

Suppose your boss asks you to write a program solving a problem which you can't come up with an efficient solution.

1. Email ask the prof who taught CSE531
2. Give up
3. Spend the next 6 months working on the problem
4. Give the boss a brute-force algorithm which takes a century to finish
5. Mathematically show the boss that this problem does not have a poly time solution
   - Highly unlikely, it is very hard to give such a proof.
   - For the hard problems, the best lower bound people have found is $\Omega(n)$, which is totally useless!
6. Mathematically show that your problem is "equivalent" to some problem which no body knows how to solve

Showing that Solving Your Problem is Mission Impossible

Main questions are

- What do we mean by an algorithm?
- What do we mean by “hard”?
- What do we mean by “equivalently hard”?

To answer these questions,

- We need a computational model, which is a formal tool to model computation.
- Let’s go back to ... Cantor, Russell, Hilbert, Gödel, Church, Turing, Cook/Levin, Karp, etc.

Georg Cantor (1845–1918): Father of Set Theory

Cantor's Set Theory

In later decades of the 19th century, Cantor developed his set theory

- Two sets A and B have the same cardinality iff $\exists$ a one-to-one correspondence (bijection) between them. We write $|A| = |B|$
  - This is subtle! E.g., let $\mathbb{E}$ be the set of even natural numbers, then $|\mathbb{E}| = |\mathbb{N}|$
- $|A| < |B|$ iff $\exists$ an injection from A into B but no bijection
  - For any set S, $|S| < |2^S|$; where 2^S is the power set of S
  - $|\mathbb{N}| < |\mathbb{R}| = c$
  - Both facts are shown with the celebrated diagonal argument

(Transfinite) Cardinal Numbers

- Cardinal Numbers are numbers which are cardinalities of sets.
- Finite Cardinal Numbers: $0, 1, 2, \ldots$
- Transfinite Cardinal Numbers: $\aleph_0 = |\{0, 1, 2, \ldots\}| = |\mathbb{N}|$, $\aleph_1$ is the next larger cardinal number, followed by $\aleph_2$, $\aleph_3$, ...
- Diagonal argument gives $\aleph_i < 2^{\aleph_i}$
- Generalized Continuum Hypothesis: $\aleph_{i+1} = 2^{\aleph_i}$
- Continuum Hypothesis (CH): $\aleph_1 = 2^{\aleph_0}$
  - We can show $c = |\mathbb{R}| = 2^{\aleph_0}$, thus the continuum problem basically asks where $|\mathbb{R}|$ is in the hierarchy of the $\aleph_i$.

Ordinal Numbers

$$0, 1, 2, 3, 4, 5, \ldots$$

$$0, 1, 2, 3, 4, 5, \ldots, \omega$$

$$0, \ldots, \omega, \omega + 1, \omega + 2, \ldots$$

$$0, \ldots, \omega, \omega + 1, \omega + 2, \ldots, \omega \cdot 2$$

$$0, \ldots, \omega, \omega + 1, \ldots, \omega \cdot 2, \ldots, \omega \cdot 3, \ldots, \omega^2$$

$$0, \ldots, \omega, \ldots, \omega \cdot 2, \ldots, \omega^2, \ldots, \omega^3$$

$$0, \ldots, \omega, \ldots, \omega^2, \ldots, \omega^3, \ldots, \omega^\omega$$

$$0, \ldots, \omega, \ldots, \omega^2, \ldots, \omega^3, \ldots, \omega^\omega, \ldots, \omega^{\omega^\omega}$$

Now he ran out of names, so he invented a new notation

$$\epsilon_0 = \omega^{\omega^{\omega^{\cdots}}}$$

where the number of times we take ω-power is ... ω!

Back to the Cardinal Numbers

- Now we have a way to index the cardinal numbers

$$\aleph_0, \aleph_1, \ldots, \aleph_\omega$$

  why stop there?

$$\aleph_0, \aleph_1, \ldots, \aleph_\omega, \ldots, \aleph_{\omega^2}, \ldots, \aleph_{\omega^\omega}, \ldots, \aleph_{\epsilon_0}$$

  shall we go on?
- We can also do arithmetics on the ordinal and cardinal numbers!

All That Leads to ... The Great Debate

- Two of the greatest mathematicians of the later half of the 19th century and the beginning of the 20th century:
  - David Hilbert: "no one shall expel us from the paradise which Cantor has created for us!"
  - Henri Poincaré: "later generations will regard set theory as a disease from which one has recovered!"
- Others: "that's not mathematics, it's theology!"
- Still, many just fell in love with Cantor's work.
- Cantor ended his life in a mental hospital.

The Paradoxes of Set Theory

Berry's paradox

The first natural number which cannot be named in less than fifteen English words

Russell's paradox

Consider the set of all sets which are not members of themselves

The Greek already knew the difficulty of self-reference:

Liar Paradox

This statement is false!

Barber Paradox

In a village, a barber shaves everyone who does not shave himself

Talking about Self-Reference

A Self-Referential Sentence

This sentence has three a's, two c's, two d's, twenty-eight e's, four f's, four g's, ten h's, eight i's, two l's, eleven n's, six o's, seven r's, twenty-seven s's, eighteen t's, three u's, five v's, six w's, three x's, and three y's.

A Self-Referential Puzzle

Write a program in C,Java,Perl,Scheme, Python, ... that prints an exact replica of its source code.

Solutions to the Paradoxes

Three main schools of thoughts

- Logicism: mathematics is, in a significant sense, mostly reducible to logic.
- Intuitionism (Brouwer): "the only way to prove that something exists is to exhibit it or to provide a method for calculating it!"
- Formalism (Hilbert): let's eliminate from mathematics the uncertainties and ambiguities of natural language; Hilbert Program: let's formalize all existing theories to a finite set of axioms, and then prove that the axioms are complete and consistent.
  It should be possible to devise a proof-checking algorithm which, given a set of axioms and inference rules, shall be able to decide if a proof is correct!

David Hilbert (1862–1943)

Hilbert's Problems

Totally 23 problems. Ten were presented at the Second International Congress of Mathematics (Paris, Aug 8, 1900)

- Problem 1: is there a transfinite number between $\aleph_0$ and *the continuum*? (CH said NO.)
- Problem 2: Can it be proven that the axioms of logic are consistent?
- Problem 8: Riemann hypothesis. (Remember John Nash in *the Beautiful Mind?*)
- Problem 10: Does there exist an algorithm to solve Diophantine equations?

In 1928, he also asked: "is mathematics decidable, i.e., is there an algorithm which decides if a mathematical statement has a proof or not?" (Entscheidungsproblem)

Kurt Gödel (1906-1978)

On Hilbert's second problem, Gödel showed (1931) that any consistent axiomatic system capable of doing arithmetics is necessarily incomplete! (Diagonal argument is key.)

Alonzo Church (1903–1995)

In 1936, Church gave two λ-calculus expressions whose equivalence cannot be computed (i.e. cannot be expressed as a recursive function). (Note: λ-calculus greatly influenced Lisp, ML, Haskell.)

Alan Turing (1912–1954)

Turing machine (1936) captures "algorithm" in Entscheidungsproblem and Hilbert's 10th problem. Answer to Entscheidungsproblem: the halting problem is undecidable. (Diagonal argument is key.)

Church-Turing Thesis

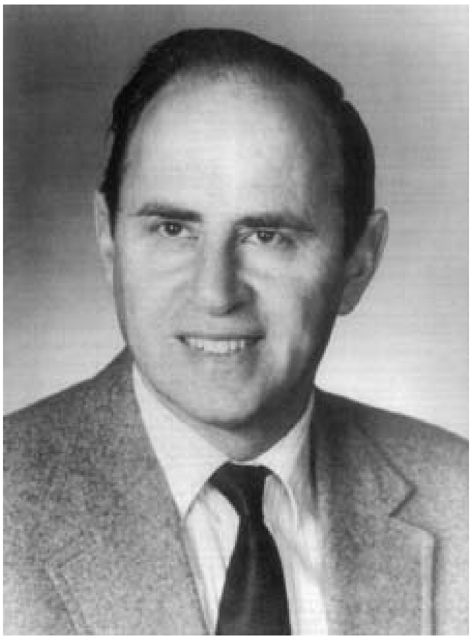
Church-Turing Thesis

The intuitive notion of computations and algorithms is captured by the Turing machine model

- In other words, anything computable is computable by a Turing machine.
- Holds true for all known computational models: Random Access Machines (RAM – von Neumann), Post system, Markov algorithms, combinatory logic, λ-calculus, parallel computers, quantum computers, DNA computers, unlimited register machines, etc.

A Remark on the Continuum Hypothesis

Paul Cohen (April 2, 1934 - March 23, 2007) showed that the Continuum Hypothesis and the Axiom of Choice are independent of Zermelo-Fraenkel (ZF) set theory



Easy Problems and Hard Problems

- In the 60s, computational resources (time, space) are scarce.
- Rabin (1960), Hartmanis and Stearns (1965), Blum (1967) introduced and studied problem complexity measured by the number of steps required to solve it with an algorithm
- Cobham (1964), Edmonds (1965), Rabin (1966) proposed the class $\mathbf{P}$ as the class of "easy problems"

Informally

$\mathbf{P}$ is the class of all problems which can be solved with a polynomial-time algorithm. Problems in $\mathbf{P}$ are easy.

Informally

Problems not in $\mathbf{P}$ are hard.

Why Polynomials?

- Polynomials typify "slowly growing" functions, closed under addition, multiplications, and compositions
- Practically, if a problem is in $\mathbf{P}$, it is extremely likely that it can be solved in time $O(n^4)$ or less

Some well-known examples of problems in $\mathbf{P}$

- PRIMALITY TESTING
- LINEAR PROGRAMMING ($\Rightarrow$ MAXIMUM FLOWS)
- SHORTEST PATHS, MINIMUM SPANNING TREES, MAXIMUM MATCHING

Efficient Verification

Motivating Examples

- The 67th Mersenne's number: in 1903, Frank Nelson Cole (1861 - 1926) gave a "lecture" to the American Mathematical Society entitled "On the Factorization of Large Numbers." Without saying a word, Cole proceeded to write on a blackboard the elementary calculations leading to

$$\begin{aligned} 2^{67} - 1 &= 147,573,952,588,676,412,927 \\ &= 193,707,721 \times 761,838,257,287 \end{aligned}$$

- Read a typical math paper: we can often verify that the proof is correct. Finding the proof is a much different ball game.
- Theme: there are problems which have short and efficiently verifiable proofs

The Class NP

Informally

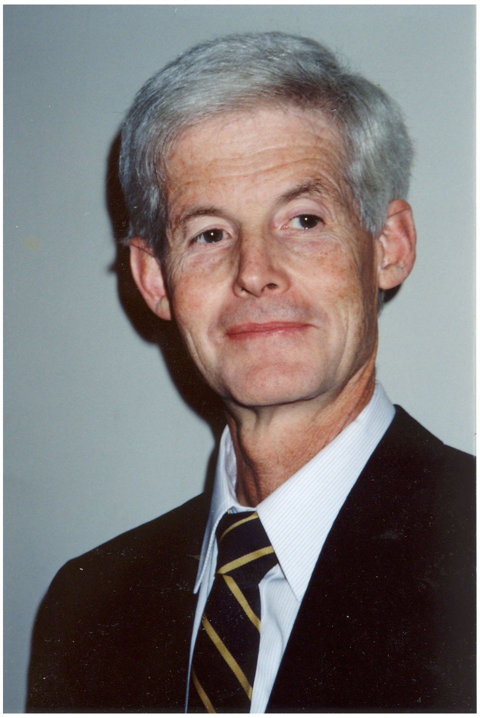
NP consists of all problems which have short and efficiently verifiable solutions.

- Efficiently verifiable = verification is in **P**
- Examples:
  - VERTEX COVER: we don't know how to efficiently decide if a graph G has a vertex cover of size at most a given k; but, the verification that a set of vertices is a VC of size at most k is very easy
  - COLORING, TSP, HAMILTONIAN CIRCUIT, etc.

Corollary

$\mathbf{P} \subseteq \mathbf{NP}$

NP was Introduced by Cook (1971) and Levin (1973)



NP in "Real" Life

- **NP** captures many tasks of human endeavor for which successful completion can be easily recognized (we can tell a good solution when we see one)
- Mathematician: given a math statement, come up with a proof
- Scientist: given a collection of data on some phenomenon, come up with a theory explaining it
- Engineer: given a set of constraints (physical laws, cost, etc.), come up with a design (of an engine, bridge, laptop, ...) meeting these constraints
- Detective: given the crime scene, find the criminal

Informal Definitions of "Harder" and "Equally Hard"

Harder

Problem B is harder than problem A, written as $A \leq B$, if an efficient algorithm for B can be used as a sub-routine to design an efficient algorithm for A. We also say A can be efficiently reduced to B.

- E.g., Bipartite Max Matching $\leq$ Maximum Flow
- If B is easy then A is easy
- If A is hard then B is hard

Equally hard

A is as hard as B if $A \leq B$ and $B \leq A$

Informal definition of $\mathbf{NP}$-completeness

$\mathbf{NP}$-complete

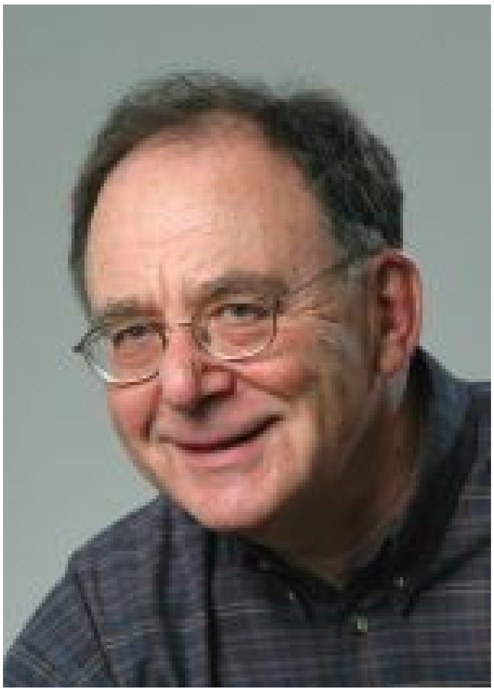
A problem C is $\mathbf{NP}$-complete if it is in $\mathbf{NP}$ and it is harder than all problems in $\mathbf{NP}$; In other words, $A \leq C$ for all $A \in \mathbf{NP}$; In particular, all $\mathbf{NP}$-complete problems are equivalently hard!

Theorem (Cook-Levin)

SATISFIABILITY *(or* SAT*) is* $\mathbf{NP}$*-complete.*

Richard Karp

In 1972, Karp showed that 21 other well-known problems (with no known efficient solutions) are $\mathbf{NP}$-complete also.



The $\mathbf{P}$ vs $\mathbf{NP}$ Problem

- Today, there are $\approx$ 10,000 $\mathbf{NP}$-complete problems
- None of them is known to be in $\mathbf{P}$!
- We do not know how to show that they are not in $\mathbf{P}$ either!
- Any $\mathbf{NP}$-complete problem is in $\mathbf{P}$ iff $\mathbf{P} = \mathbf{NP}$

The Conjecture of Computer Science

$\mathbf{P} \neq \mathbf{NP}$

Millennium Prize Problems; Clay Research Institute offered one million dollars to whoever solves one of a few outstanding problems, including

- $\mathbf{P} = \mathbf{NP}$?
- The Riemann hypothesis (Hilbert's 8th problem)
- The Poincaré conjecture (Perelman did it last year!)

Why do We Believe $\mathbf{P} \neq \mathbf{NP}$?

- For decades, no less-than-exponential algorithm is known for any of the thousands of $\mathbf{NP}$-complete problems; even though there are great financial incentives for coming up with a solution to any of these problems.
- Philosophically, creativity cannot be automated
  - coming up with a proof should be harder than checking the correctness of the proof
  - designing a bridge should be much harder than checking its safety features
  - etc.
- Philosophically, $\mathbf{P} = \mathbf{NP}$ implies the proof of $\mathbf{P} = \mathbf{NP}$ is easy to find! is hard
- Plus a myriad of technical reasons ...

What Do We Do Now That $\mathbf{P} \neq \mathbf{NP}$

- Approximation algorithms
- Randomized algorithms
- New computational models and physical realizations, e.g., DNA and/or quantum computers (probably still equivalent to Turing Machine)
- ...

Ideas to be Formalized

- $\mathbf{P}$ is the class of problems whose solutions can be found in polynomial time
- $\mathbf{NP}$ is the class of problems which have short and efficiently verifiable solutions
- $\mathbf{NP}$-complete is the class of problems in $\mathbf{NP}$ which are harder than all problems in $\mathbf{NP}$;
  In other words, a problem is $\mathbf{NP}$-complete iff any problem in $\mathbf{NP}$ is reducible to it

Examples of What We Call "Problems"

- MAXIMUM MATCHING: given a graph G, find a matching of maximum size
- GRAPH COLORING: given a graph G, find a coloring of vertices using the minimum number of colors such that adjacent vertices are colored differently
- KNAPSACK: given a collection of n items, each with a value and a weight, and a weight upper bound W, find a maximum-valued subset of items with total weight at most W
- SATISFIABILITY: given a boolean formula φ, find a truth assignment satisfying φ; e.g.,

$$\varphi(x_1,\dots,x_6) = (\bar{x}_1 \vee x_4 \vee x_6) \wedge (x_2 \vee \bar{x}_4 \vee x_5) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_3 \vee x_5 \vee \bar{x}_6)$$

Variations of a Problem

Consider the COLORING problem. (A *proper coloring* is a coloring of vertices such that adjacent vertices have different colors.)

- Optimization version: given a graph G, find a proper coloring of G using the minimum number of colors
- Search version: given a graph G and an upper bound $b \le |V|$, find a proper coloring of G using at most b colors, or report that no proper coloring exists

Optimization vs. Search

The optimization version and the search version of COLORING are equally hard (or equally easy).

There's also another variation

- Decision Version: given a graph G and an upper bound $b \le |V|$, decide whether or not G has a proper b coloring

Variations of a Problem

Consider SATISFIABILITY

- Search Version: given a boolean formula φ, find a truth assignment satisfying φ
- Decision Version: given a boolean formula φ, decide whether or not φ can be satisfied at all

Satisfiable formula:

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_3)$$

Unsatisfiable formula:

$$\varphi = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$

Decision vs. Search

- Decision version is clearly easier than search version. Specifically, solving search version $\Rightarrow$ solving decision version
- Turns out that in most cases in $\mathbf{NP}$ the decision version is equivalent to the search version
- Thus, we will focus on decision problems

Decision Problems

- A decision problem X is a set of instances. For examples:
  - $X = \textsc{Graph Coloring}$, an instance consists of a graph G and an upper bound $b \leq |V|$
  - $X = \textsc{Satisfiability}$, an instance consists of just a boolean formula φ
- X can be partitioned into YES-instances and NO-instances

$$X = X_{\text{YES}} \cup X_{\text{NO}}$$

  - X_{YES} is the subset of instances whose answers are YES
  - X_{NO} is the subset of instances whose answers are NO.
- Examples:
  - $X = \textsc{Graph Coloring}$, YES-instances are graphs G which have a proper coloring with $\leq b$ colors
  - $X = \textsc{Satisfiability}$, NO-instances are boolean formulas which cannot be satisfied

Encodings and Input Sizes

- To serve as input to an algorithm, instances need to be encoded
- The encoding decides the input size
- $\textsc{Graph Coloring}$
  - G could be encoded with an adjacency matrix
  - b is encoded in binary format
  - Input size is thus roughly $n^2 + \lg k = \Theta(n^2)$
- $\textsc{Knapsack}$
  - Values and weights encoded with a table of two rows: one row for v_i in binary, another row for w_i in binary
  - W is encoded in binary
  - Input size is thus roughly $\lg W + \sum_{i=1}^{n}(\lg v_i + \lg w_i)$

Reasonable Encodings

- Encoding has a huge effect on running time (poynomial time or not, e.g.)
  - KNAPSACK has a dynamic programming algorithm run in time $O(nW)$
  - This running time is polynomial if W is encoded in unary
  - This running time is exponential if W is encoded in binary
- We will assume that all our problems use "reasonable encodings"
  - Graphs are encoded with adjacency matrices
  - Sets are encoded with a sequence of 0's and 1's
  - Numbers (weights, costs, etc.) are encoded in binary

P

Definition

$X \in \mathbf{P}$ if there is a polynomial time algorithm $A(\cdot)$ such that, for any instance $x \in X$,

$$x \in X_{\text{YES}} \iff A(x) = \text{YES}$$

Example (BIPARTITE MATCHING)

Given a bipartite graph G and a bound b, decide if there is a matching in G of size at least b.

Example (2-SATISFIABILITY)

Given a boolean formula φ in 2-CNF, decide if φ can be satisfied. E.g.,

$$\varphi = (x_1 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_5) \wedge (x_2 \vee x_4) \wedge (\bar{x}_2 \vee x_4) \wedge (\bar{x}_3 \vee x_5)$$

NP

Definition

$X \in \mathbf{NP}$ if there exists a polynomial time verification algorithm $V(\cdot, \cdot)$, such that for any instance $x \in X$,

$$x \in X_{\text{YES}} \iff \exists \text{ certificate } y, |y| = \text{poly}(|x|), V(x, y) = \text{YES}$$

Example (Graph Coloring)

Given a graph $G = (V, E)$ and a bound $b \leq |V|$, decide if there is a proper coloring of G using at most b colors.

Example (3-Satisfiability)

Given a boolean formula φ in 3-CNF, decide if φ can be satisfied. E.g.,

$$\varphi = (x_1 \vee \bar{x}_2 \vee \bar{x}_5) \wedge (\bar{x}_3 \vee \bar{x}_5 \vee x_6) \wedge (x_2 \vee x_4 \vee \bar{x}_6) \wedge (\bar{x}_3 \vee x_5 \vee x_6)$$

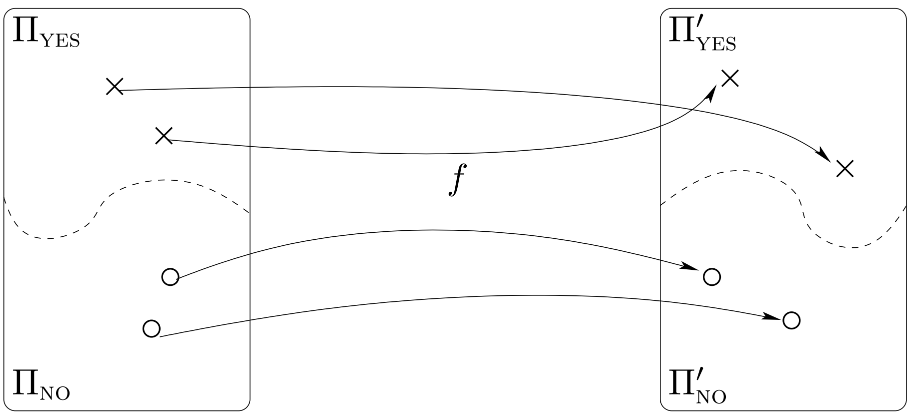
Polynomial Time Reduction

Definition (Karp Reduction)

A problem X is polynomial time reducible to a problem Y if there is a polynomial time algorithm F computing a mapping $f : X \to Y$ such that

$$\forall x \in X, \quad x \in X_{\text{YES}} \iff f(x) \in Y_{\text{YES}}$$

We write $X \leq_p Y$, and think X is not harder than Y.



VERTEX COVER $\leq_p$ INDEPENDENT SET

A vertex cover of a graph G is a subset S of vertices of G such that each edge of G is incident to at least one vertex in S.

Definition (VERTEX COVER – VC)

Instance: A graph $G = (V, E)$, and a bound $b \in \mathbb{N}$, $1 \leq b \leq |V|$.
Question: Is there a vertex cover of size at most b?

An independent set of a graph G is a subset of vertices no two of which are adjacent.

Definition (INDEPENDENT SET – IS)

Instance: A graph $G = (V, E)$, and a bound $b \in \mathbb{N}$, $1 \leq b \leq |V|$.
Question: Is there an independent set of G of size at least b?

INDEPENDENT SET $\leq_p$ CLIQUE

A clique of a graph G is a subset of vertices *every* two of which are adjacent.

Definition (CLIQUE)

Instance: A graph $G = (V, E)$, and a bound $b \in \mathbb{N}$, $1 \leq b \leq |V|$.
Question: Is there a clique of G of size at least b?

NP-Complete Problems

Definition

X is **NP**-hard iff *every* problem in **NP** is reducible to X.
X is **NP**-complete iff $X \in \mathbf{NP}$ and X is **NP**-hard.

Lemma (Transitivity)

If $X \le_p Y$ and $Y \le_p Z$, then $X \le_p Z$.

Lemma (Y Easy $\Rightarrow$ X Easy)

If $X \le_p Y$ and $Y \in \mathbf{P}$, then $X \in \mathbf{P}$.

Lemma (X Hard $\Rightarrow$ Y Hard)

*If $X \le_p Y$ and X is **NP**-hard, then Y is **NP**-hard.*
*In particular, If $X \le_p Y$, X is **NP**-complete, and $Y \in \mathbf{NP}$, then Y is **NP**-complete.*

Cook-Levin Theorem

- For any Boolean variable x, x and $\bar{x}$ are called literals
- A clause is a disjunction of literals, e.g.

$$C = (x \vee y \vee \bar{z} \vee w \vee \bar{t})$$

- A boolean formula is in conjunctive normal form (CNF) if it is a conjunction of clauses, e.g.

$$\varphi(x_1, \dots, x_6) = (\bar{x}_1 \vee x_4) \wedge (x_2 \vee \bar{x}_4 \vee x_5) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge \bar{x}_3$$

- SATISFIABILITY (or SAT): given a CNF formula φ, decide if it is satisfiable, i.e. if $\exists$ an assignment of TRUE/FALSE to all variables such that $\varphi =$ TRUE

Theorem (Cook-Levin)

SATISFIABILITY *is **NP**-complete.*

The Problems

Definition (3-SATISFIABILITY – 3-SAT)

Instance: A CNF formula φ with clauses $C_1, \dots, C_m$ over variables $x_1, \dots, x_n$, where each clause C_i consists of exactly 3 literals.
Question: Is there a truth assignment satisfying φ

Definition (SET COVER – SC)

Instance: A family $\mathcal{S}$ of m subsets of a universe set U of size n, and a bound $b \in \mathbb{N}$, $1 \leq b \leq m$.
Question: Is there a collection of at most b members of $\mathcal{S}$ whose union is the universe?
(Such a collection is called a *set cover* of U.)

The Problems

Definition (3-COLORABILITY)

Instance: A graph $G = (V, E)$.
Question: Is there a proper coloring of G using at most 3 colors?

Definition (k-COLORABILITY)

Instance: A graph $G = (V, E)$, a positive integer $k \leq |V|$.
Question: Is there a proper coloring of G using at most k colors?

The Problems

An Hamiltonian cycle of a graph G is a cycle containing all vertices of G.

Definition (Hamiltonian cycle – HC)

Instance: A graph $G = (V, E)$.
Question: Does G contain a Hamiltonian cycle?

A TSP tour is just another name for a Hamiltonian cycle.

Definition (Traveling Salesman Problem – TSP)

Instance: A complete graph $G = (V, E)$, a cost function $c : E \to \mathbb{Z}^+$, and a bound $b \in \mathbb{Z}^+$.
Question: Is there a TSP tour with total cost at most b?

The Problems

Definition (Subset Sum – SS)

Instance: A finite set S of natural numbers, and a target number $t \in \mathbb{N}$.
Question: Is there a subset $T \subseteq S$, whose elements sum up to t?

Definition (Dominating Set – DS)

Instance: A graph $G = (V, E)$, a bound $b \in \mathbb{N}$, $1 \le b \le |V|$.
Question: Is there a subset $S \subseteq V$ of size at least b such that every vertex not in S is incident to some vertex in S.
(The vertices in S dominates all vertices in V.)

The Reductions