We've done

- Dynamic Programming

Now

- In $\mathbf{P}$ or not in $\mathbf{P}$, that's the question!
  (A million dollar question, by the way.)

Next

- Approximation algorithms

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 $O(n^5)$

The question is

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

A few 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.

Note: we have seen a DP algorithm for 01-knapsack which runs in $O(nW)$, but this is NOT poly-time as we shall discuss.

So

$$\boxed{\text{What can (or should) we do?}}$$

Dealing with "hard" problems

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

1. Email ask the prof
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!
   - 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

The questions are

> What exactly do we mean by "hard"?
>
> How do we show that two problems are equivalently hard?

Hard and "Equivalently" Hard

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.

(Part of the following discussion is based on Chaitin's book "The Unknowable".)

Georg Ferdinand Ludwig Philipp Cantor (1845–1918)

Cantor

In later decades of the 19th century, he considered:

The ordinal numbers:

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

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

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

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

$$0, \dots, \omega, \omega + 1, \dots, 2\omega, \dots, 3\omega, \dots, \omega^2$$

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

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

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

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!

Cantor (cont)

He did not stop there. Let 2^S be the set of all subsets of a set S. Cantor showed that $|S| < |2^S|$ using the **diagonalization argument**.

He also considered: **The cardinal numbers:**

$$\aleph_0 = |\mathbb{N}|$$

$$\aleph_1 = |2^{\mathbb{N}}|$$

$$\aleph_2 = |2^{2^{\mathbb{N}}}|$$

continue this way, we get

$$\aleph_0, \aleph_1, \dots, \aleph_\omega$$

why stop there?

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

So the ordinal numbers were used to index the cardinal numbers!

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 others just loved Cantor’s work.

Cantor ended his life in a mental hospital.

Why the debate? The Paradoxes of Set Theory

Liar paradox:

“This statement is false!”

Barber paradox:

“in a village, a barber shaves everyone who does not shave himself”

Russell’s paradox:

“consider the set of all sets which are not members of themselves”

Examples:
- Set of all conceivable concepts
- Set of all Bush supporters

Berry’s paradox:

“the first natural number which cannot be named in less than fifteen English words”

So, what is the solution?

- Use symbolic logic to do math (Peano, Russell, Whitehead, ...) In fact, an entire volume of Russell's and Whitehead's 3-volume "Principia Mathematica" was needed to show that ... $1 + 1 = 2$!
- Intuitionism (Brouwer): "the only thing 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.

  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!

  Plus a few other things, this idea is referred to as the Hilbert's program: formalizing Mathematics (as if it is not formal enough).

David Hilbert (1962–1943)

Hilbert's Problems

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

- Problem 1a: is there a transfinite number between that of a enumerable set and the numbers of *the continuum*?
- 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?

He also asked: is mathematics decidable, i.e., is there an algorithm which decides if a statement is provable? (Entscheidungsproblem)

Kurt Gödel (1906-1978)

On Hilbert's second problem, Gödel showed (1931) that (any axiomatic system of) "mathematics" is either inconsistent or incomplete!

Alan Turing

Turing machine (1936): models "algorithm" in Entscheidungsproblem. Also, Turing machine models "algorithm" in Hilbert's 10th problem.

Answer to Entscheidungsproblem: the "halting problem" is undecidable. Turing used, again, the diagonalization argument.

Church-Turing Thesis

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

(other computational models can only be as strong as the Turing machine model)

Alonzo Church (1903–1995)

Easy and Hard Problems

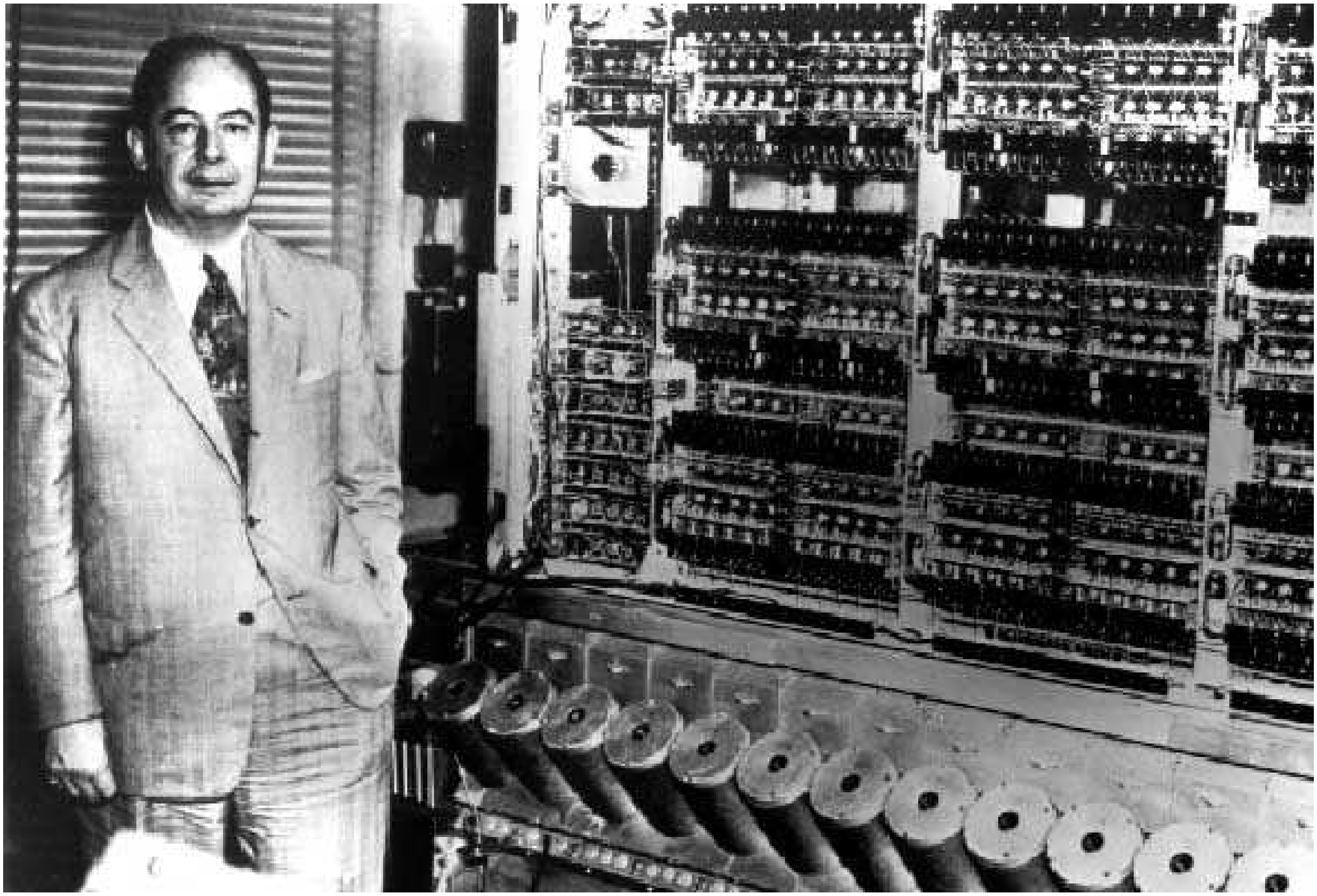
- In the 60's, people noticed some algorithms require longer time to run, i.e. harder, while some are easier
- Polynomial-time computation: von Neumann (1953), Cobham (1964), Edmonds (1965)
- Edmonds called poly-time algo. a "good algo."
- Informally, we define

  $\mathbf{P}$:= the class of problems which have a poly-time algorithm

- Problems in $\mathbf{P}$ are considered to be "easier" than problems not in $\mathbf{P}$

  (Formally, $\mathbf{P}$ is the set of languages each of which is recognized by some Deterministic Turing Machine in poly-time.)

John von Neumann (1903–1957)



Jack Edmonds

The classes of problems which are respectively known and not known to have good algorithms are of great theoretical interest ... I conjecture that there is no good algorithm for the traveling salesman problem. My reasons are the same as for any mathematical conjecture: (1) it is a legitimate mathematical possibility; and (2) I do not know.

Decision Problems

For technical reasons, we only consider decision problems: YES-NO questions.

- Given n cities, is there a TSP tour of length at most l?
- Given a graph G, is there a vertex cover of size at most k?

Note: a problem is at least as hard as its the decision version.

- If we can solve TSP, then we only have to check if $\text{OPT}(\text{TSP}) \leq l$
- If we can solve VC, then we only have to check if $\text{OPT}(\text{VC}) \leq k$

Encoding instances of a problem

- Technically, a problem Π is a set of its instances
- An algorithm for Π runs on instances of Π
- Actually, an algorithm runs on encoded instances

Example: in the VERTEX-COVER problem

- a particular graph is an instance,
- the graph's adjacency matrix is an encoding for the instance.
- an algorithm for finding a minimum VC has adjacency matrices as inputs

Note:

- The encoding decides the size of the inputs
- With the adjacency matrix encoding, the input size is actually $\Theta(n^2)$
- Polynomial time or not depends on the input size, i.e. on the encoding scheme

More on input sizes and encodings

- We have been informal on what input size of a problem is
- If the input size is $\Theta(n^4)$, and the algorithm runs in $\Theta(n^{100})$, then it is still a poly-time algorithm
- In fact, if the input size is $f(n)$, a polynomial in n, and the running time is $g(n)$, another polynomial in n, then the running time is polynomial!

The encoding scheme could make a huge difference, e.g.

- Primality testing: given a number n, check if n is a prime
- Suppose for each $k < n$, we check if n is divisible by k which takes time about $O(\lg n)$
- The total running time is $O(n \lg n)$, right?
- The answer is: it depends on how we encode n.
- If we encode n in unary format (n 1-bits), then the answer is YES
- In binary format, the input size is $m \approx \lg n$, and hence the running time is $O(2^m m)$, exponential!

Reasonable encodings

We shall assume that we use only **reasonable** encodings.

In particular, numbers are encoded in binary format.

Thus, our DP solution to 01-Knapsack, which was $O(nW)$, is not a poly-time algorithm. (Why?)

Decision problems again

Think of each problem Π as a set of instances.

Π_{YES} is the subset of Π whose answer is YES.

Π_{NO} is the subset of Π whose answer is NO.

Thus,

$$\Pi = \Pi_{\text{YES}} \cup \Pi_{\text{NO}}.$$

P

$\Pi \in \mathbf{P}$ (read "solvable in polynomial time") if there is a poly-time algorithm $A(\cdot)$, such that for any instance $x \in \Pi$

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

NP

$\Pi \in \mathbf{NP}$ (read "solvable in nondeterministic polynomial time") if there is a poly-time verification algorithm $V(\cdot, \cdot)$, such that for any instance $x \in \Pi$,

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

Examples,

- VC: $V(x, y)$ interprets x as the graph G, y as a set of vertices, and check if $|y| \leq k$ and y is a VC.
- TSP: $V(x, y)$ interprets x as the cities, y as a tour T, and check if the length of T is $\leq l$.

What about 01-KNAPSACK, what's the decision problem and the verification procedure?

Polynomial time reduction

A problem Π is **polynomial time reducible** to a problem Π' if there is a **polynomial time computable function** $f : \Pi \to \Pi'$ such that for any $x \in \Pi$,

$$x \in \Pi_{\text{YES}} \iff f(x) \in \Pi'_{\text{YES}}$$

We write $\Pi \leq_p \Pi'$, and think Π is not harder than Π'.

Example:

- VERTEX-COVER and CLIQUE
- CLIQUE and INDEPENDENT SET

Lemma 1. *If* $\Pi' \in \mathbf{P}$, *and* Π *is reducible to* Π', *then* $\Pi \in \mathbf{P}$.

NP-Complete Problems

Π is **NP-hard** if every problem in **NP** is reducible to Π.

$$\boxed{\begin{array}{c}\Pi \text{ is } \textbf{NP-complete} \text{ if and only if} \\ \Pi \in \mathbf{NP} \text{ and } \Pi \text{ is } \mathbf{NP}\text{-hard.}\end{array}}$$

Lemma 2. *Suppose* $\Pi \in \mathbf{NP}$, *and* $\Pi' \leq_p \Pi$ *where* Π' *is* **NP**-*complete, then* Π *is* **NP**-*complete.*

- Let $X = \{x_1, \dots, x_n\}$ be a set of Boolean variables.
- A *truth assignment* for X is a function $t : X \to \{\text{TRUE, FALSE}\}$.
- $\bar{x}$ denote the negation of x.
- x and $\bar{x}$ are called *literals*.
- A *clause* over X is a set C of literals, e.g. $C = \{x_1, \bar{x}_3, x_4\}$ is a clause.
- A clause C is *satisfied* by a truth assignment t iff at least one of its member is TRUE under t.

SATISFIABILITY (SAT)

INSTANCE: A set X of variables and a collection $\mathcal{C}$ of clauses over X.

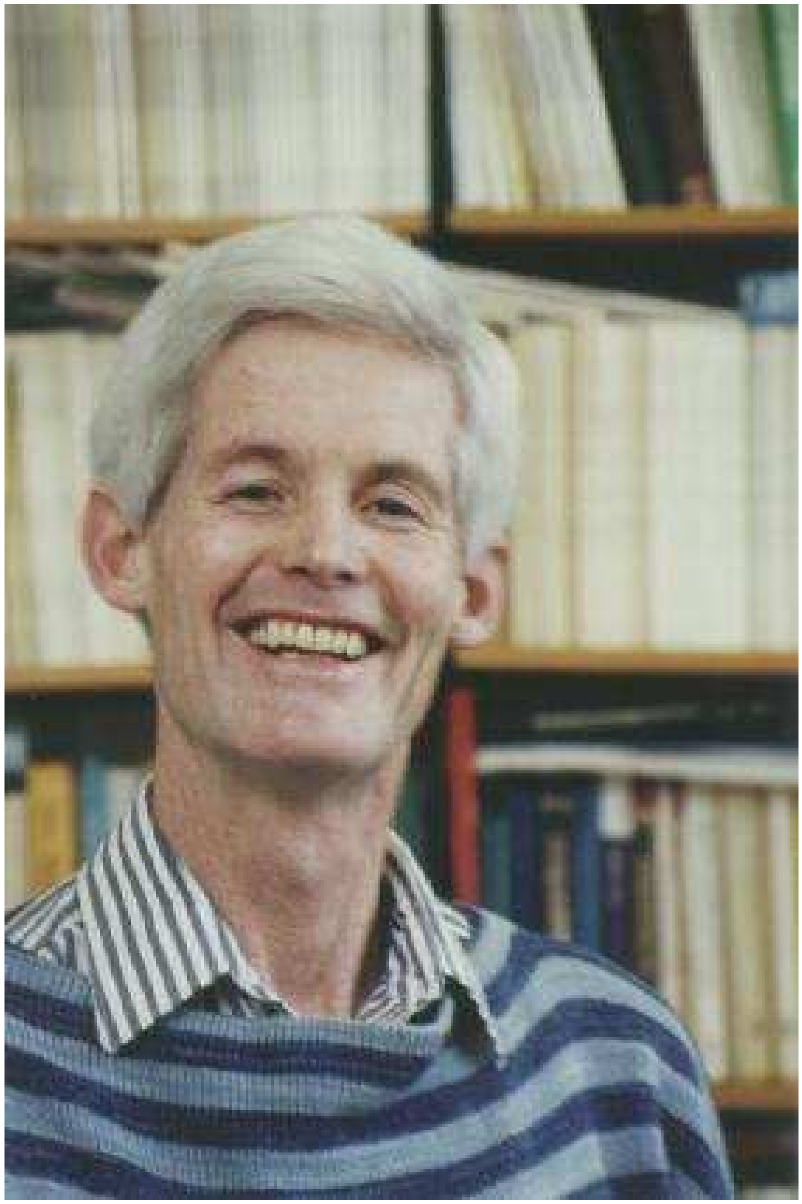
QUESTION: Is there a truth assignment which satisfies all clauses in $\mathcal{C}$.

Intuitively, we want a truth assignment for which $f = \text{TRUE}$, given f under *conjunctive normal form* (CNF), e.g.

$$f(x_1, \dots, x_n) = (x_1 + \bar{x}_3 + x_4)(x_2 + x_3)(\bar{x_1} + x_2 + \bar{x_3})$$

Stephen Cook

In 1971 he showed that SAT, 3-SAT, and SUBGRAPH ISOMORPHISM are **NP**-complete.



Leonid Levin

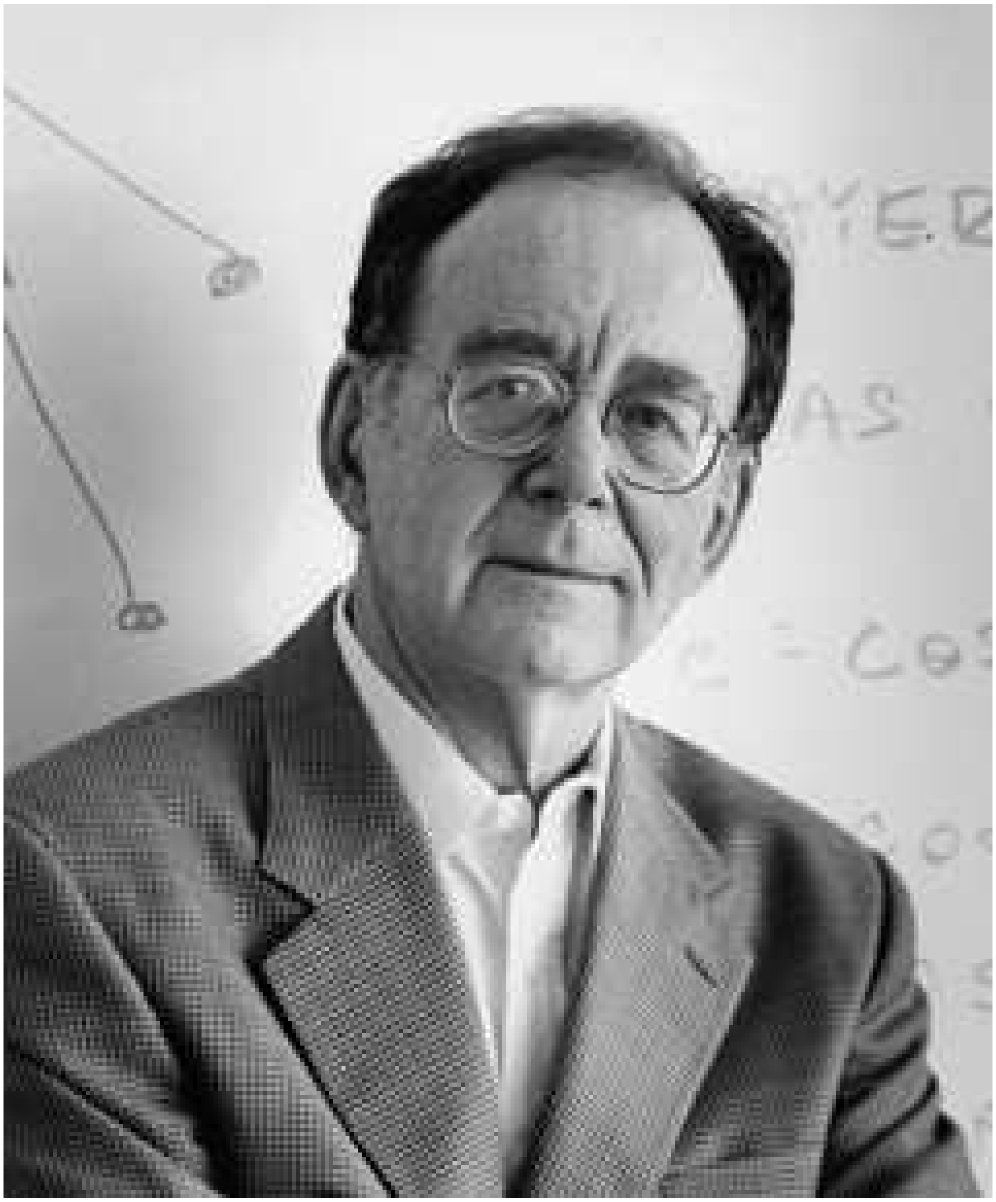
Wrote his doctoral thesis in 1971 under Kolmogorov

Was denied his Ph.D. for political reasons

Published a paper in 1973 showing the same Cook's result.

Richard Karp

In 1972, he showed that 20 other problems are **NP**-complete also, including VC, TSP, CLIQUE, INDEPENDENT SET,



Knuth

1974, settled the terminologies we are using today.

A Connection to Turing

Cook, Karp, Knuth got Turing awards.

Levin did not.

The Millennium Prize Problems

In the spirit of Hilbert, Clay Research Institute offered one million dollars award to whoever solves one of a few outstanding problems, including

- $\mathbf{P} = \mathbf{NP}$?
- The Riemann hypothesis (Hilbert's 8th problem)

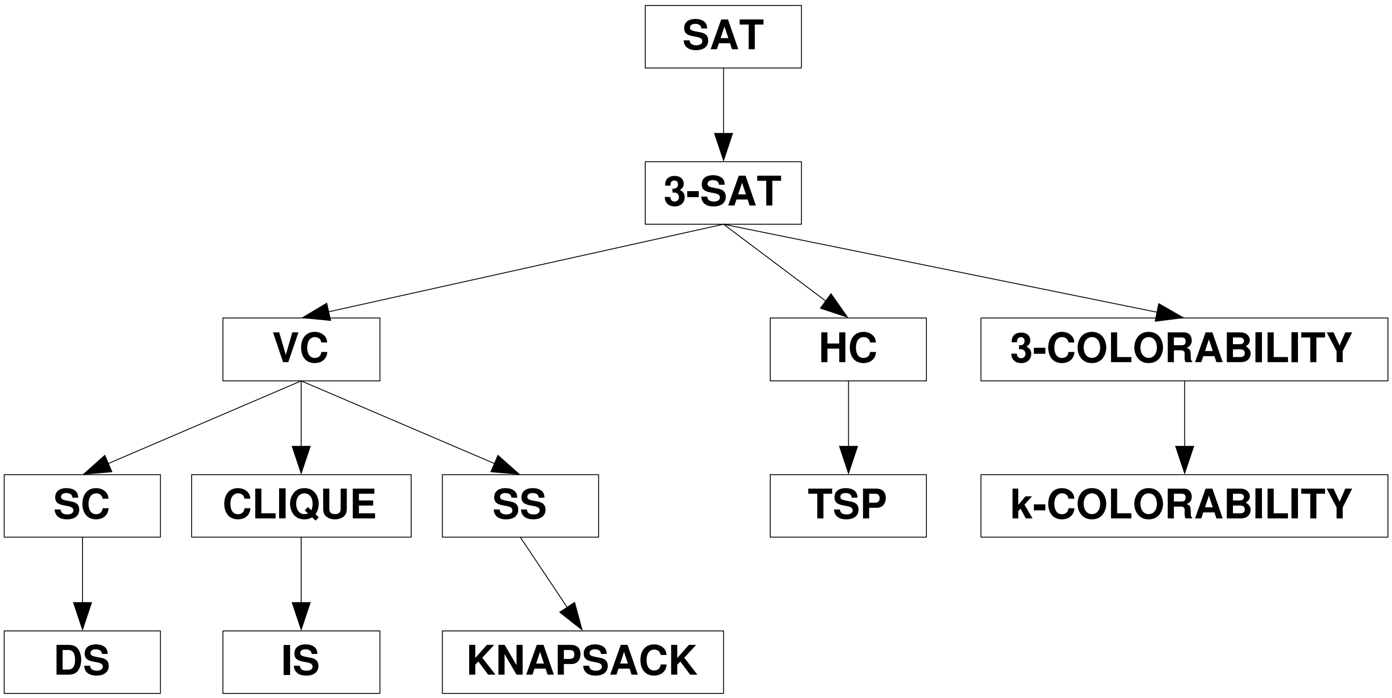
We believe in two possibilities

- $\mathbf{P} = \mathbf{NP}$? is independent (unlikely)
- $\mathbf{P} \neq \mathbf{NP}$, because natural evolution takes a long time to optimize "natural things", it should take computers a long time to solve "natural problems"

What do we do next?

- New computational models and physical computers, e.g. Quantum computers (probably still equivalent to Turing Machine)
- Randomized algorithms
- Approximation algorithms
- ...

Basic NP-complete problems - Our road map



Other problems are defined as we go along.

3-SAT

INSTANCE: A collection $\mathcal{C}$ of clauses $\{C_1, \ldots, C_m\}$ over $X = \{x_1, \ldots, x_n\}$, where each clause C_i consists of 3 literals.

QUESTION: Is there a truth assignment satisfying all of $\mathcal{C}$?

VERTEX COVER (VC)

INSTANCE: A graph $G = (V, E)$, and a bound $b \in \mathbb{Z}^+$.

QUESTION: Is there a vertex cover of size at most b?

CLIQUE

INSTANCE: A graph $G = (V, E)$, and a bound $b \in \mathbb{Z}^+$.

QUESTION: Is there a clique in G clique of size at least b?

INDEPENDENT SET (IS)

INSTANCE: A graph $G = (V, E)$, and a bound $b \in \mathbb{N}$.

QUESTION: Is there an independent set of G of size at least b?

SAT $\leq_p$ 3-SAT

Since 3-SAT $\in$ **NP** (why?) - 3-SAT is **NP**-complete.

Given an instance $\mathcal{C}$ of SAT, we would like to construct an instance $\mathcal{C}'$ of 3-SAT in polynomial time such that $\mathcal{C}$ is satisfiable if and only if $\mathcal{C}'$ is satisfiable.

Example:

$$\mathcal{C} = \{\{\bar{x}_3\}, \{x_1, \bar{x}_4\}, \{x_1, \bar{x}_2, \bar{x}_3, x_4\}\}$$

Recall that we interpret this as

$$\phi_{\mathcal{C}} = \bar{x}_3(x_1 + \bar{x}_4)(x_1 + \bar{x}_2 + \bar{x}_3 + x_4 + x_5 + \bar{x}_6)$$

In $\mathcal{C}'$, there are 4 clauses to make up for $\bar{x}_3$:

$$(\bar{\mathbf{x}}_\mathbf{3} + a + b)(\bar{\mathbf{x}}_\mathbf{3} + \bar{a} + b)(\bar{\mathbf{x}}_\mathbf{3} + a + \bar{b})(\bar{\mathbf{x}}_\mathbf{3} + \bar{a} + \bar{b})$$

2 clauses for $(x_1 + \bar{x}_4)$:

$$(\mathbf{x}_\mathbf{1} + \bar{\mathbf{x}}_\mathbf{4} + c)(\mathbf{x}_\mathbf{1} + \bar{\mathbf{x}}_\mathbf{4} + \bar{c})$$

4 clauses for $(x_1 + \bar{x}_2 + \bar{x}_3 + x_4 + x_5 + \bar{x}_6)$:

$$(\mathbf{x}_\mathbf{1} + \bar{\mathbf{x}}_\mathbf{2} + d_1)(\bar{d}_1 + \bar{\mathbf{x}}_\mathbf{3} + d_2)(\bar{d}_2 + \mathbf{x}_\mathbf{4} + d_3)(\bar{d}_3 + \mathbf{x}_\mathbf{5} + \bar{\mathbf{x}}_\mathbf{6})$$

3-SAT $\leq_p$ VC

VC is in **NP** obviously.

Given an instance $\mathcal{C} = \{C_1, \dots, C_m\}$ of 3-SAT, we would like to construct an instance (G, b) of VC in poly time such that $\mathcal{C}$ is satisfiable if and only if G has a VC of size at most b.

VC $\leq_p$ IS

IS is in **NP** obviously.

Given $G = (V, E)$. A subset $S \subseteq V$ is a vertex cover of size $|S| \leq b$ of G iff $V - S$ is an independent set of size at least $|V| - b$.

IS $\leq_p$ CLIQUE

Let $G = (V, E)$, $S \subseteq V$ is an independent set if and only if S is a clique of $\bar{G} = (V, \bar{E})$.

SET COVER (SC)

INSTANCE: A family $\mathcal{S}$ of subsets $\{S_1, \dots, S_m\}$ of a finite universe U ($|U| = n$), and a bound $b \in \mathbb{Z}^+$.

QUESTION: Is there $I \subseteq \{1, \dots, m\}$, $|I| \leq b$, such that

$$U = \bigcup_{i \in I} S_i$$

DOMINATING SET (DS)

INSTANCE: A graph $G = (V, E)$, a bound $b \in \mathbb{Z}^+$.

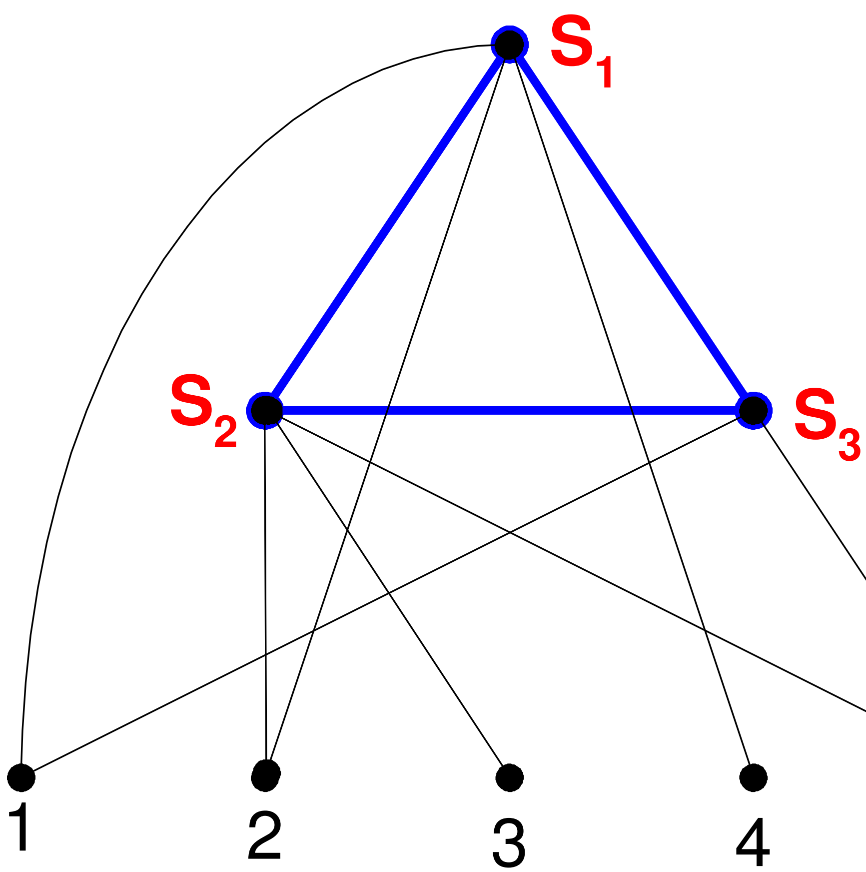
QUESTION: Is there $S \subseteq V$, $|S| \leq b$, such that every vertex v not in S is incident to some vertex in S

VC $\leq_p$ SC

Given an instance (G, b) of VC where $G = (V, E)$, let $U = E$, S_v be the set of edges incident to $v \in V$. Then a set $C \subseteq V$ of vertices cover all edges of G if and only if $E = \cup_{v \in C} S_v$.

SC $\leq_p$ DS

$S_1 = \{1, 2, 4\}, S_2 = \{2, 3\}, S_3 = \{1, 5\}$.



3-COLORABILITY

INSTANCE: A graph $G = (V, E)$.

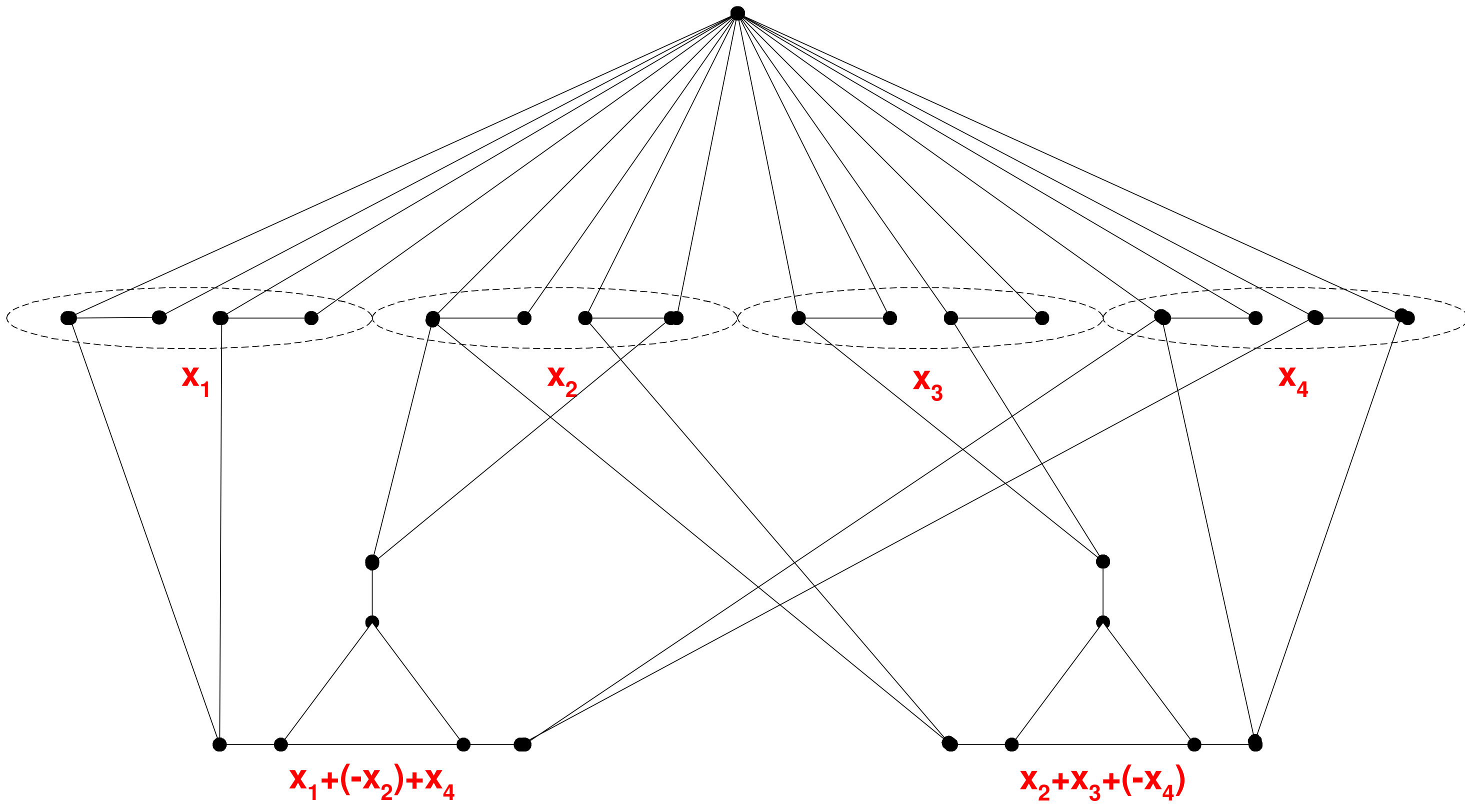
QUESTION: Is G 3-colorable, i.e. is there a way to assign each vertex of G one of 3 colors such that two adjacent vertices have different colors.

k-COLORABILITY

INSTANCE: A graph $G = (V, E)$, $k \geq 3$.

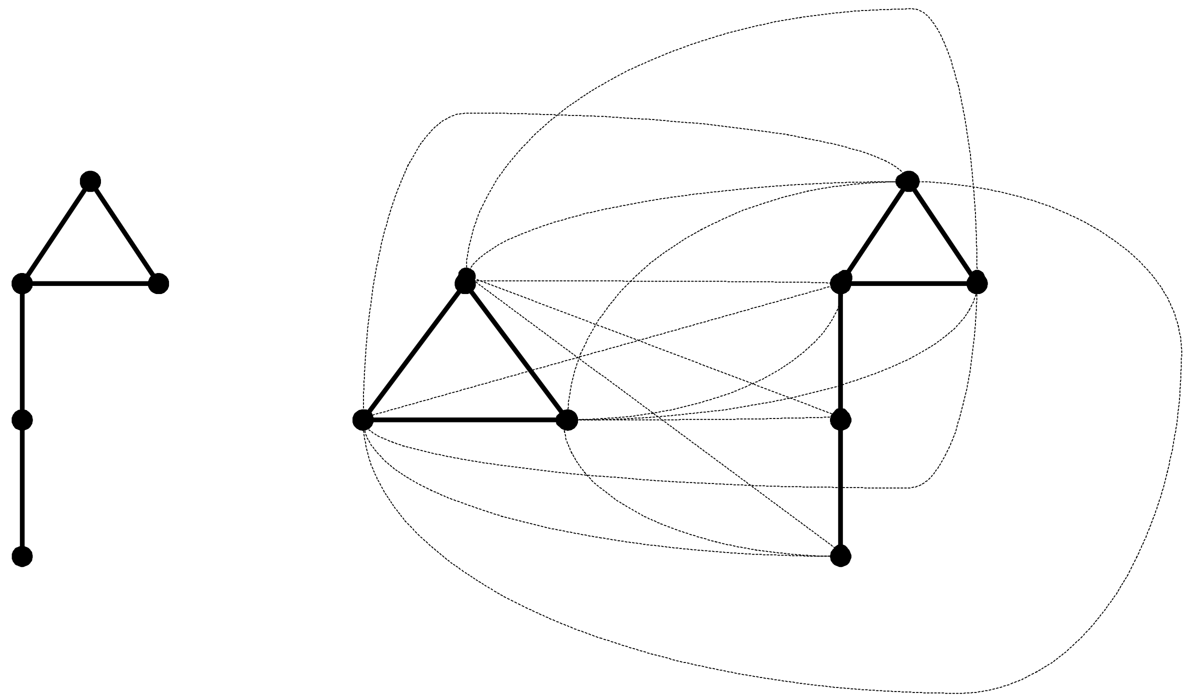
QUESTION: Is G k-colorable, i.e. is there a way to assign each vertex of G one of k colors $\{1, \dots, k\}$ such that two adjacent vertices have different colors.

3-SAT $\leq_p$ 3-COLORABILITY



Example for $(x_1 + \bar{x}_2 + x_4)(x_2 + x_3 + \bar{x}_4)$.

3-COLORABILITY $\leq_p$ k-COLORABILITY



Example for $k = 6$.

HAMILTONIAN CIRCUIT (HC)

INSTANCE: A graph $G = (V, E)$.

QUESTION: Does G contain a Hamiltonian Circuit? (An HC is a cycle containing all vertices of G.)

TRAVELING SALESMAN (TSP)

INSTANCE: A finite set C of n cities $\{1, \dots, n\}$, and their distances $d(i, j) \in \mathbb{Z}^+$, and a bound $b \in \mathbb{Z}^+$.

QUESTION: Is there a TSP tour with total length at most b.

3-SAT $\leq_p$ HC

This requires a good figure.

HC $\leq_p$ TSP

This is quite simple.

SUBSET-SUM (SS)

INSTANCE: A finite set S of natural numbers, and a target $t \in \mathbb{N}$.

QUESTION: Is there a subset $S' \subseteq S$, whose elements sum up to t.

KNAPSACK

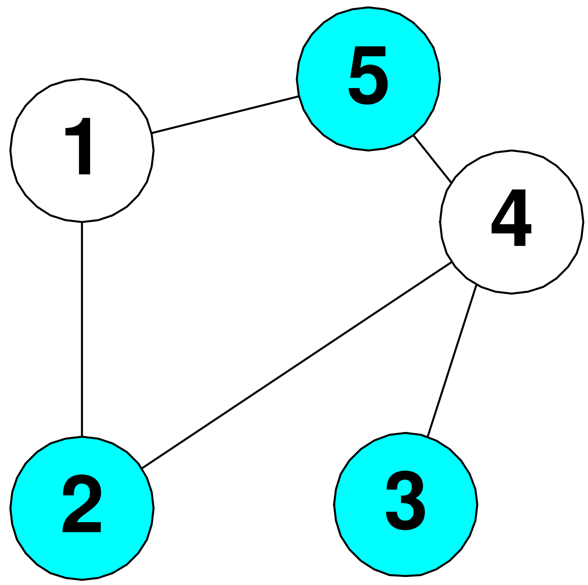
INSTANCE: n items, their values $v_i \in \mathbb{Z}^+$, their corresponding weights $w_i \in \mathbb{Z}^+$, a weight limit $W \in \mathbb{Z}^+$, and a value limit $V \in \mathbb{Z}^+$.

QUESTION: Is there a subset of items with total weight at most W, and total value at least V?

SS $\leq_p$ KNAPSACK

This is quite simple!

VC $\leq_p$ SS



Question: is there a VC of size ≤ 3?

S		12	15	24	34	45
a_1	1	1	1	0	0	0
a_2	1	1	0	1	0	0
a_3	1	0	0	0	1	0
a_4	1	0	0	1	1	1
a_5	1	0	1	0	0	1
b_{12}		1	0	0	0	0
b_{15}		0	1	0	0	0
b_{24}		0	0	1	0	0
b_{34}		0	0	0	1	0
b_{45}		0	0	0	0	1
c_1	1	0	0	0	0	0
c_2	1	0	0	0	0	0
c_3	1	0	0	0	0	0
c_4	1	0	0	0	0	0
c_5	1	0	0	0	0	0
t	**3**	2	2	2	2	2