

8. INTRACTABILITY II

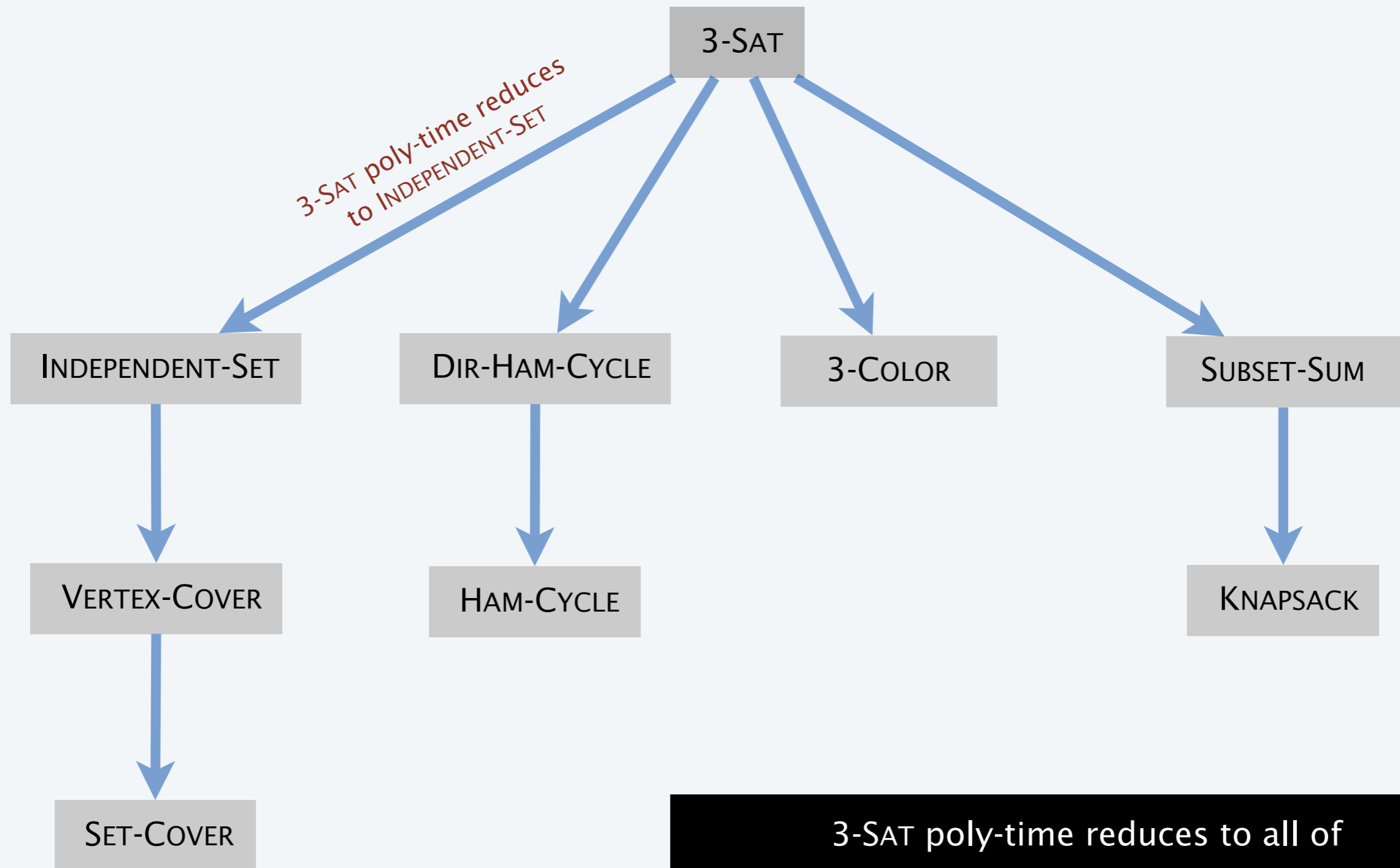
- ▶ P vs. NP
- ▶ NP -complete
- ▶ co - NP
- ▶ NP -hard

Lecture slides by Kevin Wayne

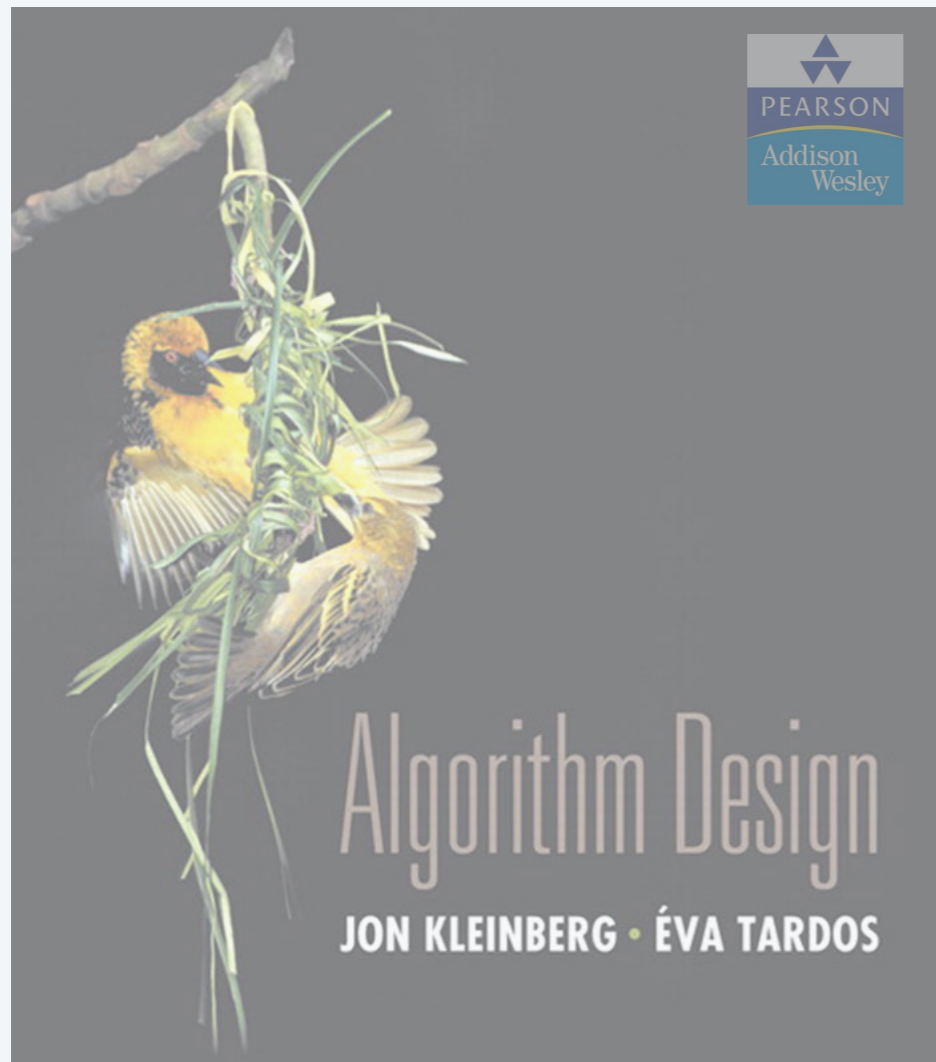
Copyright © 2005 Pearson–Addison Wesley

<http://www.cs.princeton.edu/~wayne/kleinberg-tardos>

Recap



3-SAT poly-time reduces to all of these problems (and many, many more)



SECTION 8.3

8. INTRACTABILITY II

- ▶ *P vs. NP*
- ▶ *NP-complete*
- ▶ *co-NP*
- ▶ *NP-hard*

P

Decision problem.

- Problem X is a set of strings.
- Instance s is one string.
- Algorithm A solves problem X : $A(s) = \begin{cases} \text{yes} & \text{if } s \in X \\ \text{no} & \text{if } s \notin X \end{cases}$

Def. Algorithm A runs in **polynomial time** if for every string s , $A(s)$ terminates in $\leq p(|s|)$ “steps,” where $p(\cdot)$ is some polynomial function.

↑
length of s

Def. \mathbf{P} = set of decision problems for which there exists a poly-time algorithm.

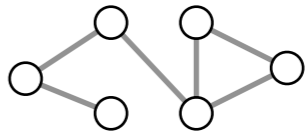
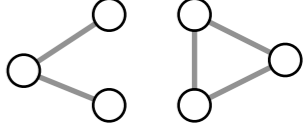
problem PRIMES: $\{ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, \dots \}$

instance s : 592335744548702854681

algorithm: Agrawal–Kayal–Saxena (2002)

Some problems in P

P. Decision problems for which there exists a poly-time algorithm.


| problem | description | poly-time algorithm | yes | no |
|---------------|---|---------------------------|---|--|
| MULTIPLE | Is x a multiple of y ? | grade-school division | 51, 17 | 51, 16 |
| REL-PRIME | Are x and y relatively prime? | Euclid's algorithm | 34, 39 | 34, 51 |
| PRIMES | Is x prime? | Agrawal–Kayal–Saxena | 53 | 51 |
| EDIT-DISTANCE | Is the edit distance between x and y less than 5? | Needleman–Wunsch | niether neither | acgggt ttttta |
| L-SOLVE | Is there a vector x that satisfies $Ax = b$? | Gauss–Edmonds elimination | $\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ |
| U-CONN | Is an undirected graph G connected? | depth-first search |  |  |

NP

Def. Algorithm $C(s, t)$ is a **certifier** for problem X if for every string s :
 $s \in X$ iff there exists a string t such that $C(s, t) = \text{yes}$.

Def. **NP** = set of decision problems for which there exists a poly-time certifier.

- $C(s, t)$ is a poly-time algorithm.
- Certificate t is of polynomial size: $|t| \leq p(|s|)$ for some polynomial $p(\cdot)$.


“certificate” or “witness”

| | |
|----------------------------|--|
| problem COMPOSITES: | { 4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, } |
| instance s: | 437669 |
| certificate t: | 541 ← 437,669 = 541 × 809 |
| certifier C(s, t): | grade-school division |

Certifiers and certificates: satisfiability

SAT. Given a CNF formula Φ , does it have a satisfying truth assignment?

3-SAT. SAT where each clause contains exactly 3 literals.

Certificate. An assignment of truth values to the Boolean variables.

Certifier. Check that each clause in Φ has at least one true literal.

$$\text{instance } s \quad \Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

$$\text{certificate } t \quad x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{false}, x_4 = \text{false}$$

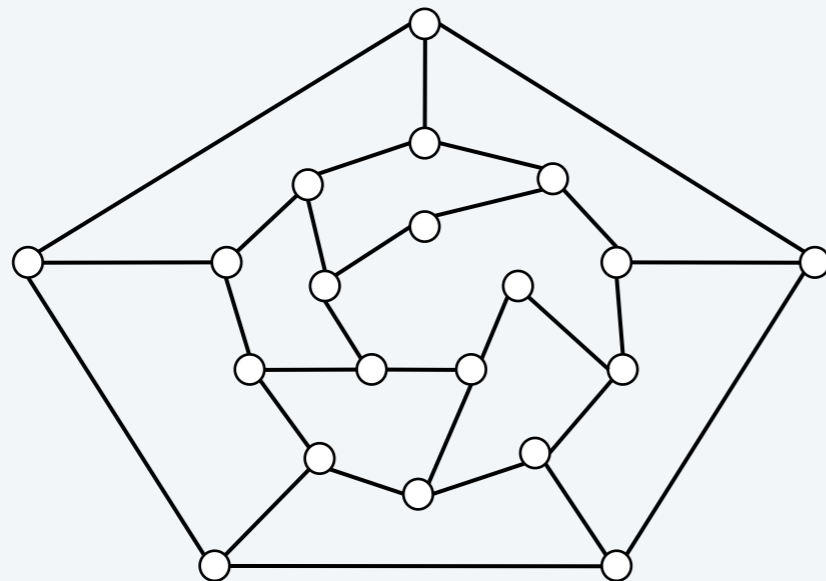
Conclusions. SAT \in **NP**, 3-SAT \in **NP**.

Certifiers and certificates: Hamilton path

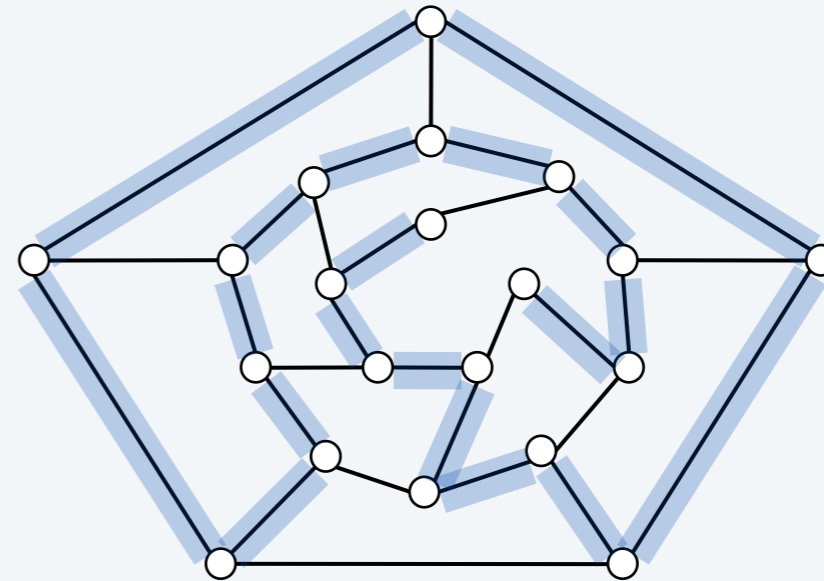
HAMILTON-PATH. Given an undirected graph $G = (V, E)$, does there exist a simple path P that visits every node?

Certificate. A permutation π of the n nodes.

Certifier. Check that π contains each node in V exactly once, and that G contains an edge between each pair of adjacent nodes.



instance s

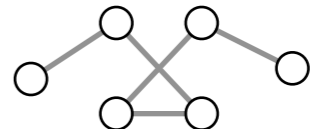
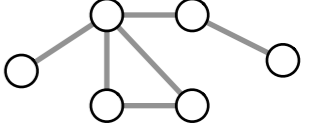


certificate t

Conclusion. HAMILTON-PATH \in **NP**.

Some problems in NP

NP. Decision problems for which there exists a poly-time certifier.

| problem | description | poly-time algorithm | yes | no |
|---------------|--|---------------------------|---|--|
| L-SOLVE | Is there a vector x that satisfies $Ax = b$? | Gauss-Edmonds elimination | $\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ |
| COMPOSITES | Is x composite? | Agrawal-Kayal-Saxena | 51 | 53 |
| FACTOR | Does x have a nontrivial factor less than y ? | ??? | (56159, 50) | (55687, 50) |
| SAT | Given a CNF formula, does it have a satisfying truth assignment? | ??? | $\neg x_1 \vee x_2 \vee \neg x_3$ $x_1 \vee \neg x_2 \vee x_3$ $\neg x_1 \vee \neg x_2 \vee x_3$ | $\neg x_2$ $x_1 \vee x_2$ $\neg x_1 \vee x_2$ |
| HAMILTON-PATH | Is there a simple path between u and v that visits every node? | ??? |  |  |



Which of the following graph problems are known to be in NP?

- A.** Is the length of the longest simple path $\leq k$?
- B.** Is the length of the longest simple path $\geq k$?
- C.** Is the length of the longest simple path $= k$?
- D.** Find the length of the longest simple path.
- E.** All of the above.



In complexity theory, the abbreviation NP stands for...

- A.** Nope.
- B.** No problem.
- C.** Not polynomial time.
- D.** Not polynomial space.
- E.** Nondeterministic polynomial time.

Significance of NP

NP. Decision problems for which there exists a poly-time certifier.

“ NP captures vast domains of computational, scientific, and mathematical endeavors, and seems to roughly delimit what mathematicians and scientists have been aspiring to compute feasibly. ” — Christos Papadimitriou

“ In an ideal world it would be renamed P vs VP. ” — Clyde Kruskal

P, NP, and EXP

P. Decision problems for which there exists a poly-time algorithm.

NP. Decision problems for which there exists a poly-time certifier.

EXP. Decision problems for which there exists an exponential-time algorithm.

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

Pf. Consider any problem $X \in \mathbf{P}$.

- By definition, there exists a poly-time algorithm $A(s)$ that solves X .
- Certificate $t = \varepsilon$, certifier $C(s, t) = A(s)$. ■

Proposition. $\mathbf{NP} \subseteq \mathbf{EXP}$.

Pf. Consider any problem $X \in \mathbf{NP}$.

- By definition, there exists a poly-time certifier $C(s, t)$ for X , where certificate t satisfies $|t| \leq p(|s|)$ for some polynomial $p(\cdot)$.
- To solve instance s , run $C(s, t)$ on all strings t with $|t| \leq p(|s|)$.
- Return *yes* iff $C(s, t)$ returns *yes* for any of these potential certificates. ■

Fact. $\mathbf{P} \neq \mathbf{EXP} \Rightarrow$ either $\mathbf{P} \neq \mathbf{NP}$, or $\mathbf{NP} \neq \mathbf{EXP}$, or both.

The main question: P vs. NP

Q. How to solve an instance of 3-SAT with n variables?

A. Exhaustive search: try all 2^n truth assignments.

Q. Can we do anything substantially more clever?

Conjecture. No poly-time algorithm for 3-SAT.

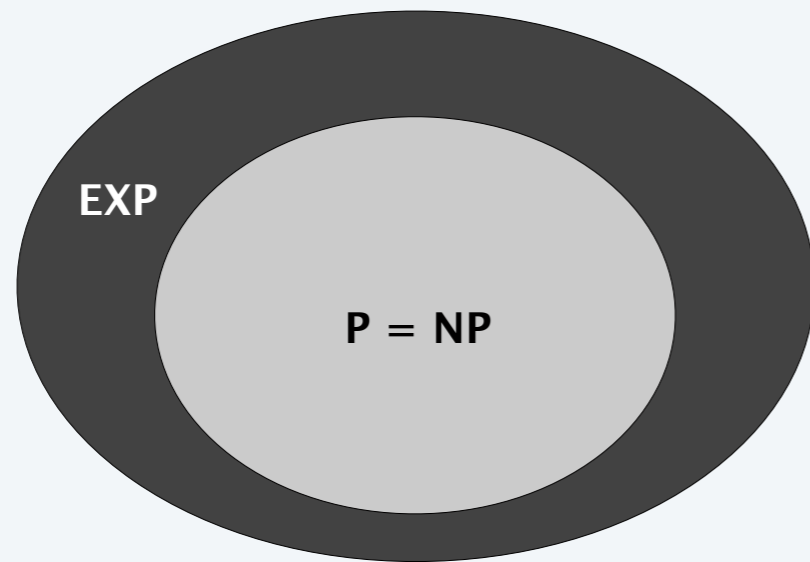
“intractable”



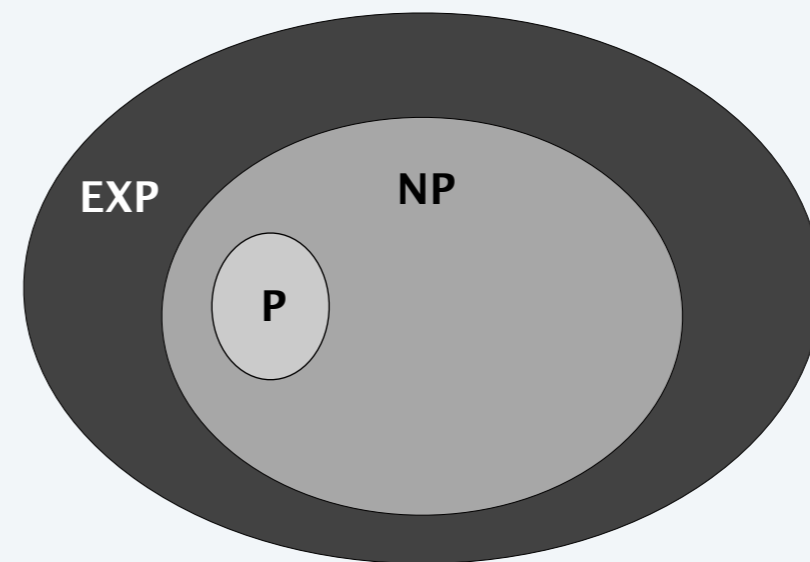
The main question: P vs. NP

Does $P = NP$? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]

Is the decision problem as easy as the certification problem?



If $P = NP$



If $P \neq NP$

If yes... Efficient algorithms for 3-SAT, TSP, VERTEX-COVER, FACTOR, ...

If no... No efficient algorithms possible for 3-SAT, TSP, VERTEX-COVER, ...

Consensus opinion. Probably no.



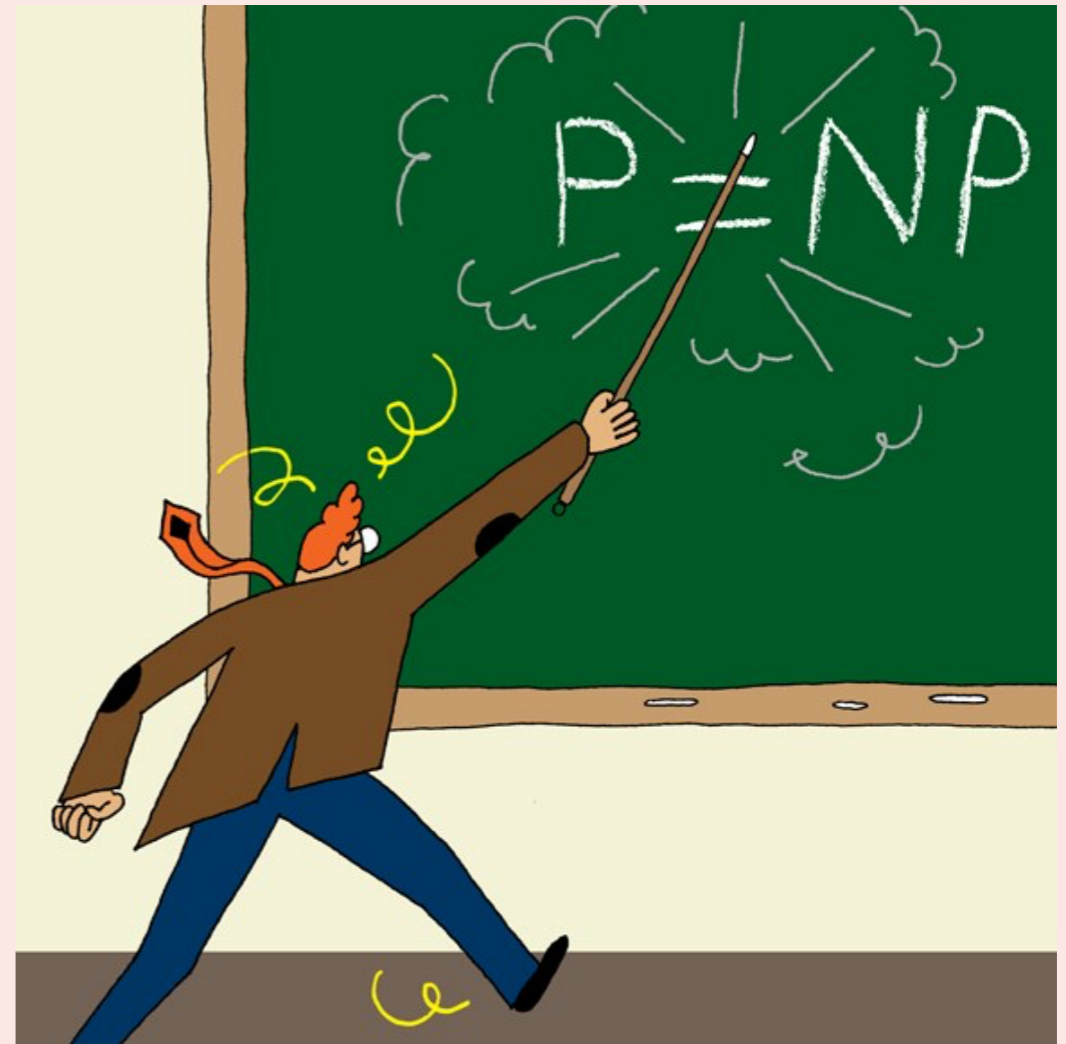
Suppose $P \neq NP$. Which of the following are still possible?

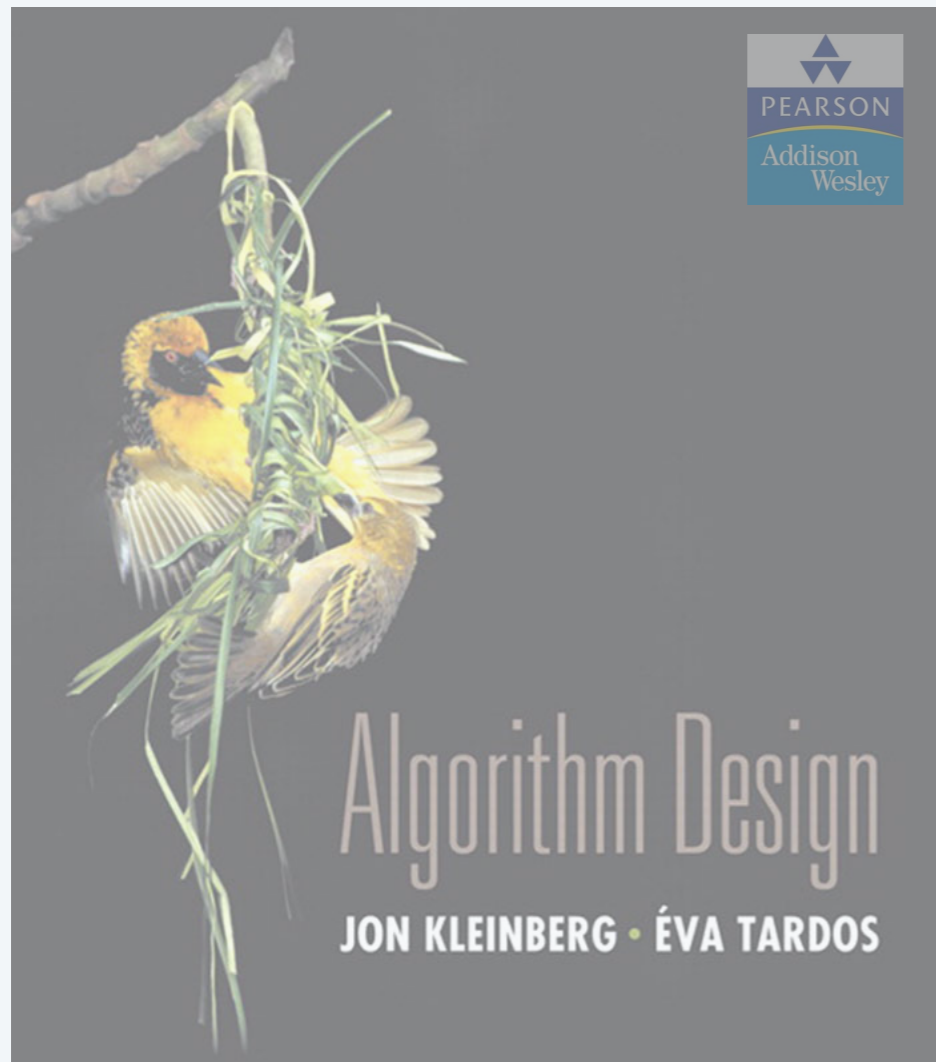
- A.** $O(n^3)$ algorithm for factoring n -bit integers.
- B.** $O(1.657^n)$ time algorithm for HAMILTON-CYCLE.
- C.** $O(n^{\log \log \log n})$ algorithm for 3-SAT.
- D.** All of the above.



Does $P = NP$?

- A. Yes.
- B. No.
- C. None of the above.





SECTION 8.4

8. INTRACTABILITY II

- ▶ *P vs. NP*
- ▶ ***NP-complete***
- ▶ *co-NP*
- ▶ *NP-hard*

NP-complete

NP-complete. A problem $Y \in \mathbf{NP}$ with the property that for every problem $X \in \mathbf{NP}$, $X \leq_p Y$.

Proposition. Suppose $Y \in \mathbf{NP}$ -complete. Then, $Y \in \mathbf{P}$ iff $\mathbf{P} = \mathbf{NP}$.

Pf. \Leftarrow If $\mathbf{P} = \mathbf{NP}$, then $Y \in \mathbf{P}$ because $Y \in \mathbf{NP}$.

Pf. \Rightarrow Suppose $Y \in \mathbf{P}$.

- Consider any problem $X \in \mathbf{NP}$. Since $X \leq_p Y$, we have $X \in \mathbf{P}$.
- This implies $\mathbf{NP} \subseteq \mathbf{P}$.
- We already know $\mathbf{P} \subseteq \mathbf{NP}$. Thus $\mathbf{P} = \mathbf{NP}$. ■

Fundamental question. Are there any “natural” **NP**-complete problems?

The "first" NP-complete problem

Theorem. [Cook 1971, Levin 1973] $SAT \in NP$ -complete.

The Complexity of Theorem-Proving Procedures

Stephen A. Cook

University of Toronto

Summary

It is shown that any recognition problem solved by a polynomial time-bounded nondeterministic Turing machine can be "reduced" to the problem of determining whether a given propositional formula is a tautology. Here "reduced" means, roughly speaking, that the first problem can be solved deterministically in polynomial time provided an oracle is available for solving the second. From this notion of reducible, polynomial degrees of difficulty are defined, and it is shown that the problem of determining tautologyhood has the same polynomial degree as the problem of determining whether the first of two given graphs is isomorphic to a subgraph of the second. Other examples are discussed. A method of measuring the complexity of proof procedures for the predicate calculus is introduced and discussed.

Throughout this paper, a set of strings means a set of strings on some fixed, large, finite alphabet Σ . This alphabet is large enough to include symbols for all sets described here. All Turing machines are deterministic recognition devices, unless the contrary is explicitly stated.

1. Tautologies and Polynomial Reducibility.

Let us fix a formalism for the propositional calculus in which formulas are written as strings on Σ . Since we will require infinitely many proposition symbols (atoms), each such symbol will consist of a member of Σ followed by a number in binary notation to distinguish that symbol. Thus a formula of length n can only have about $n/\log n$ distinct function and predicate symbols. The logical connectives are $\&$ (and), \vee (or), and \neg (not).

The set of tautologies (denoted by {tautologies}) is a

certain recursive set of strings on this alphabet, and we are interested in the problem of finding a good lower bound on its possible recognition times. We provide no such lower bound here, but theorem 1 will give evidence that {tautologies} is a difficult set to recognize, since many apparently difficult problems can be reduced to determining tautologyhood. By reduced we mean, roughly speaking, that if tautologyhood could be decided instantly (by an "oracle") then these problems could be decided in polynomial time. In order to make this notion precise, we introduce query machines, which are like Turing machines with oracles in [1].

A query machine is a multitape Turing machine with a distinguished tape called the query tape, and three distinguished states called the query state, yes state, and no state, respectively. If M is a query machine and T is a set of strings, then a T-computation of M is a computation of M in which initially M is in the initial state and has an input string w on its input tape, and each time M assumes the query state there is a string u on the query tape, and the next state M assumes is the yes state if $u \in T$ and the no state if $u \notin T$. We think of an "oracle", which knows T , placing M in the yes state or no state.

Definition

A set S of strings is P-reducible (P for polynomial) to a set T of strings iff there is some query machine M and a polynomial $Q(n)$ such that for each input string w , the T-computation of M with input w halts within $Q(|w|)$ steps ($|w|$ is the length of w), and ends in an accepting state iff $w \in S$.

It is not hard to see that P-reducibility is a transitive relation. Thus the relation E on

ПРОБЛЕМЫ ПЕРЕДАЧИ ИНФОРМАЦИИ

Том IX

1973

Вып. 3

КРАТКИЕ СООБЩЕНИЯ

УДК 519.14

УНИВЕРСАЛЬНЫЕ ЗАДАЧИ ПЕРЕБОРА

Л. А. Левин

В статье рассматривается несколько известных массовых задач «переборного типа» и доказывается, что эти задачи можно решать лишь за такое время, за которое можно решать вообще любые задачи указанного типа.

После уточнения понятия алгоритма была доказана алгоритмическая неразрешимость ряда классических массовых проблем (например, проблем тождества элементов групп, гомеоморфности многообразий, разрешимости диофантовых уравнений и других). Тем самым был снят вопрос о нахождении практического способа их решения. Однако существование алгоритмов для решения других задач не снимает для них аналогичного вопроса из-за фантастически большого объема работы, предсказываемого этими алгоритмами. Такова ситуация с так называемыми переборными задачами: минимизации булевых функций, поиска доказательств ограниченной длины, выяснения изоморфности графов и другими. Все эти задачи решаются тривиальными алгоритмами, состоящими в переборе всех возможностей. Однако эти алгоритмы требуют экспоненциального времени работы и у математиков сложилось убеждение, что более простые алгоритмы для них невозможны. Был получен ряд серьезных аргументов в пользу его справедливости (см. [1, 2]), однако доказать это утверждение не удалось никому. (Например, до сих пор не доказано, что для нахождения математических доказательств нужно больше времени, чем для их проверки.)

Однако если предположить, что вообще существует какая-нибудь (хотя бы искусственно построенная) массовая задача переборного типа, неразрешимая простыми (в смысле объема вычислений) алгоритмами, то можно показать, что этим же свойством обладают и многие «классические» переборные задачи (в том числе задача минимизации, задача поиска доказательств и др.). В этом и состоят основные результаты статьи.

Функции $f(n)$ и $g(n)$ будем называть сравнимыми, если при некотором k

$$f(n) \leq (g(n) + 2)^k \text{ и } g(n) \leq (f(n) + 2)^k.$$

Аналогично будем понимать термин «меньше или сравнимо».

О п р е д е л е н и е. Задачей переборного типа (или просто переборной задачей) будем называть задачу вида «по данному x найти какое-нибудь y длины, сравнимой с длиной x , такое, что выполняется $A(x, y)$ », где $A(x, y)$ — какое-нибудь свойство, проверяемое алгоритмом, время работы которого сравнимо с длиной x . (Под алгоритмом здесь можно понимать, например, алгоритмы Колмогорова — Успенского или машины Тьюринга, или нормальные алгоритмы; x, y — двоичные слова). Квазипереборной задачей будем называть задачу выяснения, существует ли такое y .

Мы рассмотрим шесть задач этих типов. Рассматриваемые в них объекты кодируются естественным образом в виде двоичных слов. При этом выбор естественной кодировки не существен, так как все они дают сравнимые длины кодов.

Задача 1. Заданы список конечное множество и покрытие его 500-элементными подмножествами. Найти подпокрытие заданной мощности (соответственно выяснить существует ли оно).

Задача 2. Таблично задана частичная булева функция. Найти заданного размера дизъюнктивную нормальную форму, реализующую эту функцию в области определения (соответственно выяснить существует ли она).

Задача 3. Выяснить, выводима или опровержима данная формула исчисления высказываний. (Или, что то же самое, равна ли константе данная булева формула.)

Задача 4. Даны два графа. Найти гомоморфизм одного на другой (выяснить его существование).

Задача 5. Даны два графа. Найти изоморфизм одного в другой (на его часть).

Задача 6. Рассматриваются матрицы из целых чисел от 1 до 100 и некоторое условие о том, какие числа в них могут соседствовать по вертикали и какие по горизонтали. Заданы числа на границе и требуется продолжить их на всю матрицу с соблюдением условия.

Establishing NP-completeness

Remark. Once we establish first “natural” **NP**-complete problem, others fall like dominoes.

Recipe. To prove that $Y \in \mathbf{NP}$ -complete:

- Step 1. Show that $Y \in \mathbf{NP}$.
- Step 2. Choose an **NP**-complete problem X .
- Step 3. Prove that $X \leq_p Y$.

Proposition. If $X \in \mathbf{NP}$ -complete, $Y \in \mathbf{NP}$, and $X \leq_p Y$, then $Y \in \mathbf{NP}$ -complete.

Pf. Consider any problem $W \in \mathbf{NP}$. Then, both $W \leq_p X$ and $X \leq_p Y$.

- By transitivity, $W \leq_p Y$.
- Hence $Y \in \mathbf{NP}$ -complete. ■

↑
by definition of
NP-complete

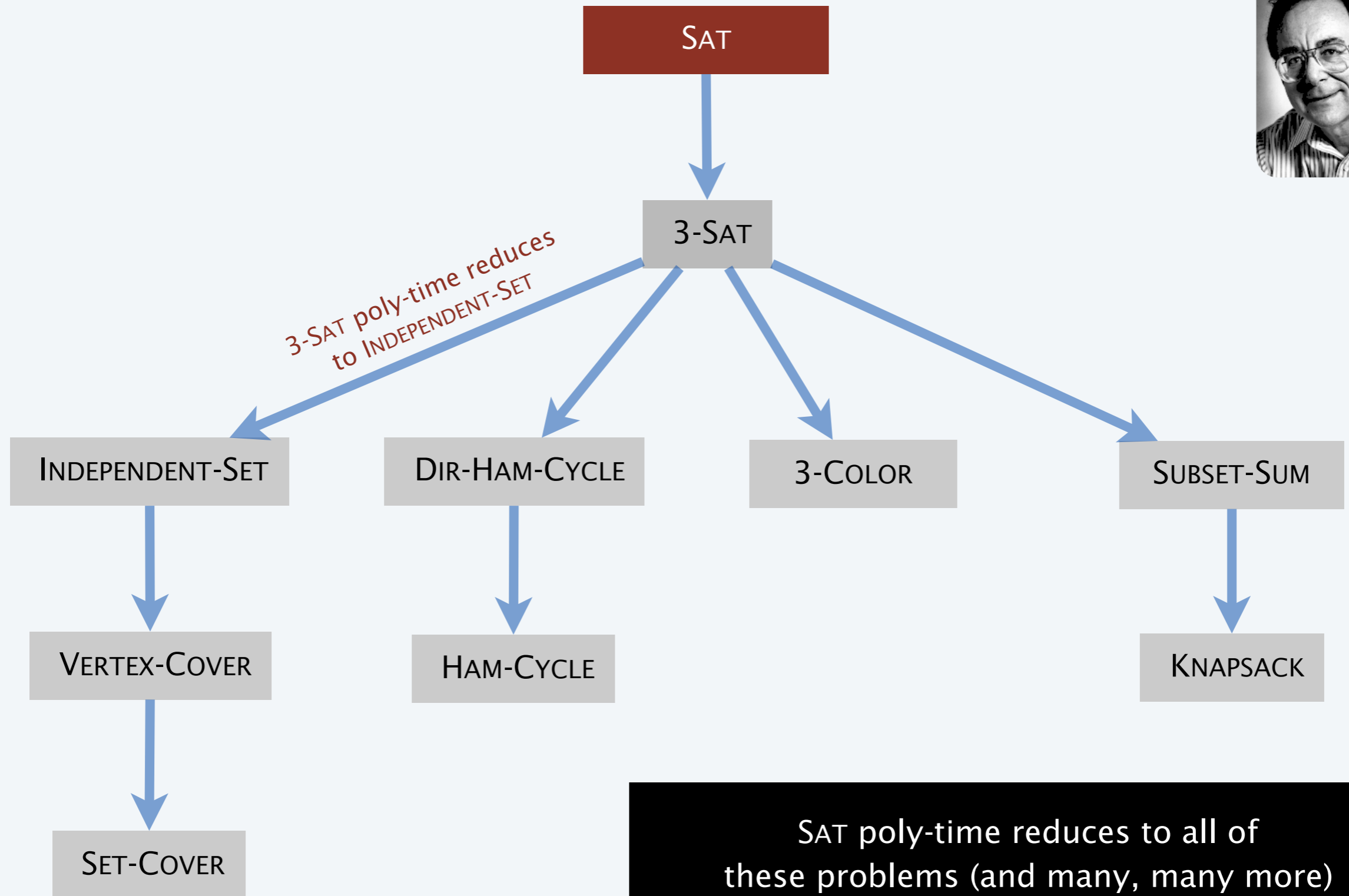
↑
by assumption



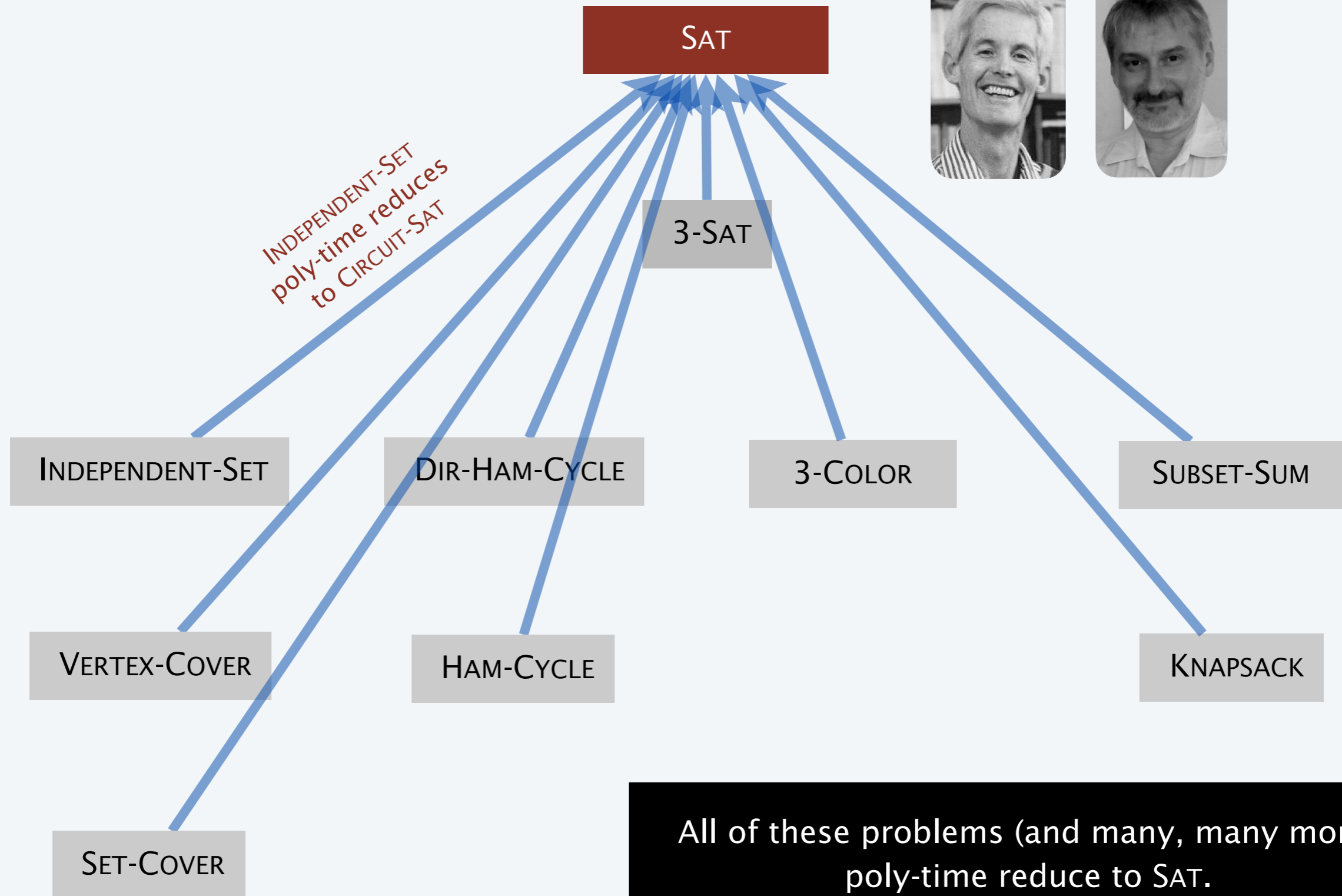
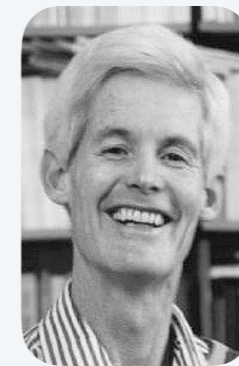
Suppose that $X \in \text{NP-COMplete}$, $Y \in \text{NP}$, and $X \leq_P Y$. Which can you infer?

- A. Y is NP-complete.
- B. If $Y \notin \text{P}$, then $\text{P} \neq \text{NP}$.
- C. If $\text{P} \neq \text{NP}$, then neither X nor Y is in P .
- D. All of the above.

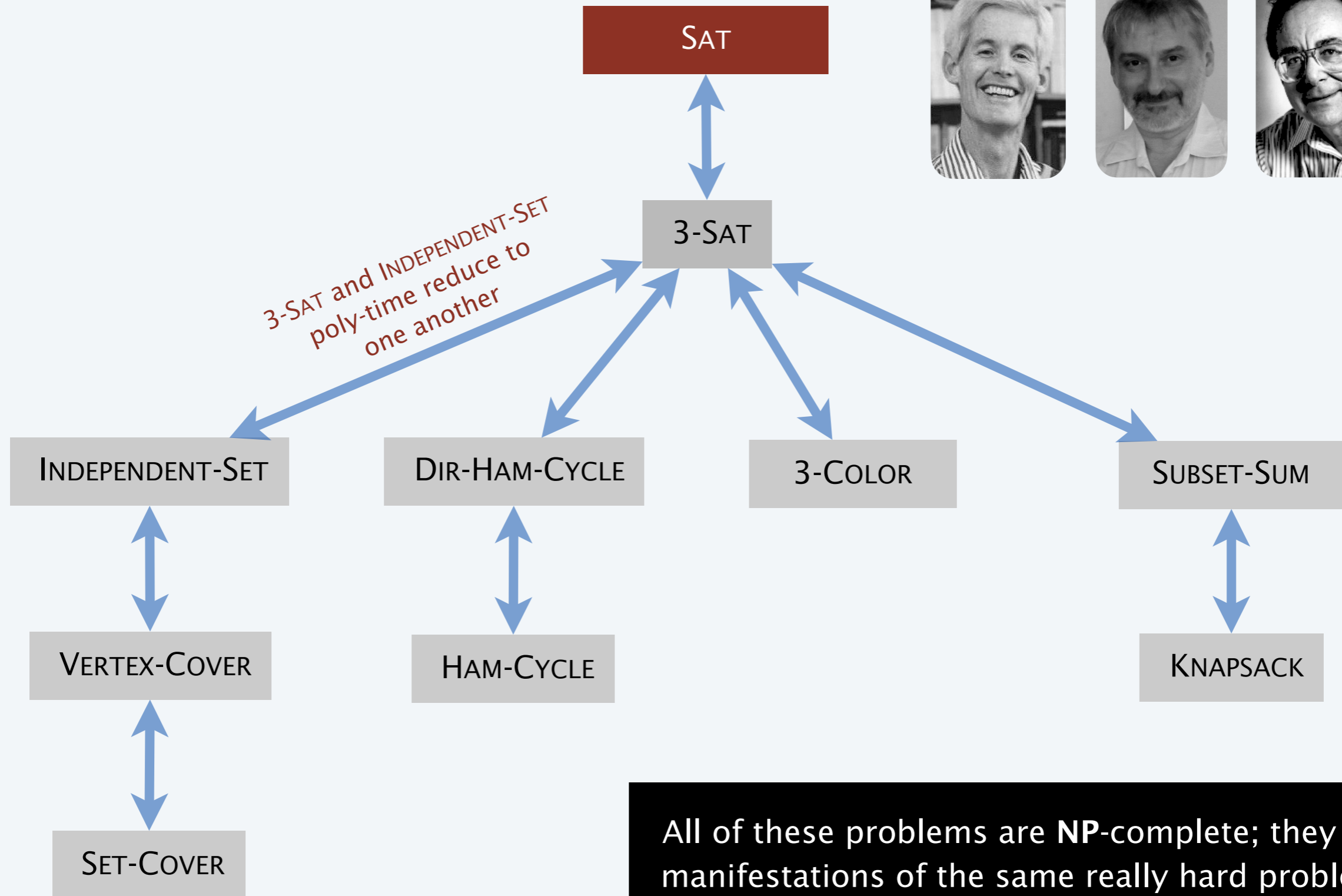
Implications of Karp



Implications of Cook-Levin



Implications of Karp + Cook-Levin



**I'D TELL YOU ANOTHER
NP-COMPLETE JOKE,
BUT ONCE YOU'VE HEARD ONE,**

YOU'VE HEARD THEM ALL.

Some NP-complete problems

Basic genres of NP-complete problems and paradigmatic examples.

- Packing/covering problems: SET-COVER, VERTEX-COVER, INDEPENDENT-SET.
- Constraint satisfaction problems: CIRCUIT-SAT, SAT, 3-SAT.
- Sequencing problems: HAMILTON-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING, 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.

Practice. Most **NP** problems are known to be in either **P** or **NP**-complete.

NP-intermediate? FACTOR, DISCRETE-LOG, GRAPH-ISOMORPHISM,

Theorem. [Ladner 1975] Unless **P** = **NP**, there exist problems in **NP** that are in neither **P** nor **NP**-complete.

On the Structure of Polynomial Time Reducibility

RICHARD E. LADNER

University of Washington, Seattle, Washington

More hard computational problems

Garey and Johnson. Computers and Intractability.

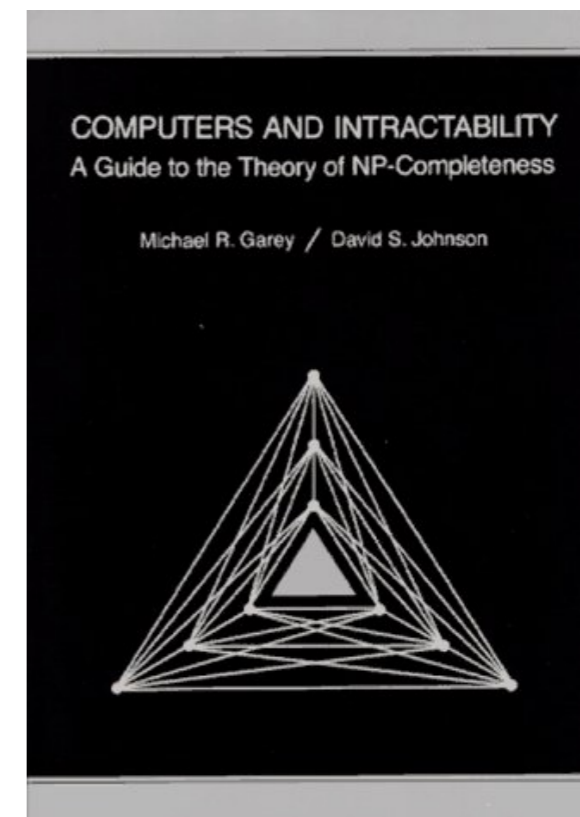
- Appendix includes over 300 **NP**-complete problems.
- Most cited reference in computer science literature.

Most Cited Computer Science Citations

This list is generated from documents in the CiteSeer^x database as of January 17, 2013. This list is automatically generated and may contain errors. The list is generated in batch mode and citation counts may differ from those currently in the CiteSeer^x database, since the database is continuously updated.

[All Years](#) | [1990](#) | [1991](#) | [1992](#) | [1993](#) | [1994](#) | [1995](#) | [1996](#) | [1997](#) | [1998](#) | [1999](#) | [2000](#) | [2001](#) | [2002](#) | [2003](#) | [2004](#) | [2005](#) | [2006](#) | [2007](#) | [2008](#) | [2009](#) | [2010](#) | [2011](#) | [2012](#) | [2013](#)

1. M R Garey, D S Johnson
[Computers and Intractability. A Guide to the Theory of NP-Completeness](#) 1979
8665
2. T Cormen, C E Leiserson, R Rivest
[Introduction to Algorithms](#) 1990
7210
3. V N Vapnik
[The nature of statistical learning theory](#) 1998
6580
4. A P Dempster, N M Laird, D B Rubin
[Maximum likelihood from incomplete data via the EM algorithm.](#) Journal of the Royal Statistical Society, 1977
6082
5. T Cover, J Thomas
[Elements of Information Theory](#) 1991
6075
6. D E Goldberg
[Genetic Algorithms](#) in Search, Optimization, and Machine Learning, 1989
5998
7. J Pearl
[Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference](#) 1988
5582
8. E Gamma, R Helm, R Johnson, J Vlissides
[Design Patterns: Elements of Reusable Object-Oriented Software](#) 1995
4614
9. C E Shannon
[A mathematical theory of communication](#) Bell Syst. Tech. J, 1948
4118
10. J R Quinlan
[C4.5: Programs for Machine Learning](#) 1993
4018



More hard computational problems

Aerospace engineering. Optimal mesh partitioning for finite elements.

Biology. Phylogeny reconstruction.

Chemical engineering. Heat exchanger network synthesis.

Chemistry. Protein folding.

Civil engineering. Equilibrium of urban traffic flow.

Economics. Computation of arbitrage in financial markets with friction.

Electrical engineering. VLSI layout.

Environmental engineering. Optimal placement of contaminant sensors.

Financial engineering. Minimum risk portfolio of given return.

Game theory. Nash equilibrium that maximizes social welfare.

Mathematics. Given integer a_1, \dots, a_n , compute $\int_0^{2\pi} \cos(a_1\theta) \times \cos(a_2\theta) \times \dots \times \cos(a_n\theta) d\theta$

Mechanical engineering. Structure of turbulence in sheared flows.

Medicine. Reconstructing 3d shape from biplane angiocardiogram.

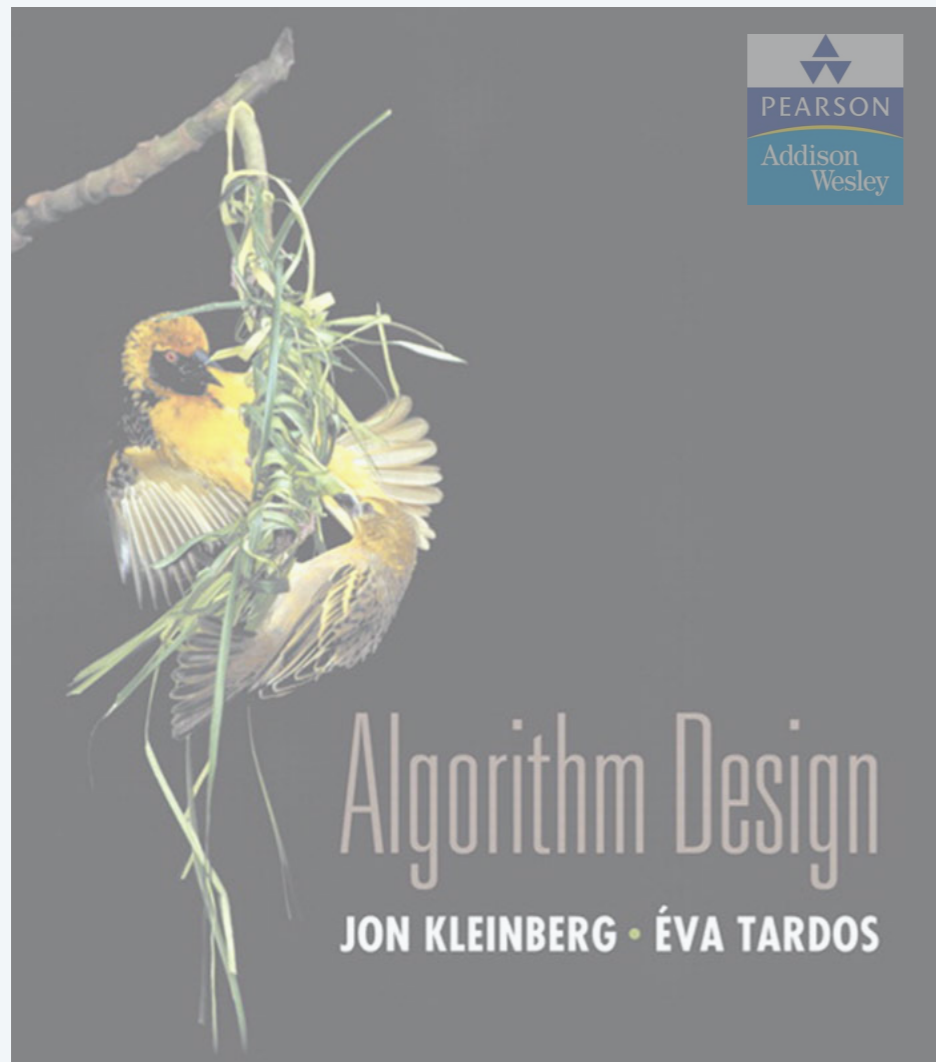
Operations research. Traveling salesperson problem.

Physics. Partition function of 3d Ising model.

Politics. Shapley–Shubik voting power.

Recreation. Versions of Sudoku, Checkers, Minesweeper, Tetris, Rubik's Cube.

Statistics. Optimal experimental design.



SECTION 8.9

8. INTRACTABILITY II

- ▶ *P vs. NP*
- ▶ *NP-complete*
- ▶ ***co-NP***
- ▶ *NP-hard*

Asymmetry of NP

Asymmetry of NP. We need short certificates only for *yes* instances.

Ex 1. SAT vs. UN-SAT.

- Can prove a CNF formula is satisfiable by specifying an assignment.
- How could we prove that a formula is not satisfiable?

SAT. Given a CNF formula Φ , is there a satisfying truth assignment?

UN-SAT. Given a CNF formula Φ , is there no satisfying truth assignment?

Asymmetry of NP

Asymmetry of NP. We need short certificates only for *yes* instances.

Ex 2. HAMILTON-CYCLE vs. NO-HAMILTON-CYCLE.

- Can prove a graph is Hamiltonian by specifying a permutation.
- How could we prove that a graph is not Hamiltonian?

HAMILTON-CYCLE. Given a graph $G = (V, E)$, is there a simple cycle Γ that contains every node in V ?

NO-HAMILTON-CYCLE. Given a graph $G = (V, E)$, is there no simple cycle Γ that contains every node in V ?

Asymmetry of NP

Asymmetry of NP. We need short certificates only for *yes* instances.

Q. How to classify UN-SAT and NO-HAMILTON-CYCLE ?

- SAT \in **NP**-complete and SAT \equiv_P UN-SAT.
- HAMILTON-CYCLE \in **NP**-complete and HAMILTON-CYCLE \equiv_P NO-HAMILTON-CYCLE.
- But neither UN-SAT nor NO-HAMILTON-CYCLE are known to be in **NP**.

NP and co-NP

NP. Decision problems for which there is a poly-time certifier.

Ex. SAT, HAMILTON-CYCLE, and COMPOSITES.

Def. Given a decision problem X , its **complement** \bar{X} is the same problem with the *yes* and *no* answers reversed.

Ex. $X = \{ 4, 6, 8, 9, 10, 12, 14, 15, \dots \}$

$\bar{X} = \{ 2, 3, 5, 7, 11, 13, 17, 23, 29, \dots \}$

← ignore 0 and 1
(neither prime nor composite)

co-NP. Complements of decision problems in **NP**.

Ex. UN-SAT, NO-HAMILTON-CYCLE, and PRIMES.

NP = co-NP ?

Fundamental open question. Does **NP = co-NP**?

- Do *yes* instances have succinct certificates iff *no* instances do?
- Consensus opinion: no.

Theorem. If **NP \neq co-NP**, then **P \neq NP**.

Pf idea.

- **P** is closed under complementation.
- If **P = NP**, then **NP** is closed under complementation.
- In other words, **NP = co-NP**.
- This is the contrapositive of the theorem.

Good characterizations

Good characterization. [Edmonds 1965] **NP** \cap **co-NP**.

- If problem X is in both **NP** and **co-NP**, then:
 - for *yes* instance, there is a succinct certificate
 - for *no* instance, there is a succinct disqualifier
- Provides conceptual leverage for reasoning about a problem.

Ex. Given a bipartite graph, is there a perfect matching?

- If yes, can exhibit a perfect matching.
- If no, can exhibit a set of nodes S such that $|neighbors(S)| < |S|$.

JOURNAL OF RESEARCH of the National Bureau of Standards—B. Mathematics and Mathematical Physics
Vol. 69B, Nos. 1 and 2, January–June 1965

Minimum Partition of a Matroid Into Independent Subsets¹

Jack Edmonds

(December 1, 1964)

A matroid M is a finite set M of elements with a family of subsets, called independent, such that (1) every subset of an independent set is independent, and (2) for every subset A of M , all maximal independent subsets of A have the same cardinality, called the rank $r(A)$ of A . It is proved that a matroid can be partitioned into as few as k sets, each independent, if and only if every subset A has cardinality at most $k \cdot r(A)$.

Good characterizations

Observation. $\mathbf{P} \subseteq \mathbf{NP} \cap \mathbf{co-NP}$.

- Proof of max-flow min-cut theorem led to stronger result that max-flow and min-cut are in \mathbf{P} .
- Sometimes finding a good characterization seems easier than finding an efficient algorithm.

Fundamental open question. Does $\mathbf{P} = \mathbf{NP} \cap \mathbf{co-NP}$?

- Mixed opinions.
- Many examples where problem found to have a nontrivial good characterization, but only years later discovered to be in \mathbf{P} .

Linear programming is in $\mathbf{NP} \cap \mathbf{co-NP}$

LINEAR-PROGRAMMING. Given $A \in \mathfrak{R}^{m \times n}$, $b \in \mathfrak{R}^m$, $c \in \mathfrak{R}^n$, and $\alpha \in \mathfrak{R}$, does there exist $x \in \mathfrak{R}^n$ such that $Ax \leq b$, $x \geq 0$ and $c^T x \geq \alpha$?

Theorem. [Gale–Kuhn–Tucker 1948] $\mathbf{LINEAR-PROGRAMMING} \in \mathbf{NP} \cap \mathbf{co-NP}$.

Pf sketch. If (P) and (D) are nonempty, then $\max = \min$.

$$\begin{array}{ll} \text{(P)} & \max c^T x \\ & \text{s. t. } Ax \leq b \\ & \quad x \geq 0 \end{array} \qquad \begin{array}{ll} \text{(D)} & \min y^T b \\ & \text{s. t. } A^T y \geq c \\ & \quad y \geq 0 \end{array}$$

CHAPTER XIX

LINEAR PROGRAMMING AND THE THEORY OF GAMES¹

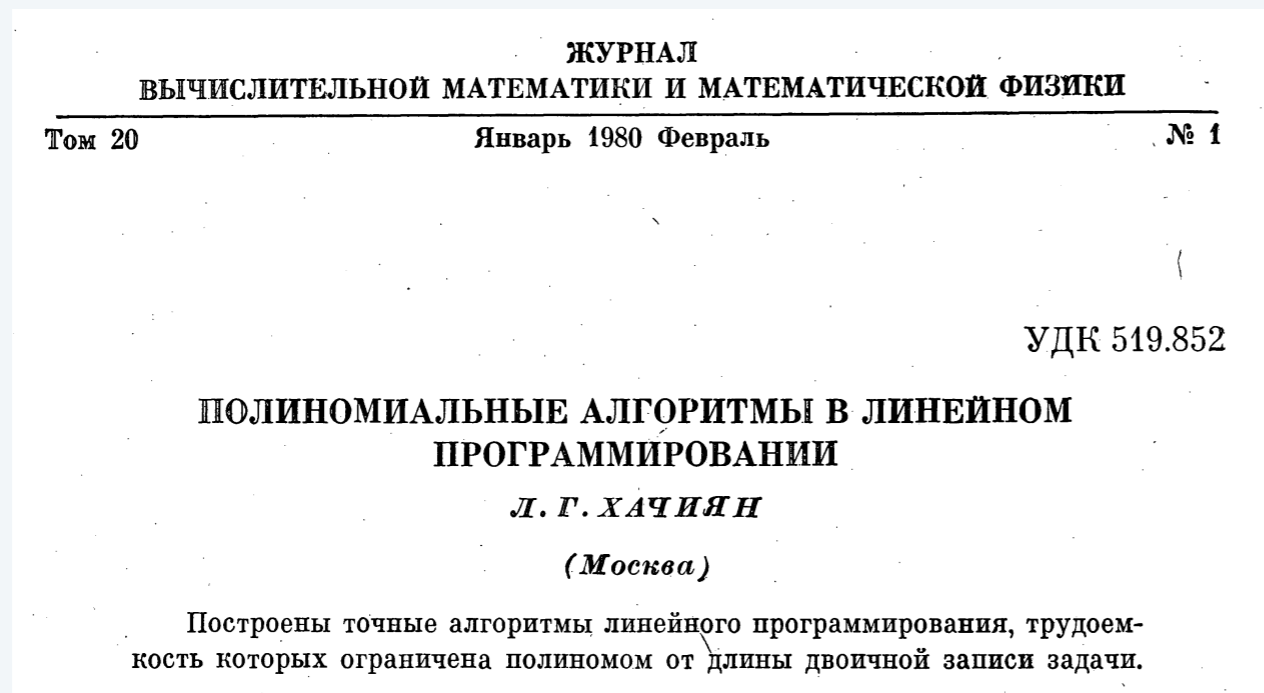
BY DAVID GALE, HAROLD W. KUHN, AND ALBERT W. TUCKER²

The basic “scalar” problem of *linear programming* is to maximize (or minimize) a linear function of several variables constrained by a system of linear inequalities [Dantzig, II]. A more general “vector” problem calls for maximizing (in a sense of partial order) a system of linear functions of several variables subject to a system of linear inequalities and, perhaps, linear equations [Koopmans, III]. The purpose of this chapter is to establish theorems of duality and existence for general “matrix” problems of linear programming which contain the “scalar” and “vector” problems as special cases, and to relate these general problems to the theory of zero-sum two-person games.

Linear programming is in $\text{NP} \cap \text{co-NP}$

LINEAR-PROGRAMMING. Given $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}$, does there exist $x \in \mathbb{R}^n$ such that $Ax \leq b$, $x \geq 0$ and $c^T x \geq \alpha$?

Theorem. [Khachiyan 1979] $\text{LINEAR-PROGRAMMING} \in \mathbf{P}$.



Primality testing is in $\text{NP} \cap \text{co-NP}$

Theorem. [Pratt 1975] $\text{PRIMES} \in \text{NP} \cap \text{co-NP}$.

SIAM J. COMPUT.
Vol. 4, No. 3, September 1975

EVERY PRIME HAS A SUCCINCT CERTIFICATE*

VAUGHAN R. PRATT†

Abstract. To prove that a number n is composite, it suffices to exhibit the working for the multiplication of a pair of factors. This working, represented as a string, is of length bounded by a polynomial in $\log_2 n$. We show that the same property holds for the primes. It is noteworthy that almost no other set is known to have the property that short proofs for membership or nonmembership exist for all candidates without being known to have the property that such proofs are easy to come by. It remains an open problem whether a prime n can be recognized in only $\log_2^\alpha n$ operations of a Turing machine for any fixed α .

The proof system used for certifying primes is as follows.

AXIOM. $(x, y, 1)$.

INFERENCE RULES.

$R_1: (p, x, a), q \vdash (p, x, qa)$ provided $x^{(p-1)/q} \not\equiv 1 \pmod{p}$ and $q|(p-1)$.

$R_2: (p, x, p-1) \vdash p$ provided $x^{p-1} \equiv 1 \pmod{p}$.

THEOREM 1. p is a theorem $\equiv p$ is a prime.

THEOREM 2. p is a theorem $\supset p$ has a proof of $\lceil 4 \log_2 p \rceil$ lines.

Primality testing is in $\mathbf{NP} \cap \mathbf{co-NP}$

Theorem. [Pratt 1975] $\mathbf{PRIMES} \in \mathbf{NP} \cap \mathbf{co-NP}$.

Pf sketch. An odd integer s is prime iff there exists an integer $1 < t < s$ s.t.

$$t^{s-1} \equiv 1 \pmod{s}$$

$$t^{(s-1)/p} \not\equiv 1 \pmod{s}$$

for all prime divisors p of $s-1$

instance s 437677

certificate t 17, $2^2 \times 3 \times 36473$



prime factorization of $s-1$
also need a recursive certificate
to assert that 3 and 36,473 are prime

CERTIFIER (s)

CHECK $s - 1 = 2 \times 2 \times 3 \times 36473$.

CHECK $17^{s-1} = 1 \pmod{s}$.

CHECK $17^{(s-1)/2} \equiv 437676 \pmod{s}$.

CHECK $17^{(s-1)/3} \equiv 329415 \pmod{s}$.

CHECK $17^{(s-1)/36,473} \equiv 305452 \pmod{s}$.



use repeated squaring

Primality testing is in P

Theorem. [Agrawal–Kayal–Saxena 2004] $\text{PRIMES} \in \mathbf{P}$.

Annals of Mathematics, **160** (2004), 781–793

PRIMES is in P

By MANINDRA AGRAWAL, NEERAJ KAYAL, and NITIN SAXENA*

Abstract

We present an unconditional deterministic polynomial-time algorithm that determines whether an input number is prime or composite.

Factoring is in $\mathbf{NP} \cap \mathbf{co-NP}$

FACTORIZE. Given an integer x , **find** its prime factorization.

FACTOR. Given two integers x and y , does x have a nontrivial factor $< y$?

Theorem. $\mathbf{FACTOR} \equiv_P \mathbf{FACTORIZE}$.

Pf.

- \leq_P trivial.
- \geq_P binary search to find a factor; divide out the factor and repeat. ■

Theorem. $\mathbf{FACTOR} \in \mathbf{NP} \cap \mathbf{co-NP}$.

Pf.

- Certificate: a factor p of x that is less than y .
- Disqualifier: the prime factorization of x (where each prime factor is less than y), along with a Pratt certificate that each factor is prime. ■

Is factoring in P ?

Fundamental question. Is FACTOR $\in P$?

Challenge. Factor this number.

74037563479561712828046796097429573142593188889231289
08493623263897276503402826627689199641962511784399589
43305021275853701189680982867331732731089309005525051
16877063299072396380786710086096962537934650563796359

RSA-704

(\$30,000 prize if you can factor)

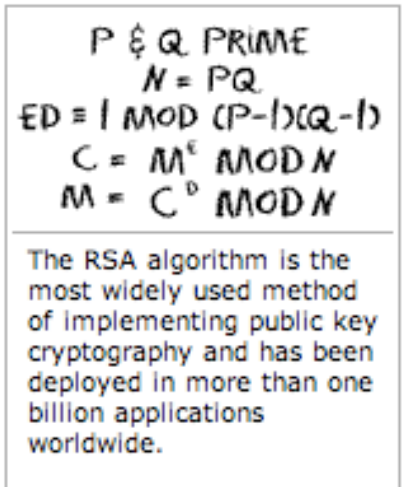
Exploiting intractability

Modern cryptography.

- Ex. Send your credit card to Amazon.
- Ex. Digitally sign an e-document.
- Enables freedom of privacy, speech, press, political association.

RSA. Based on dichotomy between complexity of two problems.

- To use: generate two random n -bit primes and multiply.
- To break: suffices to factor a $2n$ -bit integer.



$P \in Q$ PRIME
 $N = PQ$
 $ED \equiv 1 \pmod{(P-1)(Q-1)}$
 $C = M^E \pmod{N}$
 $M = C^D \pmod{N}$

The RSA algorithm is the most widely used method of implementing public key cryptography and has been deployed in more than one billion applications worldwide.

RSA algorithm



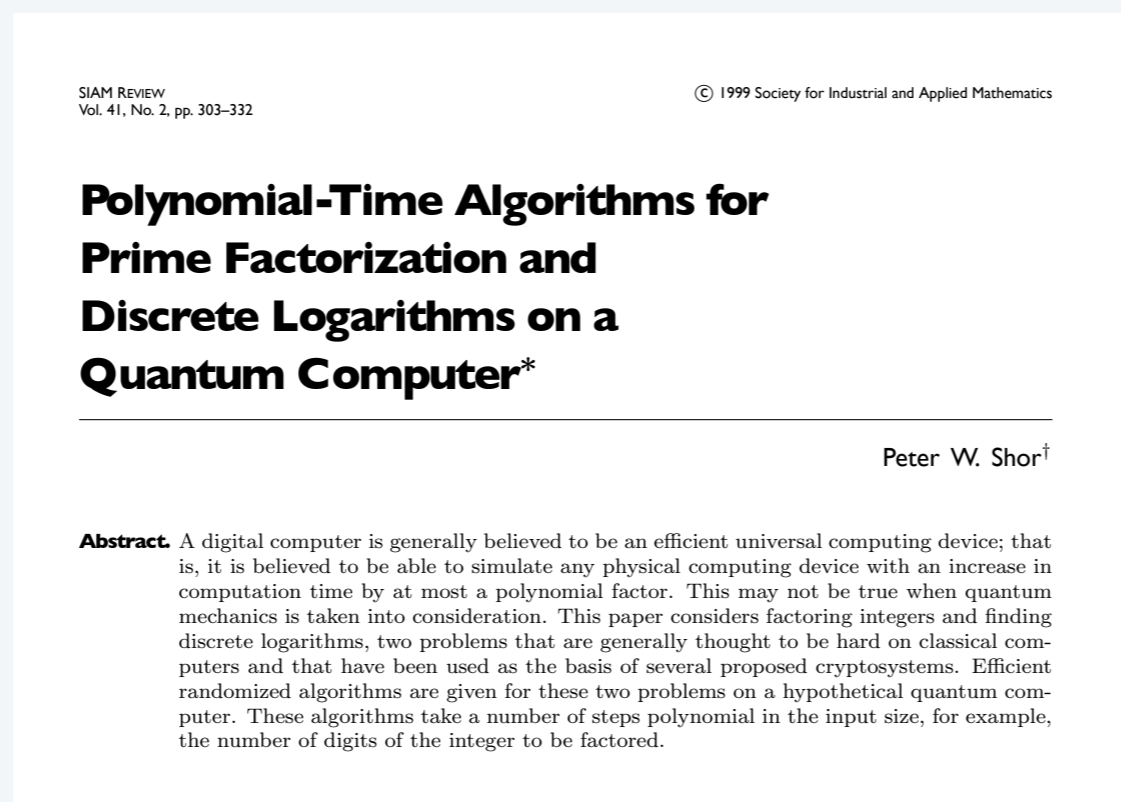
RSA sold
for \$2.1 billion



or design a t-shirt

Factoring on a quantum computer

Theorem. [Shor 1994] Can factor an n -bit integer in $O(n^3)$ steps on a “quantum computer.”



2001. Factored $15 = 3 \times 5$ (with high probability) on a quantum computer.

2012. Factored $21 = 3 \times 7$.

Fundamental question. Does $P = BQP$?

← quantum analog of P
(bounded error quantum polynomial time)

Quantum Factorizations: D-Wave Quantum Computer

A long way to go: Universal quantum computers

- John Martinis & Matthias Troyer: it would be years before achieving some practical applications, including the code-cracking^{1,2}.
- Factor n bit integers require about 2n qubits³.
- Jan. 8 2019, IBM released IBM Q System One™ with 20 qubits that can factor up to 10-bit integers in theory.



PUBLIC RELEASE: 3-APR-2019

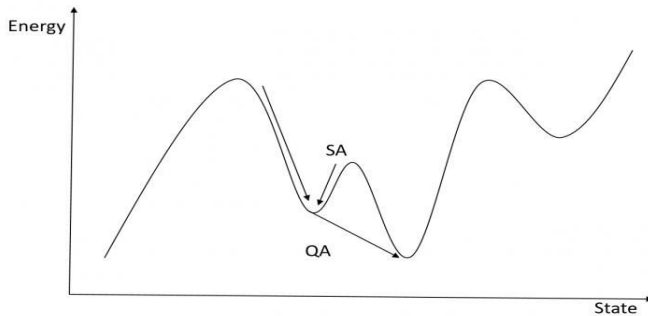
A new hope of quantum computers for factorizations of RSA with a thousand-fold excess

$$p = (1p_{k_1-1}p_{k_1-2} \cdots p_1 1)_2$$

$$q = (1q_{k_2-1}q_{k_2-2} \cdots q_1 1)_2$$

$$n = p \times q \rightarrow \min\{(n - pq)^2\}$$

A new way for factorization⁴: Quantum Annealing in D-Wave



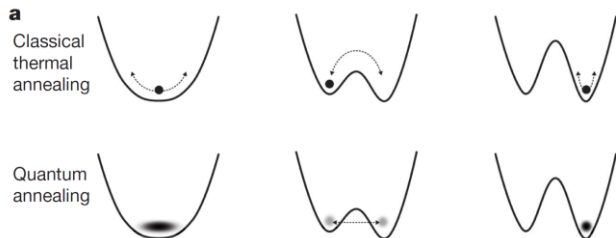
- Near absolute zero (15 mK).
- Quantum tunneling effects.
- As thermal fluctuations turn off, quantum fluctuations work.
- Develop over 100 applications.



Quantum Computing Model

- potentials for factorizations:

$$H(t) = \sum_i h_i(t) \sigma_i^z + \sum_{i,j} J_{ij}(t) \sigma_i^z \sigma_j^z$$



- C. Wang et al. shows the potentials of QA and DWave annealer for deciphering the RSA cryptosystem in future.
- DWave 2000Q System can solve the factorizations on **20 bit** integers, a thousand-fold surpassing the IBM version.



8. INTRACTABILITY II

- ▶ *P vs. NP*
- ▶ *NP-complete*
- ▶ *co-NP*
- ▶ *NP-hard*

A note on terminology

SIGACT News

12

January 1974

A TERMINOLOGICAL PROPOSAL

D. F. Knuth

While preparing a book on combinatorial algorithms, I felt a strong need for a new technical term, a word which is essentially a one-sided version of polynomial complete. A great many problems of practical interest have the property that they are at least as difficult to solve in polynomial time as those of the Cook-Karp class NP. I needed an adjective to convey such a degree of difficulty, both formally and informally; and since the range of practical applications is so broad, I felt it would be best to establish such a term as soon as possible.

The goal is to find an adjective x that sounds good in sentences like this:

The covering problem is x .

It is x to decide whether a given graph has a Hamiltonian circuit.

It is unknown whether or not primality testing is an x problem.

Note. The term x does not necessarily imply that a problem is in NP, just that every problem in NP poly-time reduces to x .

A note on terminology: consensus

NP-complete. A problem in **NP** such that every problem in **NP** poly-time reduces to it.

NP-hard. [Bell Labs, Steve Cook, Ron Rivest, Sartaj Sahni]

A problem such that every problem in **NP** poly-time reduces to it.

One final criticism (which applies to all the terms suggested) was stated nicely by Vaughan Pratt: "If the Martians know that $P = NP$ for Turing Machines and they kidnap me, I would lose face calling these problems 'formidable'." Yes; if $P = NP$, there's no need for any term at all. But I'm willing to risk such an embarrassment, and in fact I'm willing to give a prize of one live turkey to the first person who proves that $P = NP$.

PSPACE

P. Decision problems solvable in polynomial **time**.

PSPACE. Decision problems solvable in polynomial **space**.

Observation. $\mathbf{P} \subseteq \mathbf{PSPACE}$.



poly-time algorithm
can consume
only polynomial space

PSPACE

Binary counter. Count from 0 to $2^n - 1$ in binary.

Algorithm. Use n bit odometer.

Claim. 3-SAT \in **PSPACE**.

Pf.

- Enumerate all 2^n possible truth assignments using counter.
- Check each assignment to see if it satisfies all clauses. ■

Theorem. **NP** \subseteq **PSPACE**.

Pf. Consider arbitrary problem $Y \in$ **NP**.

- Since $Y \leq_p$ 3-SAT, there exists algorithm that solves Y in poly-time plus polynomial number of calls to 3-SAT black box.
- Can implement black box in poly-space. ■

Quantified satisfiability

QSAT. Let $\Phi(x_1, \dots, x_n)$ be a boolean CNF formula. Is the following propositional formula true?

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \dots \forall x_{n-1} \exists x_n \Phi(x_1, \dots, x_n)$$

↑
assume n is odd

Intuition. Amy picks truth value for x_1 , then Bob for x_2 , then Amy for x_3 , and so on. Can Amy satisfy Φ no matter what Bob does?

Ex. $(x_1 \vee x_2) \wedge (x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)$

Yes. Amy sets x_1 true; Bob sets x_2 ; Amy sets x_3 to be same as x_2 .

Ex. $(x_1 \vee x_2) \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)$

No. If Amy sets x_1 false; Bob sets x_2 false; Amy loses;

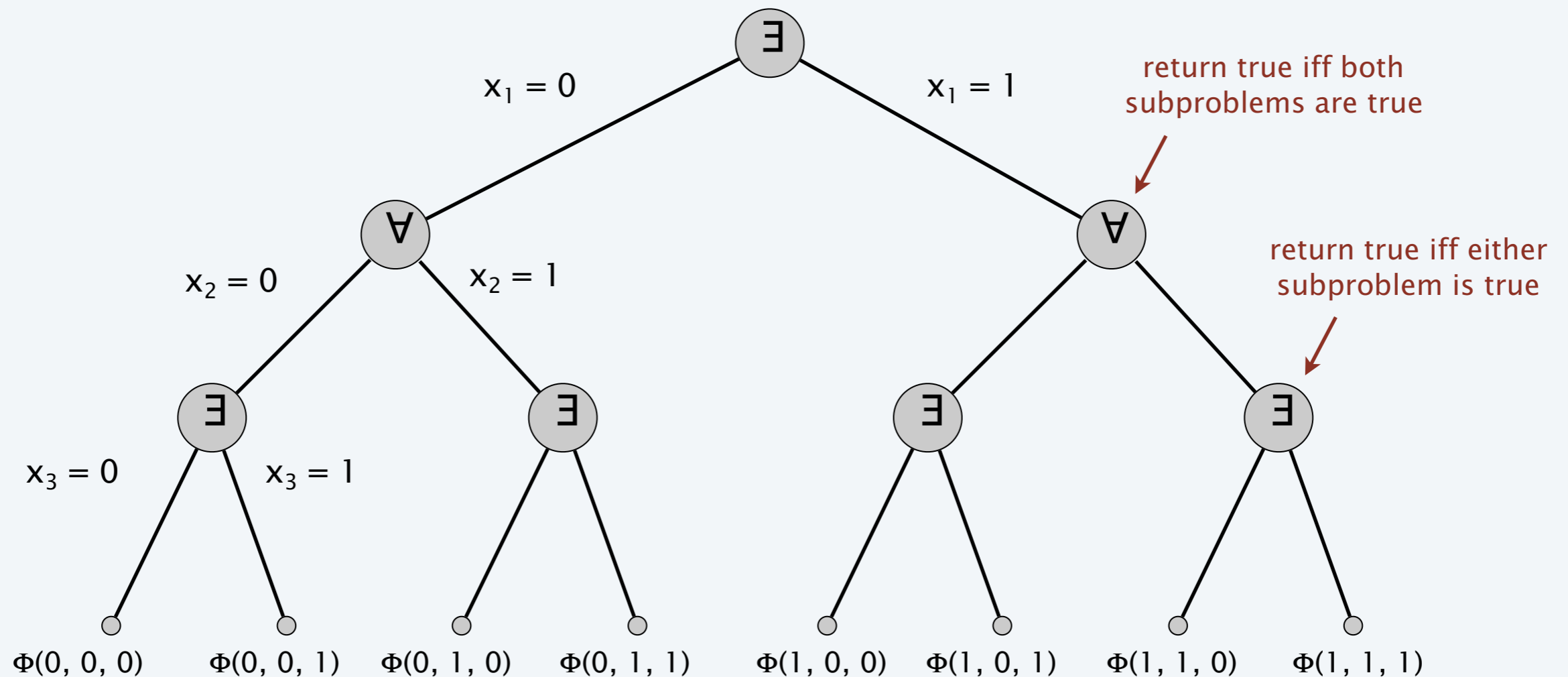
No. if Amy sets x_1 true; Bob sets x_2 true; Amy loses.

Quantified satisfiability is in PSPACE

Theorem. $Q\text{-SAT} \in \text{PSPACE}$.

Pf. Recursively try all possibilities.

- Only need one bit of information from each subproblem.
- Amount of space is proportional to depth of function call stack.



Planning problem: 8-puzzle

Planning example. Can we solve the 8-puzzle?

Conditions. $C_{ij}, 1 \leq i, j \leq 9.$ ← C_{ij} means tile i is in square j

Initial state. $c_0 = \{C_{11}, C_{22}, \dots, C_{66}, C_{78}, C_{87}, C_{99}\}.$

Goal state. $c^* = \{C_{11}, C_{22}, \dots, C_{66}, C_{77}, C_{88}, C_{99}\}.$

Operators.

- Precondition to apply $O_i = \{C_{11}, C_{22}, \dots, C_{66}, C_{78}, C_{87}, C_{99}\}.$
- After invoking O_i , conditions C_{79} and C_{97} become *true*.
- After invoking O_i , conditions C_{78} and C_{99} become *false*.

Solution. No solution to 8-puzzle or 15-puzzle!

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 8 | 7 | 9 |



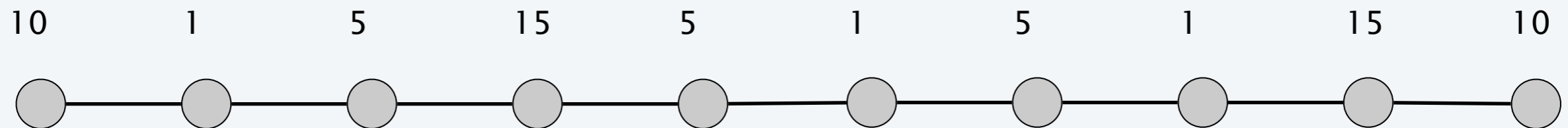
| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 8 | 9 | 7 |

Competitive facility location

Input. Graph $G = (V, E)$ with positive edge weights, and target B .

Game. Two competing players alternate in selecting nodes. Not allowed to select a node if any of its neighbors has been selected.

Competitive facility location. Can second player guarantee at least B units of profit?



yes if $B = 20$;
no if $B = 25$

PSPACE-complete

PSPACE. Decision problems solvable in polynomial space.

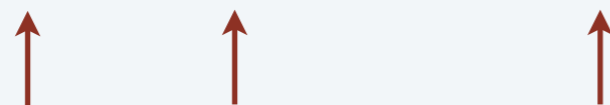
PSPACE-complete. Problem $Y \in \mathbf{PSPACE}$ -complete if (i) $Y \in \mathbf{PSPACE}$ and (ii) for every problem $X \in \mathbf{PSPACE}$, $X \leq_p Y$.

Theorem. [Stockmeyer–Meyer 1973] $\text{QSAT} \in \mathbf{PSPACE}$ -complete.

Theorem. $\mathbf{PSPACE} \subseteq \mathbf{EXPTIME}$.

Pf. Previous algorithm solves QSAT in exponential time; and QSAT is **PSPACE-complete**. ■

Summary. $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXPTIME}$.



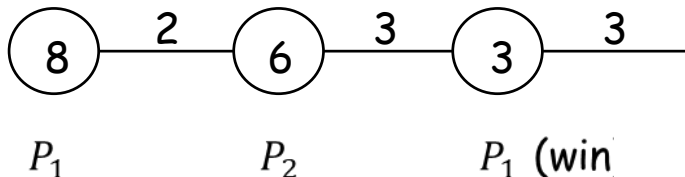
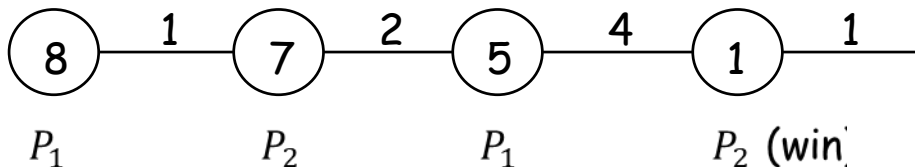
it is known that $\mathbf{P} \neq \mathbf{EXPTIME}$,
but unknown which inclusion is strict;
conjectured that all are

Competitive Facility Allocation: Card Game

Card game

- Two players alternatively pick cards from a pile of n cards.
- The first player can pick $i: 1, 2, \dots, n-1$ cards.
- At each round, a player can select $j: 1, 2, \dots, 2i$ cards, where i is the pick of the other player in the previous round.
- Whoever picks the last card wins the game.

Can the first player always win for a given n cards? If so, how?



Competitive Facility Allocation: Fighting for Lava (April 2019, CACM)

last byte



Dennis Shasha

DOI:10.1145/3314071

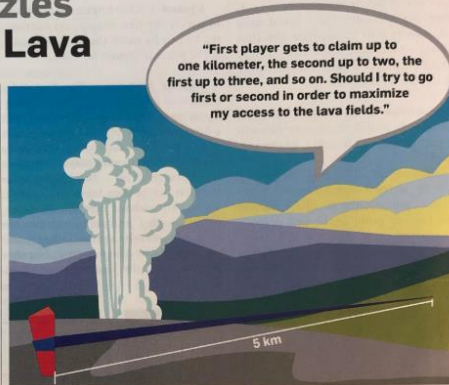
Upstart Puzzles Fighting for Lava

THE VAST UNDERGROUND lava fields in the western U.S. feature a photogenic geyser called Old Faithful. Eruptions send approximately 15,000 liters of steaming water 50 meters into the air approximately every hour. Unfortunately, what is underground is not nearly as appealing. If the lava fields erupted in a major way, they could cause ferocious firestorms that would destroy a large portion of the western U.S. and Canada and substantially cool the planet.

Now imagine a pair of tunnel-boring energy-extraction companies are competing to cool the lava, make some money, and provide carbon-free energy besides. The idea is to tunnel from a power plant outside the lava fields to near the lava, but not too close, to avoid accidental eruptions. A pipeline could in theory then take cool water from the power plant to the end of the tunnel where the water would be heated into steam and the steam would power the turbines of the power plant. The whole system could be designed to recycle the steam back into water in a closed loop.

In this scenario, the federal government, which owns the land, steps in to lease the energy rights, identifying cross-sections underground to which tunnels can be drilled. The government identifies those underground cross-sections based on their more-or-less linear segments aboveground, so leasing a segment would confer the right to tap the lava in the vertical cross-section below that segment.

To encourage participation in the project while achieving equity for both companies, government mathematicians design a game-style



protocol, whereby each company (player) takes turns to acquire non-overlapping sections of a full segment. The first player may take up

Imagine a pair of tunnel-boring energy-extraction companies are competing to cool the lava, make some money, and provide carbon-free energy besides.

to one kilometer in the first turn, the second player then takes up to two kilometers, the first player then gets up to three kilometers, the second then gets up to four kilometers, and so on.

Warm-up. Suppose the line segment is five kilometers long from a stake at kilometer 0 to a stake at kilometer 5. Suppose the first player takes between 0 and 1 kilometer. Which player would get more of the line segment, assuming each plays optimally?

Answer to warm-up. Player 2. Player 1 takes kilometer 0 to 1. Player 2 then takes kilometers 2 to 4. The first player then takes one of the two remaining kilometers—1 to 2 or 4 to 5—and the second player then takes the other remaining kilometer. The second player ends up with lease rights to three of the [CONTINUED ON P. 143]

Another 2-player game on 1-D and 2-D

last byte

[CONTINUED FROM P. 144] five kilometers, thus one more kilometer than the first player.

Question 1. By playing differently, beginning with the first move, could the first player acquire the rights to more of the line segment than the second player?

Solution to question 1. Yes. If the first player takes kilometers 2 to 3, then the second player could take kilometers 0 to 2, but then the first player would take kilometers 3 to 5. The first player would thus get three of the five kilometers.

Question 2. Is there a minimal amount by which one player can win regardless of the length L of the segment?

Answer. Yes. The first player can win by at least one kilometer every time by going in the middle, meaning the halfway point of the first player's kilometer is at position $L/2$. After that, the first player would mirror the second player's moves. So if the second player takes x to $x+2$ to the left

of the middle kilometer, then the first player takes $(x + L/2)$ to $(x + 2 + L/2)$ on the right of the middle. The net effect is the first player can always guarantee to capture at least as much territory as the second player on the two sides of that middle kilometer. The first player wins by at least the kilometer of the first move.

Upstart 1. Characterize situations in which the first player can guarantee to win by more than one kilometer or prove it cannot be done.

Upstart 2. Suppose the line segment is of length L , but there are now k players instead of two. The rules are a direct generalization of the original game; the first player may take one kilometer, the second player two, the third player three, ... the k^{th} player k , the first player then takes $k+1$... and so on, all without overlap. Is there some length L and some number of players k whereby a player other than the first player can guarantee to capture more of the line segment than anyone else?

Upstart 3. Suppose the government leased out vertical cross-sectional squares belowground. Each player would thus take squares, with the side length of each square increasing by one kilometer with each move. The first player takes one kilometer squared. The second player then gets two kilometers squared. The first player then gets three kilometers squared, and so on, again without overlap. Does either player have a winning strategy if the area available to lease could be an arbitrary rectangular cross-section belowground? How would this generalize to more players?

All are invited to submit their solutions to upstartpuzzles@cacm.acm.org; solutions to upstarts and discussion will be posted at <http://cs.nyu.edu/cs/faculty/shasha/papers/cacmpuzzles.html>.

Dennis Shasha (dennisshasha@yahoo.com) is a professor of computer science in the Computer Science Department of the Courant Institute at New York University, New York, USA, as well as the chronicler of his good friend the omniscient Dr. Ecco.

Copyright held by author.