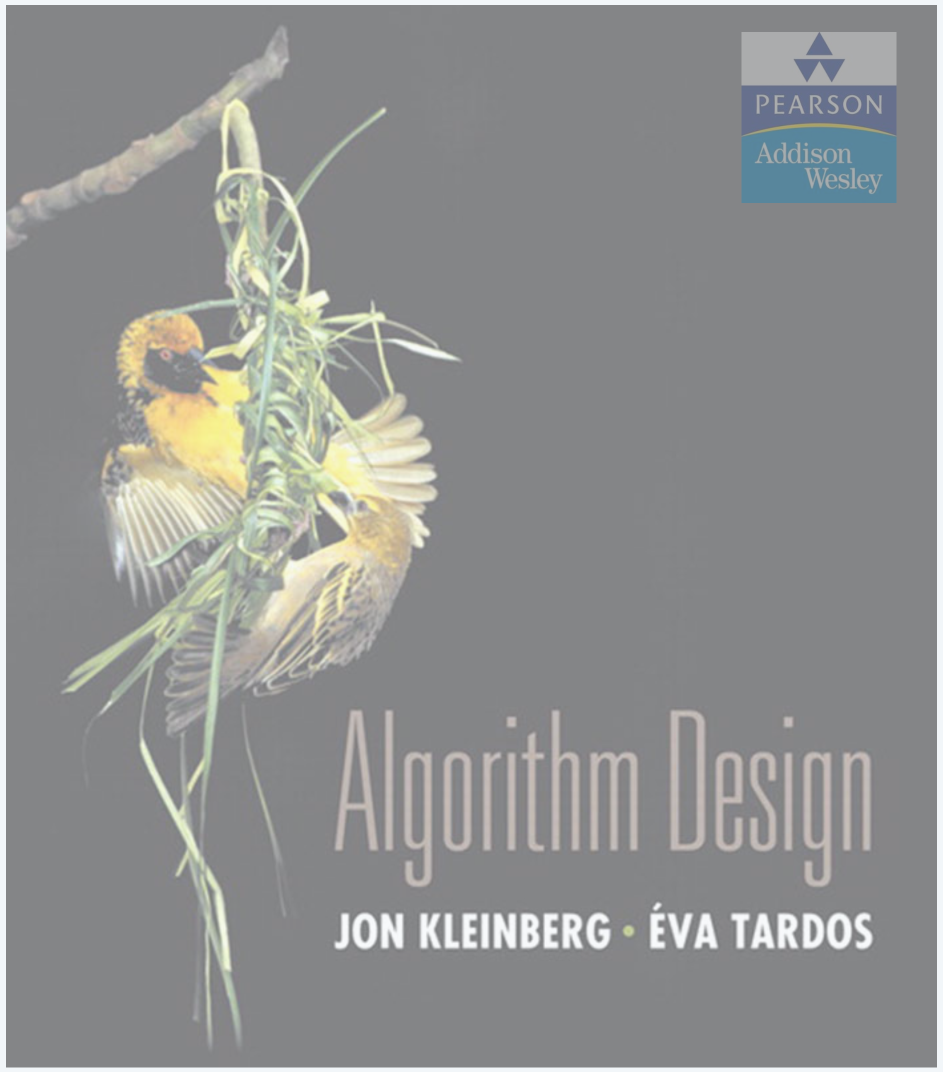


SECTION 8.9

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 ∈ **NP**-complete and SAT $\equiv_P$ UN-SAT. (under $\leq_T^P$ Cook Reductions)
- HAMILTON-CYCLE ∈ **NP**-complete and HAMILTON-CYCLE $\equiv_P$ NO-HAMILTON-CYCLE.
- But neither UN-SAT nor NO-HAMILTON-CYCLE are known to be in **NP**.

Problem X polynomial-time (Cook) reduces to problem Y

if arbitrary instances of problem X can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem Y.

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 $\overline{X}$ is the same problem with the *yes* and *no* answers reversed.

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

$\overline{X} = \{ 2, 3, 5, 7, 11, 13, 17, 23, 29, \ldots \}$ ← 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** ≠ **co-NP**, then **P** ≠ **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.

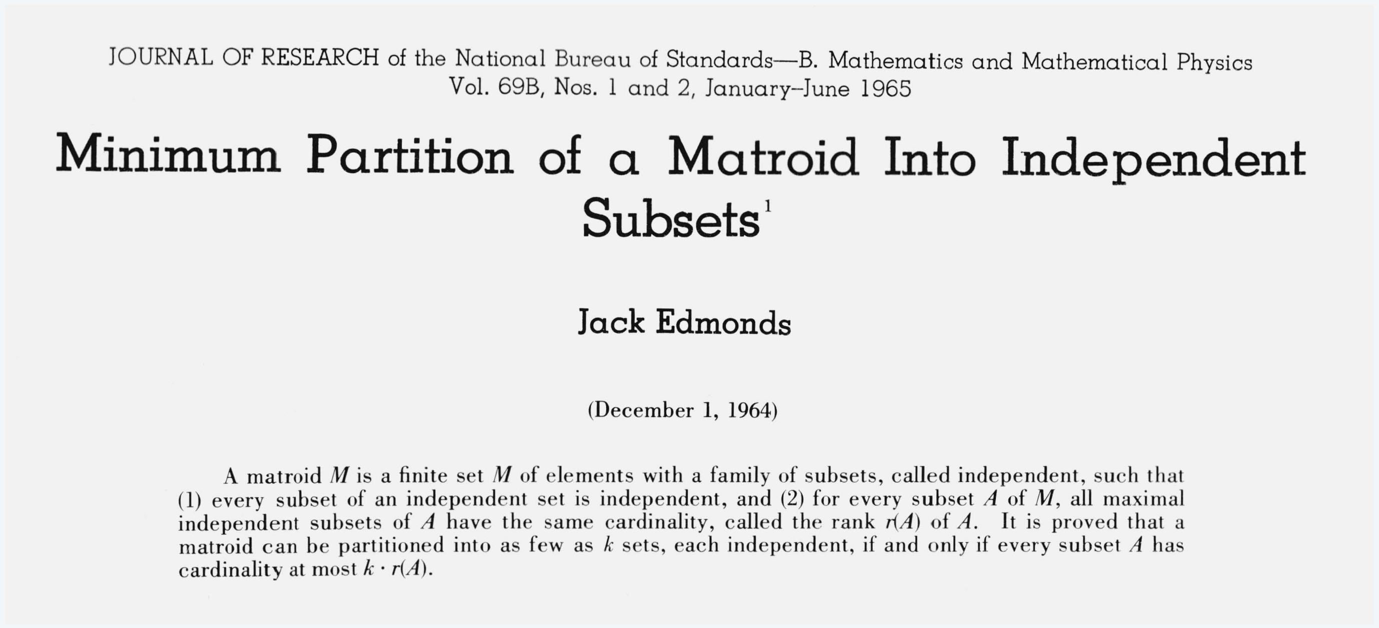
Good characterizations

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 Subsets[1]

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. **P** $\subseteq$ **NP** $\cap$ **co-NP**.

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

Fundamental open question. Does **P** = **NP** $\cap$ **co-NP**?

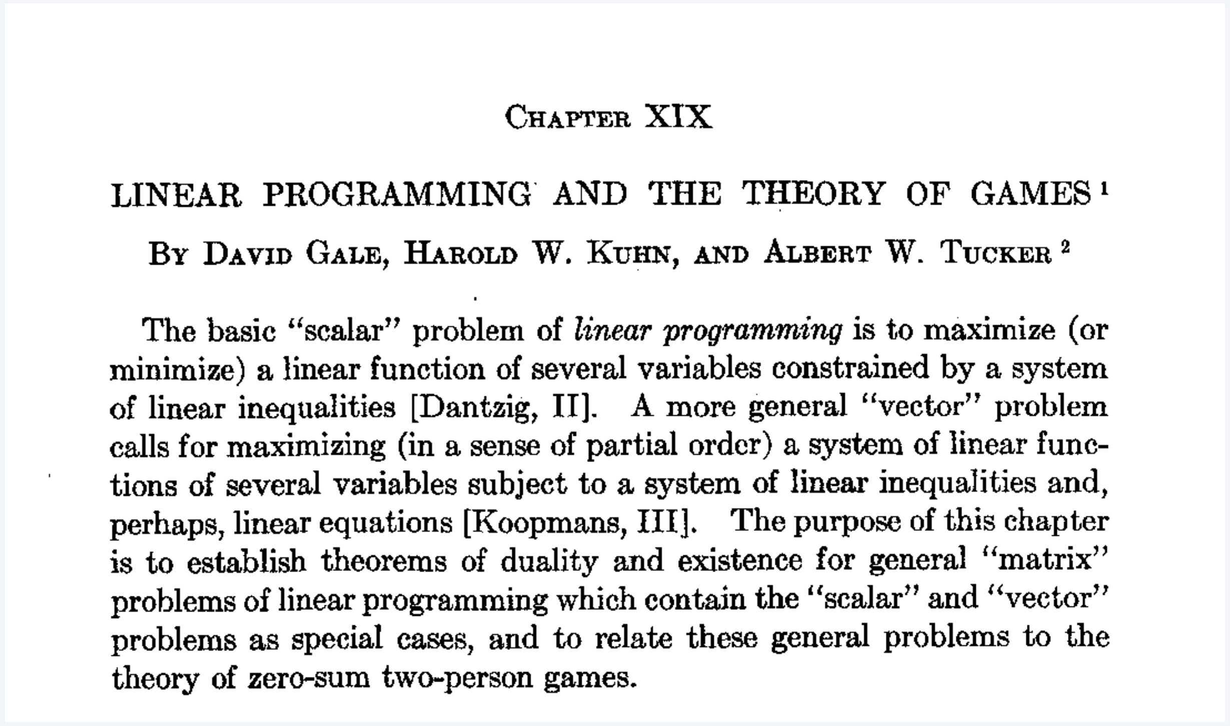
- Mixed opinions.
- Many examples where problem found to have a nontrivial good characterization, but only years later discovered to be in **P**.

Linear programming is in NP ∩ co-NP

LINEAR-PROGRAMMING. Given $A \in \Re^{m \times n}$, $b \in \Re^m$, $c \in \Re^n$, and $\alpha \in \Re$, does there exist $x \in \Re^n$ such that $Ax \le b$, $x \ge 0$ and $c^{\mathrm{T}} x \ge \alpha$?

Theorem. [Gale–Kuhn–Tucker 1948] LINEAR-PROGRAMMING $\in$ **NP** $\cap$ **co-NP**.
Pf sketch. If (P) and (D) are nonempty, then max = min.

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


CHAPTER XIX

LINEAR PROGRAMMING AND THE THEORY OF GAMES[1]

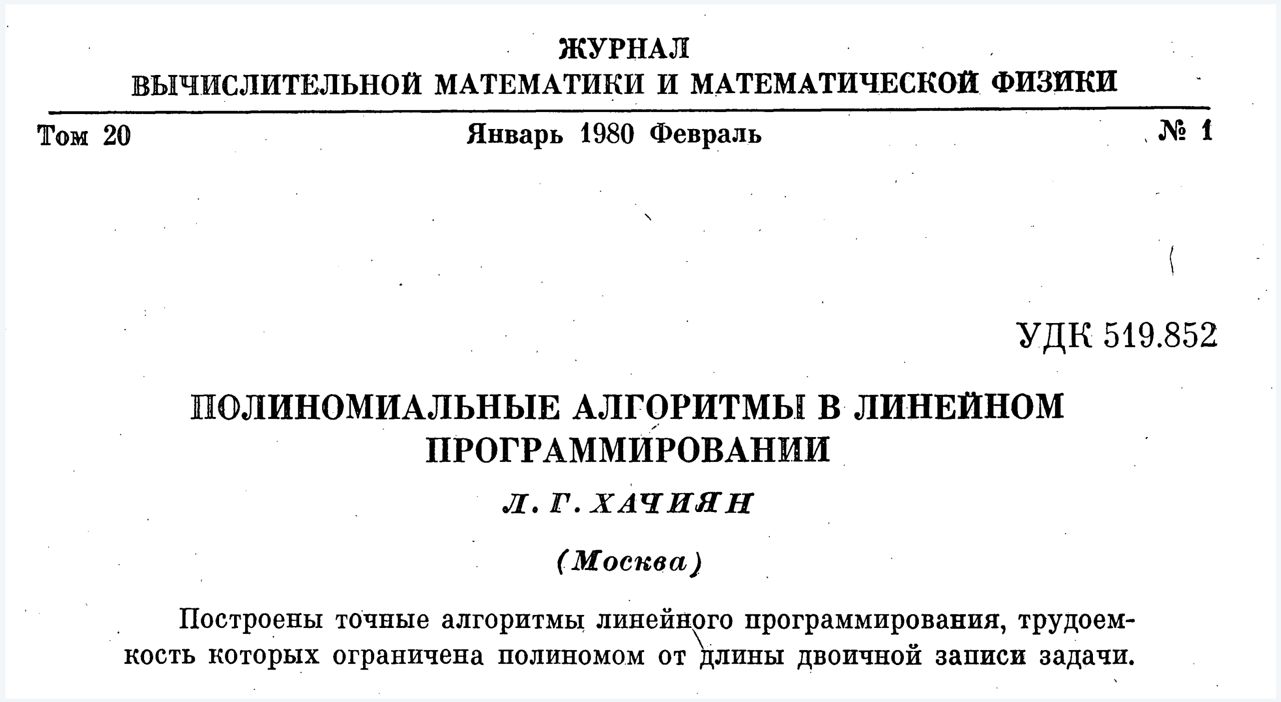
BY DAVID GALE, HAROLD W. KUHN, AND ALBERT W. TUCKER[2]

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 NP ∩ co-NP

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

Theorem. [Khachiyan 1979] LINEAR-PROGRAMMING $\in$ **P.**


ЖУРНАЛ
ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И МАТЕМАТИЧЕСКОЙ ФИЗИКИ

Том 20 Январь 1980 Февраль № 1

УДК 519.852

ПОЛИНОМИАЛЬНЫЕ АЛГОРИТМЫ В ЛИНЕЙНОМ ПРОГРАММИРОВАНИИ

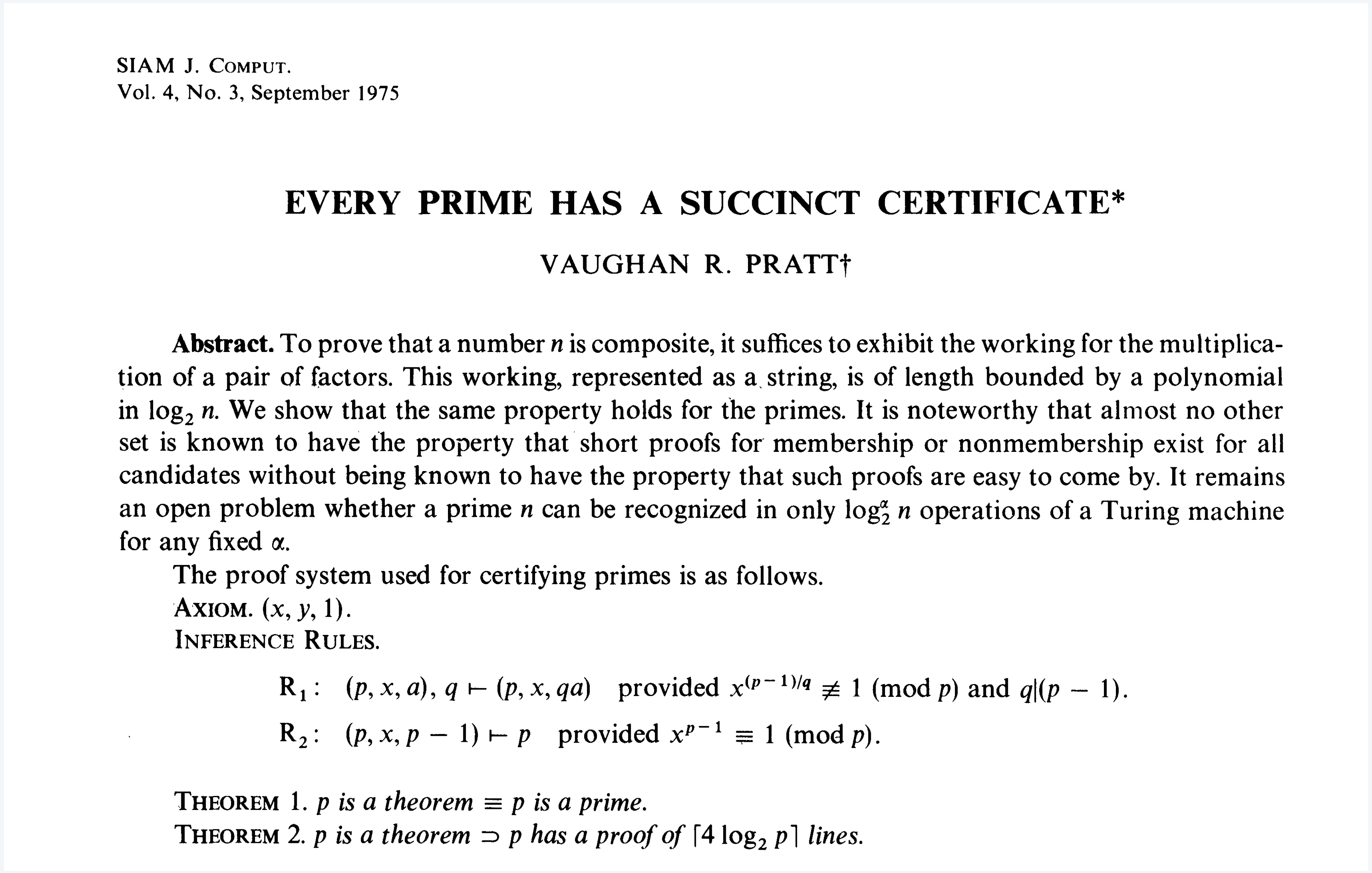
Л. Г. ХАЧИЯН

(Москва)

Построены точные алгоритмы линейного программирования, трудоемкость которых ограничена полиномом от длины двоичной записи задачи.

Primality testing is in NP ∩ co-NP

Theorem. [Pratt 1975] PRIMES ∈ **NP** ∩ **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.

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

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

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 NP ∩ co-NP

Theorem. [Pratt 1975] PRIMES ∈ **NP ∩ 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, $2^2 \times 3 \times 36473$

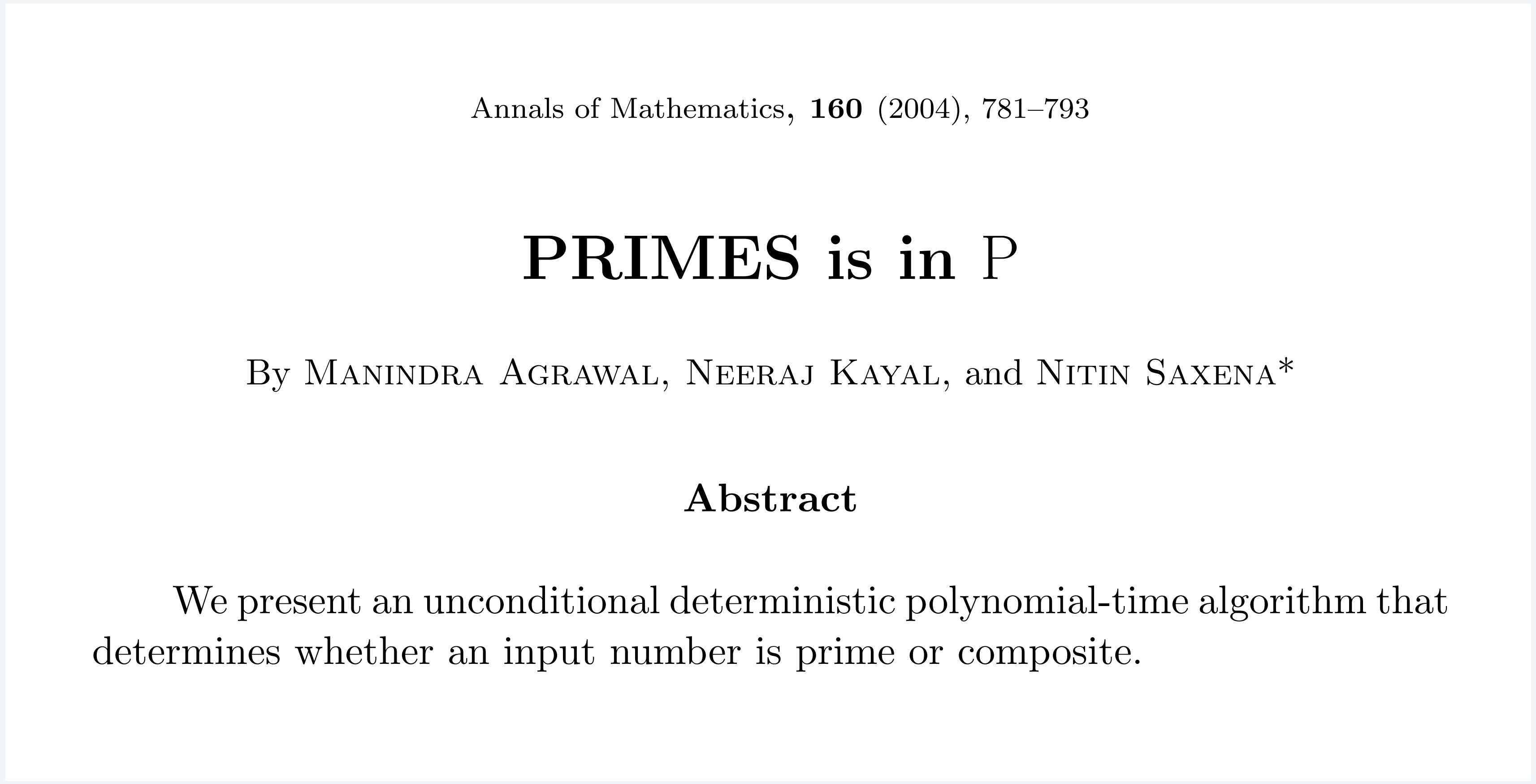
prime factorization of s–1
also need a recursive certificate
to assert that 3 and 36,473 are prime

```
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] PRIMES ∈ **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 NP ∩ 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. FACTOR $\equiv_P$ FACTORIZE.

Pf.

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

Theorem. FACTOR ∈ **NP** ∩ **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.

7403756347956171282804679609742957314259318888923128
9084936232638972765034028266276891996419625117843995
8943305021275853701189680982867331732731089309005525
0511687706329907239638078671008609696253793465056379
6359

RSA-704
($30,000 prize if you can factor)

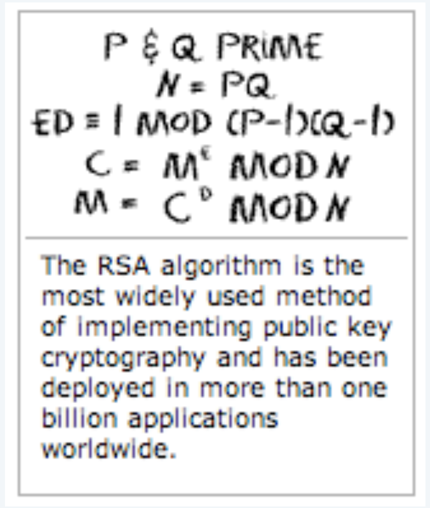
Exploiting intractability

Modern cryptography.

- Ex. Send your credit card number 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 & Q PRIME

$N = PQ$

$ED \equiv 1 \text{ MOD } (P-1)(Q-1)$

$C = M^E \text{ MOD } N$

$M = C^D \text{ MOD } N$

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.

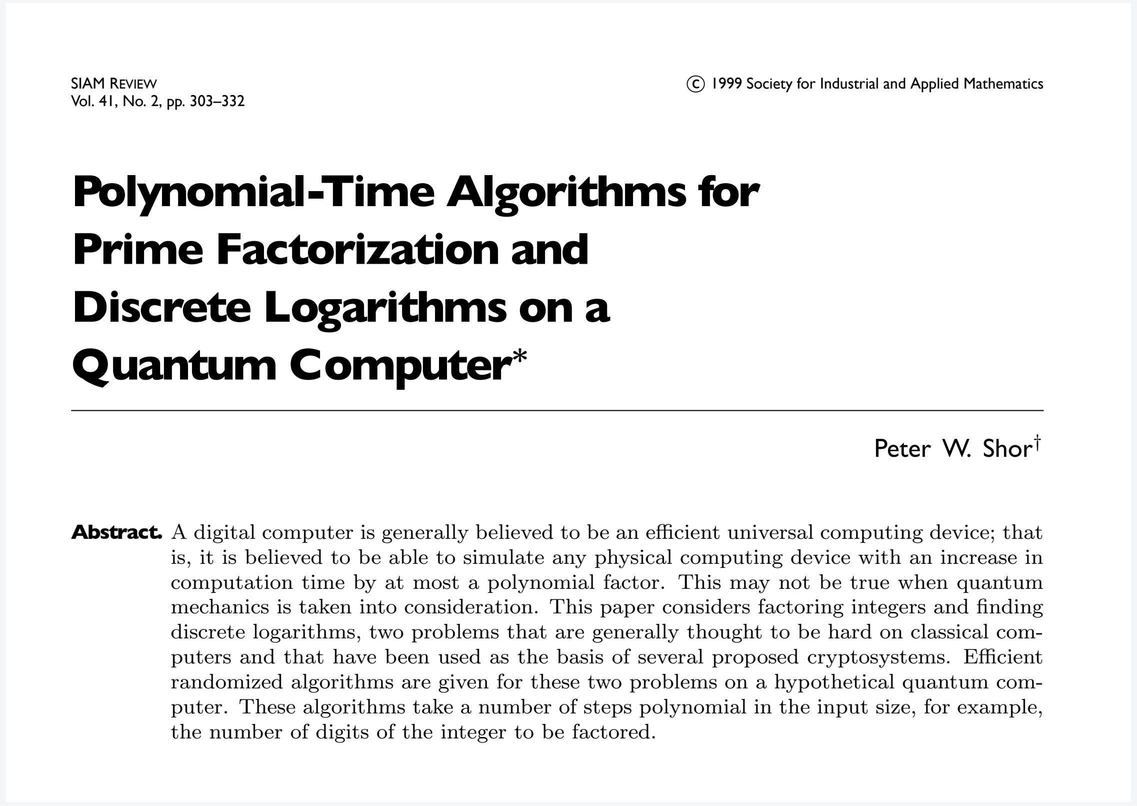
RSA algorithm

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."



SIAM REVIEW
Vol. 41, No. 2, pp. 303–332

© 1999 Society for Industrial and Applied Mathematics

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

Abstract. A digital computer is generally believed to be an efficient universal computing device; that is, it is believed to be able to simulate any physical computing device with an increase in computation time by at most a polynomial factor. This may not be true when quantum mechanics is taken into consideration. This paper considers factoring integers and finding discrete logarithms, two problems that are generally thought to be hard on classical computers and that have been used as the basis of several proposed cryptosystems. Efficient randomized algorithms are given for these two problems on a hypothetical quantum computer. These algorithms take a number of steps polynomial in the input size, for example, the number of digits of the integer to be factored.

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)