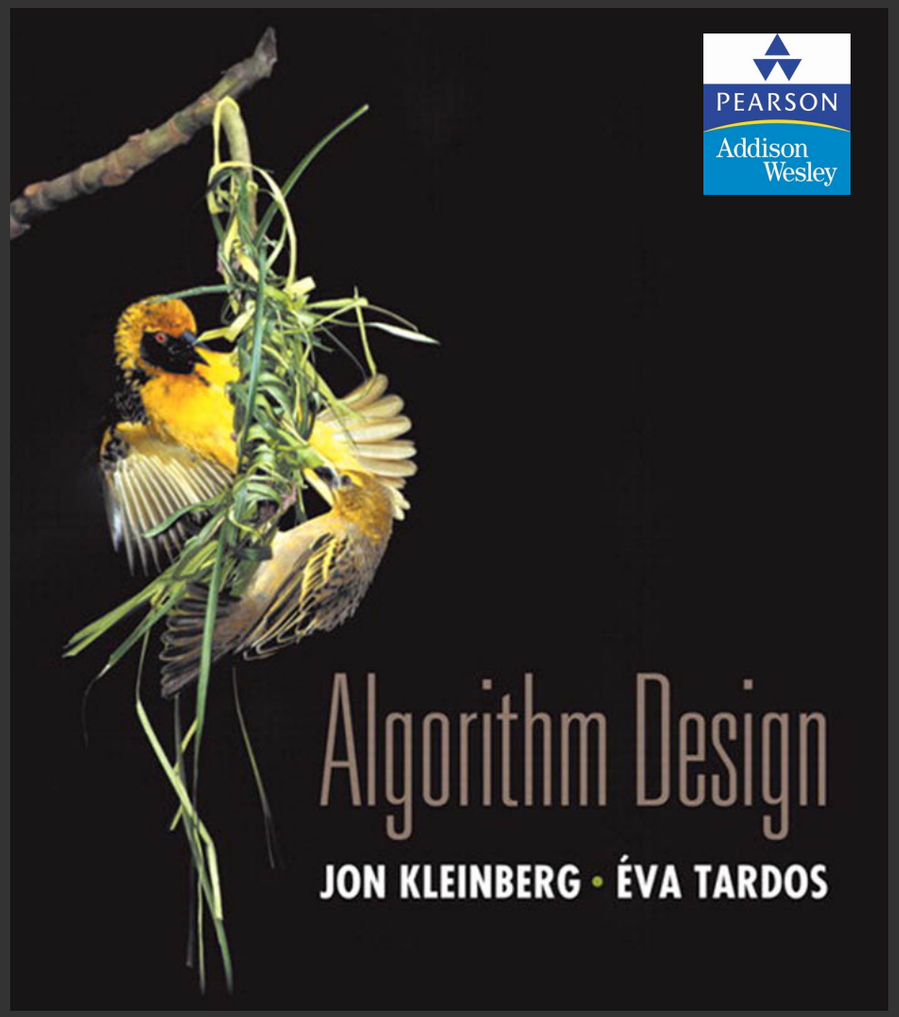


8. Intrattabilità II

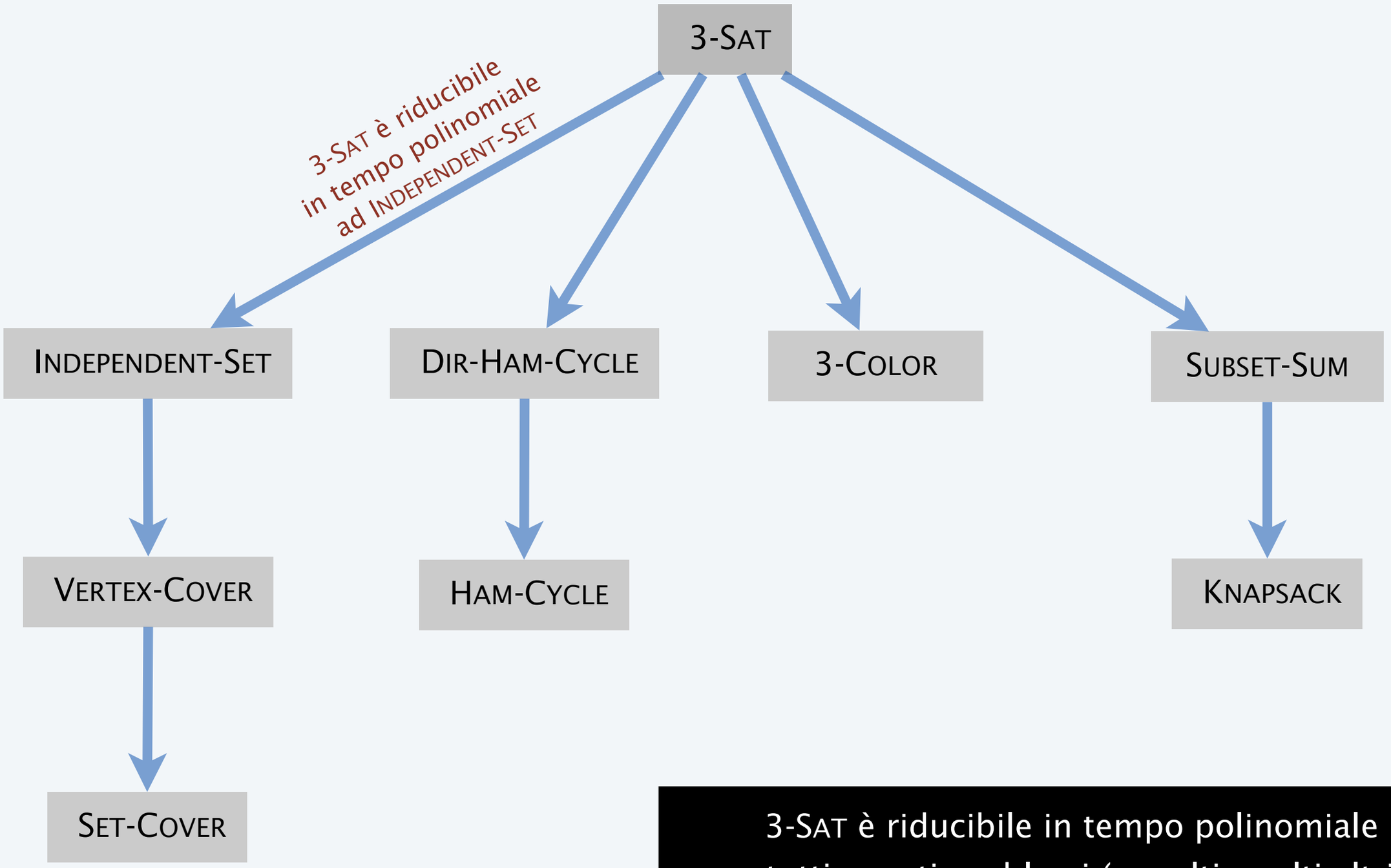
- *P vs. NP*
- *NP-completo*
- *co-NP*
- *NP-arduo*

Traduzione e adattamento di Vincenzo Bonifaci
Original lecture slides by Kevin Wayne

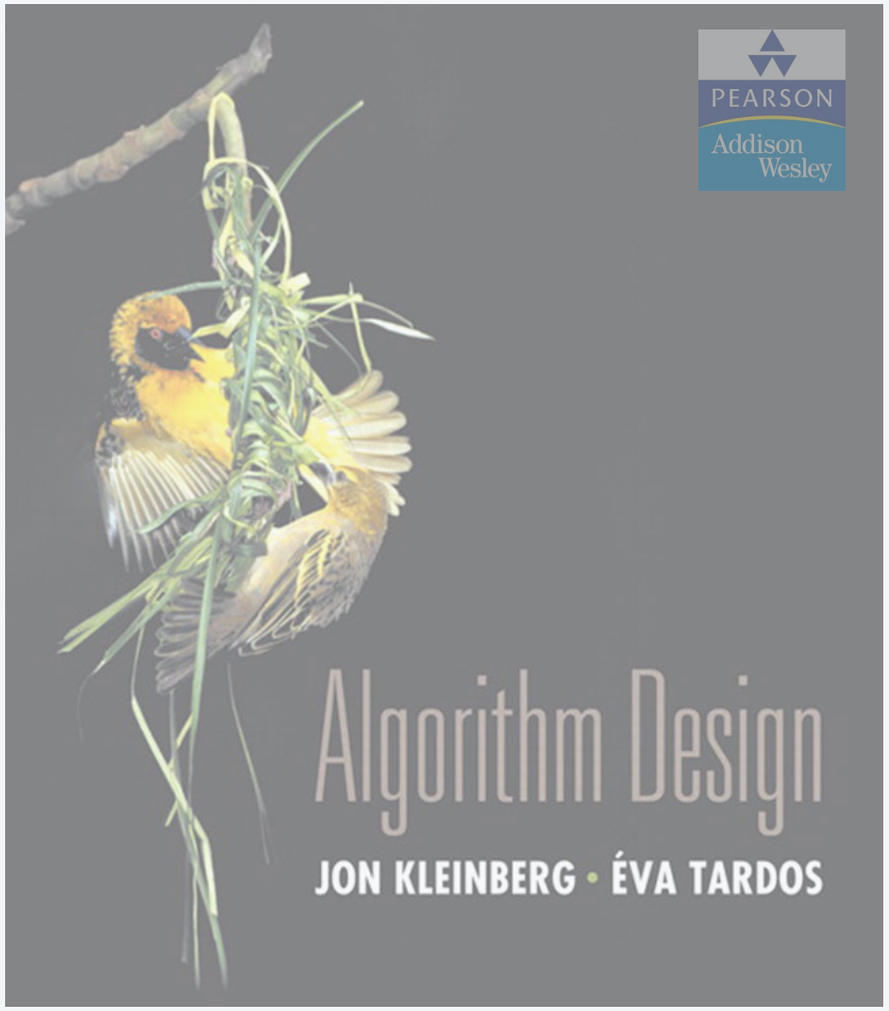
http://www.cs.princeton.edu/~wayne/kleinberg-tardos

Last updated on 10/06/2020 11:53

Riepilogo



3-SAT è riducibile in tempo polinomiale
a tutti questi problemi (e molti, molti altri)



SECTION 8.3

8. INTRATTABILITÀ II

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

P

Problemi di decisione.

- Un problema X è un insieme di stringhe.
- Un'istanza s è una stringa.
- Algoritmo A risolve X se:

$$A(s) = \begin{cases} yes & \text{if } s \in X \\ no & \text{if } s \notin X \end{cases}$$

Def. Algoritmo A è tempo polinomiale se per ogni stringa s, $A(s)$ termina in $\leq p(|s|)$ "passi," dove $p(\cdot)$ è un qualche polinomio.

lunghezza di s

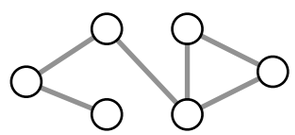
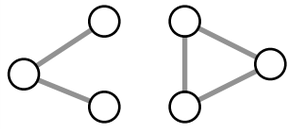
Def. **P** = insieme di problemi di decisione per i quali esiste un algoritmo polinomiale.

su una macchina di Turing deterministica

problema PRIMES:	$\{ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, \ldots \}$
istanza s:	592335744548702854681
algoritmo:	Agrawal–Kayal–Saxena (2002)

Alcuni problemi in P

P. Problemi di decisione per i quali esiste un algoritmo polinomiale.

problema	descrizione	algoritmo tempo-polinomiale	istanza yes	istanza no
MULTIPLE	x è un multiplo di y ?	divisione elementare	51, 17	51, 16
REL-PRIME	x e y sono primi tra loro ?	algoritmo di Euclide	34, 39	34, 51
PRIMES	x è primo ?	Agrawal–Kayal–Saxena	53	51
EDIT-DISTANCE	La edit distance tra x ed y è al più 5 ?	Needleman–Wunsch	niether neither	acgggt ttttta
L-SOLVE	Esiste un vettore x che soddisfa $Ax = b$?	eliminazione Gaussiana	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$
U-CONN	Il grafo G è connesso ?	visita in profondità		

NP

Def. Un algoritmo $C(s, t)$ è un certificatore per il problema X se per ogni stringa s si ha: $s \in X$ sse esiste una stringa t tale che $C(s, t) = yes$.

Def. **NP** = insieme dei problemi di decisione che ammettono certificatori polinomiali:

- $C(s, t)$ è un algoritmo tempo-polinomiale.
- Il certificato t ha taglia polinomiale: $|t| \leq p(|s|)$ per qualche polinomio $p(\cdot)$.

"certificato" o "testimone"

problema COMPOSITES:	$\{ 4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, \ldots. \}$
istanza s:	437669
certificato t:	541 ⟵ 437,669 = 541 × 809
certificatore C(s, t):	divisione elementare

Certificatori e certificati: Satisfiability

SAT. Data una formula CNF Φ, esiste un'assegnazione di verità valida?

3-SAT. SAT in cui ogni clause ha esattamente 3 letterali.

Certificato. Un'assegnazione di valori di verità alle variabili booleane.

Certificatore. Verifica che ogni clausola di Φ abbia almeno un letterale vero.

istanza s $\Phi = \left(\overline{x_1} \vee x_2 \vee x_3 \right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3 \right) \wedge \left(\overline{x_1} \vee x_2 \vee x_4 \right)$

certificato t $x_1 = true,\ x_2 = true,\ x_3 = false,\ x_4 = false$

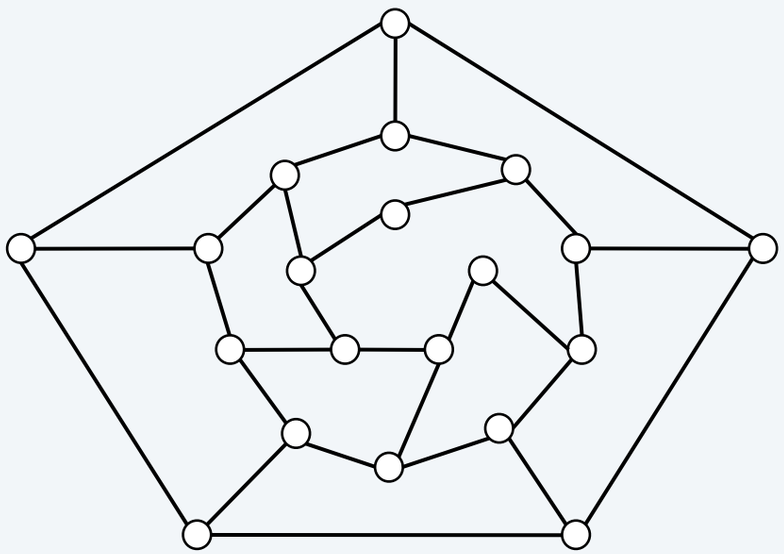
Conclusioni. SAT $\in$ **NP**, 3-SAT $\in$ **NP**.

Certificatori e certificati: Cammini hamiltoniani

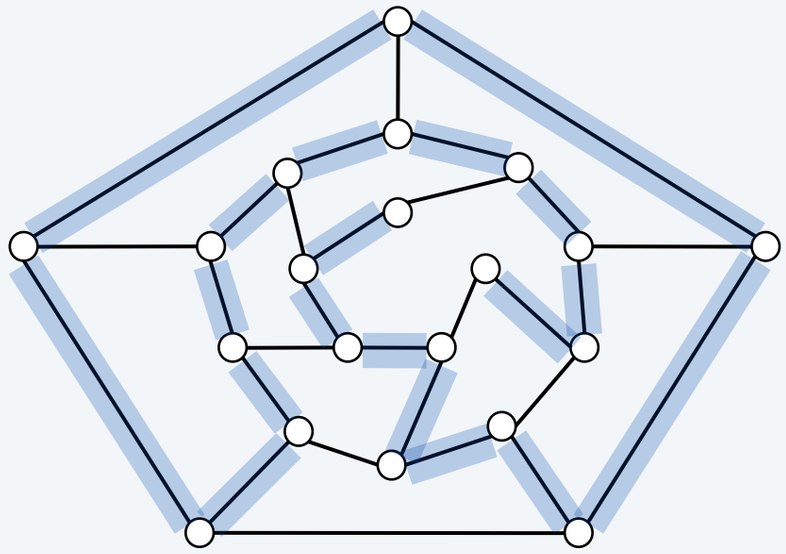
HAMILTON-PATH. Dato un grafo non orientato $G = (V, E)$, esiste un cammino semplice P che visita ogni nodo?

Certificato. Una permutazione π degli n nodi.

Certificatore. Verifica che π contenga ogni nodo di V esattamente una volta, e che G contenga un arco tra ogni coppia di nodi adiacenti in π.



istanza s

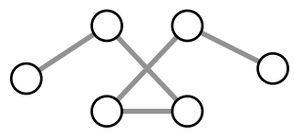
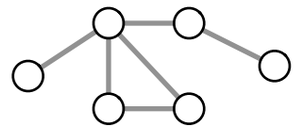


certificato t

Conclusione. HAMILTON-PATH $\in$ **NP.**

Alcuni problemi in NP

NP. Problemi di decisione per i quali esiste un certificatore tempo-polinomiale.

problema	descrizione	algoritmo tempo-polinomiale	istanza yes	istanza no
L-SOLVE	Esiste un vettore x che soddisfa $Ax = b$?	eliminazione gaussiana	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$
COMPOSITES	x è composto ?	Agrawal–Kayal–Saxena	51	53
FACTOR	x ha un fattore non banale minore o uguale a y ?	???	(56159, 50)	(55687, 50)
SAT	Data una formula CNF, esiste un'assegnazione valida?	???	$\neg x_1 \vee x_2 \vee \neg x_3$ $x_1 \vee \neg x_2 \vee x_3$ $\neg x_1 \vee \neg x_2 \vee x_3$	$\neg x_2$ $x_1 \vee x_2$ $\neg x_1 \vee x_2$
HAMILTON-PATH	Esiste un cammino semplice tra u e v che visita ogni nodo?	???		

Intrattabilità: quiz 1

Quali dei seguenti problemi su grafi sono noti appartenere ad NP?

A. La lunghezza del cammino semplice più lungo è $\leq k$?

B. La lunghezza del cammino semplice più lungo è $\geq k$?

C. La lunghezza del cammino semplice più lungo è $= k$?

D. Trova la lunghezza del cammino semplice più lungo.

E. Tutte le precedenti.

Intrattabilità: quiz 2

Nella teoria della complessità, l'abbreviazione NP sta per…

A. Nope.

B. Nessun problema.

C. Tempo non polinomiale.

D. Spazio non polinomiale.

E. Tempo nondeterministico polinomiale.

L'importanza di NP

NP. Problemi di decisione per i quali esiste un certificatore tempo-polinomiale.

> *“ NP cattura vasti domini di imprese computazionali, scientifiche, e matematiche, e appare delimitare all'incirca ciò che i matematici e gli scienziati aspirano a calcolare in modo efficiente.”* — *Christos Papadimitriou*

> *“ In un mondo ideale andrebbe rinominato P vs VP. ”* — *Clyde Kruskal*

P, NP, ed EXP

P. Problemi di decisione per cui esiste un algoritmo tempo-polinomiale.

NP. Problemi di decisione per cui esiste un certificatore tempo-polinomiale.

EXP. Problemi di decisione per cui esiste un algoritmo tempo-esponenziale.

Proposizione. **P** $\subseteq$ **NP**.

Dim. Considera un qualunque problema $X \in$ **P**.

- Per definizione, esiste un algoritmo tempo-polinomiale $A(s)$ che risolve X.
- Certificato $t = \varepsilon$ (stringa vuota), certificatore $C(s, t) = A(s)$. ▪

Proposizione. **NP** $\subseteq$ **EXP**.

Dim. Considera un qualunque problema $X \in$ **NP**.

- Per definizione, esiste un certificatore tempo-polinomiale $C(s, t)$ per X, dove il certificato t soddisfa $|t| \leq p(|s|)$ per qualche polinomio $p(\cdot)$.
- Per risolvere l'istanza s, esegui $C(s, t)$ su tutte le stringhe t con $|t| \leq p(|s|)$.
- Ritorna *yes* sse $C(s, t)$ ritorna *yes* per qualcuno di questi potenziali certificati. ▪

Fatto. **P** $\neq$ **EXP** $\Rightarrow$ o si ha **P** $\neq$ **NP**, o si ha **NP** $\neq$ **EXP**, o entrambe le cose.

La questione principale: P vs. NP

D. Come risolvere un'istanza di 3-SAT con n variabili?

R. Ricerca esaustiva: prova tutte le 2^n assegnazioni di verità.

D. Possiamo fare qualcosa di sostanzialmente più intelligente?

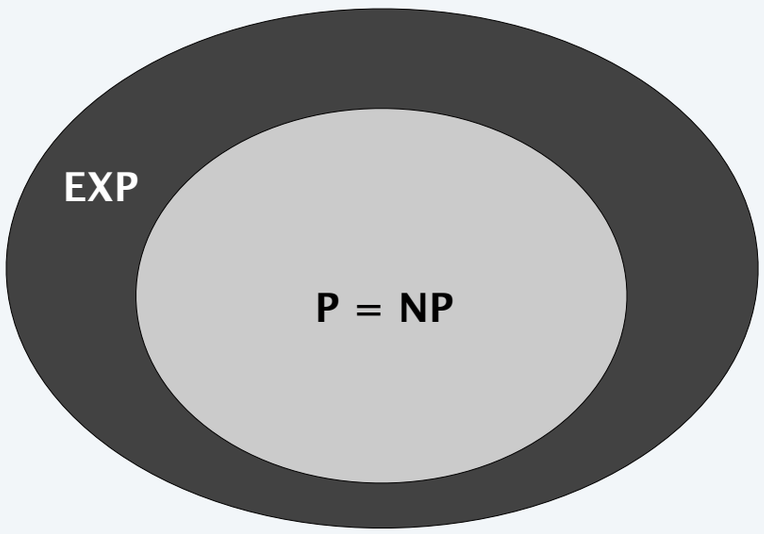
Congettura. Non esiste un algoritmo tempo-polinomiale per 3-SAT.

"intrattabile"

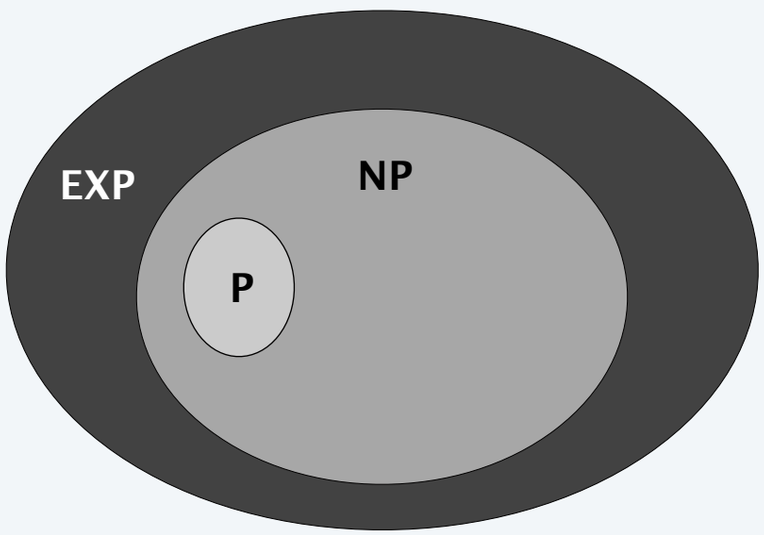
La questione principale: P vs. NP

Si ha P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]

Il problema di decisione è facile quanto il problema di certificazione?



Se P = NP



Se P ≠ NP

Se sì... Algoritmi efficienti per 3-SAT, TSP, VERTEX-COVER, FACTOR, ...

Se no... Nessun algoritmo efficiente possibile per 3-SAT, TSP, VERTEX-COVER, ...

Opinione di maggioranza. Probabilmente no.

Riduzioni: quiz 3

Supponiamo P ≠ NP. Quale dei seguenti algoritmi è ancora possibile?

A. Algoritmo $O(n^3)$ per fattorizzare interi ad n-bit.

B. Algoritmo $O(1.657^n)$ per HAMILTON-CYCLE.

C. Algoritmo $O(n^{\log \log \log n})$ per 3-SAT.

D. Tutte le precedenti.

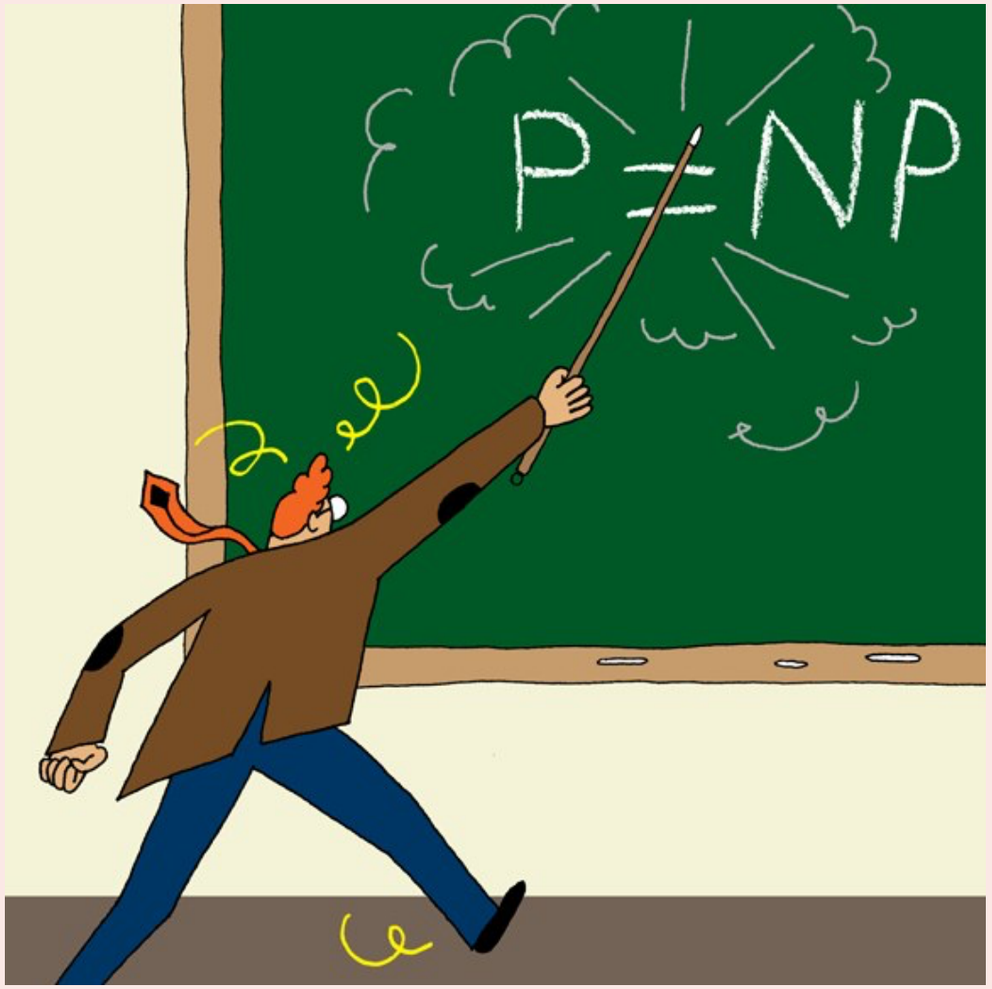
Intrattabilità: quiz 4

Si ha P = NP?

A. Sì.

B. No.

C. Nessuna delle precedenti.



Possibili esiti

P ≠ NP

“ Io congetturo che non vi siano algoritmi efficienti per il problema del commesso viaggiatore. Le mie motivazioni sono le stesse dietro ogni congettura matematica: (i) È uno scenario possibile e (ii) non lo so.

— *Jack Edmonds 1966*

“ Secondo me, non si può tirare a indovinare in modo intelligente sulla risposta a queste domande. Se dovessi scommettere ora, scommetterei che P non è uguale ad NP. Stimo l'emivita di questo problema a 25–50 anni, ma non scommetterei che venga risolto prima del 2100. ”

— *Bob Tarjan (2002)*

Possibili esiti

P ≠ NP

“ Apparentemente ci manca anche la comprensione più basilare della natura della difficoltà della questione… Tutti gli approcci tentati sinora probabilmente (in alcuni casi, dimostrabilmente) hanno fallito. In tal senso, P=NP è diverso da molti altri importanti problemi matematici sui quali si è avuto un progresso costante (talvolta per secoli) dopo il quale essi sono stati risolti, completamente o parzialmente. ”

— *Alexander Razborov (2002)*

Possibili esiti

P = NP

“ Su questa questione credo di essere tra le fila dei pazzoidi della comunità matematica: penso (non troppo ardentemente!) che P=NP e che questo sarà dimostrato entro vent'anni. Alcuni anni fa, Charles Read ed io ci abbiamo lavorato un bel po', ed abbiamo anche festeggiato con una cena prima di accorgerci di un errore assolutamente fatale. ”

— *Béla Bollobás (2002)*

“ Secondo me non dovrebbe trattarsi di un problema difficile; è solo che siamo arrivati tardi a questa teoria, e non abbiamo ancora sviluppato tecniche per dimostrare che alcune computazioni sono difficili. Alla fine, sarà solo una nota a piè di pagina in un libro. ” — *John Conway*

Altri possibili esiti

P = **NP**, ma tutti gli algoritmi per 3-SAT sono $\Omega(n^{100})$.

P ≠ **NP**, ma con un algoritmo $O(n^{\log^* n})$ per 3-SAT.

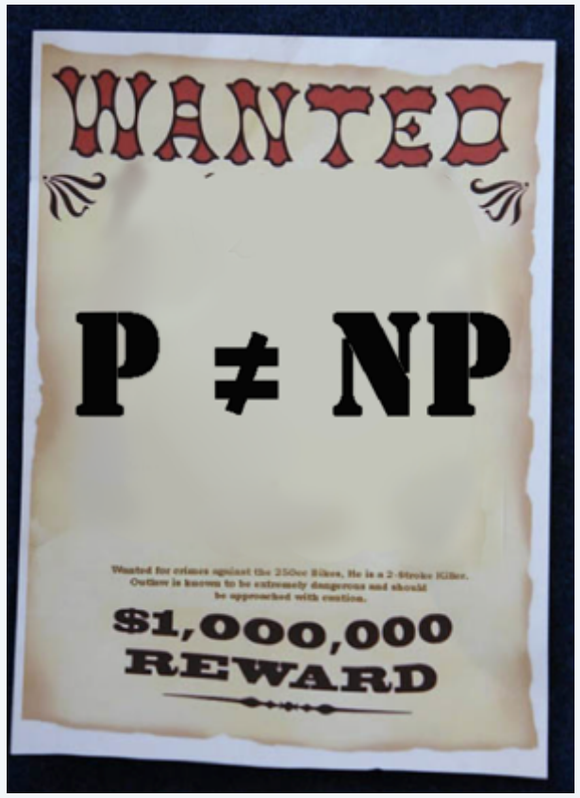
P = **NP** è indipendente (dalla teoria assiomatica degli insiemi ZFC).

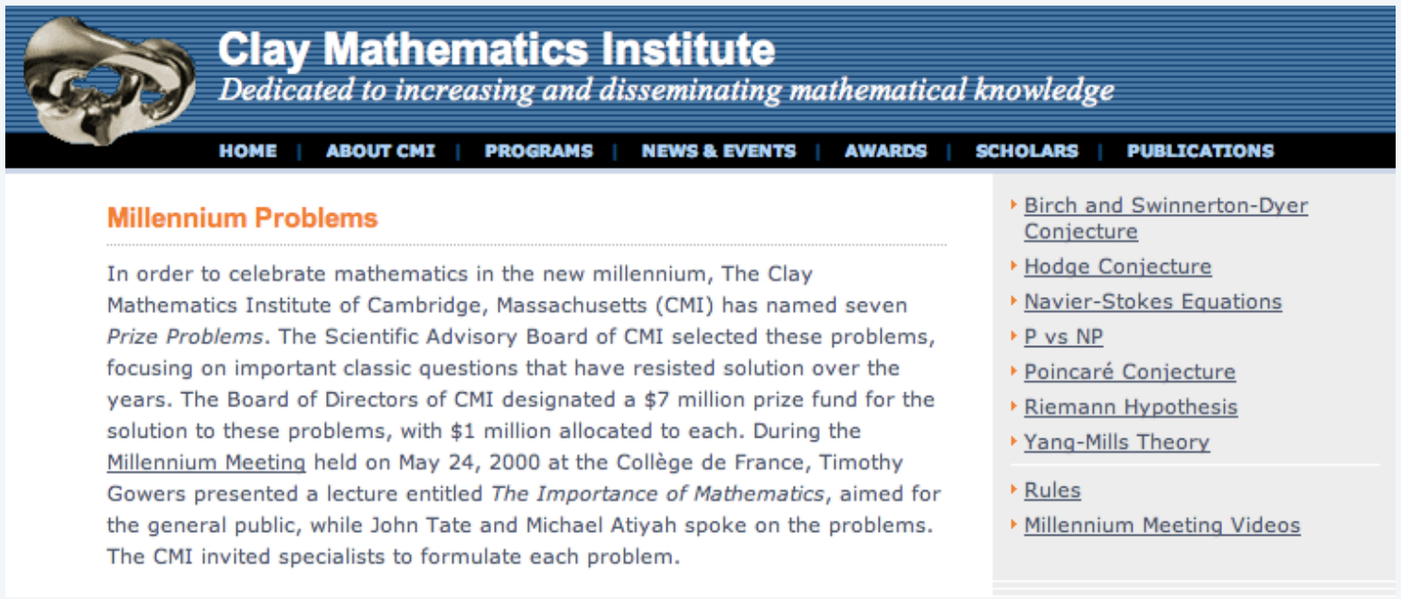
“ Sarà risolto o entro il 2048 o entro il 4096. Attualmente sono un po' pessimista. L'esito sarà lo scenario davvero peggiore: qualcuno dimostrerà che P = NP perché ci sono solo un numero finito di ostruzioni per l'ipotesi opposta; quindi esiste un algoritmo tempo-polinomiale per SAT ma non conosceremo mai la sua complessità! ”

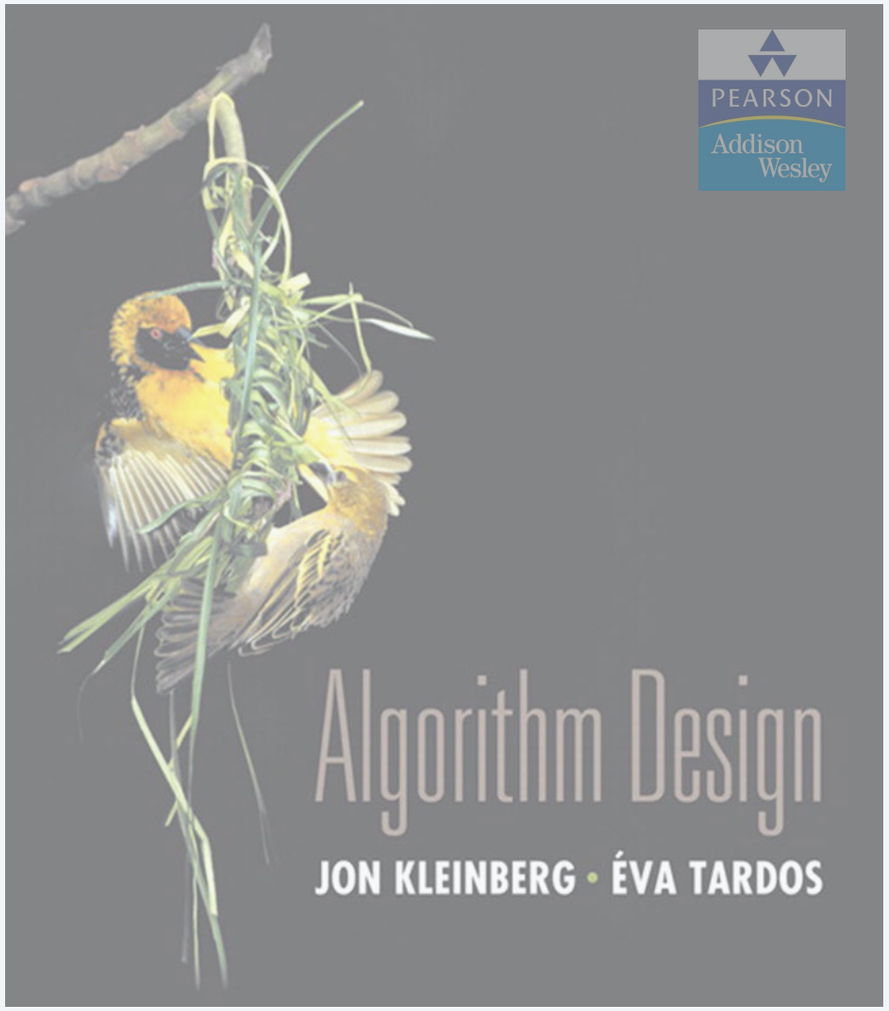
— *Donald Knuth*

Premio del millennio

Premio del millennio. 1 milione di dollari per la risoluzione della questione **P ≠ NP**.







SECTION 8.4

8. INTRATTABILITÀ II

- *P vs. NP*
- *NP-completo*
- *co-NP*
- *NP-hard*

Trasformazioni polinomiali

Def. Problema X (Cook)-riduce al problema Y se istanze arbitrarie del problema X possono essere risolte utilizzando:

- Un numero polinomiale di passi computazionali standard, più
- Un numero polinomiale di chiamate ad un oracolo per il problema Y.

Def. Problema X (Karp)-trasforma nel problema Y se data una istanza x di X, si può costruire un'istanza y di Y tale che x è un'istanza yes di X sse y è un'istanza yes di Y.

inoltre $|y|$ deve avere taglia polinomiale in $|x|$

Nota. Una trasformazione polinomiale è una riduzione polinomiale con una sola chiamata all'oracolo per Y, esattamente alla fine dell'algoritmo per X. Quasi tutte le riduzioni da noi viste in precedenza avevano questa forma.

Questione aperta. Questi due concetti sono equivalenti rispetto ad **NP**?

abusiamo la notazione $\leq_P$ evitando di distinguere i due tipi

NP-completo

NP-completo. Un problema $Y \in$ **NP** con la proprietà che per ogni problema $X \in$ **NP**, si ha $X \leq_P Y$.

Proposizione. Supponi $Y \in$ **NP**-completo. Allora, $Y \in$ **P** sse **P** = **NP**.

Dim. $\Leftarrow$ Se **P** = **NP**, allora $Y \in$ **P** perché $Y \in$ **NP**.

Dim. $\Rightarrow$ Supponi $Y \in$ **P**.

- Considera qualunque problema $X \in$ **NP**. Poiché $X \leq_P Y$, abbiamo $X \in$ **P**.
- Ciò implica **NP** $\subseteq$ **P**.
- Già sappiamo che **P** $\subseteq$ **NP**. Quindi **P** = **NP**. ▪

Questione fondamentale. Esistono dei problemi **NP**-completi "naturali"?

Il "primo" problema NP-completo

Teorema. [Cook 1971, Levin 1973] SAT $\in$ **NP**-completi.

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/logn 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, предписываемого 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

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

Аналогично будем понимать термин «меньше или сравнимо».

Определение. 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 списком 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.

Come dimostrare l'NP-completezza

Nota. Una volta trovato un problema **NP**-completo "naturale", gli altri seguono come tessere di un domino.

Ricetta. Per dimostrare $Y \in$ **NP**-completi:

- Passo 1. Mostra che $Y \in$ **NP**.
- Passo 2. Scegli un qualunque problema **NP**-completo X.
- Passo 3. Dimostra che $X \leq_P Y$.

Proposizione. Se $X \in$ **NP**-completi, $Y \in$ **NP**, e $X \leq_P Y$, allora $Y \in$ **NP**-completi.

Dim. Considera un problema $W \in$ **NP**. Allora, vale sia $W \leq_P X$ che $X \leq_P Y$.

- Per transitività, $W \leq_P Y$.
- Quindi $Y \in$ **NP**-completi. ▪

per definizione di **NP**-completo

per assunzione

Riduzioni: quiz 4

Supponi che $X \in$ NP-Complete, $Y \in$ NP, e $X \leq_{\mathrm{P}} Y$. Cosa si può concludere?

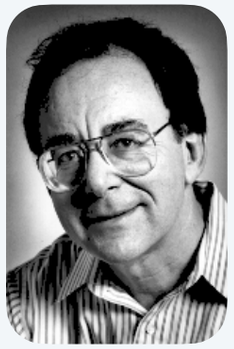
A. Y è **NP**-completo.

B. Se $Y \notin$ **P**, allora **P** ≠ **NP**.

C. Se **P** ≠ **NP**, allora né X né Y sono in **P**.

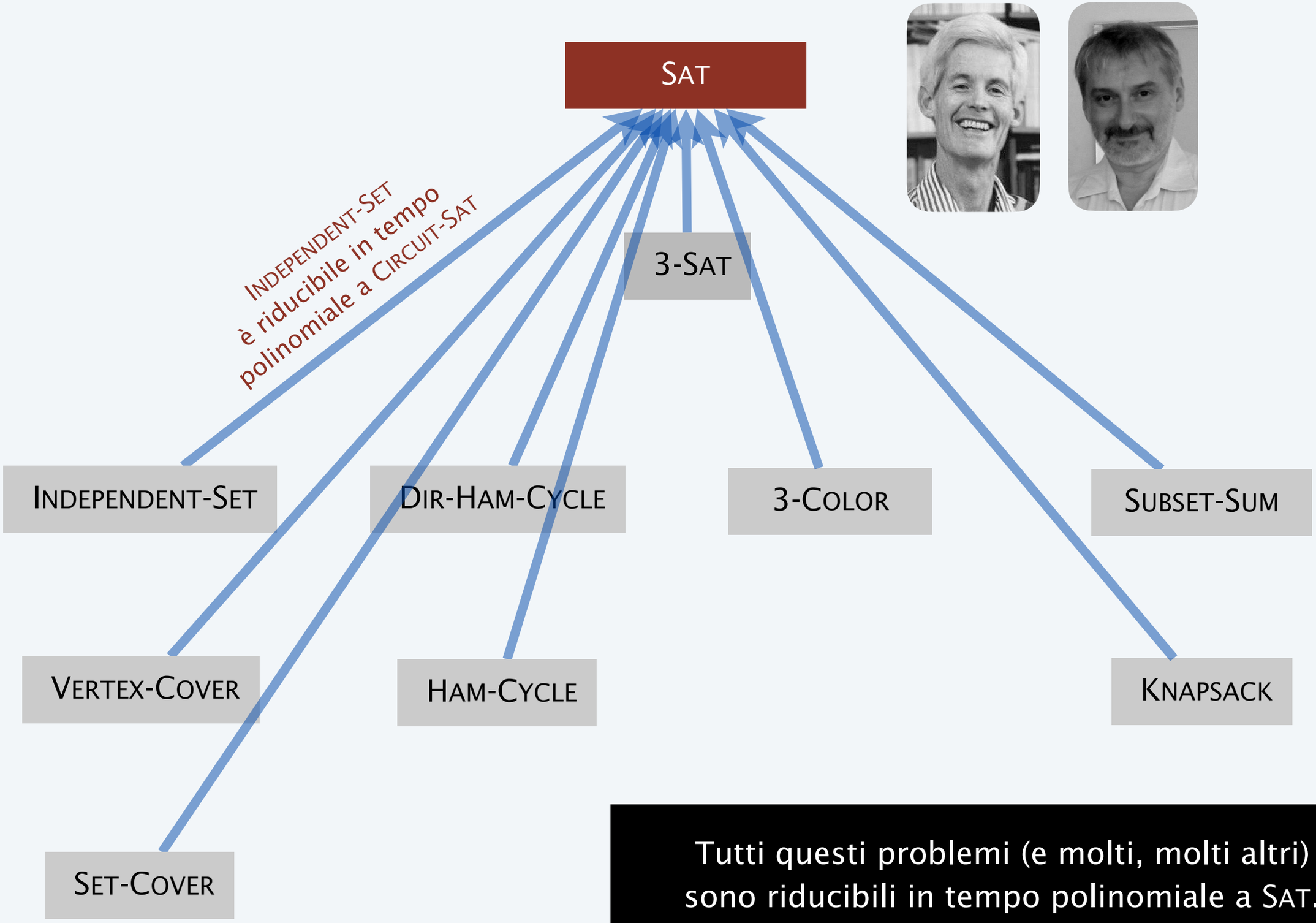
D. Tutte le precedenti.

Implicazioni dovute a Karp



SAT è riducibile in tempo polinomiale a tutti questi problemi (e molti, molti altri)

Implicazioni dovute a Cook–Levin



Tutti questi problemi (e molti, molti altri) sono riducibili in tempo polinomiale a SAT.

Implicazioni dovute a Karp + Cook–Levin

Tutti questi problemi sono **NP**-complete; sono manifestazioni dello stesso problema difficile.

Problemi NP-completi

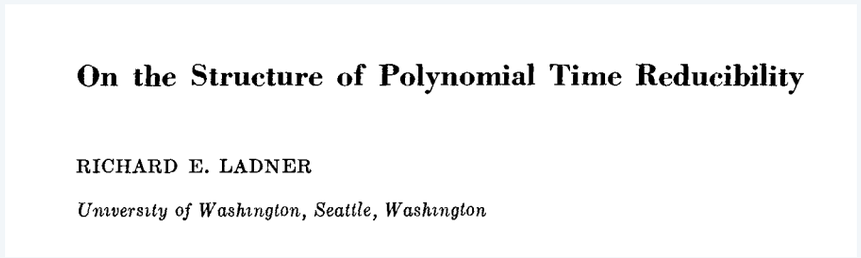
Generi fondamentali di problemi NP-completi ed esempi paradigmatici.

- Copertura/impaccamento: SET-COVER, VERTEX-COVER,INDEPENDENT-SET.
- Soddisfacimento di vincoli: CIRCUIT-SAT, SAT, 3-SAT.
- Sequenziamento: HAMILTON-CYCLE, TSP.
- Partizionamento: 3D-MATCHING, 3-COLOR.
- Numerici: SUBSET-SUM, KNAPSACK.

In pratica. Il più dei problemi in **NP** sono noti essere in **P** o **NP**-completi.

NP-intermedi? FACTOR, DISCRETE-LOG, GRAPH-ISOMORPHISM, ...

Teorema. [Ladner 1975] A meno che **P** = **NP**, esistono problemi in **NP** che non sono né in **P** né **NP**-completi.



On the Structure of Polynomial Time Reducibility

RICHARD E. LADNER

University of Washington, Seattle, Washington

Altri problemi computazionalmente difficili

Garey e Johnson. *Computers and Intractability*.

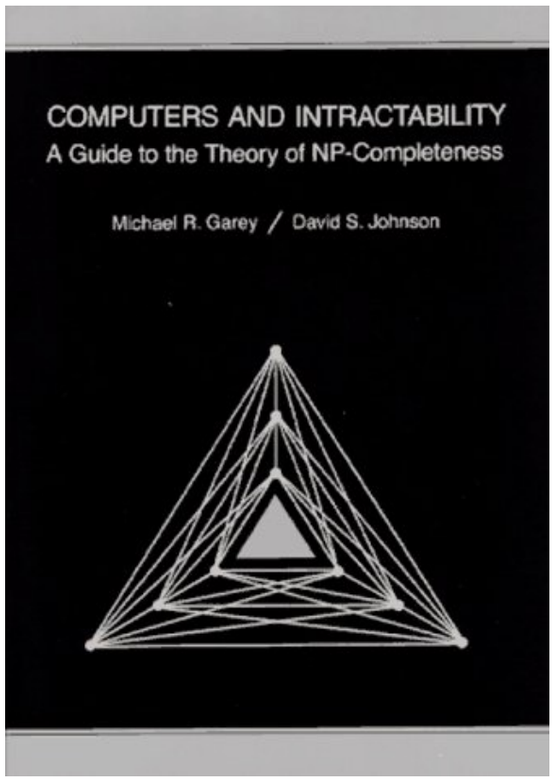
- L'appendice include oltre 300 problemi **NP**-completi.
- Il riferimento più citato della letteratura scientifica in informatica.

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



Altri problemi computazionalmente difficili

Ingegneria aerospaziale. Partizionamento in mesh ottimo per elementi finiti.

Biologia. Ricostruzione filogenetica.

Ingegneria chimica. Sintesi di reti di scambiatori di calore.

Chimica. Ripiegamento (folding) di proteine.

Ingegneria civile. Equilibrio dei flussi di traffico urbano.

Economia. Calcolo dell'arbitraggio in mercati finanziari con frizione.

Ingegneria elettrica. Progettazione di circuiti integrati VLSI.

Ingegneria ambientale. Piazzamento ottimo di sensori per la contaminazione.

Ingegneria finanziaria. Portafoglio a rischio minimo per un dato ricavo.

Teoria dei giochi. Equilibrio di Nash che massimizza il benessere sociale.

Matematica. Dati interi a_1, ..., a_n, calcolare $\int_0^{2\pi} \cos(a_1\theta) \times \cos(a_2\theta) \times \cdots \times \cos(a_n\theta)\, d\theta$

Ingegneria meccanica. Struttura della turbolenza in flussi di torsione.

Medicina. Ricostruzione di forme 3D da angiocardiogrammi biplanari.

Ricerca operativa. Problema del commesso viaggiatore.

Fisica. Funzione di partizione del modello Ising 3D.

Politica. Indice di potere di voto Shapley–Shubik.

Ricreazione. Versioni di Sudoku, Dama, Campo Minato, Tetris, Cubo di Rubik.

Statistica. Progetto ottimo di esperimenti.

Ambito e impatto della NP-completezza

Ambito della NP-completezza. [Papadimitriou 1995]

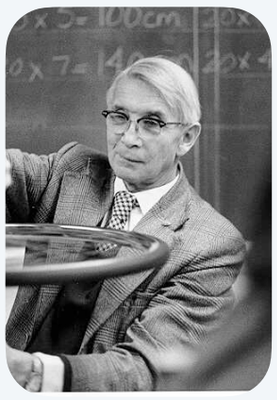
- Principale export intellettuale dell'informatica alle altre discipline.
- 6000 citazioni l'anno (più di “compilatore”, “OS”, “database”).
- Larga applicabilità e potere di classificazione.

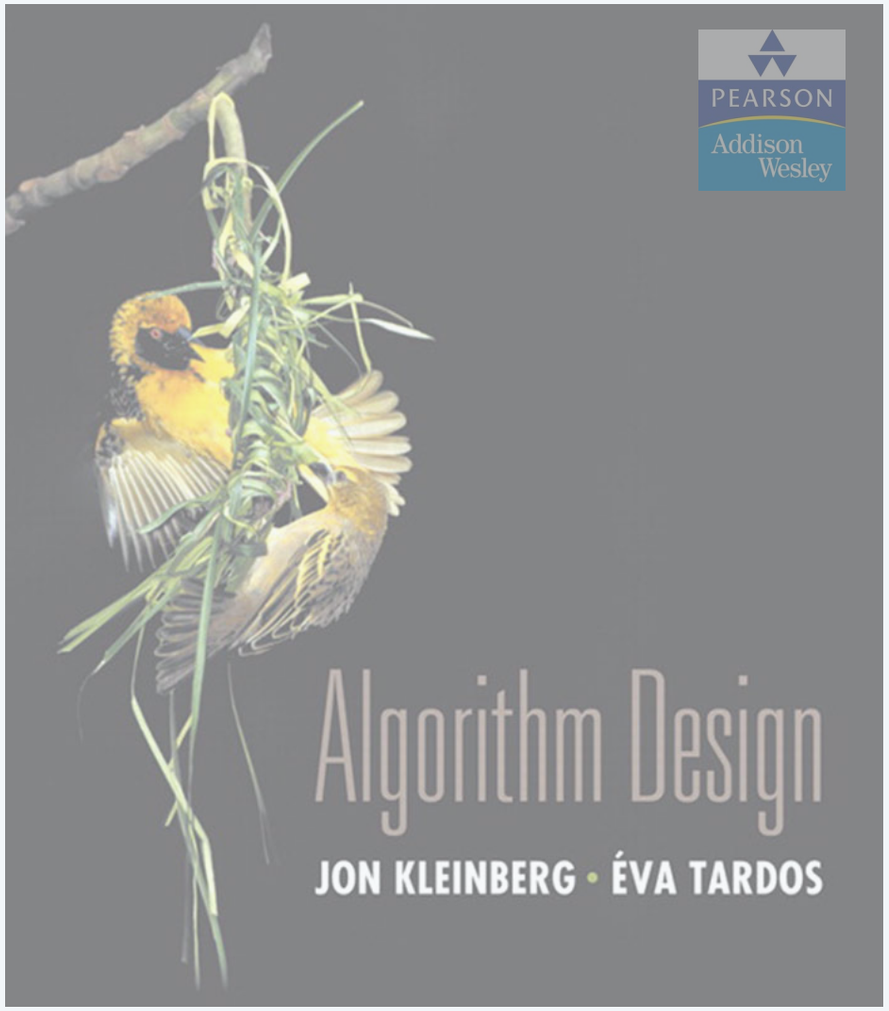
L'NP-completezza può guidare l'indagine scientifica.

- 1926: Ising introduce un semplice modello per le transizioni di fase.
- 1944: Onsager trova una soluzione in forma chiusa a 2D-ISING.
- 19xx: Feynman e altre grandi menti cercano di risolvere 3D-ISING.
- 2000: Istrail dimostra che 3D-ISING ∈ **NP**-completi.

un santo graal della meccanica statistica

la ricerca di una forma chiusa appare destinata a fallire



8. INTRATTABILITÀ II

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

SECTION 8.9

Asimmetria di NP

Asimmetria di NP. Sono richiesti certificati brevi solo per istanze *yes*.

Es 1. SAT vs. UN-SAT.

- Una formula CNF può essere dimostrata soddisfacibile esibendo una assegnazione.
- Come potremmo dimostrare che una formula non è soddisfacibile?

SAT. Data una formula CNF Φ, esiste un'assegnazione valida?

UN-SAT. Data una formula CNF Φ, è vero che nessuna assegnazione è valida?

Asimmetria di NP

Asimmetria di NP. Sono richiesti certificati brevi solo per istanze yes.

Es 2. HAMILTON-CYCLE vs. NO-HAMILTON-CYCLE.

- Un grafo può essere dimostrato hamiltoniano esibendo una permutazione.
- Come potremmo dimostrare che un grafo non è hamiltoniano?

HAMILTON-CYCLE. Dato un grafo $G = (V, E)$, esiste un ciclo semplice Γ che contiene ogni nodo di V ?

NO-HAMILTON-CYCLE. Dato un grafo $G = (V, E)$, è vero che nessun ciclo semplice Γ contiene ogni nodo di V ?

Asimmetria di NP

Asimmetria di NP. Sono richiesti certificati brevi solo per istanze yes.

D. Come classifichiamo UN-SAT e NO-HAMILTON-CYCLE ?

- SAT $\in$ **NP**-completi e SAT $\equiv_P$ UN-SAT.
- HAMILTON-CYCLE $\in$ **NP**-completi e HAMILTON-CYCLE $\equiv_P$ NO-HAMILTON-CYCLE.
- Ma né UN-SAT né NO-HAMILTON-CYCLE sono noti essere in **NP**.

NP e co-NP

NP. Problemi di decisione per i quali esiste un certificatore polinomiale.

Es. SAT, HAMILTON-CYCLE, e COMPOSITES.

Def. Dato un problema di decisione X, il suo complemento $\overline{X}$ è lo stesso problema con le risposte *yes* e *no* rovesciate.

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

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

ignora 0 e 1
(né primi né composti)

co-NP. Complementi di problemi di decisione in **NP**.

Es. UN-SAT, NO-HAMILTON-CYCLE, e PRIMES.

NP = co-NP ?

Fondamentale questione aperta. Si ha **NP** = **co-NP**?

- Le istanze *yes* hanno certificati succinti sse li hanno le istanze *no*?
- Opinione di maggioranza: no.

Teorema. Se **NP** ≠ **co-NP**, allora **P** ≠ **NP**.

Pf idea.

- **P** è chiusa rispetto al complemento.
- Se **P** = **NP**, allora **NP** è chiusa rispetto al complemento.
- In altre parole, allora **NP** = **co-NP**.
- Questa asserzione è logicamente equivalente alla tesi.

Buone caratterizzazioni

Buone caratterizzazioni. [Edmonds 1965] **NP** ∩ **co-NP**.

- Se un problema X sia in **NP** che **co-NP**, allora:
  - per le istanze yes, esiste un certificato succinto
  - per le istanze no, esiste un "certificato di squalifica" succinto
- Fornisce una leva concettuale per ragionare su un problema.

Es. Dato un grafo bipartito, esiste un abbinamento perfetto?

- Se sì, si può esibire l'abbinamento perfetto.
- Se no, si può esibire un insieme di nodi S tale che $|vicini(S)| < |S|$.

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 $k \cdot r(A)$.

Buone caratterizzazioni

We seek a good characterization of the minimum number of independent sets into which the columns of a matrix of M_F can be partitioned. As the criterion of "good" for the characterization we apply the "principle of the absolute supervisor." The good characterization will describe certain information about the matrix which the supervisor can require his assistant to search out along with a minimum partition and which the supervisor can then use with ease to verify with mathematical certainty that the partition is indeed minimum. Having a good characterization does not mean necessarily that there is a good algorithm. The assistant might have to kill himself with work to find the information and the partition.

Buone caratterizzazioni

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

- La dimostrazione del teorema massimo flusso – minimo taglio ha portato ad un risultato più forte: che sia il massimo flusso che il minimo taglio sono problemi in in **P**.
- Talvolta trovare una buona caratterizzazione sembra più semplice che trovare un algoritmo efficiente.

Fondamentale questione aperta. Si ha **P** = **NP** $\cap$ **co-NP**?

- Opinioni variegate.
- Molti esempi di problemi per cui si è trovata una buona caratterizzazione non banale, che solo anni dopo sono stati dimostrati essere in **P**.

Ottimizzazione lineare [Linear programming] è in NP ∩ co-NP

LINEAR-PROGRAMMING. Dati $A \in \Re^{m \times n}$, $b \in \Re^m$, $c \in \Re^n$, e $\alpha \in \Re$, esiste $x \in \Re^n$ tale che $Ax \le b$, $x \ge 0$ e $c^T x \ge \alpha$?

Teorema. [Gale–Kuhn–Tucker 1948] LINEAR-PROGRAMMING ∈ **NP ∩ co-NP.**

Pf sketch. Se (P) e (D) sono non-vuoti, allora max = min.

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

CHAPTER XIX

LINEAR PROGRAMMING AND THE THEORY OF GAMES [1]

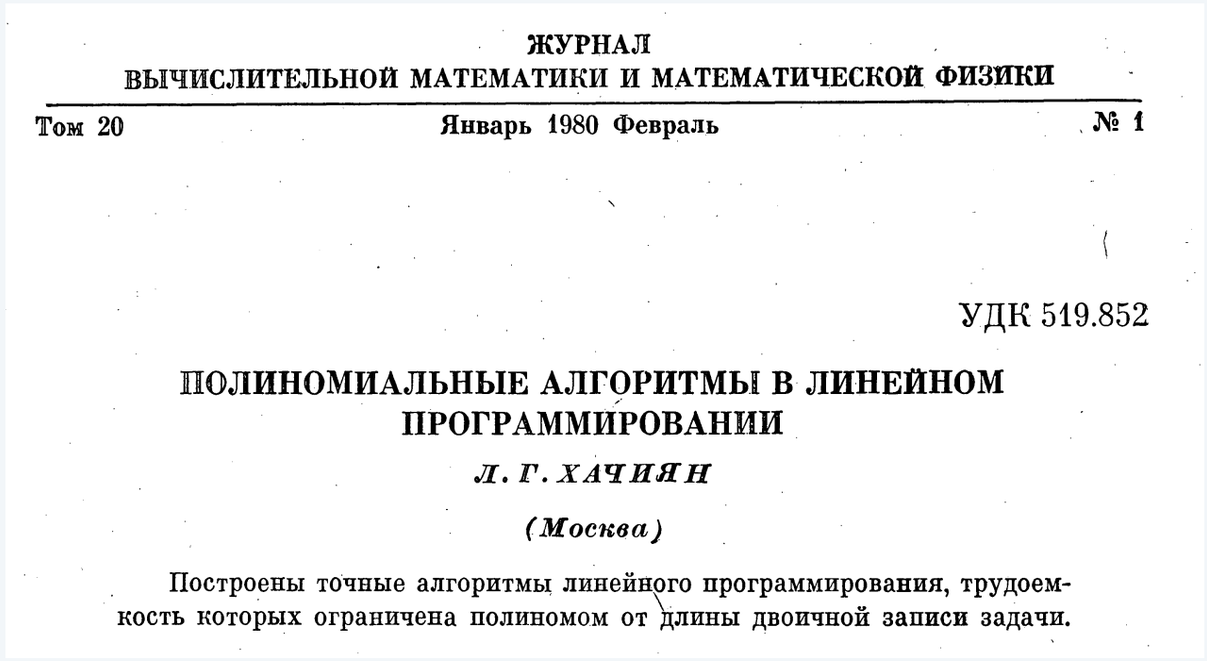
BY DAVID GALE, HAROLD W. KUHN, AND ALBERT W. TUCKER [2]

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.

Ottimizzazione lineare [Linear programming] è in NP ∩ co-NP

LINEAR-PROGRAMMING. Dati $A \in \Re^{m \times n}$, $b \in \Re^m$, $c \in \Re^n$, e $\alpha \in \Re$, esiste $x \in \Re^n$ tale che $Ax \le b$, $x \ge 0$ e $c^T x \ge \alpha$?

Teorema. [Khachiyan 1979] LINEAR-PROGRAMMING ∈ **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.

Verifica di primalità è in NP ∩ co-NP

Teorema. [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: $(p, x, a), q \vdash (p, x, qa)$ 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.*

Verifica di primalità è in NP ∩ co-NP

Teorema. [Pratt 1975] PRIMES ∈ **NP** ∩ **co-NP**.

Pf sketch. Un intero dispari s è primo sse esiste un intero $1 < t < s$ t.c.

$$t^{s-1} \equiv 1 \pmod{s}$$
$$t^{(s-1)/p} \not\equiv 1 \pmod{s}$$
$$\text{for all prime divisors } p \text{ of } s\text{-}1$$

istanza s	437677
certificato t	$17,\ 2^2 \times 3 \times 36473$

anche la fattorizzazione prima di s–1
ha bisogno di certificato (ricorsivo)
per poter asserire che 3 e 36,473 sono primi

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

usa elevamento a quadrato ripetuto

Verifica di primalità è in NP ∩ co-NP

Teorema. [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.

Il problema della fattorizzazione è in NP ∩ co-NP

FACTORIZE. Dato un intero x, trovare la sua fattorizzazione prima.

FACTOR. Dati due interi x e y, è vero che x ha un fattore (non-banale) $< y$?

Teorema. FACTOR $\equiv_P$ FACTORIZE.

Dim.

- $\leq_P$ banale.
- $\geq_P$ ricerca binaria per trovare un fattore; dividi per il fattore e ripeti. ▪

Teorema. FACTOR $\in$ **NP** $\cap$ **co-NP**.

Dim.

- Certificato: un fattore p di x che sia minore di y.
- Certificato di squalifica: la fattorizzazione prima di x (in cui ogni fattore primo è minore di y), insieme ad un certificato di Pratt che ogni numero della fattorizzazione è primo. ▪

La fattorizzazione è in P ?

Domanda fondamentale. FACTOR $\in$ **P** ?

Sfida. Fattorizzare questo numero.

7403756347956171282804679609742957314259318888923128908493623263897276503402826627689199641962511784399589433050212758537011896809828673317327310893090055250511687706329907239638078671008609696253793465056379635

RSA-704
(premio di $30,000 per chi lo fattorizza)

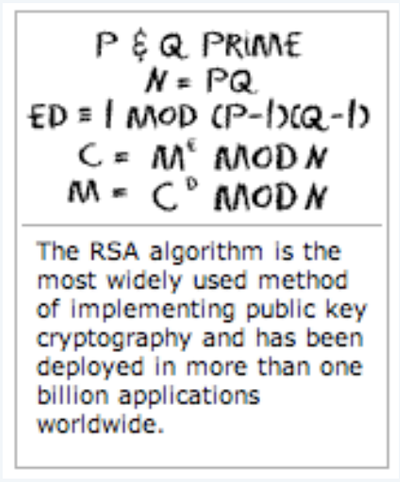
Sfruttare l'intrattabilità

Crittografia moderna.

- Es. Mandare il numero della vostra carta di credito ad Amazon.
- Es. Firmare digitalmente un documento elettronico.
- Libertà di privacy, di parola, di stampa, di associazione politica.

RSA. Basato su una dicotomia tra le complessità di due problemi.

- Per utilizzarlo: generare due primi casuali ad n-bit e moltiplicarli.
- Per romperlo: fattorizzare un intero a $2n$-bit.



algoritmo RSA

RSA venduta per $2.1 miliardi

t-shirt

Fattorizzazione su un computer quantistico

Teorema. [Shor 1994] Si può fattorizzare un intero a n-bit in $O(n^3)$ passi su un "computer quantistico."

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

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. Fattorizzato $15 = 3 \times 5$ (con alta probabilità) su un quantum computer.

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

Domanda fondamentale. Si ha **P** = **BQP** ?

analogo quantistico di **P**
(bounded error quantum polynomial time)

8. INTRATTABILITÀ II

- *P vs. NP*
- *NP-complete*
- *co-NP*
- *NP-arduo*

Una nota terminologica

SIGACT News 12 January 1974

A TERMINOLOGICAL PROPOSAL

D. E. 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.

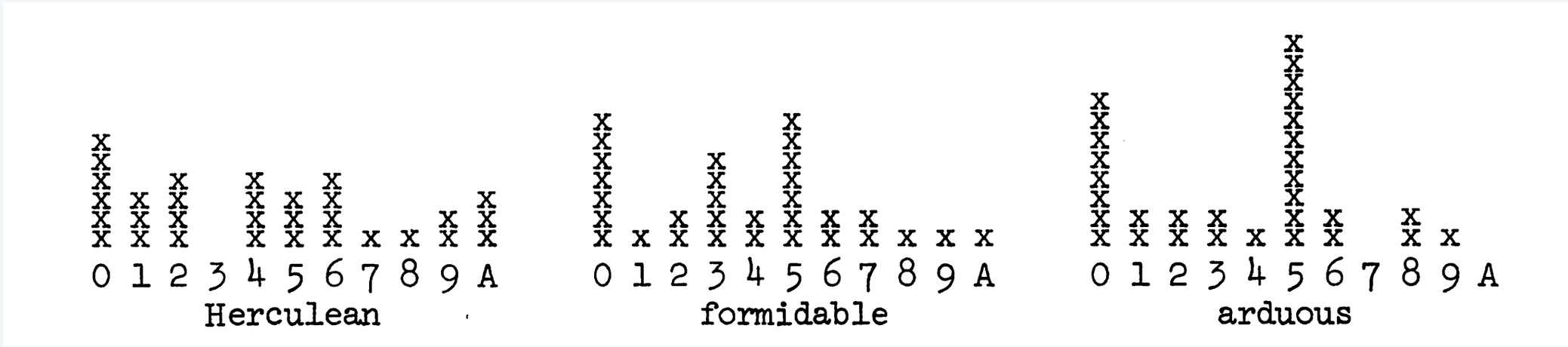
Nota. Il termine x non implica necessariamente che un problema è in **NP**, ma solo che ogni problema in **NP** riduce in tempo-polinomiale ad x.

Una nota terminologica

Suggerimenti originali di Knuth.

- Difficile.
- Tosto.
- Erculeo.
- Formidabile.
- Arduo.

← così comuni che non è chiaro quando vengono usate in senso tecnico



assegna un numero reale tra 0 ed 1 ad ogni scelta

Una nota terminologica: opinione della maggioranza

NP-completo. Un problema in **NP** tale che ogni problema in **NP** è riducibile in tempo polinomiale ad esso.

NP-arduo [NP-hard]. [Bell Labs, Steve Cook, Ron Rivest, Sartaj Sahni]
Un problema tale che ogni problema in **NP** è riducibile in tempo polinomiale ad esso.

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 .