Automi e Linguaggi Formali

Parte 17 – Problemi trattabili e problemi intrattabili

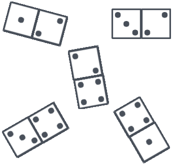
Davide Bresolin
Ultimo aggiornamento: 5 giugno 2023

Università degli Studi di Padova

Giochiamo a Domino[1]

Disponete in fila le tessere del domino che vi sono state consegnate:

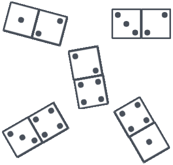
1. in modo da usare tutte le tessere



Giochiamo a Domino[1]

Disponete in fila le tessere del domino che vi sono state consegnate:

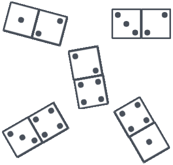
1. in modo da usare tutte le tessere



Giochiamo a Domino[1]

Disponete in fila le tessere del domino che vi sono state consegnate:

1. in modo da usare tutte le tessere



Domanda:

È un problema facile o difficile da risolvere?

Come mostrare che un problema è facile?

Definizione

Un problema è trattabile (facile) se esiste un algoritmo efficiente per risolverlo.

- Gli algoritmi efficienti sono algoritmi con complessità polinomiale:
  - il loro tempo di esecuzione è $O(n^k)$ per qualche costante k.
- Avere complessità polinomiale è una condizione minima per considerare un algoritmo efficiente
- Un algoritmo con complessità più che polinomiale (p.es. esponenziale) è un algoritmo non efficiente perché non è scalabile.

Mostriamo che Domino[1] è trattabile

Obiettivo

Trovare un algoritmo polinomiale per Domino[1]

1. Formulazione del problema in termini di linguaggio
2. Definizione di una Macchina di Turing che lo decide
3. Analisi di complessità della macchina di Turing (o dell'algoritmo)

Un linguaggio e una riduzione

$$D_1 = \{\langle B \rangle \mid B \text{ è un insieme di tessere del domino, ed esiste un allineamento che usa tutte le tessere}\}$$

- Usiamo una riduzione mediante funzione per trovare l'algoritmo polinomiale
- Riduciamo D_1 ad un problema su grafi ...
- ... per il quale sappiamo che esiste un algoritmo polinomiale

Dalle tessere al grafo

Definition (Grafo)

Un grafo (non orientato) G è una coppia (V, E) dove:

- $V = \{v_1, v_2, \ldots, v_n\}$ è un insieme finito e non vuoto di vertici;
- $E \subseteq \{\{u, v\} \mid u, v \in V\}$ è un insieme di coppie non ordinate, ognuna delle quali corrisponde ad un arco del grafo.

Dalle tessere al grafo

Definition (Grafo)

Un grafo (non orientato) G è una coppia (V, E) dove:

- $V = \{v_1, v_2, \dots, v_n\}$ è un insieme finito e non vuoto di vertici;
- $E \subseteq \{\{u, v\} \mid u, v \in V\}$ è un insieme di coppie non ordinate, ognuna delle quali corrisponde ad un arco del grafo.

Grafo del domino

- Vertici: i numeri che si trovano sulle tessere
  - $V = \{$⚀, ⚁, ⚂, ⚃$\}$
- Archi: le tessere del domino
  - $E = \{$⚂⚃, ⚀⚁, ⚃⚁, ⚃⚀, ⚂⚁$\}$

Domino[1] è un problema su grafi!

- **Cammino Euleriano**: percorso in un grafo che attraversa tutti gli archi una sola volta

Il problema del Cammino Euleriano

$EULER = \{\langle G \rangle \mid G$ è un grafo che possiede un cammino Euleriano$\}$

- *EULER* è un problema classico di teoria dei grafi
- Esistono algoritmi polinomiali per risolverlo

Algoritmo di Fleury

Su input $\langle G \rangle$, con G grafo non orientato:

1. Scegliere un vertice con grado dispari (un vertice qualsiasi se tutti pari)
2. Scegliere un arco tale che sua cancellazione non sconnetta il grafo. Se tale arco non esiste, RIFIUTA
3. Passare al vertice nell'altra estremità dell'arco scelto
4. Cancellare l'arco dal grafo
5. Ripetere da 2 finché non sono stati eliminati tutti gli archi
6. Se tutti gli archi sono stati eliminati, ACCETTA

Complessità

Su un grafo con n archi, l'algoritmo di Fleury impiega tempo $O(n^2)$

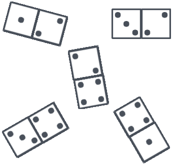
Complessità di Domino[1]

- L'algoritmo di Fleury risolve *EULER* in tempo polinomiale
- La riduzione ci dice che $D_1 \leq_m EULER$
- Quanto tempo serve per risolvere il problema D_1?

Giochiamo a domino[2]

Disponete in fila le tessere del domino che vi sono state consegnate:

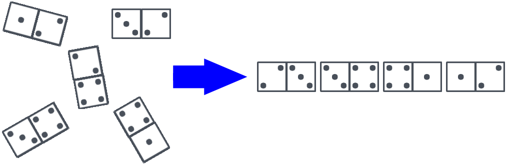
2. in modo che ogni numero compaia esattamente due volte (potete usare meno tessere di quelle che avete).



Giochiamo a domino[2]

Disponete in fila le tessere del domino che vi sono state consegnate:

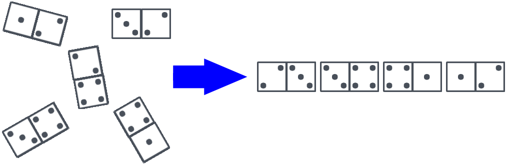
2. in modo che ogni numero compaia esattamente due volte (potete usare meno tessere di quelle che avete).



Giochiamo a domino[2]

Disponete in fila le tessere del domino che vi sono state consegnate:

2. in modo che ogni numero compaia esattamente due volte (potete usare meno tessere di quelle che avete).



Domanda:

È un problema facile o difficile da risolvere?

Una riduzione in senso opposto!

UNIVERSITÀ DEGLI STUDI DI PADOVA

$D_2 = \{\langle B \rangle \mid B$ insieme di tessere del domino, ed esiste allineamento dove ogni numero compare due volte$\}$

- **Circuito Hamiltoniano**: ciclo nel grafo che attraversa tutti i vertici una sola volta

Il problema del Circuito Hamiltoniano

$HAMILTON = \{\langle G \rangle \mid G$ è un grafo con un circuito Hamiltoniano$\}$

- Come facciamo a dimostrare che $HAMILTON \leq_m D_2$?

HAMILTON è un problema difficile!

- Il problema del circuito Hamiltoniano è un problema classico di teoria dei grafi
- Un algoritmo polinomiale per risolverlo non è mai stato trovato
- Se qualcuno mi dà una possibile soluzione, è facile verificare se è corretta

Problemi trattabili e problemi intrattabili

- I problemi per i quali esiste un algoritmo polinomiale vengono considerati trattabili
- quelli che richiedono un algoritmo più che polinomiale sono detti intrattabili.
- Sappiamo che ci sono problemi che non possono essere risolti da nessun algoritmo:
  - "Halting Problem" di Turing
- Ci sono problemi che richiedono un tempo esponenziale:
  - il gioco della Torre di Hanoi

Problemi trattabili e problemi intrattabili

- I problemi per i quali esiste un algoritmo polinomiale vengono considerati trattabili
- quelli che richiedono un algoritmo più che polinomiale sono detti intrattabili.
- Sappiamo che ci sono problemi che non possono essere risolti da nessun algoritmo:
  - "Halting Problem" di Turing
- Ci sono problemi che richiedono un tempo esponenziale:
  - il gioco della Torre di Hanoi

Stabilire con precisione qual'è il confine tra problemi trattabili ed intrattabili è piuttosto difficile

Facili da risolvere

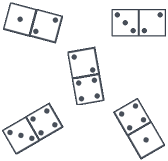
Facili da verificare

	P
Facili da risolvere	✓
Facili da verificare	✓
Esempi	Domino[1], Euler, Path, Relprime, ...

	P	NP
Facili da risolvere	✓	?
Facili da verificare	✓	✓
Esempi	Domino[1], Euler, Path, Relprime, ...	Domino[2], Hamilton, Sudoku, Protein folding, Crittografia, ...

Domino[1] è in P

Istanza
del problema

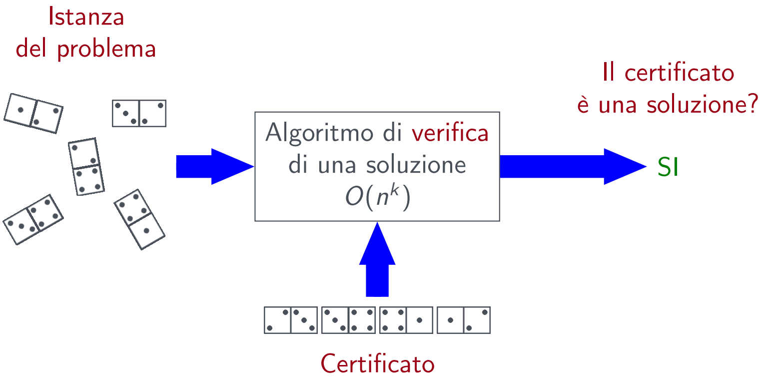


Macchina di Turing
che decide
in tempo $O(n^k)$

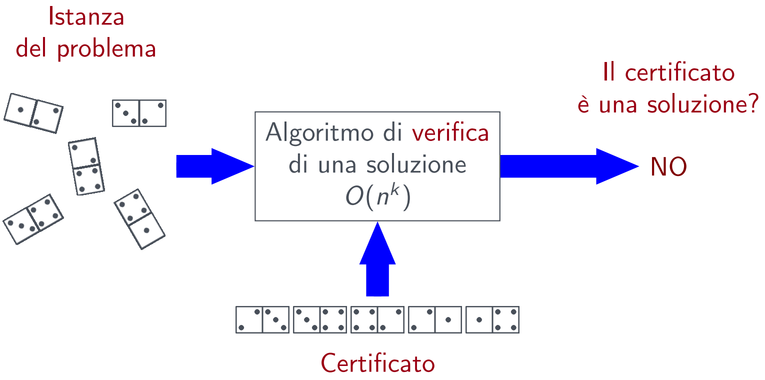
Esiste una soluzione?

SI / NO

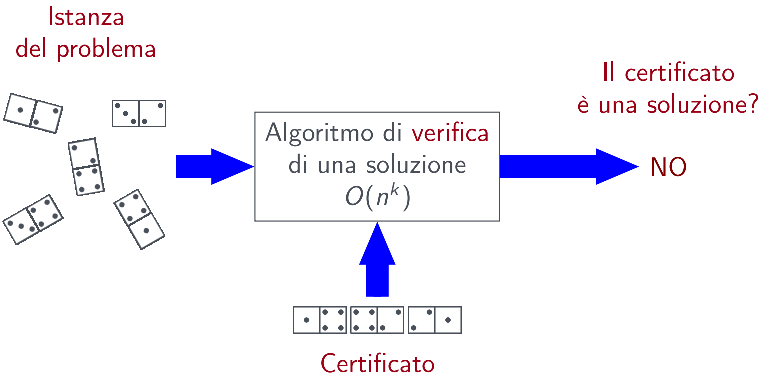
Domino[2] è in NP



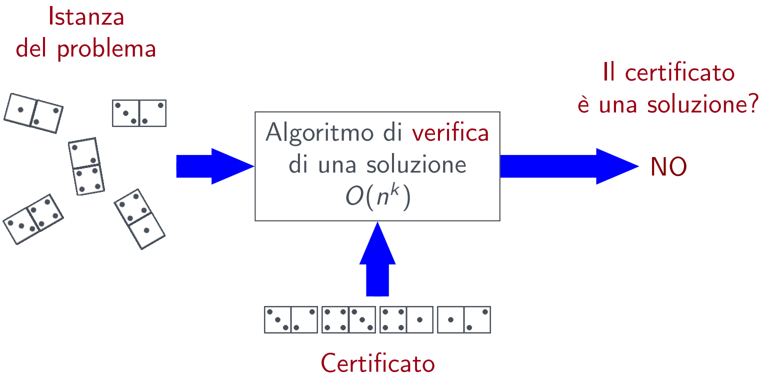
Domino[2] è in NP



Domino[2] è in NP



Domino[2] è in NP



Definition

Un verificatore per un linguaggio A è un algoritmo V tale che

$$A = \{w \mid V \text{ accetta } \langle w, c\rangle \text{ per qualche stringa } c\}$$

- il verificatore usa ulteriori informazioni per stabilire se w appartiene al linguaggio
- questa informazione è il certificato c

Problemi P ed NP

- **P** è la classe dei linguaggi che possono essere decisi da una macchina di Turing deterministica che impiega tempo polinomiale.

Problemi P ed NP

- **P** è la classe dei linguaggi che possono essere decisi da una macchina di Turing deterministica che impiega tempo polinomiale.

- **NP** è la classe dei linguaggi che ammettono un verificatore che impiega tempo polinomiale.

Problemi P ed NP

- **P** è la classe dei linguaggi che possono essere decisi da una macchina di Turing deterministica che impiega tempo polinomiale.

- **NP** è la classe dei linguaggi che ammettono un verificatore che impiega tempo polinomiale.
- **Equivalente:** è la classe dei linguaggi che possono essere decisi da una macchina di Turing non deterministica che impiega tempo polinomiale.

Due problemi in P ...

Raggiungibilità in un grafo

$PATH = \{\langle G, s, t \rangle \mid G \text{ grafo che contiene un cammino da } s \text{ a } t\}$

Numeri relativamente primi

$RELPRIME = \{\langle x, y \rangle \mid 1 \text{ è il massimo comun divisore di } x \text{ e } y\}$

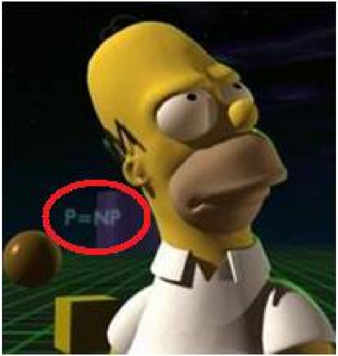
Problema del circuito Hamiltoniano

$$HAMILTON = \{\langle G\rangle \mid G \text{ è un grafo con un circuito Hamiltoniano}\}$$

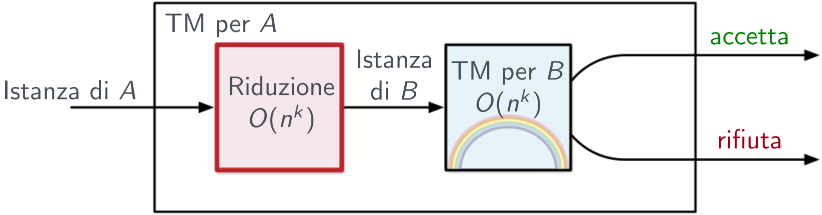
Numeri composti

$$COMPOSITES = \{\langle x\rangle \mid x = pq, \text{ per gli interi } p, q > 1\}$$

P = NP ?



Se $A \leq_P B$, e $B \in P$, allora $A \in P$:



Sommario

- una formula Booleana come

$$(a \vee b \vee c \vee \overline{d}) \leftrightarrow ((b \wedge \overline{c}) \vee \overline{(\overline{a} \rightarrow d)} \vee (c \neq a \wedge b)),$$

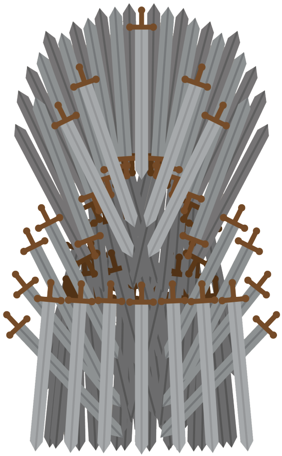
- è soddisfacibile se è possibile assegnare dei valori booleani (Vero/Falso) alle variabili $a, b, c, \ldots,$ in modo che il valore di verità della formula sia Vero

$$\textbf{SAT} = \{\langle \varphi \rangle \mid \varphi \text{ è una formula booleana soddisfacibile}\}$$

SAT è un problema NP

1. Trovare un certificato di appartenenza:
   $\Rightarrow$ l'assegnamento di verità alle variabili $a, b, c, \ldots$

2. Trovare un algoritmo polinomiale per verificare il certificato.



SAT è Re tra i problemi in NP:

- Ogni problema NP può essere trasformato in una istanza di **SAT** in tempo polinomiale
- **SAT** può essere usato per risolvere tutti i problemi NP
- chi scopre un algoritmo polinomiale per **SAT** sa risolvere tutti i problemi NP in tempo polinomiale!

Teorema di Cook-Levin

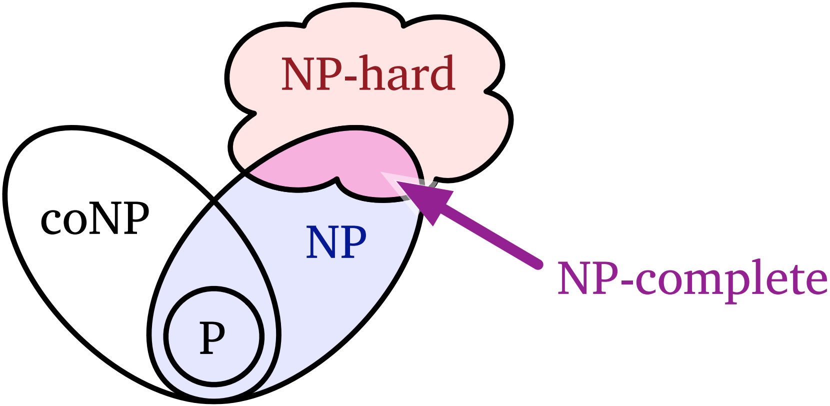
Theorem (Cook e Levin, 1973)

L'esistenza di una macchina di Turing polinomiale per risolvere ***SAT*** *implica che $P = NP$.*

Problemi NP-hard

- Un problema è NP-hard se l'esistenza di un algoritmo polinomiale per risolverlo implica l'esistenza di un algoritmo polinomiale per ogni problema in NP.
- Se siamo in grado di risolvere un problema NP-hard in modo efficiente, allora possiamo risolvere in modo efficiente ogni problema di cui possiamo verificare facilmente una soluzione, usando la soluzione del problema NP-hard come sottoprocedura.
- Un problema è NP-completo se è sia NP-hard che appartenente alla classe NP (o "NP-easy").
  - Esempio: **CircuitSAT**!

Come pensiamo sia fatto il mondo



Perché studiare la NP-completezza?

- Progettare ed implementare algoritmi richiede la conoscenza dei principi di base della teoria della complessità
- Stabilire che un problema è NP-completo costituisce una prova piuttosto forte della sua intrattabilità
- Conviene cercare di risolverlo con un approccio diverso . . .
  - identificare un caso particolare trattabile
  - cercare una soluzione approssimata
- . . . invece di cercare un algoritmo efficiente per il caso generale che probabilmente non esiste nemmeno

Dimostrare che un problema è NP-completo

Ogni dimostrazione di NP-completezza di compone di due parti:

1. dimostrare che il problema appartiene alla classe NP;
2. dimostrare che il problema è NP-hard.

Dimostrare che un problema è NP-completo

Ogni dimostrazione di NP-completezza di compone di due parti:

1. dimostrare che il problema appartiene alla classe NP;
2. dimostrare che il problema è NP-hard.

- Dimostrare che un problema è in NP vuol dire dimostrare l'esistenza di un verificatore polinomiale.

Dimostrare che un problema è NP-completo

Ogni dimostrazione di NP-completezza di compone di due parti:

1. dimostrare che il problema appartiene alla classe NP;
2. dimostrare che il problema è NP-hard.

- Dimostrare che un problema è in NP vuol dire dimostrare l'esistenza di un verificatore polinomiale.
- Le tecniche che si usano per dimostrare che un problema è NP-hard sono fondamentalmente diverse.

Riduzioni tra problemi diversi

- Per dimostrare che un certo problema è NP-hard si procede tipicamente con una dimostrazione per riduzione polinomiale

Riduzioni tra problemi diversi

- Per dimostrare che un certo problema è NP-hard si procede tipicamente con una dimostrazione per riduzione polinomiale

Per dimostrare che B è NP-hard dobbiamo ridurre un problema NP-hard a B.

Riduzioni tra problemi diversi

- Per dimostrare che un certo problema è NP-hard si procede tipicamente con una dimostrazione per riduzione polinomiale

Per dimostrare che B è NP-hard dobbiamo ridurre un problema NP-hard a B.

- Abbiamo bisogno di un problema NP-hard da cui partire: SAT

Schema di riduzione polinomiale

Università degli Studi di Padova

Per dimostrare che un problema B è NP-hard:

1. Scegli un problema A che sai essere NP-hard.
2. Descrivi una riduzione polinomiale da A a B:
   - data un'istanza di A, trasformala in un'istanza di B,
   - con una funzione che opera in tempo polinomiale.
3. Dimostra che la riduzione è corretta:
   - Dimostra che la funzione trasforma istanze "buone" di A in istanze "buone" di B.
   - Dimostra che la funzione trasforma istanze "cattive" di A in istanze "cattive" di B.
     Equivalente: se la tua funzione produce un'istanza "buona" di B, allora era partita da un'istanza "buona" di A.
4. Mostra che la funzione impiega tempo polinomiale.

Una versione ristretta di soddisfacibilità

Università degli Studi di Padova

- Intendiamo dimostrare l'NP-completezza di un'ampia gamma di problemi
- Dovremmo procedere per riduzione polinomiale da **SAT** al problema in esame
- Esiste però un importante problema "intermedio", detto **3SAT**, molto più facile da ridurre ai problemi tipici rispetto a **SAT**:
  - anche **3SAT** è un problema di soddisfacibilità di formule booleane
  - **3SAT** però richiede che le formule siano di una forma ben precisa, formate cioè da congiunzione logica di clausole ognuna delle quali è disgiunzione logica di tre variabili (anche negate)

Forme normali di formule booleane

- Un letterale è una variabile o una variabile negata, ad es. x, $\overline{y}$
- Una clausola è una disgiunzione logica (OR) di uno o più letterali, ad es. x, $x \vee \overline{y}$
- Una formule booleana si dice in forma normale congiuntiva (CNF), se è la congiunzione logica (AND) di una o più clausole:
  - $(x \vee y) \wedge (\overline{x} \vee z)$, e $x \wedge y$ sono in CNF
  - mentre $(x \vee y \vee z) \wedge (\overline{y} \vee \overline{z}) \vee (x \vee y \wedge z)$ non è in CNF
- Un formula si dice in forma normale 3-congiuntiva (3-CNF) se è composta di clausole che hanno esattamente 3 letterali distinti

Il problema 3SAT

$\textbf{3SAT} = \{\langle\varphi\rangle \mid \varphi \text{ è una formula booleana in 3-CNF soddisfacibile}\}$

Il problema 3SAT

3SAT $= \{\langle\varphi\rangle \mid \varphi$ è una formula booleana in 3-CNF soddisfacibile$\}$

- **3SAT** è in NP:

Il problema 3SAT

Università degli Studi di Padova

3SAT $= \{\langle\varphi\rangle \mid \varphi$ è una formula booleana in 3-CNF soddisfacibile$\}$

- **3SAT** è in NP:
  il certificato è l'assegnamento di verità alle variabili $a, b, c, \ldots$

Il problema 3SAT

3SAT $= \{\langle\varphi\rangle \mid \varphi$ è una formula booleana in 3-CNF soddisfacibile$\}$

- **3SAT** è in NP:
  il certificato è l'assegnamento di verità alle variabili $a, b, c, \ldots$
- **3SAT** è NP-hard:

Il problema 3SAT

3SAT $= \{\langle\varphi\rangle \mid \varphi$ è una formula booleana in 3-CNF soddisfacibile$\}$

- **3SAT** è in NP:
  il certificato è l'assegnamento di verità alle variabili $a, b, c, \ldots$
- **3SAT** è NP-hard:
  dimostrazione per riduzione di **SAT** a **3SAT**

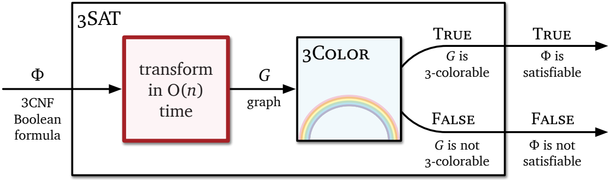
Sommario

Il problema della colorazione

- Sia $G = (V, E)$ un grafo non orientato.
- "Colorare" G significa assegnare etichette ("colori"), ai vertici del grafo in modo tale che nessuna coppia di vertici collegati da un arco condivida lo stesso colore.
- Chiamiamo 3-**Color** il problema di trovare, se esiste, una colorazione di un grafo che usa 3 colori diversi.

3-Color è NP-hard!

Dimostriamo che 3-**Color** è NP-hard con una riduzione da **3SAT**



Sommario

Perché studiare la NP-completezza?

- Progettare ed implementare algoritmi richiede la conoscenza dei principi di base della teoria della complessità
- Stabilire che un problema è NP-completo costituisce una prova piuttosto forte della sua intrattabilità
- Conviene cercare di risolverlo con un approccio diverso . . .
  - identificare un caso particolare trattabile
  - cercare una soluzione approssimata
- . . . invece di cercare un algoritmo efficiente per il caso generale che probabilmente non esiste nemmeno

Altri esempi di dualità

Problemi NP-Completi

Problemi in P

Altri esempi di dualità

Problemi NP-Completi

- circuito Hamiltoniano
  Trovare un ciclo che visita ogni vertice esattamente una volta

Problemi in P

Altri esempi di dualità

Problemi NP-Completi

- circuito Hamiltoniano
  Trovare un ciclo che visita ogni vertice esattamente una volta

Problemi in P

- Circuito Euleriano
  Trovare un ciclo che visita ogni arco esattamente una volta

Altri esempi di dualità

Problemi NP-Completi

- circuito Hamiltoniano
  Trovare un ciclo che visita ogni vertice esattamente una volta
- Copertura di vertici
  Trovare il minimo sottoinsieme S di vertici tali che ogni arco ha almeno un'estremità in S

Problemi in P

- Circuito Euleriano
  Trovare un ciclo che visita ogni arco esattamente una volta

Altri esempi di dualità

Problemi NP-Completi

- circuito Hamiltoniano
  Trovare un ciclo che visita ogni vertice esattamente una volta
- Copertura di vertici
  Trovare il minimo sottoinsieme S di vertici tali che ogni arco ha almeno un'estremità in S

Problemi in P

- Circuito Euleriano
  Trovare un ciclo che visita ogni arco esattamente una volta
- Copertura di archi
  Trovare il minimo sottoinsieme M di archi tali che ogni vertice è adiacente ad un arco in M

Altri esempi di dualità

Problemi NP-Completi

- circuito Hamiltoniano
  Trovare un ciclo che visita ogni vertice esattamente una volta
- Copertura di vertici
  Trovare il minimo sottoinsieme S di vertici tali che ogni arco ha almeno un'estremità in S
- 3-colorazione di un grafo
  Trovare un modo per colorare i vertici di un grafo con tre colori tale che vertici adiacenti sono di colore diverso

Problemi in P

- Circuito Euleriano
  Trovare un ciclo che visita ogni arco esattamente una volta
- Copertura di archi
  Trovare il minimo sottoinsieme M di archi tali che ogni vertice è adiacente ad un arco in M

Altri esempi di dualità

Problemi NP-Completi

- circuito Hamiltoniano
  Trovare un ciclo che visita ogni vertice esattamente una volta
- Copertura di vertici
  Trovare il minimo sottoinsieme S di vertici tali che ogni arco ha almeno un'estremità in S
- 3-colorazione di un grafo
  Trovare un modo per colorare i vertici di un grafo con tre colori tale che vertici adiacenti sono di colore diverso

Problemi in P

- Circuito Euleriano
  Trovare un ciclo che visita ogni arco esattamente una volta
- Copertura di archi
  Trovare il minimo sottoinsieme M di archi tali che ogni vertice è adiacente ad un arco in M
- 2-colorazione di un grafo
  Trovare un modo per colorare i vertici di un grafo con due colori tale che vertici adiacenti sono di colore diverso

Come scegliere il problema giusto

- Se il problema richiede di assegnare bit agli oggetti: **SAT**
- Se il problema richiede di assegnare etichette agli oggetti prese un piccolo insieme, o di partizionare gli oggetti in un numero costante di sottoinsiemi: **3Color** o k**Color**
- Se il problema richiede di organizzare un insieme di oggetti in un ordine particolare: **Circuito Hamiltoninano**
- Se il problema richiede di trovare un piccolo sottoinsieme che soddisfi alcuni vincoli: **MinVertexCover**
- Se il problema richiede di trovare un sottoinsieme grande che soddisfi alcuni vincoli: **MaxIndependentSet**
- Se il numero 3 appare in modo naturale nel problema, provare **3SAT** o **3Color** (No, questo non è uno scherzo.)
- Se tutto il resto fallisce, prova **3SAT** o anche **SAT**!