# Sommario

- **Caratterizzazione alternativa di NP: il verificatore polinomiale**
- **esempi di problemi in NP**

**"I conjecture that there is no good algorithm for the traveling salesman problem. My reasons are the same as for any mathematical conjecture:**
**(1) It is a legitimate mathematical possibility, and**
**(2) I do not know."**

**Jack Edmonds, 1966**

# Nuova definizione di NP

**Con l'introduzione del concetto di verificatore diamo una nuova definizione di NP, come la classe dei problemi per i quali si può verificare se una potenziale soluzione lo è, in tempo polinomiale.**

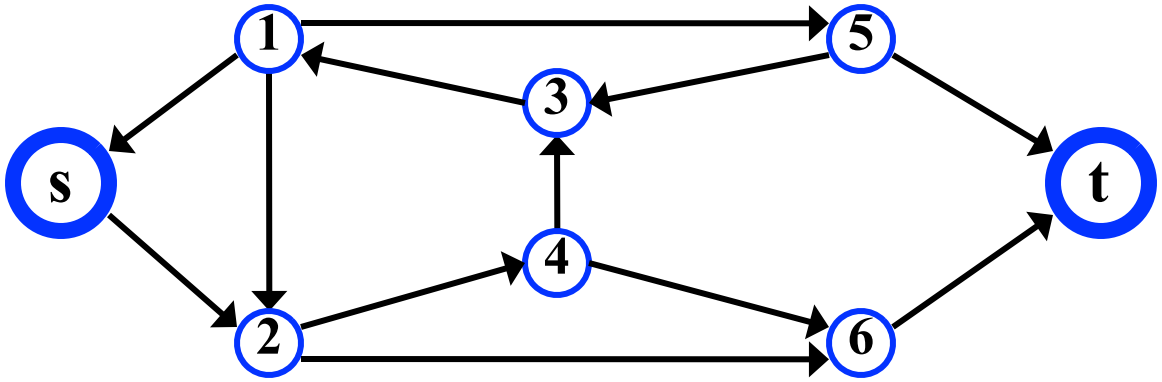
**L'interesse per questa nuova definizione è duplice:**
**1. non fa riferimento al concetto di determinismo**

**2. supporta la tesi che P sia contenuto strettamente in NP perché questo riflette l'intuizione che trovare una soluzione è più difficile che verificare una soluzione.**

# HAMPATH: il problema

**HAMPATH = {<G,s,t> | G è un grafo diretto con un cammino hamiltoniano da s a t (cioè un cammino che attraversa tutti i nodi di G una e una sola volta)}**

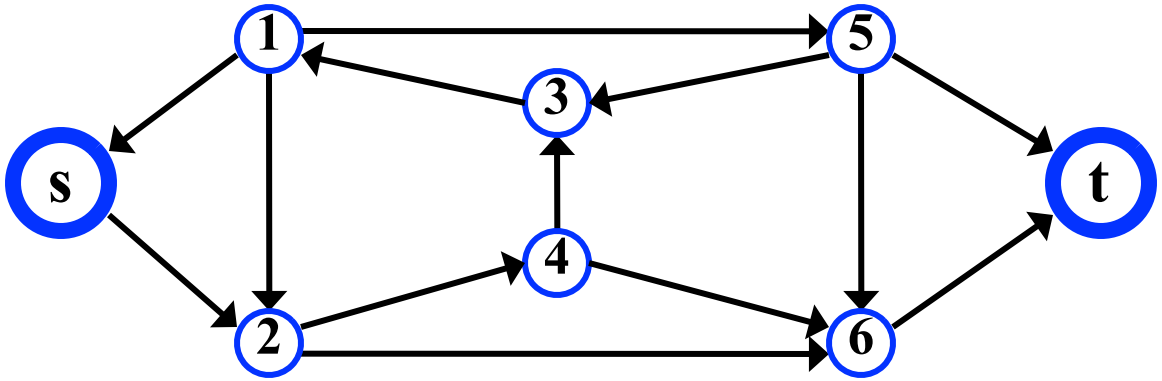
**Istanza 1**




**$\in$ HAMPATH?**

**NO**

**Istanza 2**




**$\in$ HAMPATH?**

**SI**

# HAMPATH: un verificatore

**HAMPATH = {<G,s,t> | G è un grafo diretto con un cammino hamiltoniano da s a t (cioè un cammino che attraversa tutti i nodi di G una e una sola volta)}**

**Osserviamo che se p = $v_1,...,v_n$ è una sequenza di n vertici in un grafo diretto G, con n vertici, si può controllare che si tratta di un cammino hamiltoniano da s a t verificando che:**

1. **$v_1$= s, $v_n$ = t e che i vertici in p siano diversi tra loro**
2. **($v_i$,$v_{i+1}$ ) è un arco in G, per i = 1,...,n-1**

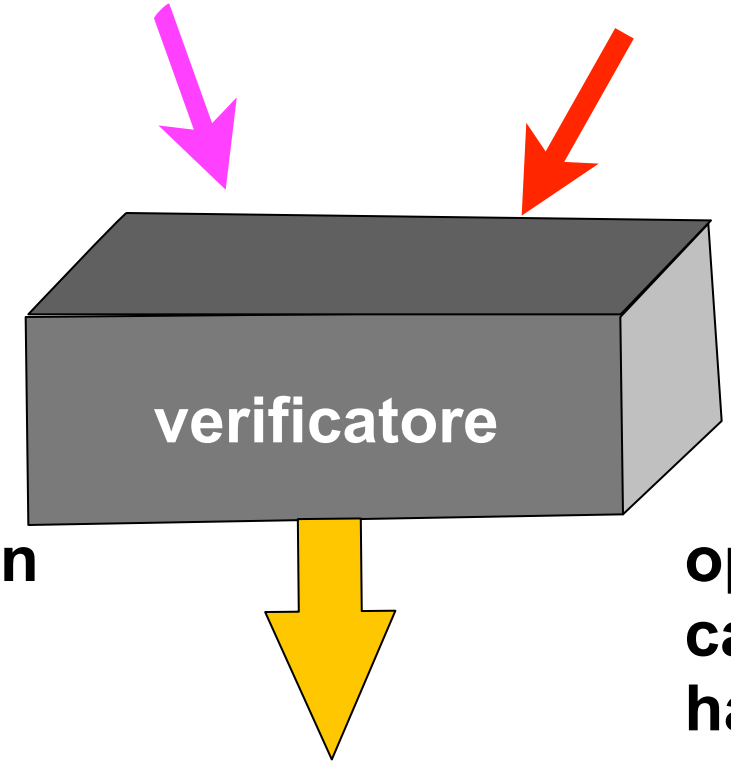
**Questi controlli sono semplici computazionalmente, cioè eseguibili in tempo polinomiale**

# L’esempio del ciclo hamiltoniano

**HAMPATH = {<G,s,t> | G è un grafo diretto con un cammino hamiltoniano da s a t (cioè un cammino che attraversa tutti i nodi di G una e una sola volta)}**

**input del problema: <G,s,t>**

**certificato: p= $v_1,\ldots,v_n$**




**NO: p non è un cammino hamiltoniano**

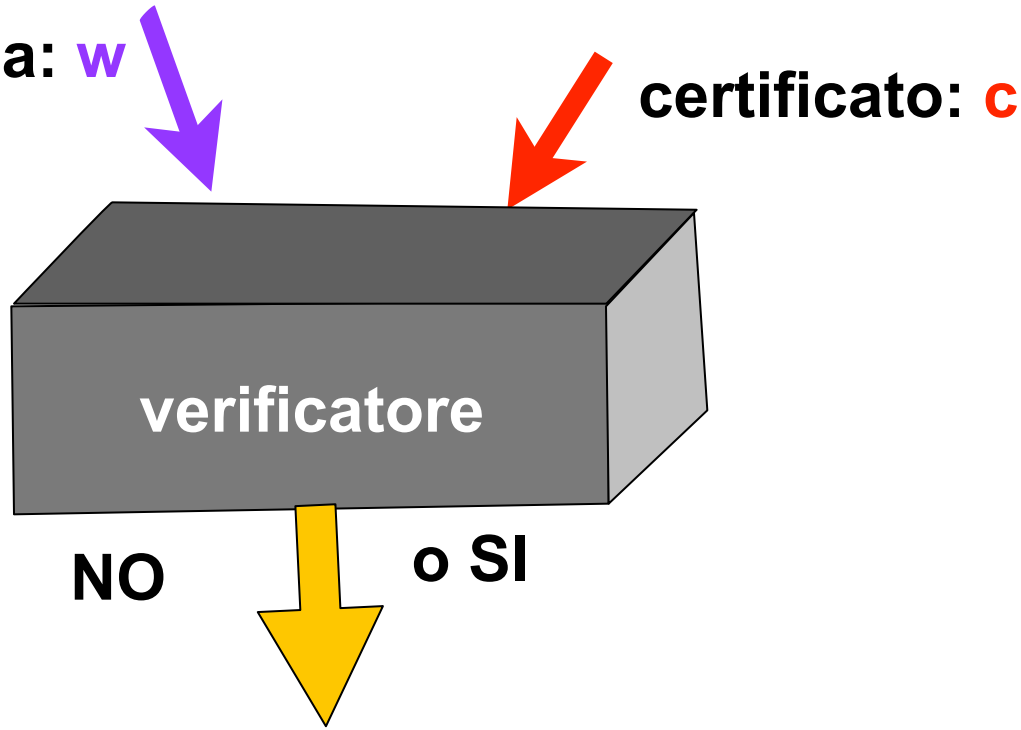
**oppure SI: p è un cammino hamiltoniano**

# Il verificatore

**Un verificatore per A è un algoritmo V (una TM) tale che**

**A = { w | V accetta < w,c> per qualche stringa c}**

**La stringa c è detta il certificato o la prova.**
**Un verificatore V è detto polinomiale se accetta una stringa w in tempo polinomiale nella lunghezza di w.**



# Il verificatore

**Un linguaggio A è polinomialmente verificabile se ha un verificatore polinomiale.**

**Nota: il certificato c deve essere di lunghezza polinomiale nella lunghezza di w perchè il verificatore sia polinomiale, deve infatti almeno leggerlo.**

# Un verificatore per HAMPATH

**Un verificatore per**

**HAMPATH = {<G,s,t> | G è un grafo diretto con un cammino hamiltoniano da s a t (cioè un cammino che attraversa tutti i nodi di G una e una sola volta)}**

**è un algoritmo $V_H$ tale che**

**HAMPATH = {<G,s,t> | $V_H$ accetta <<G,s,t>,<$v_1$,...,$v_n$>>, $v_i$ è un vertice di G, i=1,...,n}**

**Quindi $V_H$ è un algoritmo che effettua i controlli che abbiamo visto per decidere se $v_1$,...,$v_n$ è un cammino hamiltoniano o no. Il verificatore $V_H$ è polinomiale.**

**Non sappiamo se c'è un algoritmo polinomiale che decide HAMPATH.**

# Un verificatore per ¬HAMPATH

**Un verificatore per**

**¬HAMPATH = {<G,s,t> | G è un grafo diretto senza un cammino hamiltoniano da s a t (cioè un cammino attraversa tutti i nodi di G una e una sola volta)}**

**Non conosciamo un verificatore per $V_{\neg H}$ che lavori in tempo polinomiale.**

**Sappiamo solo che dando come certificato la sequenza di tutte le permutazioni di n-2 vertici, possiamo verificare che nessuna rappresenti un cammino da s a t. Ma questo è un verificatore esponenziale**

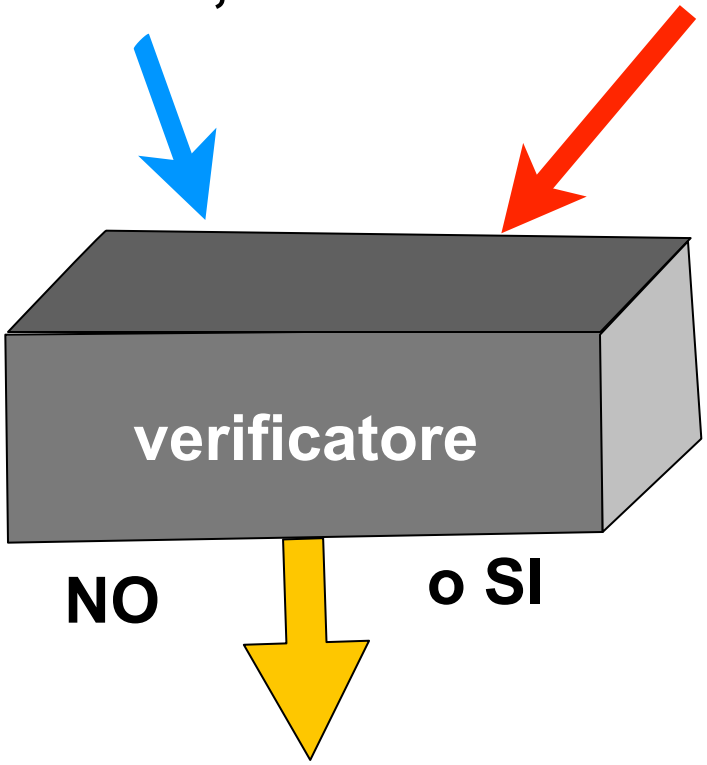
# Un verificatore per SUBSET-SUM

**SUBSET-SUM = { <S,t> | S è un insieme di interi dotato di un sottoinsieme a somma t}**

**Es.: S={10,-3,5,-1,-8,4}, t=0 è in SUBSET-SUM perchè Y ={-3,-1,4} soddisfa la condizione.**

**input del problema: <S,t>**

**certificato: <Y>**




**Il verificatore deve controllare che Y sia un sottoinsieme di S la cui somma è t.**

# Un verificatore per SUBSET-SUM

**Un verificatore per SUBSET-SUM è un algoritmo V tale che**

**SUBSET-SUM = { <S,t> | V decide < <S,t>,Y >}**

**V deve controllare che Y sia un sottoinsieme di S la cui somma è t.**

**Questo controllo si può fare in tempo polinomiale, mentre non sappiamo se c'è un algoritmo polinomiale per decidere SUBSET-SUM.**

# NP e verificatori polinomiali

**Teorema. Un linguaggio L è deciso da una NMT in tempo polinomiale $\Leftrightarrow$ L ha un verificatore polinomiale.**

**Così potremo dire che NP è l'insieme dei problemi verificabili in tempo polinomiale, cioè dei problemi per i quali si può decidere in tempo polinomiale se un dato elemento è una soluzione del problema.**

**Scompare dalla definizione ogni riferimento al nondeterminismo!**

# NP e verificatori polinomiali

**Prova:**

**Un linguaggio L è deciso da una NMT in tempo polinomiale $\Rightarrow$ L ha un verificatore polinomiale.**

**Sia L un linguaggio per il quale esiste una NTM T che lo accetta in tempo polinomiale, limitato da $n^k$.**

**Sia r il massimo grado di non determinismo in T e sia c una stringa su {1,...,r} di lunghezza $n^k$**

**Considera il seguente verificatore:**

**$V_T$= input <w,c>**

1. **esegui T su w seguendo il cammino computazionale descritto da c**
2. **se il cammino è di accettazione, accetta altrimenti rifiuta**

## Un linguaggio L è deciso da una NMT in tempo polinomiale $\Rightarrow$ L ha un verificatore polinomiale.

La costruzione del verificatore è corretta perchè

$$L(T) = \{w \mid V_T \text{ accetta } \langle w, c \rangle \text{ per qualche stringa } c\}$$

infatti per ogni parola w accettata dalla NTM T esiste un cammino di accettazione per w, dunque esiste un certificato c per cui $V_T$ accetta. Mentre se una parola è rifiutata da T ogni certificato porta al rifiuto in $V_T$.

Inoltre la complessità è polinomiale perchè la sequenza c è di lunghezza polinomiale.

# NP e verificatori polinomiali

**Un linguaggio L ha un verificatore polinomiale $\Longrightarrow$**

**L è deciso da una NMT in tempo polinomiale**

**Se V è un verificatore polinomiale per L vuol dire che L = { w | V accetta <w,c> per qualche stringa c}.**
**V è una TM di complessità $n^k$, dove |w|=n.**
**Quindi possiamo costruire una NTM $T_V$ per L.**

**$T_V$ = su input w**
**scegli non deterministicamente una stringa c di lunghezza $n^k$**
**esegui V sull'input <w,c> se V accetta, accetta e se V rifiuta, rifiuta**

## Un linguaggio L ha un verificatore polinomiale $\Rightarrow$ L è deciso da una NMT in tempo polinomiale

**Si tratta di fa vedere che**

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

**è accettato dalla NTM $T_V$.**

**Per ogni parola w tale che <w,c> è accettata da V esiste un cammino di accettazione di $T_V$ per w, visto che li "tenta" tutti. D'altro canto se una parola non ha un certificato di accettazione in V, tutti i cammini di $T_V$ su w saranno di rifiuto e quindi la parola non è accettata.**

# NP

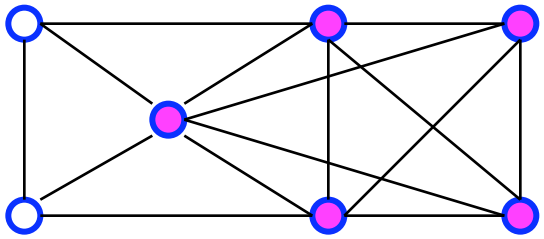
**In conclusione NP si può definire anche come la classe dei problemi che ammettono un verificatore polinomiale**

# Problemi in NP - CLIQUE

**Dato un grafo non diretto un clique del grafo è un sottografo indotto completo.**

**Un k-clique è un clique di k nodi**

**CLIQUE = {<G,k> | G è un grafo non diretto con un k-clique} è un linguaggio in NP**



# Prova con verificatore

**CLIQUE è in NP.**

**Dato un grafo G=(V,E) un certificato per il k-clique è semplicemente un sottoinsieme c di k vertici di G.**

**Verificatore per CLIQUE:**

**Input <<G,k>,c>**

1. **verifica che c sia un sottoinsieme di k elementi di V**
2. **verifica se tutti i vertici in c sono connessi tra loro.**
3. **se entrambi i tests hanno avuto successo, accetta altrimenti rifiuta.**

# Prova con una NTM

**CLIQUE è in NP.**

**Dato un grafo G=(V,E) un certificato per il k-clique è semplicemente un sottoinsieme c di k vertici di G.**

**NTM per CLIQUE:**

**Input <G,k>**

1. **nondeterministicamente genera un sottoinsieme c di k elementi di V**
2. **verifica se tutti i vertici in c sono connessi tra loro.**
3. **se sì, accetta altrimenti rifiuta.**

# Problemi in NP - SAT

**SAT={$\phi$ | $\phi$ è una formula booleana soddisfacibile}**

**Es. (A $\vee$ ¬B) $\wedge$ (¬C $\vee$ ¬B) $\wedge$ (C $\vee$ A$\vee$ B)**

# Prova con verificatore

**Verificatore per SAT:**

**Input <<φ>,c>**

1. **verifica che c sia un assegnamento di valori di verità per tutte le variabili di φ**
2. **verifica se φ risulta vera e in tal caso accetta, altrimenti rifiuta.**

# Prova con una NTM

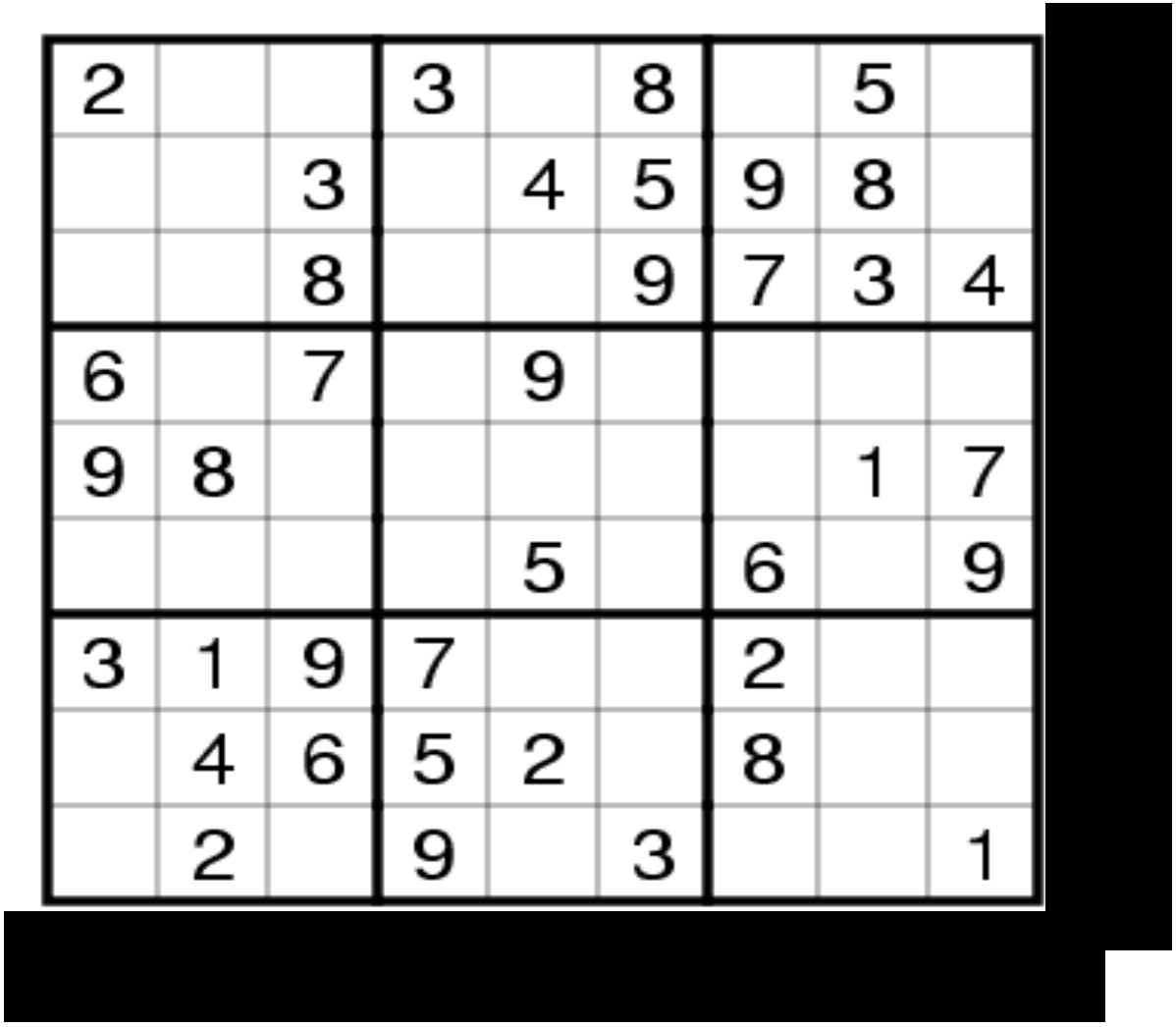
**Per SAT abbiamo una NTM $T_{SAT}$ di complessità di tempo polinomiale così definita:**

**$T_{SAT}$ = INPUT φ**

1. **non deterministicamente genera un possibile assegnamento di valori 0 e 1 per le variabili di φ**

2. **verifica se φ vale 1, se sì accetta altrimenti rifiuta.**

# 3-Sudoku



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

**Regole:**

**la tavola è suddivisa in 9 quadranti di 3x3 caselle alcune delle quali sono già fissate, le altre vanno riempite con numeri dall'1 al 9**

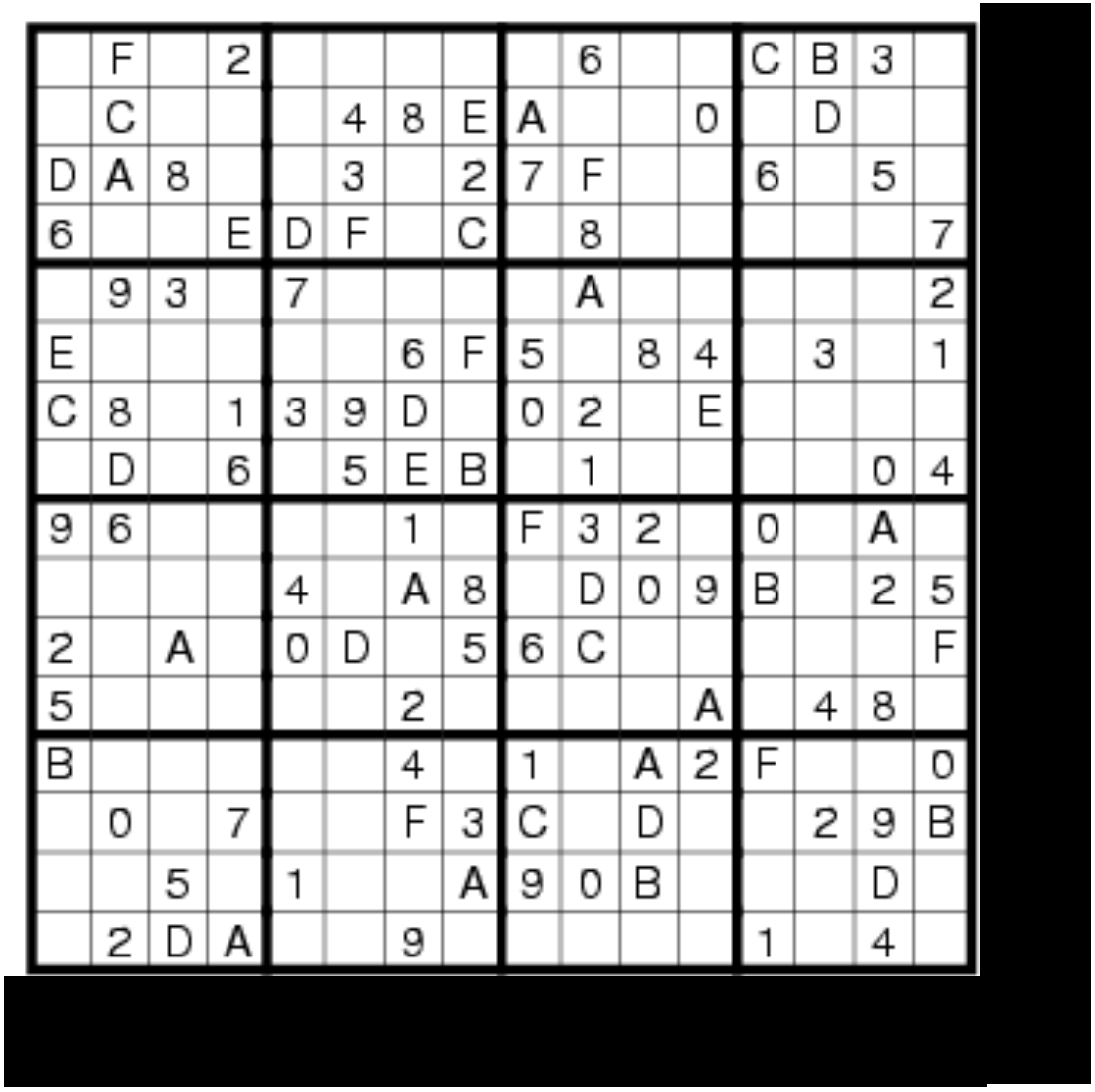
**su ogni quadrante devono essere messi tutti e 9 i numeri, senza ripetizioni**

**inoltre, ogni riga orizzontale e ogni riga verticale dell'intera tavola non deve contenere ripetizioni di numeri**

# Una soluzione

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

# 4-Sudoku



| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F | | 2 | | | | | | 6 | | | C | B | 3 | |
| | C | | | | 4 | 8 | E | A | | | 0 | | D | | |
| D | A | 8 | | | 3 | | 2 | 7 | F | | | 6 | | 5 | |
| 6 | | | E | D | F | | C | | 8 | | | | | | 7 |
| | 9 | 3 | | 7 | | | | | A | | | | | | 2 |
| E | | | | | | 6 | F | 5 | | 8 | 4 | | 3 | | 1 |
| C | 8 | | 1 | 3 | 9 | D | | 0 | 2 | | E | | | | |
| | D | | 6 | | 5 | E | B | | 1 | | | | | 0 | 4 |
| 9 | 6 | | | | | 1 | | F | 3 | 2 | | 0 | | A | |
| | | | | 4 | | A | 8 | | D | 0 | 9 | B | | 2 | 5 |
| 2 | | A | | 0 | D | | 5 | 6 | C | | | | | | F |
| 5 | | | | | | 2 | | | | | A | | 4 | 8 | |
| B | | | | | | 4 | | 1 | | A | 2 | F | | | 0 |
| | 0 | | 7 | | | F | 3 | C | | D | | | 2 | 9 | B |
| | | 5 | | 1 | | | A | 9 | 0 | B | | | | D | |
| | 2 | D | A | | | 9 | | | | | | 1 | | 4 | |

**Regole:**

**•la tavola è suddivisa in 16 quadranti di 4x4 caselle, alcune delle quali sono già fissate, le altre vanno riempite con i numeri da 0 a 15 (da 0 a 9 e da A a F: numeri in base 16).**

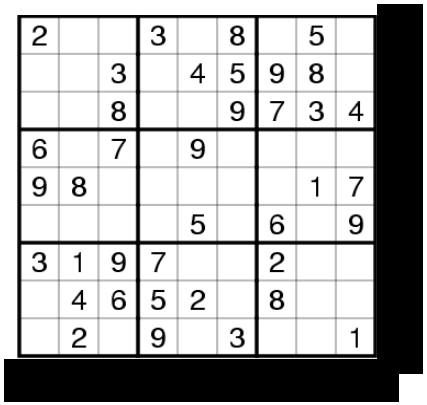
**•su ogni quadrante devono essere messi tutti e 16 i numeri senza ripetizioni**

**•inoltre, ogni riga orizzontale e ogni riga verticale dell'intera tavola non deve contenere ripetizioni di numeri**

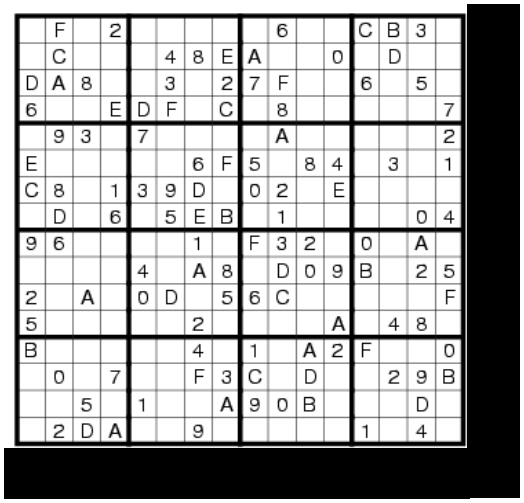
# Una soluzione

| 0 | F | 9 | 2 | A | 7 | 5 | 1 | 4 | 6 | E | D | C | B | 3 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | C | 1 | 3 | 6 | 4 | 8 | E | A | B | 5 | 0 | 2 | D | F | 9 |
| D | A | 8 | 4 | 9 | 3 | B | 2 | 7 | F | C | 1 | 6 | 0 | 5 | E |
| 6 | 5 | B | E | D | F | 0 | C | 2 | 8 | 9 | 3 | 4 | A | 1 | 7 |
| 4 | 9 | 3 | 5 | 7 | 1 | C | 0 | D | A | F | B | 8 | E | 6 | 2 |
| E | B | 7 | 0 | 2 | A | 6 | F | 5 | 9 | 8 | 4 | D | 3 | C | 1 |
| C | 8 | F | 1 | 3 | 9 | D | 4 | 0 | 2 | 6 | E | 5 | 7 | B | A |
| A | D | 2 | 6 | 8 | 5 | E | B | 3 | 1 | 7 | C | 9 | F | 0 | 4 |
| 9 | 6 | 4 | 8 | E | B | 1 | 7 | F | 3 | 2 | 5 | 0 | C | A | D |
| 3 | 7 | C | F | 4 | 6 | A | 8 | E | D | 0 | 9 | B | 1 | 2 | 5 |
| 2 | 1 | A | B | 0 | D | 3 | 5 | 6 | C | 4 | 8 | 7 | 9 | E | F |
| 5 | E | 0 | D | F | C | 2 | 9 | B | 7 | 1 | A | 3 | 4 | 8 | 6 |
| B | 3 | 6 | 9 | C | E | 4 | D | 1 | 5 | A | 2 | F | 8 | 7 | 0 |
| 1 | 0 | E | 7 | 5 | 8 | F | 3 | C | 4 | D | 6 | A | 2 | 9 | B |
| 8 | 4 | 5 | C | 1 | 2 | 7 | A | 9 | 0 | B | F | E | 6 | D | 3 |
| F | 2 | D | A | B | 0 | 9 | 6 | 8 | E | 3 | 7 | 1 | 5 | 4 | C |

# Problemi in NP: n-Sudoku



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



| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F | | 2 | | | | | | 6 | | | C | B | 3 | |
| | C | | | | 4 | 8 | E | A | | | 0 | | D | | |
| D | A | 8 | | | 3 | | 2 | 7 | F | | | 6 | | 5 | |
| 6 | | | E | D | F | | C | | 8 | | | | | | 7 |
| | 9 | 3 | | 7 | | | | | A | | | | | | 2 |
| E | | | | | | 6 | F | 5 | | 8 | 4 | | 3 | | 1 |
| C | 8 | | 1 | 3 | 9 | D | | 0 | 2 | | E | | | | |
| | D | | 6 | | 5 | E | B | | 1 | | | | | 0 | 4 |
| 9 | 6 | | | | | 1 | | F | 3 | 2 | | 0 | | A | |
| | | | | 4 | | A | 8 | | D | 0 | 9 | B | | 2 | 5 |
| 2 | | A | | 0 | D | | 5 | 6 | C | | | | | | F |
| 5 | | | | | | 2 | | | | | A | | 4 | 8 | |
| B | | | | | | 4 | | 1 | | A | 2 | F | | | 0 |
| | 0 | | 7 | | | F | 3 | C | | D | | | 2 | 9 | B |
| | | 5 | | 1 | | | A | 9 | 0 | B | | | | D | |
| | 2 | D | A | | | 9 | | | | | | 1 | | 4 | |

Supponiamo che prenda tempo $t_S(n)$ risolvere un'istanza di dimensione n

Sia V(n) il tempo per verificare la soluzione

$$V(n) = O(n^2 \times n^2)$$

C'è una costante c tale che

$$t_S(n) \leq n^c ?$$

In altri termini c'è un algoritmo polinomiale per il sudoku?