

# Informatica Teorica



Anno Accademico 2022/2023  
Fabio Zanasi  
<https://www.unibo.it/sitoweb/fabio.zanasi>  
Dodicesima lezione

# Nelle puntate precedenti

Abbiamo definito cosa si intende per complessità di tempo di un algoritmo, come si misura usando la big-O notation, e quali sono le criticità di questa metodologia.

Abbiamo introdotto le classi P e NP, e discusso la domanda se P sia uguale a NP.

# In questa lezione

Riduzioni efficienti: Poly (mapping) reduction

NP-completezza: il teorema di Cook-Levin

# Introduzione

# Verso il teorema di Cook-Levin

Il nostro obiettivo di oggi sarà dimostrare il seguente teorema

**Teorema (Cook-Levin 1971)**  
**Se  $SAT \in P$  allora  $P = NP$ .**

$SAT$  è un problema (che definiremo) in  $NP$ . Il teorema dà a  $SAT$  uno statuto speciale tra i problemi in  $NP$ . Diremo che  $SAT$  è **NP-completo**: se fosse in  $P$ , allora sarebbero in  $P$  tutti gli altri problemi  $NP$ .

# Verso il teorema di Cook-Levin

Il metodo utilizzato per dimostrare questo risultato é di interesse indipendentemente dal teorema stesso. É una forma di **riduzione** (*poly reduction*) tra problemi che tiene conto della complessità di tempo.

La dimostrazione consiste nel dimostrare che qualsiasi problema in NP é riducibile in tempo polinomiale a SAT.

# Poly reduction

# Poly reduction

**Definizione.** Siano  $L$  e  $L'$  linguaggi sull'alfabeto  $\Sigma$ . Diciamo che  $L'$  è poly (mapping-)riducibile a  $L$ , scritto  $L' \leq_p L$ , se esiste una TM che computa *in tempo polinomiale* una funzione (totale)  $f : \Sigma^* \rightarrow \Sigma^*$  tale che

$$x \in L' \Leftrightarrow f(x) \in L$$

In altre parole,  $L' \leq_p L$  se  $L' \leq L$  e la riduzione che lo testimonia è computabile in tempo polinomiale.

# Poly reduction e P

**Teorema** Se  $L' \leq_p L$  e  $L \in P$ , allora  $L' \in P$ .

**Dimostrazione** Alla lavagna.

# Poly reduction e complemento

**Teorema** Per  $L$  in  $P$ ,  $L^{\perp} \leq_p L$ .

**Dimostrazione** Alla lavagna.

**Domanda** Se  $L$  é in  $NP$ ,  $L^{\perp} \leq_p L$ ?

# Esempio: 3SAT e CLIQUE

Dimostreremo  $3SAT \leq_p CLIQUE$ .

Prima definiamo 3SAT.

# Formule e soddisfabilità

Una **formula booleana** è costruita a partire da variabili  $x, y, z, \dots$ , loro negazione  $\bar{x}, \bar{y}, \bar{z} \dots$  (chiamati collettivamente *letterali*) e combinazioni di letterali tramite congiunzione  $\wedge$  e disgiunzione  $\vee$ . Le variabili possono ricevere valore vero (1) o falso (0), da cui deriviamo il valore di verità dell'intera formula.

Esempio:  $(\bar{x} \vee y) \wedge (x \vee z)$

Una formula è **soddisfacibile** se esiste un assegnamento di valore alle sue variabili che le dia valore 1.

La formula di esempio è soddisfacibile, come testimoniato dall'assegnamento  $x = 1, y = 1, z = 0$ .

# 3cnf

Una formula booleana é una **clausola** se é una disgiunzione di variabili (positive o negate).

$$\bar{x} \vee y \vee z \vee \bar{z}$$

Una formula booleana é in **forma normale congiunta (cnf)** se é una congiunzione di clausole.

$$(x \vee y) \wedge (\bar{z} \vee z) \wedge (\bar{x} \vee y \vee z \vee \bar{z})$$

Una formula booleana é **3cnf** se é in cnf e ogni clausola contiene esattamente tre letterali.

# 3SAT e SAT

**SAT = { $\langle F \rangle \mid F$  è una formula booleana soddisfacibile}**

**3SAT = { $\langle F \rangle \mid F$  è una formula booleana 3cnf soddisfacibile}**

Perché SAT e 3SAT sono importanti?

Ragioni storiche:

Quesiti di logica come questi hanno dato il via al lavoro fondazionale di Turing, Church, Gödel, etc. sulla teoria della computabilità.

Ragioni pratiche:

Numerosi problemi in informatica (esempio: *constraint programming*) possono essere formulati come istanze di 3SAT/SAT.

# Esempio: 3SAT e CLIQUE

Dimostreremo  $3SAT \leq_p CLIQUE$ .

Ricordiamo CLIQUE:

$CLIQUE = \{<G,k> \mid \text{Il grafo } G \text{ contiene un clique di } k \text{ nodi.}\}$

La riduzione è interessante perché collega problemi all'apparenza molto diversi (uno sulle formule logiche e l'altro sui grafi).

# $3SAT \leq_p CLIQUE$ : Dimostrazione

$$\langle F \rangle \in 3SAT \Leftrightarrow f(\langle F \rangle) \in CLIQUE$$

L'idea della dimostrazione è tradurre formule in grafi, dove  $f(\langle F \rangle)$  sarà costruito in modo tale da 'mimare' il comportamento di variabili e clausole. Un clique in  $f(\langle F \rangle)$  corrisponderà ad un assegnamento che soddisfa  $\langle F \rangle$ .

# $3SAT \leq_p CLIQUE$ : Dimostrazione

$$\langle F \rangle \in 3SAT \Leftrightarrow f(\langle F \rangle) \in CLIQUE$$

Sia  $F$  una formula con  $k$  clausole:

$$F = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \dots \wedge (a_k \vee b_k \vee c_k)$$

Definiamo  $f(\langle F \rangle) = \langle G, k \rangle$ , dove  $G$  è il seguente grafo:

La costruzione di  $G$  richiede tempo polinomiale rispetto alla lunghezza di  $\langle F \rangle$ .

**Nodi:**  $G$  ha  $3k$  nodi, suddivisi in  $k$  gruppi  $t_1 \dots t_k$  chiamati triples. L'idea è che ogni tripla corrisponde ad una clausola di  $F$ : perciò etichettiamo ciascun nodo di  $G$  con il letterale corrispondente in  $F$ .

**Archi:** gli archi di  $G$  collegano tutti le coppie  $(n_1, n_2)$  di nodi tra di loro, eccetto (I) se  $n_1$  e  $n_2$  sono nella stessa tripla, oppure (II)  $n_1$  ha etichetta  $x$  e  $n_2$  ha etichetta  $\bar{x}$  per qualche variabile  $x$ .

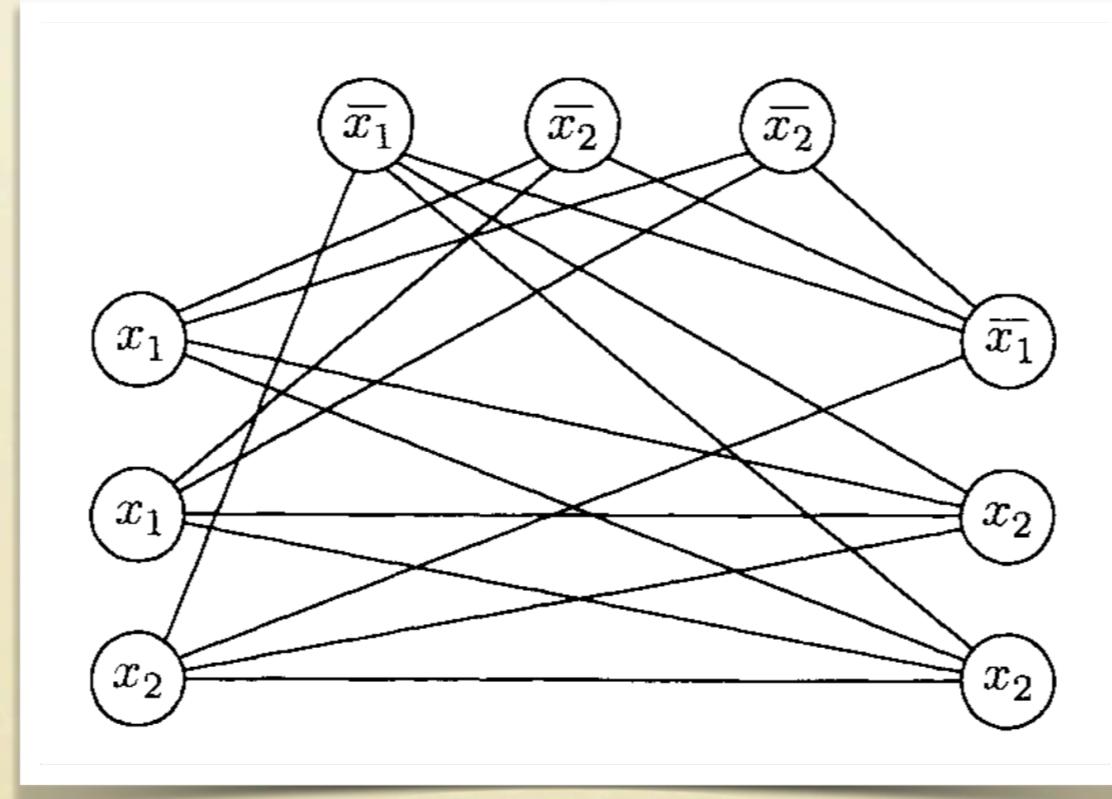
# $3SAT \leq_p CLIQUE$ : Dimostrazione

$$\langle F \rangle \in 3SAT \Leftrightarrow f(\langle F \rangle) \in CLIQUE$$

Per esempio:

$$F = (x_1 \vee x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_2)$$

$f(\langle F \rangle)$  è definito nel modo seguente:



# $3SAT \leq_p CLIQUE$ : Dimostrazione

$$\langle F \rangle \in 3SAT \Leftrightarrow f(\langle F \rangle) \in CLIQUE$$

Dimostriamo che la costruzione soddisfa l'equivalenza.

$$\langle F \rangle \in 3SAT \rightarrow f(\langle F \rangle) \in CLIQUE$$

Se  $F$  é soddisfacibile, almeno un letterale per ogni clausola é vero. Selezioniamo il nodo corrispondente nel grafo  $G$ : avremo così selezionato un nodo in ciascuna tripla. Questa selezione ci da un clique di grandezza  $k$ : notiamo infatti che (1) ci sono  $k$  triple e (2) ogni nodo selezionato é collegato agli altri della selezione, per virtù delle regole di costruzione di  $G$ .

# $3SAT \leq_p CLIQUE$ : Dimostrazione

$$\langle F \rangle \in 3SAT \Leftrightarrow f(\langle F \rangle) \in CLIQUE$$

Dimostriamo che la costruzione soddisfa l'equivalenza.

$$\langle F \rangle \in 3SAT \Leftarrow f(\langle F \rangle) \in CLIQUE$$

Se  $f(\langle F \rangle) = \langle G, k \rangle$  e  $G$  contiene un clique  $S$  di  $k$  nodi, definiamo un assegnamento  $A$  di valori di verità alle variabili in  $F$  dando semplicemente valore vero ad ogni letterale che etichetta un nodo in  $S$ . Per costruzione di  $G$ , nota che (1)  $A$  non è contraddittoria, perché nodi etichettati con non sono collegati e perciò non possono far parte di  $S$ ; (2) ogni nodo di  $S$  appartiene ad una tripla diversa rispetto agli altri in  $S$ , perciò  $A$  assegna valore vero esattamente ad un letterale per clausola. Perciò  $A$  rende vera  $F$ .

# Il teorema di Cook-Levin

# NP-completezza

## Definizione

Un linguaggio  $L$  è NP-completo se è in NP e ogni altro linguaggio  $L'$  in NP è poly-riducibile ad esso ( $L' \leq_p L$ ).

# NP-completezza

Esistono linguaggi NP-completi?

$TMSAT = \{ \langle x, w, s, t \rangle \mid x = \text{code}(M) \text{ per qualche TM } M$   
 $\text{e } M \text{ accetta } \langle w, c \rangle \text{ per qualche } c \text{ di lunghezza al più } s, \text{ in tempo al più } t. \}$

Per ogni  $L'$  in NP,  $L' \leq_p TMSAT$ .

Tuttavia, TMSAT non ci fa scoprire ‘nulla di nuovo’ su NP.

# Il teorema di Cook Levin

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

**Teorema** (Cook, Levin, 1971) SAT è NP-completo.

**Corollario** Se SAT è in P, allora  $P = NP$ .

# Il teorema di Cook Levin: Dimostrazione

## Idea della dimostrazione

Dimostrare che SAT é in NP é immediato: data una formula F, possiamo costruire una TM che scelga in maniera non-deterministica un assegnamento A, e accettare se A rende F vera.

# Il teorema di Cook Levin: Dimostrazione

## Idea della dimostrazione

Rimane da dimostrare che ogni linguaggio  $L$  in NP è riducibile a SAT.

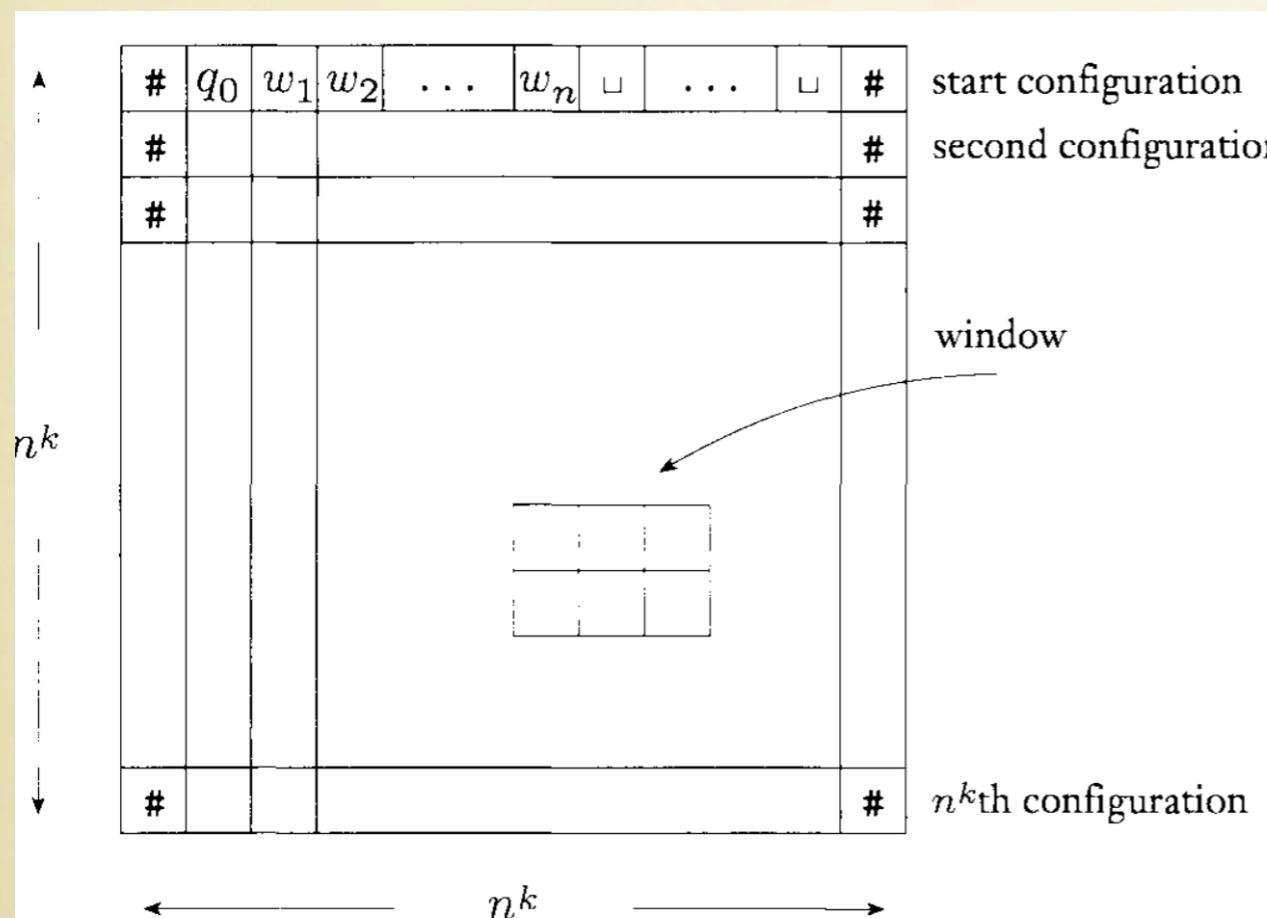
Supponi che  $M$  sia la TM non-deterministica che decide  $L$ . L'idea è di tradurre qualsiasi stringa  $w$  in una formula  $F_w$ , tale che assegnamenti di valore A a  $F_w$  rappresentano computazioni di  $M$  su  $w$ . In particolare, vogliamo che A renda  $F_w$  vera se e solo se  $M$  accetta  $w$  (quindi  $w$  è in  $L$ ).

La parte 'laboriosa' (ma concettualmente semplice) è costruire  $F_w$  con tale proprietà.

# Il teorema di Cook Levin: Dimostrazione

Supponiamo che la TM  $M = \langle \Sigma, Q, q_0, \delta, \{Y, N\} \rangle$  decida  $L$  in tempo  $n^k$  per qualche costante  $k$ .

Definiamo un **tableau** per  $M$  come una tabella  $n^k \times n^k$  dove le righe descrivono le configurazioni di un ramo di computazione di  $M$  su input  $w$ .



Osserva: convenzione su posizione di simboli speciali  $\#$ , e stato corrente.  
 $\text{cell}[i,j] \in \Sigma \cup Q \cup \{\#\}$

Il problema di stabilire se  $M$  accetta  $w$  è equivalente a stabilire se esiste un tableau accettante (cioè uno che arrivi ad una configurazione in uno stato finale).

# Il teorema di Cook Levin: Dimostrazione

La formula  $F_w$  sarà la congiunzione di quattro sottoformule.

$$F_w := F_{cell} \wedge F_{start} \wedge F_{move} \wedge F_{accept}$$

Ricordiamo la proprietà che vogliamo garantire tramite la definizione di  $F_w$ :

$F_w$  è soddisfacibile  $\leftrightarrow$  esiste un tableau accettante  
 $(\leftrightarrow$  esiste una computazione accettante di  $M$  su  $w$ )

# Il teorema di Cook Levin: Dimostrazione

L'insieme di variabili che appaiono in  $F_w$  è dato da

$$\{x_{i,j,s} \mid (i,j) \in n^k \times n^k \text{ e } s \in Q \cup \Sigma \cup \{\#\}\}$$

Intuitivamente, vogliamo assegnare valore 1 (vero) a  $x_{i,j,s}$  se cell[i,j] contiene il valore s, e 0 (falso) altrimenti.

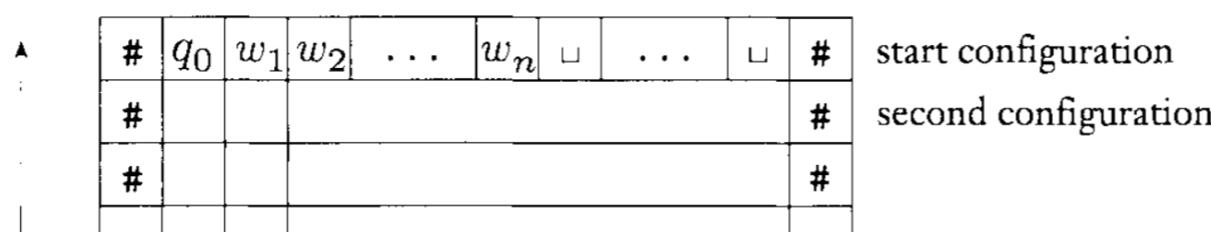
La prima sottoformula di  $F_w$ ,  $F_{cell}$ , assicura che per rendere vera  $F_w$  dobbiamo rendere vera esattamente una variabile per ogni cella del tableau.

$$F_{cell} := \bigwedge_{1 \leq i, j \leq n^k} \left[ \left( \bigvee_{s \in C} x_{i,j,s} \right) \wedge \left( \bigwedge_{\substack{s, t \in C \\ s \neq t}} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

# Il teorema di Cook Levin: Dimostrazione

La seconda sottoformula di  $F_w$ ,  $F_{start}$ , assicura che nel rendere vera  $F_w$  il tableau considerato deve avere sulla prima riga la configurazione iniziale di una computazione di  $M$  su  $w$ .

$$\begin{aligned} F_{start} := & \ x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ & x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \dots \wedge x_{1,n+2,w_n} \wedge \\ & x_{1,n+3,\sqcup} \wedge \dots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#} \end{aligned}$$



#	$q_0$	$w_1$	$w_2$	$\dots$	$w_n$	$\sqcup$	$\dots$	$\sqcup$	#
#									#
#									#

start configuration  
second configuration

# Il teorema di Cook Levin: Dimostrazione

La quarta sottoformula di  $F_w$ ,  $F_{accept}$ , assicura che nel rendere vera  $F_w$  almeno una delle configurazioni rappresentate nel tableau raggiunge lo stato accettante Y.

$$F_{accept} := \bigvee_{1 \leq i,j \leq n^k} x_{i,j,Y}$$

# Il teorema di Cook Levin: Dimostrazione

Rimane da definire la terza sottoformula di  $F_w$ ,  $F_{move}$ , la quale assicura che  $F_w$  può essere resa vera solo se i simboli nelle varie celle del tableau descrivono una computazione di  $M$  su  $w$  che rispetti la funzione di transizione  $\delta$  di  $M$ .

La computazione di una TM è locale, per cui è sufficiente esprimere una condizione che riguardi porzioni del tableau di dimensione 2x3. Le chiamiamo **finestre**.

a	$q_1$	b
a	a	$q_2$

# Il teorema di Cook Levin: Dimostrazione

Per esempio, supponiamo che  $\delta$  includa le transizioni

$$\delta(q_1, a) = \{(q_1, b, \rightarrow )\} \quad \delta(q_1, b) = \{(q_2, c, \leftarrow ), (q_2, a \rightarrow )\}$$

Le seguenti sono finestre ammesse nel tableau

(a)

a	$q_1$	b
$q_2$	a	c

(b)

a	$q_1$	b
a	a	$q_2$

(c)

a	a	$q_1$
a	a	b

(d)

#	b	a
#	b	a

(e)

a	b	a
a	b	$q_2$

(f)

b	b	b
c	b	b

Le seguenti sono finestre non ammesse nel tableau

(a)

a	b	a
a	a	a

(b)

a	$q_1$	b
$q_1$	a	a

(c)

b	$q_1$	b
$q_2$	b	$q_2$

# Il teorema di Cook Levin: Dimostrazione

‘Per ogni  
cella  $cell[i,j]$ ’

‘Intorno’ alla cella  $cell[i,j]$  si ha una finestra ammessa

$a_1$	$a_2$	$a_3$
$a_4$	$a_5$	$a_6$

$$F_{move} := \bigwedge_{1 < i \leq n^k, 1 < j < n^k} \left( \bigvee_{\substack{a_1, \dots, a_6 \\ \text{é una finestra ammessa}}} x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+i,j,a_5} \wedge x_{i+1,j+1,a_6} \right)$$

Omettiamo i dettagli (complicati, ma non complessi) di come esprimere in formule “ $a_1, \dots, a_6$  é una finestra ammessa” sulla base di  $\delta$ .

# Il teorema di Cook Levin: Dimostrazione

$$F_w := F_{cell} \wedge F_{start} \wedge F_{move} \wedge F_{accept}$$

Da verificare:

1.  $F_w$  è soddisfacibile  $\leftrightarrow$  esiste un tableau accettante



2. La costruzione di  $F_w$  avviene in tempo polinomiale rispetto alla lunghezza di  $w$ .

# Il teorema di Cook Levin: Dimostrazione

$$F_w := F_{cell} \wedge F_{start} \wedge F_{move} \wedge F_{accept}$$

2. La costruzione di  $F_w$  avviene in tempo polinomiale rispetto alla lunghezza di  $w$ . Verifichiamolo:
  - a. Il tableau è una tabella  $n^k \times n^k$ , perciò contiene  $n^{2k}$  celle. Ogni cella contiene uno tra  $m$  simboli diversi, dove  $m$  è la cardinalità di  $Q \cup \Sigma \cup \{\#\}$ . Perciò il numero di variabili  $x_{i,j,s}$  è  $m \times n^{2k}$ . In notazione Big-O,  $O(n^{2k})$ .

# Il teorema di Cook Levin: Dimostrazione

$$F_w := F_{cell} \wedge F_{start} \wedge F_{move} \wedge F_{accept}$$

$$F_{cell} := \bigwedge_{1 \leq i, j \leq n^k} \left[ \left( \bigvee_{s \in C} x_{i,j,s} \right) \wedge \left( \bigwedge_{\substack{s, t \in C \\ s \neq t}} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

b.  $F_{cell}$  contiene una sottoformula di lunghezza costante per ogni  $x_{i,j,s}$ , perciò la sua lunghezza è  $O(n^{2k})$ .

# Il teorema di Cook Levin: Dimostrazione

$$F_w := F_{cell} \wedge F_{start} \wedge F_{move} \wedge F_{accept}$$

$$\begin{aligned} F_{start} := & x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ & x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \dots \wedge x_{1,n+2,w_n} \wedge \\ & x_{1,n+3,\sqcup} \wedge \dots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#} \end{aligned}$$

c.  $F_{start}$  contiene una sottoformula di lunghezza costante per ognuna delle  $n^k$  celle nella prima riga, perciò la sua lunghezza è  $O(n^k)$ .

# Il teorema di Cook Levin: Dimostrazione

$$F_w := F_{cell} \wedge F_{start} \wedge F_{move} \wedge F_{accept}$$

$$F_{move} := \bigwedge_{1 \leq i \leq n^k, 1 \leq j \leq n^k} \left( \bigvee_{\substack{a_1, \dots, a_6 \\ \text{è una finestra} \\ \text{ammessa}}} x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+i,j,a_5} \wedge x_{i+1,j+1,a_6} \right)$$

$$F_{accept} := \bigvee_{1 \leq i, j \leq n^k} x_{i,j,Y}$$

d. Sia  $F_{move}$  che  $F_{accpet}$  contengono una sottoformula di lunghezza costante per ogni cella del tableau, perciò la loro lunghezza è  $O(n^{2k})$ .

# Il teorema di Cook Levin: Dimostrazione

$$F_w := F_{cell} \wedge F_{start} \wedge F_{move} \wedge F_{accept}$$

e. Perciò la lunghezza complessiva di  $F_w$  è polinomiale:

$$O(n^{2k}) + O(n^k) + O(n^{2k}) + O(n^{2k}) = O(n^{2k}).$$

Ora che abbiamo appurato che generare ogni carattere di  $F_w$  richiede tempo polinomiale, non è difficile concepire una TM che costruisce in tempo polinomiale  $F_w$  nel modo descritto, a partire dalla definizione di  $M$  e di  $w$ .

# Il teorema di Cook Levin: Dimostrazione

$$F_w := F_{cell} \wedge F_{start} \wedge F_{move} \wedge F_{accept}$$

Da verificare:

1.  $F_w$  è soddisfacibile  $\leftrightarrow$  esiste un tableau accettante



2. La costruzione di  $F_w$  avviene in tempo polinomiale rispetto alla lunghezza di  $w$ .



# Altri problemi NP-completi

**Teorema** 3SAT é NP-completo.

**Corollario** CLIQUE é NP-completo.

Altri esempi:

- Problema del commesso viaggiatore
- Colorare un grafo con  $k$  colori diversi
- Vincere a battaglia navale
- Fare mining di bitcoin in maniera ottimale
- Serializzabilità della cronologia di un database