1°

ESERCIZIO 1.a. La funzione di transizione è definita correttamente? **Punteggio massimo** 2	mancano due nastri	1 / 2
ESERCIZIO 1.b. Si dimostra che i linguaggi Turing-riconoscibili sono riconoscibili dalle RATM? **Punteggio massimo** 2	si	2 / 2
ESERCIZIO 1.b. La soluzione codifica correttamente il nastro della RATM? **Punteggio massimo** 2	manca la codifiica del nastro puntatore	1 / 2
ESERCIZIO 1.b. La soluzione simula correttamente l'operazione di accesso diretto? **Punteggio massimo** 4	descrizione ad alto livello, non si spiega come fa la TM a memorizzare i valor di p, l e m	1 / 4
ESERCIZIO 1.b. La soluzione simula correttamente gli spostamenti a destra e sinistra della testina? **Punteggio massimo** 2	no	0 / 2

2°

ESERCIZIO 2.a. La definizione del linguaggio SUFFIXCLOSED_TM è corretta? **Punteggio massimo** 2	condizione di appartenenza non valida	1 / 2
ESERCIZIO 2.b. il problema di riferimento è scelto correttamente? **Punteggio massimo** 2	si	2 / 2
ESERCIZIO 2.b. Input e output della riduzione sono corretti? **Punteggio massimo** 2	si	2 / 2
ESERCIZIO 2.b. La riduzione è corretta? **Punteggio massimo** 3	definizione ambiga e inconsistente di M'. Uso di u, v, L non definiti in quel contesto	1 / 3
ESERCIZIO 2.b. Si dimostra la correttezza della riduzione? **Punteggio massimo** 3	no, perché M' non è definita in modo valido	0 / 3

3°

ESERCIZIO 3.a. Il certificato è corretto? **Punteggio massimo** 2	si	2 / 2
ESERCIZIO 3.a. Il verificatore è corretto e polinomiale? **Punteggio massimo** 2	no, il verificatore non fa i controlli necessari ed è descritto in modo poco chiaro e ambiguo	0 / 2
ESERCIZIO 3.b. Input e output della riduzione sono corretti? **Punteggio massimo** 2	manca	0 / 2
ESERCIZIO 3.b. La riduzione è corretta e polinomiale? **Punteggio massimo** 3	manca	0 / 3
ESERCIZIO 3.b. Si dimostra la correttezza della riduzione? **Punteggio massimo** 3	no	0 / 3

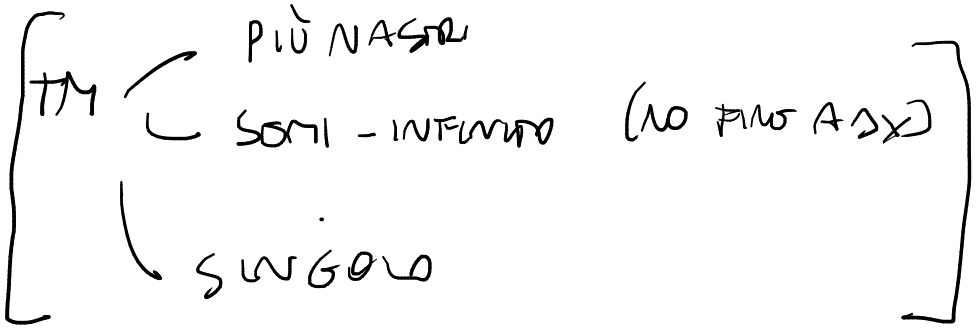
1. **(12 punti)** Una *Macchina di Turing ad accesso casuale (RATM)* è una variante della macchina di Turing che estende il modello standard introducendo un meccanismo per accedere direttamente a una qualsiasi posizione dell'input, senza dover scorrere sequenzialmente le celle del nastro. Una RATM è dotata di tre nastri:

   - un nastro di input a sola lettura, che contiene l'input;
   - un nastro di lavoro dove la macchina può leggere, scrivere e spostarsi a piacere;
   - un nastro puntatore con alfabeto binario. Anche in questo nastro la macchina può leggere, scrivere e spostarsi a piacere.

   Oltre alle consuete operazioni di lettura, scrittura e spostamento delle testine, una RATM può eseguire un'operazione aggiuntiva di accesso diretto all'input. Per eseguire questa operazione, la macchina legge il numero binario p sul nastro puntatore e poi scrive il p-esimo simbolo dell'input sulla cella corrente del nastro lavoro. I simboli dell'input sono numerati da sinistra a destra a partire dalla posizione 0.

   (a) Dai una definizione formale della funzione di transizione di una RATM.

   (b) Dimostra che le RATM riconoscono la classe dei linguaggi Turing-riconoscibili. Usa una descrizione a livello implementativo per definire le macchine di Turing.



Se devo dimostrare solo con nastro singolo:

- Dimostra con TM a nastro singolo

Dimostra che la tua variante riconosce la classe dei linguaggi Turing-riconoscibili
→ 3 scelte a disposizione

② $(\sigma : Q \times \Gamma) \mapsto$ INPUT

STATI NASTRO RATTI

$Q \times \Gamma \times \{L, R\}$ / INPUT

LISTA COSE CHE FANNO QUESTO 2

Q: STATO A INPUT

Γ: LETTO

$\{L, R\}$: LEFT, RIGHT

③ RATTI (3 NASTRI)

$(\text{SINGOLO} \rightarrow \underbrace{\text{RATTI}}_{\text{VARIANTE}})$ = ANSATZ 1°

$(\text{RATTI} \Rightarrow \text{SINGOLO})$ = RISULTATO

$\{L, R, \text{INPUT}\}$ $\{L, 2\}$

(INPUT - NASTRO 1)

(LAVORO - NASTRO 2)

$\{L, R, W\}$ WRITE

$=$

$\{\text{PUNTATORE}\}\{0, 1\}$

NASTRO 3

TM CON PIÙ NASTRI

① $= [\underbrace{ABC}] \rightarrow$ NASTRO 1 (INPUT)

② $= \{L, R, \text{INPUT}\}$

NASTRO SINGOLO: S = SU INPUT "w"

↓

SIMULA

NASTRO 1 : w

NASTRO 2 : COPIA DI SOPRA

NASTRO 3 : PUNTATORE

3 SIMBOLI INPUT

$\delta(q, a, b, c) = (r, b, R)$

1 — DOVETE = TRANSIZIONE = MOSSA

q — STATO IN

r — STATO OUT

= SCRIVI "b" ANDANDO R

- PARTO DA q, APRILO E SCRIVIAMO B SUL NASTRO 2
- LEGGENDO NASTRO 3 → MI RICORDO DOVE ORA CONTROLLO
- VADO AVANTI

INPUT → SCORRI SEQUENZIALMENTE

S : SU INPUT "w";

- $\delta(q, a) = (r, b, L)$

Per simulare una mossa del tipo δ(q,a,b,c) = (r,d,m₁,e,m₂,m₃)
senza accesso diretto:

- Leggi il simbolo a dal primo nastro, b dal secondo nastro, c dal terzo nastro
- Scrivi d sulla cella corrente del secondo nastro
- Scrivi e sulla cella corrente del terzo nastro
- Muovi la testina del secondo nastro secondo m₁
- Muovi la testina del terzo nastro secondo m₂
- Muovi la testina del primo nastro secondo m₃

Passa allo stato r andando verso DESTRA (R)

$\delta(q, a) = (r, b, L)$

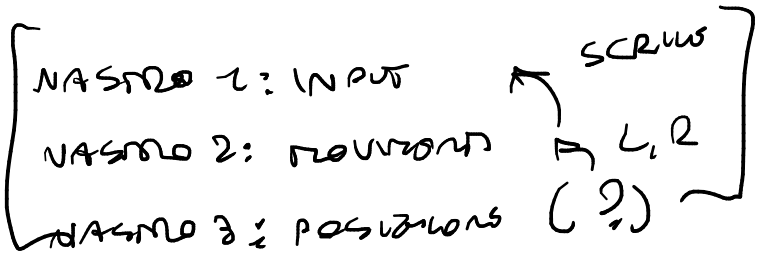
Per simulare una mossa del tipo $\delta(q,a,b,c) = (r,d,m_1,e,m_2,m_3)$
senza accesso diretto:

- Leggi il simbolo a dal primo nastro, b dal secondo nastro, c dal terzo nastro
- Scrivi d sulla cella corrente del secondo nastro
- Scrivi e sulla cella corrente del terzo nastro
- Muovi la testina del secondo nastro secondo m_1
- Muovi la testina del terzo nastro secondo m_2
- Muovi la testina del primo nastro secondo m_3

Passa allo stato r andando verso SINISTRA (SX)

$\delta(q, a) = (r, b, \text{INPUT})$

ACCESSO DIRETTO



Operazione di accesso diretto:

- Salva la posizione corrente della testina del terzo nastro (puntatore)
- Sposta la testina del terzo nastro all'inizio
- Leggi la sequenza binaria dal terzo nastro
fino al primo ⊔ (blank = vuoto) e calcola la combo di 0 e di 1
- Salva la posizione corrente della testina del primo nastro
- Sposta la testina del primo nastro alla posizione p (se p è valido)
- Leggi il simbolo in posizione p
e scrivilo sulla cella corrente del secondo nastro
- Ripristina le posizioni salvate delle testine
del primo e terzo nastro

Se in qualsiasi momento
la simulazione raggiunge
lo stato di accettazione di M,
allora S termina con accettazione.

Se in qualsiasi momento la simulazione
raggiunge lo stato di rifiuto di M,
allora S termina con rifiuto.
Negli altri casi continua
la simulazione dal punto 2."

MOSSA SEMPLICE

SINGOLO → RAM

MULTI-NASTRO

VERSO 1 - ANDATA

Sia M una TM standard.
Costruiamo una RATM R equivalente.
R = "Su input w:

Copia sequenziale: Usa il nastro puntatore
per mantenere un contatore binario inizializzato a 0.

Simulazione: Per ogni mossa di M:

Se M si sposta a destra:
incrementa il contatore binario sul nastro puntatore, usa DIRECT per leggere il simbolo successi
Se M si sposta a sinistra:
decrementa il contatore binario, usa DIRECT per accedere alla posizione precedente
Simula scrittura e cambio di stato normalmente

Terminazione: Termina negli stessi stati di M."

2. (12 punti) Data una parola w su un alfabeto Σ, si dice che $u \in \Sigma^*$ è un suffisso di w se esiste una stringa $v \in \Sigma^*$ tale che $w = vu$. Un linguaggio $L \subseteq \Sigma^*$ è *chiuso per suffisso* se per ogni parola $w \in L$, tutti i suffissi u di w appartengono anch'essi a L. Considera il problema di determinare se il linguaggio di una TM M è chiuso per suffisso.

(a) Formula questo problema come un linguaggio $\textsc{SuffixClosed}_{TM}$.

(b) Dimostra che il linguaggio $\textsc{SuffixClosed}_{TM}$ è indecidibile.

SUFFISSO = STRINGA DOPO $\exists w^{\prime\prime}$

(a) $\text{SUFFIXCLOSED}_{TM} =$

$\{ \langle M \rangle \mid M$ è una TM e $L(M)$ CHIUSO PER SUFFISSO $\}$

$\forall w \in L, \forall v \in \Sigma^* \Rightarrow wv \in L$

OUTPUT

COPIA NASTRO MACCHINA A = CONSEGNA

CHIUSO PER SUFFISSO

SUFFISSO

STRINGA DOPO OUTPUT

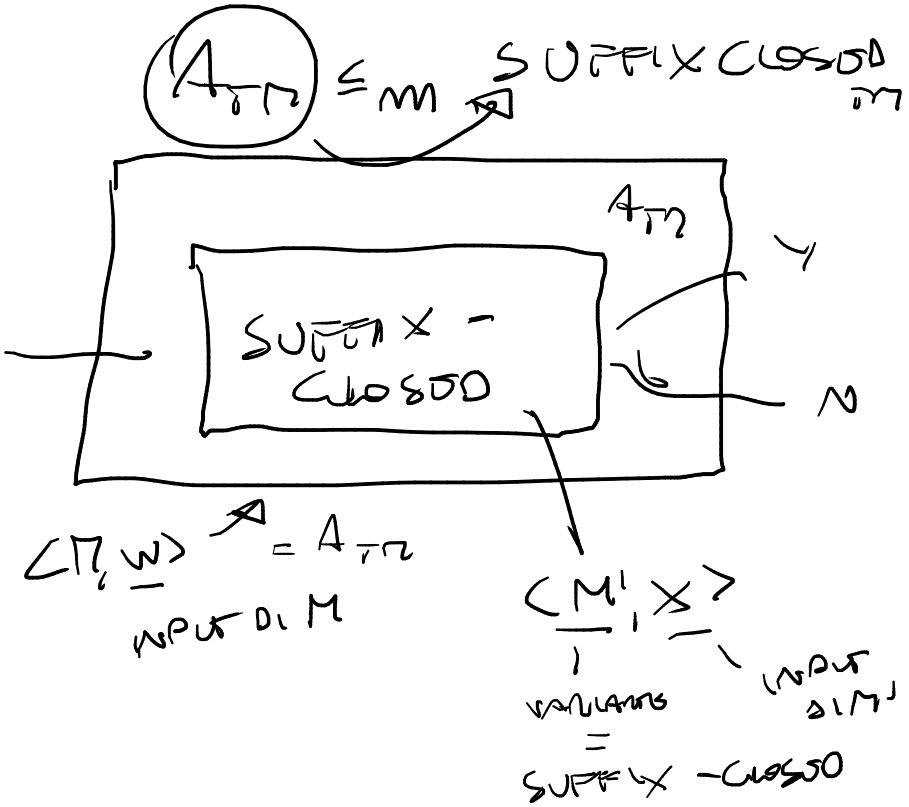
$\langle M, w, v, u \rangle$

M è TM $\rightarrow$ $w = vu$

(b)

A_{TM} → 90%

M su input w accetta



F = su input $\langle M, w \rangle$:

|

FUNZIONE DI RIDUZIONE $\leq_m$ → MAPPI LA SOLUZIONE

(0 w w w w ——— SUFFISSI) → CHIUSO PER SUFFISSO

$\langle M, w \rangle$;

- ESEGUI M su input w
- SIMULA M' su input x

∀

- $X = VU$ → RITORNA M'

- $W = VU$ → SE A_{TM} ACCETTA, È CHIUSA PER SUFFISSO

- RESTITUISCI $\langle M, w \rangle$ A_{TM}

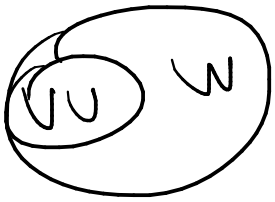
FINO:

$\langle M, w \rangle \in A_{TM}$ SSE

$\langle M' \rangle$ È SUFFIX-CLOSED

⇒ A_{TM} → $w \in M$

VU → X ↓ SUFFIX

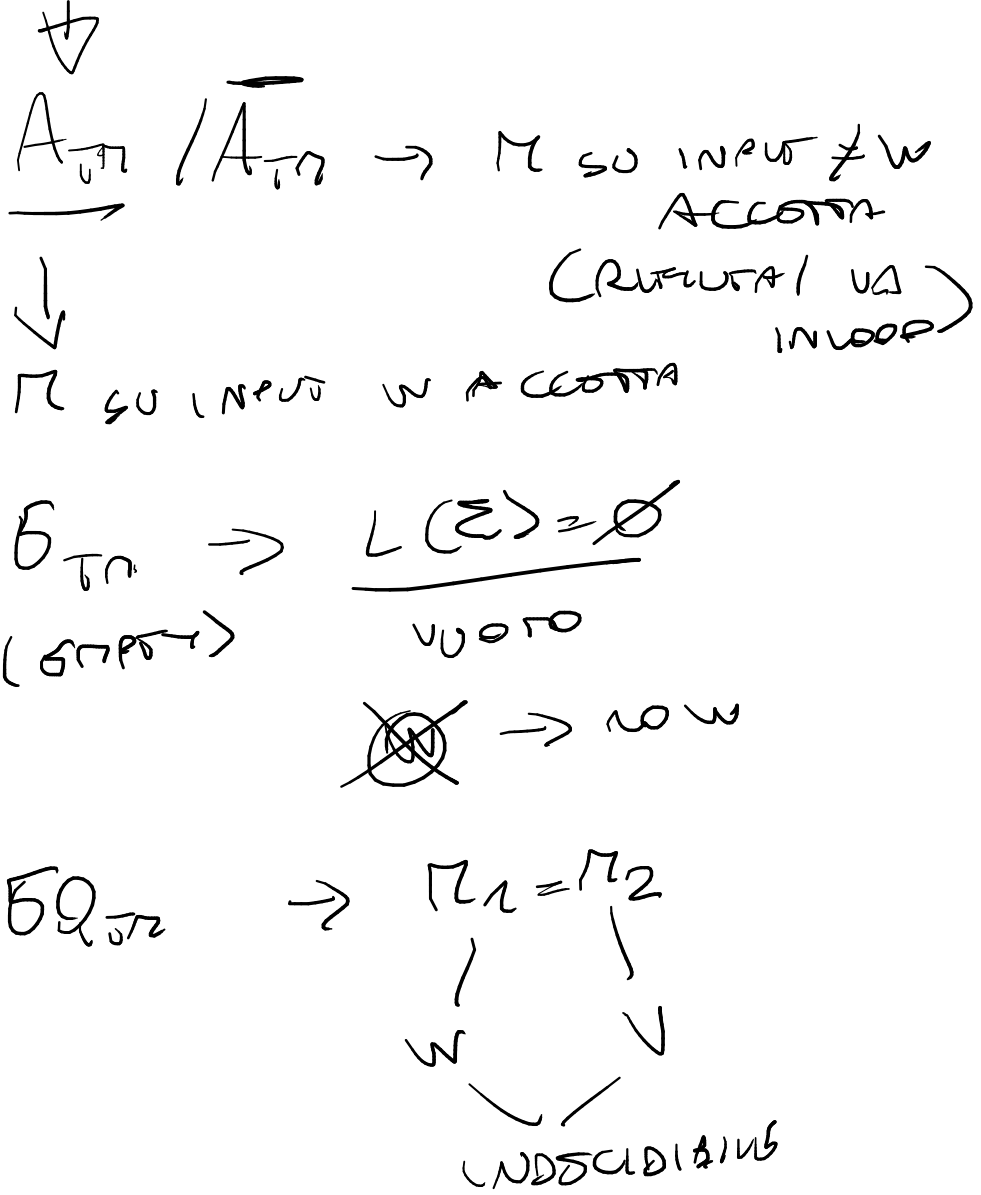


↓ CHIUSA PER SUFFISSO → ANDATA

⇐ $w' \notin L(M)$ → NON CHIUSO PER PREFISSO

$w' \neq w$

$W = VU$ → CHIUSO PER PREFISSO

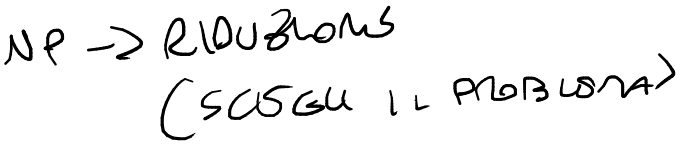


3. (12 punti) Un sottoinsieme S di vertici di un grafo non orientato $G = (V, E)$ è *quasi indipendente* se esiste esattamente un arco di G che ha entrambi gli estremi in S. Considera il seguente problema:

$$\textsc{AlmostIndependentSet} = \{\langle G, k\rangle \mid \text{esiste } S \subseteq V \text{ quasi indipendente di cardinalità } k\}$$

(a) Dimostra che AlmostIndependentSet è un problema NP.

(b) Dimostra che AlmostIndependentSet è NP-hard, usando IndependentSet come problema NP-hard di riferimento.

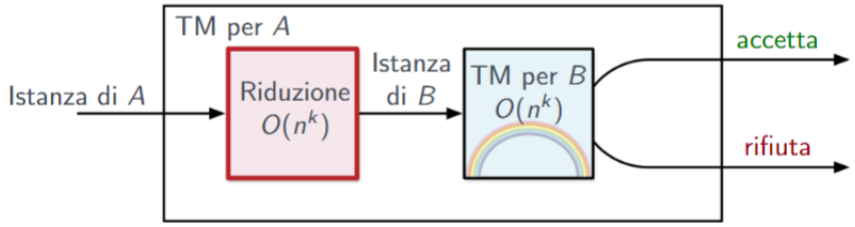


Ogni dimostrazione di NP-completezza di compone di due parti:

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.

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



3. (12 punti) Un sottoinsieme S di vertici di un grafo non orientato $G = (V, E)$ è *quasi indipendente* se esiste esattamente un arco di G che ha entrambi gli estremi in S. Considera il seguente problema:

$$\textsc{AlmostIndependentSet} = \{\langle G, k\rangle \mid \text{esiste } S \subseteq V \text{ quasi indipendente di cardinalità } k\}$$

(a) Dimostra che AlmostIndependentSet è un problema NP.

(b) Dimostra che AlmostIndependentSet è NP-hard, usando IndependentSet come problema NP-hard di riferimento.

NP → - ESISTE UN CERTIFICATO

⟨G, K⟩ → INPUT DOVE FARE CONSEGNA

- LO DOVE FARE UN T. FINITO

3. (12 punti) Un sottoinsieme S di vertici di un grafo non orientato $G = (V, E)$ è *quasi indipendente* se esiste esattamente un arco di G che ha entrambi gli estremi in S. Considera il seguente problema:

$$\textsc{AlmostIndependentSet} = \{\langle G, k\rangle \mid \text{esiste } S \subseteq V \text{ quasi indipendente di cardinalità } k\}$$

(a) Dimostra che AlmostIndependentSet è un problema NP.

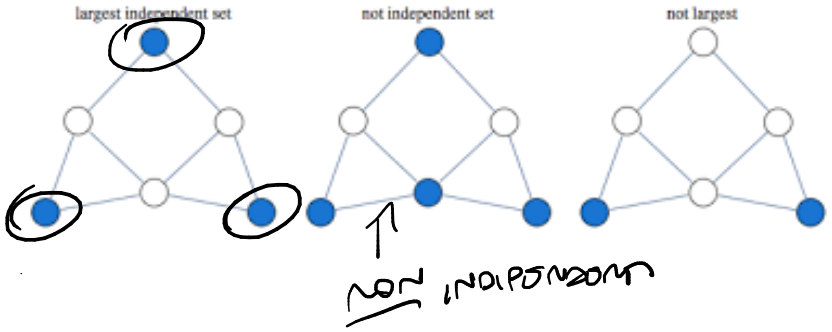
(b) Dimostra che AlmostIndependentSet è NP-hard, usando IndependentSet come problema NP-hard di riferimento.

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: **VertexCover**
- Se il problema richiede di trovare un sottoinsieme grande che soddisfi alcuni vincoli: **IndependentSet**
- 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**!

(INDIPENDENT SET) → GRAFO / VERTICI

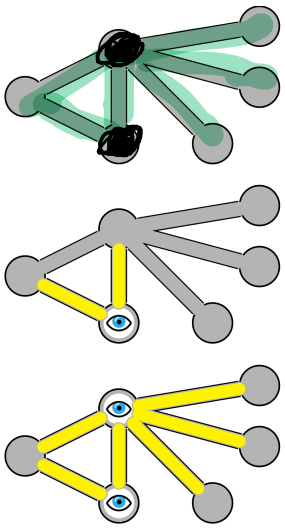
$G = (V, E)$ LATI = ARCHI



NON INDIPENDENTE

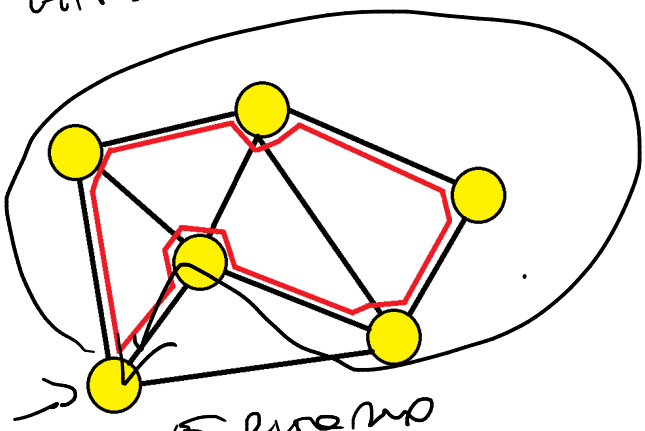
(VERTEX COVER)

TROVO NUMERO DI ARCHI PER COPRIRE GRAFO



→ PRENDO DAL VERTICE CHE COPRE IL GRAFO

(CIRCUITO HAMILTONIANO)



→ PARTO DALLO STESSO

VALID ? = TRUE ?

SAT formula

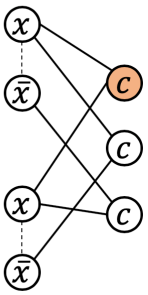
$(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$

Literals: $x_1, \neg x_1, x_2, \neg x_2$

Clauses: c_1, c_2, c_3

$\Leftrightarrow$

Bipartite graph



BOOLEAN

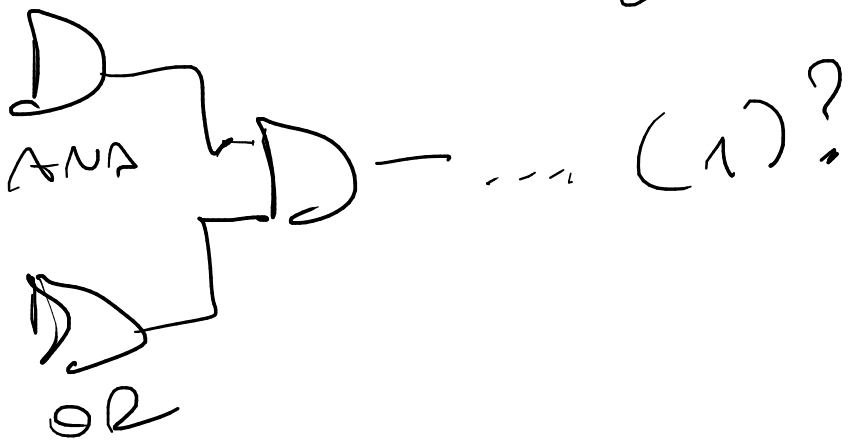
SAT = SATISFIABILITY

A OR B AND NOT C

CLAUSOLE (A V B V C)

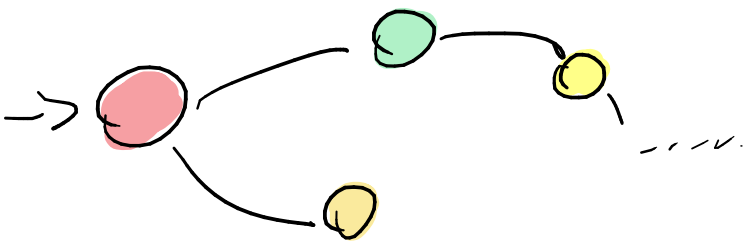
LETTERALI

CIRCUIT - SAT = PORTE LOGICHE



COLOR. DEGREE = N. DI COLORI

3 - COLOR → (K) - COLOR



3. (12 punti) Un sottoinsieme S di vertici di un grafo non orientato $G = (V, E)$ è *quasi indipendente* se esiste esattamente un arco di G che ha entrambi gli estremi in S. Considera il seguente problema:

ALMOSTINDEPENDENTSET $= \{\langle G, k\rangle \mid$ esiste $S \subseteq V$ quasi indipendente di cardinalità $k\}$

(a) Dimostra che ALMOSTINDEPENDENTSET è un problema NP.

(b) Dimostra che ALMOSTINDEPENDENTSET è NP-hard, usando INDEPENDENTSET come problema NP-hard di riferimento.

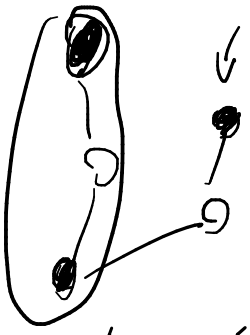
VERIFICATORE → CERTIFICATO

→ TEMPO P

INPUT BUONO DEL PROBLEMA

ALMOSTINDEPENDENTSET $= \{\langle G, k\rangle \mid$ esiste $S \subseteq V$ quasi indipendente di cardinalità k

(a) Dimostra che ALMOSTINDEPENDENTSET è un problema NP.



'QUASI - INDIPENDENTE (CARD - 2)

② → VERIFICATORE (POLY - TIME)

CARDINALITÀ → K

N. DI ELEMENTI DEL GRAFO

N. DI VERTICI

V → $\langle\langle G, k\rangle, S\rangle$

VERIFICATORE

SOTTOINSIEME

CHECKLIST

→ ① u — e — v $e = (u, v)$

→ ② CONTA GLI ARCHI DI G

→ (3) CONTROLLA SE $|S| = K$

CARDINALITÀ

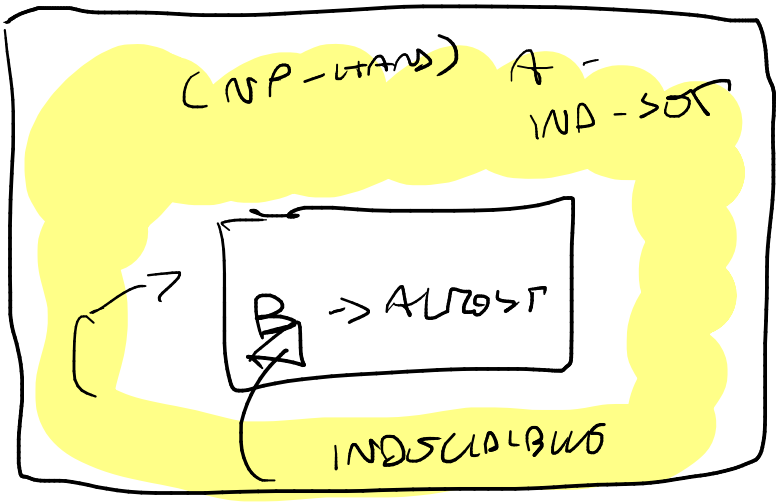
$S \subseteq S$

↓

SOTTOINSIEME → QUASI INDIPENDENTE

(4) $IND\text{-}SOT \leq_m ALLSOT$

A = NP-HARD $\quad B$

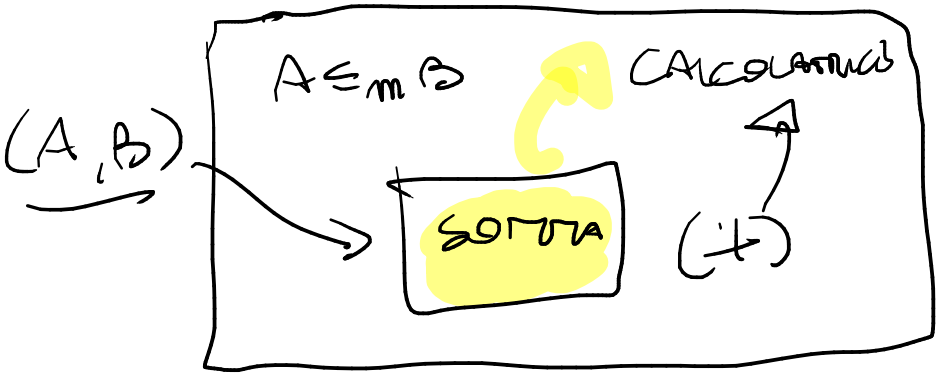


SE USO IND-SOT

⇒ SON O COSTOSO

AD USARE ALLSOT

POI RISOLVERE



DECIDIBILE ⇒ FINISCE!

ALLOST → NP-HARD

① è in NP (verifica / certif.)

② Hai usato → NP-HARD
un algo
per risolvere
NP-hard

↓

NP-completo
/ \
NP NP-hard

A $\leq_m$ B → B (piccolo)
lo chiamo!

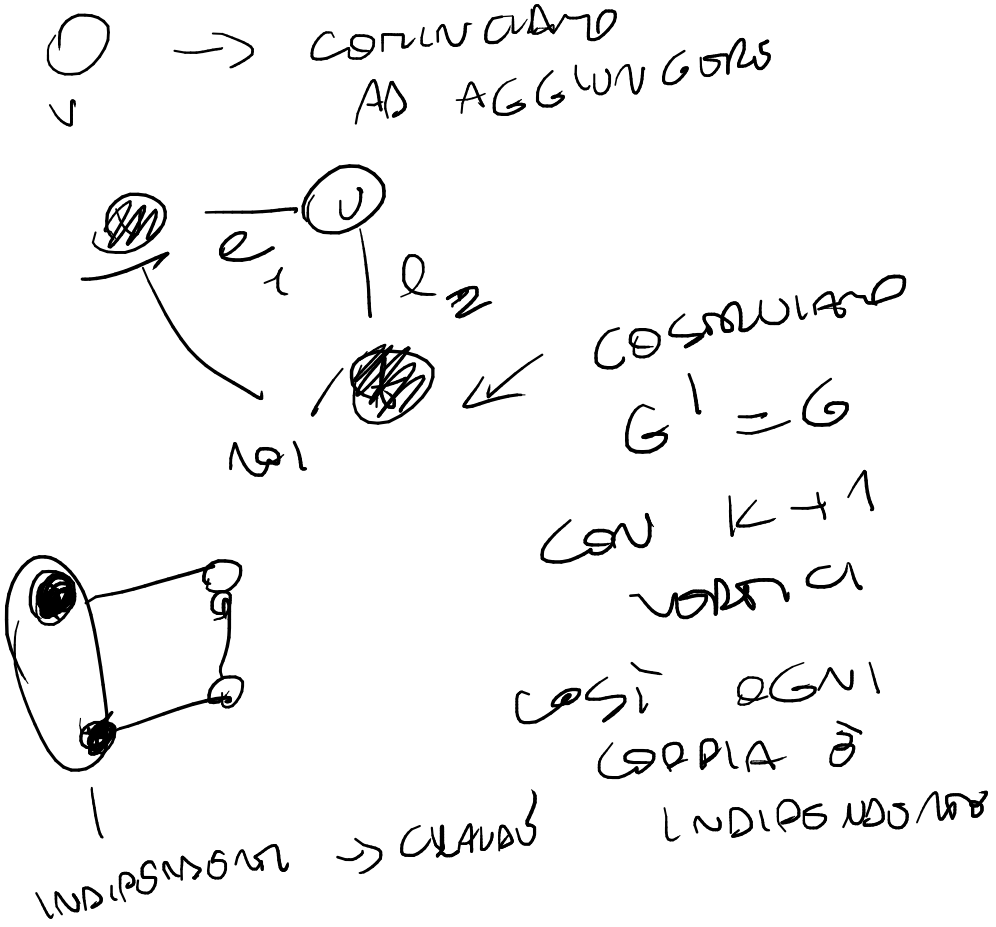
A

A è NP-hard (geared)

IND SET $\leq_m$ ALLOST

$G = (V, U)$

$\langle G, K \rangle$, S → verifica!
↓
sottoinsieme

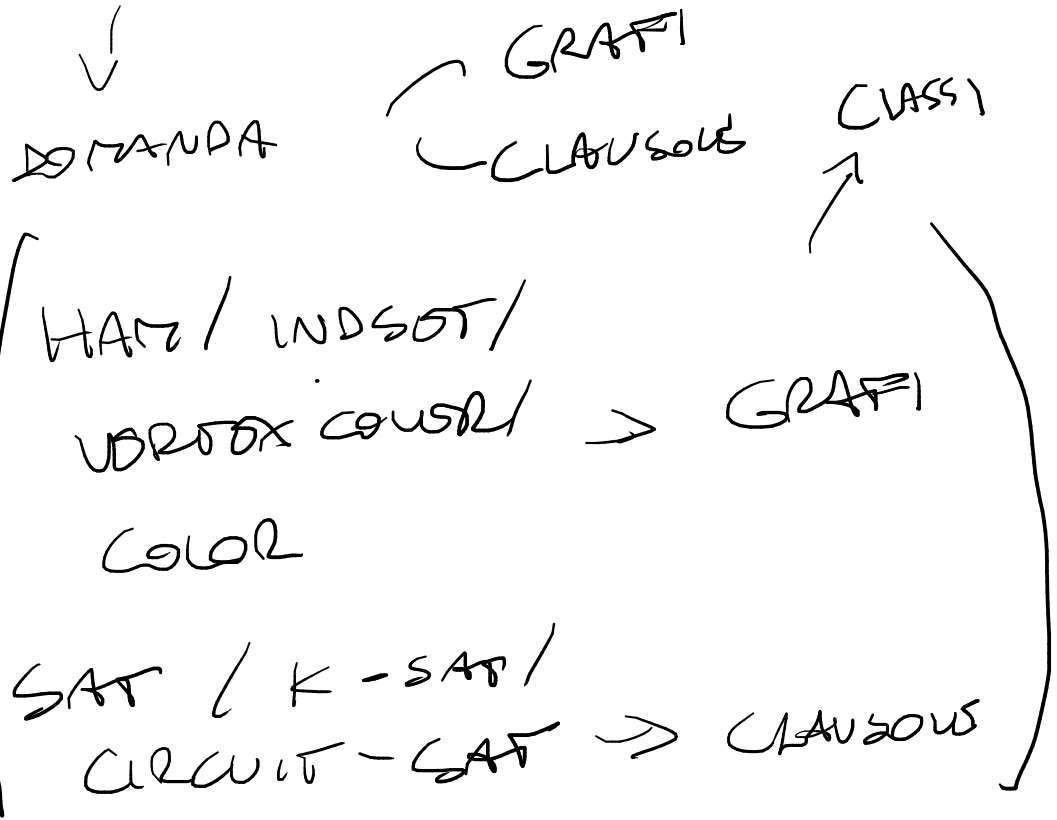


Sia R la seguente funzione di riduzione:
R = "Su input ⟨G,k⟩ dove G = (V,E):

Costruzione nuovo grafo: Crea G' = (V', E') dove:

V' = V ∪ {a, b} (aggiungi due nuovi vertici)
E' = E ∪ {(a,b)} (aggiungi arco tra i nuovi vertici)

Nuovo parametro: k' = k + 2
Restituisce ⟨G', k'⟩."



ALTRI 2 PROBLEMI

ESAMI

ESERCIZI ALGO

SUBSET - SUM SOTTOINSIEMI

$S = S_1 \cup S_2$

UNIONE

$S_1 \neq S_2$

$S = \{8\}$

$S_1 \in$ TARGET

S_1, S_2

$S_1 = \{1, 2, 4, 1\}$

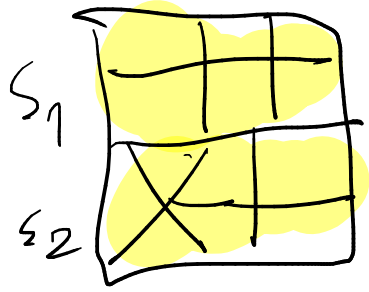
$S_2 = \{2, 2, 2, 2\}$

$\rightarrow$ STESSA SOMMA ELEMENTI DIVERSI

SET - PARTITIONING

$S_1 = S_2$

ELEMENTI DIVERSI



$S \rightarrow$

SET - PARTITIONING

SUBSET - SUM

TORNANDO A PUNTO (3)

$(\Leftrightarrow)$

SE IND. SOT ALLORA ALLOST

$(\rightarrow)$ 1

$I = K + 2$ $\quad S \subseteq I$

(K)

$(\leftarrow)$

SE ALLOST ALLORA IND.

$SE\ \exists S \rightarrow K$

$q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \rightarrow K+2 = I$