Classi fondamentali di complessità da tenere a mente

Sigla	Definizione operativa (decision problems)	Relazioni e fatti chiave	Esempi tipici
P	Esiste un algoritmo deterministico che decide l'istanza in tempo polinomiale	P ⊆ NP. Se troviamo un algoritmo polinomiale per un problema NP-completo otteniamo P = NP	*PATH* (raggiungibilità), *RELPRIME* (gcd = 1)
NP	Le istanze "YES" posseggono un certificato verificabile in tempo polinomiale (equiv.: TM nondeterministica polinomiale)	Contiene P. Non sappiamo se P = NP.	*Hamiltonian Cycle*, *COMPOSITES*
coNP	Complemento di un linguaggio in NP	NP ∩ coNP potrebbe essere stretto o coincidere con NP; aperto	*UNSAT* è in coNP
NP-hard	Ogni problema in NP si riduce in tempo polinomiale (≤_P) al problema in questione	Non richiede di essere in NP; basta la riducibilità	*Max Independent Set*, *Halting Problem* (non in NP)
NP-complete	Problema sia NP-hard sia appartenente a NP ("più difficili in NP")	Un algoritmo polinomiale per uno di essi implica P = NP	*CircuitSAT* (Cook–Levin), *SAT*, *3SAT*, *Vertex Cover*, *3-Color*

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 **kColor**
- 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**!

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

NP-HARD

① PROBLEMA IN NP → PROBLEMA NON RISOLVIBILE SEMPRE

MA VERIFICA IN TEMPO P LA SOLUZIONE

② NON SI RISOLVE IN MODO EFFICIENTE

$A \leq_m B$

↑ NP-HARD

→ SAT = BOOLEAN SATISFIABILITY

TRUE = $(a \land b \lor c)$ → ESPRESSIONE / FORMULA

CIRCUIT - SAT

Y / N

3 - SAT → a / b / c

2-SAT → a / b

SAT $\leq_m$ CIRCUIT - SAT

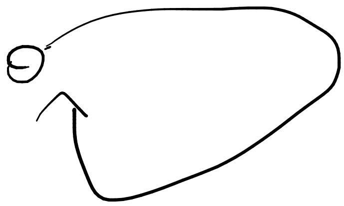
A B

NP → a/b/c Y / N

SAT

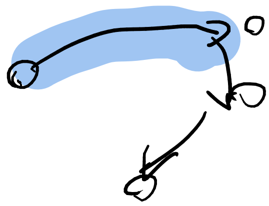
CIRCUIT

→ HAMILTONIAN CIRCUIT



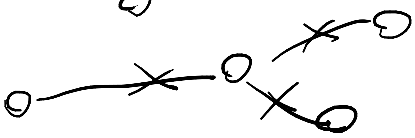
PARTO DA UN VERTICE E CI TORNO

→ VERTEX COVER



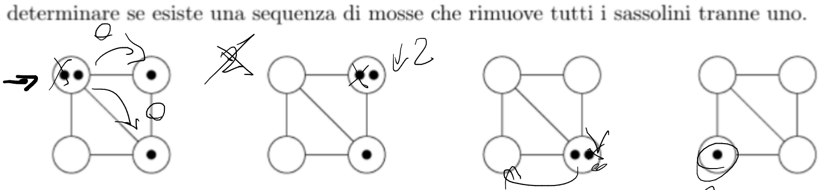
= COPRI TUTTO IL GRAFO

→ INDEPENDENT SET



= SCONNETTO IL GRAFO

4. Pebbling è un solitario giocato su un grafo non orientato G, in cui ogni vertice ha zero o più ciottoli. Una mossa del gioco consiste nel rimuovere due ciottoli da un vertice v e aggiungere un ciottolo ad un vertice u adiacente a v (il vertice v deve avere almeno due ciottoli all'inizio della mossa). Il problema PebbleDestruction chiede, dato un grafo $G = (V, E)$ ed un numero di ciottoli $p(v)$ per ogni vertice v, di determinare se esiste una sequenza di mosse che rimuove tutti i sassolini tranne uno.



Una soluzione in 3 mosse di PebbleDestruction.

Fornisci un verificatore per PebbleDestruction. → PARTE 1 → NP

NP-HARD

VERIFICA CHE PD SIA NP

VERIFICATORE

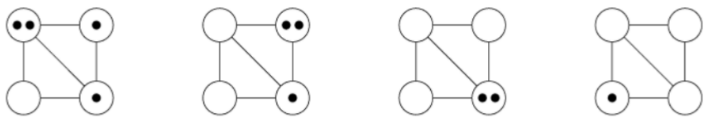
CERTIFICATO = INPUT CHE SEI IN NP

$G = \langle V, E \rangle$

DIMOSTRA CHE PD SIA NP-HARD

(1)

4. Pebbling è un solitario giocato su un grafo non orientato G, in cui ogni vertice ha zero o più ciottoli. Una mossa del gioco consiste nel rimuovere due ciottoli da un vertice v e aggiungere un ciottolo ad un vertice u adiacente a v (il vertice v deve avere almeno due ciottoli all'inizio della mossa). Il problema PEBBLEDESTRUCTION chiede, dato un grafo $G = (V, E)$ ed un numero di ciottoli $p(v)$ per ogni vertice v, di determinare se esiste una sequenza di mosse che rimuove tutti i sassolini tranne uno.



Una soluzione in 3 mosse di PEBBLEDESTRUCTION.

Fornisci un verificatore per PEBBLEDESTRUCTION.

F tempo P

CERTIFICATO → $G = (V, E)$ / $p(v)$ = INPUT BUONO PER NP

1. Dato un grafo G = (V, E)

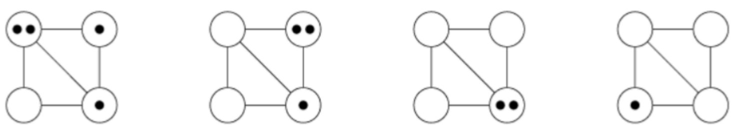
1.a. Parto da un vertice (sorgente) e con una mossa, dato il numero di sassolini p(v), mi muovo ad uno dei vertici adiacenti, rimuovendo un paio di sassolini.

1.b. Continuo finché non ho esaurito le mosse per percorrere tutti gli archi

2. Se ho esaurito tutti i vertici, ho coperto tutto il grafo e restituisce SI
3. Altrimenti NO

(2)

4. Pebbling è un solitario giocato su un grafo non orientato G, in cui ogni vertice ha zero o più ciottoli. Una mossa del gioco consiste nel rimuovere due ciottoli da un vertice v e aggiungere un ciottolo ad un vertice u adiacente a v (il vertice v deve avere almeno due ciottoli all'inizio della mossa). Il problema PEBBLEDESTRUCTION chiede, dato un grafo $G = (V, E)$ ed un numero di ciottoli $p(v)$ per ogni vertice v, di determinare se esiste una sequenza di mosse che rimuove tutti i sassolini tranne uno.



Una soluzione in 3 mosse di PEBBLEDESTRUCTION.

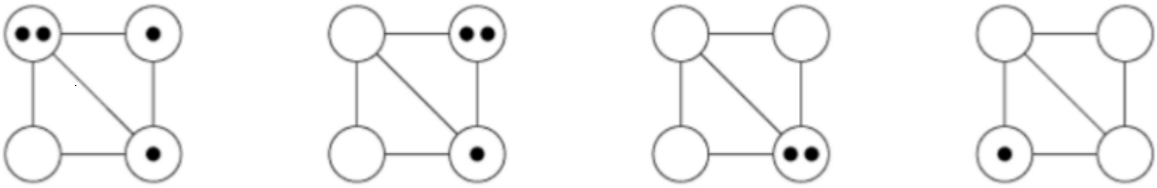
Fornisci un verificatore per PEBBLEDESTRUCTION.

→ LISTA PROBLEMI NP-HARD

$A \leq_m$ PEBBLE-DESTRUCTION

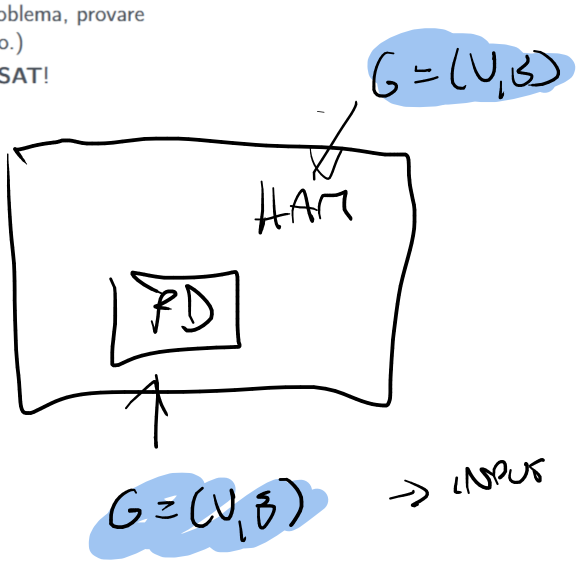
↓ TUA SCELTA

SOTTO-PROBLEMA



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**!



HAM $\leq_m$ PD

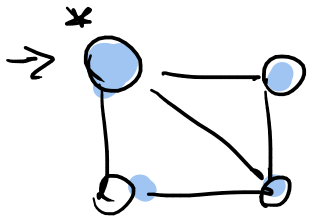
($\Longleftrightarrow$) = SE E SOLO SE

PD → P(V), G=(V,E)

→ TOGLI 2 SASSI DA UN VERTICE NE METTI 1 AGLI ADIACENTI

($\Longrightarrow$)

OGNI MOSSA DI PD EQUIVALE AD UNA MOSSA DI HAM



E VADO AVANTI PER OGNI VERTICE ADIACENTE

A → (U,V) → B

USANDO I TERMINI DI PD DESCRIVO HAM [OGNI MOSSA, GRAZIE AL CERTIFICATO, TOGLI DUE SASSI TORNANDO AD UN VERTICE ADIACENTE A *]

(⇐)

PD = G = (V, E)

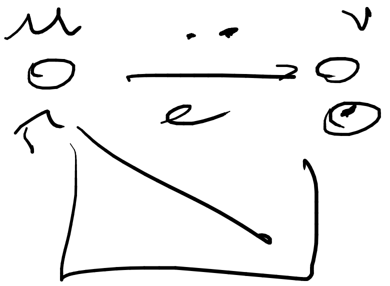
PD → HAM

MOSSA → RIMUOVO 2 SASSI

PER CORRERE TUTTO IL GRAFO (CIRCUITO)

RIMUOVO COPPIA DI SASSI

SAPENDO CHE RITORNO VICINO AL V. DI PARTENZA



AVENDO SVUOTATO TUTTO IL GRAFO QUELLO DI PARTENZA È VUOTO

[NP-COMPLETO] { È IN NP / È NP-HARD

PD È

---

4. (8 punti) *Supponiamo che un impianto industriale costituito da m linee di produzione identiche debba eseguire n lavori distinti. Ognuno dei lavori può essere svolto da una qualsiasi delle linee di produzione, e richiede un certo tempo per essere completato. Il* problema del bilanciamento del carico *(*LoadBalance*) chiede di trovare un* assegnamento *dei lavori alle linee di produzione che permetta di completare tutti i lavori entro un tempo limite k.*

$\Sigma m \leq K$

Più precisamente, possiamo rappresentare l'input del problema con una tripla $\langle m, T, k\rangle$ dove:

- *m è il numero di linee di produzione;*
- *$T[1 \dots n]$ è un array di numeri interi positivi dove $T[j]$ è il tempo di esecuzione del lavoro j;*
- *k è un limite superiore al tempo di completamento di tutti i lavori.*

Per risolvere il problema vi si chiede di trovare un array $A[1 \dots n]$ con gli assegnamenti, dove $A[j] = i$ significa che il lavoro j è assegnato alla linea di produzione i. Il tempo di completamento (o makespan*) di A è il tempo massimo di occupazione di una qualsiasi linea di produzione:*

[MAKE]SPAN = QUANTI LAVORI RIESCO A METTERE PER MAX. PRODUZIONE?

$$\text{makespan}(A) = \max_{1 \leq i \leq m} \sum_{A[j]=i} T[j] \rightarrow 5+4+3+3 \leq 15$$

Load Balance *è il problema di trovare un assegnamento con makespan minore o uguale al limite superiore k:*

ES. INPUT BUONO...

$$\text{LoadBalance} = \{\langle m, T, k\rangle \mid \textit{esiste un assegnamento } A \textit{ degli } n \textit{ lavori su } m \textit{ linee di produzione tale che } \text{makespan}(A) \leq k\}$$

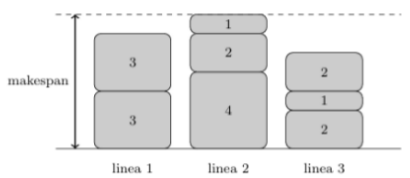


Figura 1: Esempio di assegnamento dei lavori $T = \{1, 1, 2, 2, 2, 3, 3, 4\}$ su 3 linee con makespan 7.

(a) *Dimostra che* LoadBalance *è un problema NP.*
(b) *Dimostra che* LoadBalance *è NP-hard, usando* SetPartitioning *come problema NP-hard di riferimento.*

(a) → DIMOSTRA MK ≤ NP

∀i → ASSEGNO J

↓

A[J] = i

WHILE ≤ K

→ ASSEGNA I LAVORI

A[J] = i

RISULTATI T. MAX

4. **(8 punti)** *Supponiamo che un impianto industriale costituito da m linee di produzione identiche debba eseguire n lavori distinti. Ognuno dei lavori può essere svolto da una qualsiasi delle linee di produzione, e richiede un certo tempo per essere completato. Il problema del bilanciamento del carico (*LoadBalance*) chiede di trovare un assegnamento dei lavori alle linee di produzione che permetta di completare tutti i lavori entro un tempo limite k.*

Più precisamente, possiamo rappresentare l'input del problema con una tripla (m, T, k) dove:

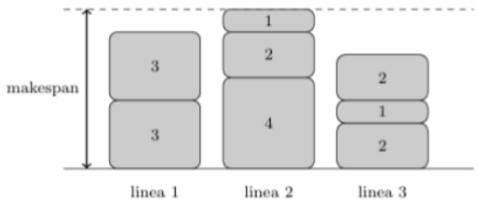
- *m è il numero di linee di produzione;*
- *$T[1 \dots n]$ è un array di numeri interi positivi dove $T[j]$ è il tempo di esecuzione del lavoro j;*
- *k è un limite superiore al tempo di completamento di tutti i lavori.*

Per risolvere il problema vi si chiede di trovare un array $A[1 \dots n]$ con gli assegnamenti, dove $A[j] = i$ significa che il lavoro j è assegnato alla linea di produzione i. Il tempo di completamento (o makespan*) di A è il tempo massimo di occupazione di una qualsiasi linea di produzione:*

$$\text{makespan}(A) = \max_{1 \le i \le m} \sum_{A[j]=i} T[j]$$

Load Balance *è il problema di trovare un assegnamento con makespan minore o uguale al limite superiore k:*

$$\text{LoadBalance} = \{\langle m, T, k\rangle \mid \textit{esiste un assegnamento } A \textit{ degli } n \textit{ lavori su } m \textit{ linee di produzione tale che } \text{makespan}(A) \le k\}$$



SOLUZIONE SEMPRE OTTIMA → SE = K → ASSEGNO SOLO MIGLIORE!

Figura 1: Esempio di assegnamento dei lavori $T = \{1, 1, 2, 2, 2, 3, 3, 4\}$ su 3 linee con makespan 7.

(a) *Dimostra che* LoadBalance *è un problema NP.*
(b) *Dimostra che* LoadBalance *è NP-hard, usando* SetPartitioning *come problema NP-hard di riferimento.*

SET-PARTITIONING → a / b = K ? → LOAD BALANCE

SOTTOINSIEMI

SUBSET SUM

↓

a + b + c - ... = K

In number theory and computer science, the **partition problem**, or **number partitioning**,[1] is the task of deciding whether a given multiset S of positive integers can be partitioned into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of the numbers in S_2. Although the partition problem is NP-complete, there is a pseudo-polynomial time dynamic programming solution, and there are heuristics that solve the problem in many instances, either optimally or approximately. For this reason, it has been called "th easiest hard problem".[2][3]

$S_1 = S_2 = K$

DUE SOTTOINSIEMI I LORO = STESSA SOMMA

4. (8 punti) *Supponiamo che un impianto industriale costituito da m linee di produzione identiche debba eseguire n lavori distinti. Ognuno dei lavori può essere svolto da una qualsiasi delle linee di produzione, e richiede un certo tempo per essere completato. Il* problema del bilanciamento del carico *(*LoadBalance*) chiede di trovare un assegnamento dei lavori alle linee di produzione che permetta di completare tutti i lavori entro un tempo limite k.*

Più precisamente, possiamo rappresentare l'input del problema con una tripla $\langle m, T, k\rangle$ *dove:*

- m *è il numero di linee di produzione;*
- $T[1 \dots n]$ *è un array di numeri interi positivi dove* $T[j]$ *è il tempo di esecuzione del lavoro* j*;*
- k *è un limite superiore al tempo di completamento di tutti i lavori.*

Per risolvere il problema vi si chiede di trovare un array $A[1 \dots n]$ *con gli assegnamenti, dove* $A[j] = i$ *significa che il lavoro* j *è assegnato alla linea di produzione* i*. Il tempo di completamento (o* makespan*) di* A *è il tempo massimo di occupazione di una qualsiasi linea di produzione:*

$$\text{makespan}(A) = \max_{1 \le i \le m} \sum_{A[j]=i} T[j]$$

Load Balance *è il problema di trovare un assegnamento con makespan minore o uguale al limite superiore k:*

$$\text{LoadBalance} = \{\langle m, T, k\rangle \mid \textit{esiste un assegnamento } A \textit{ degli } n \textit{ lavori su } m \textit{ linee di produzione tale che } \text{makespan}(A) \le k\}$$

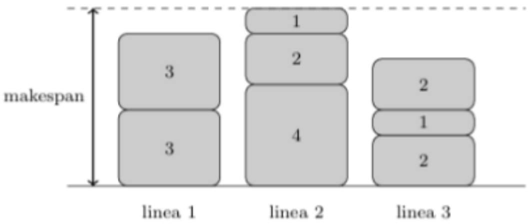


Figura 1: Esempio di assegnamento dei lavori $T = \{1, 1, 2, 2, 2, 3, 3, 4\}$ su 3 linee con makespan 7.

(a) *Dimostra che* LoadBalance *è un problema NP.*
(b) *Dimostra che* LoadBalance *è NP-hard, usando* SetPartitioning *come problema NP-hard di riferimento.*

LB ≤ NP-HARD

(b)

SET-PARTITIONING $\le_m$ LOAD BALANCE

$\langle S_1, S_2 \rangle$ $\langle m, T, k \rangle$

JOBS TIMES = TOTALE UPPER BOUND

$S_1 + S_2 = K$ A $\le_m$ B $\sum_i m \le k$

$A[\,] = S$

$S_1 \ne S_2$

(b) $(\Rightarrow)$

$m \to \underline{max}$ $[T \le K]$ $T_1 \le \frac{K}{2}$, $T_2 \le \frac{K}{2}$ S_1 S_2

$O(K)$ --- $[\underline{\quad}_{S_1} \,\|\, \underline{\quad}_{S_2}]$

MAX $\underbrace{[\qquad]}_{S_1}$ MAX $\underbrace{[\qquad]}_{S_2}$ $\le K$

SET-PARTITION

$S_1 \to$ MAX $\le K$
$+$
S_2

ATTACCHIAMO I DUE MAX DI S_1, S_2 $(\le K)$

(⇒) → S_1 → T. RANDOM ... $\leq K$

S_2 ↗

LOAD-BALANCE

MAX → O(1) $\Sigma \leq k$!

$\langle S_1, S_2 \rangle \rightarrow \langle m, T, k \rangle \rightarrow$!

---

2. (12 punti) I *grawlix* sono sequenze di simboli senza senso che sostituiscono le parolacce nei fumetti.

L M w

Un linguaggio è *volgare* se contiene almeno un grawlix. Considera il problema di determinare se il linguaggio di una TM è volgare.

(a) Formula questo problema come un linguaggio $GROSS_{TM}$.
(b) Dimostra che il linguaggio $GROSS_{TM}$ è indecidibile.

→ $A \leq_m B$

a)

$GROSS_{TM} = \{ \langle \ \rangle \mid \ \ \}$ $A_{TM} / E_{TM} / EQ_{TM}$ COMPLEMENTO

$\langle M, w \rangle$ → VERIFICA SE ∃ w PER M

$GROSS_{TM} = \{ \langle L, w \rangle \mid$ L è volgare se ∃ ! w = GRAWLIX $\}$

$\langle M \rangle$ | verifica se L volgare   RSGEX

b) DIMOSTRA $GROSS_{TM}$ INDECIDIBILE

$A \leq_m B$ ⇒

↓ INDECIDIBILE

$[A_{TM} \leq_m GROSS_{TM}]$

F A_{TM} $\langle M, w \rangle$ Y   N

GROSS $\langle N, x \rangle$ → GRAWLIX → */#... @

w = *

← SEGNA CON TM   ∀ CARATTERE

F. DI RIDUZIONE

F = SU INPUT $\langle M, w \rangle$

A_{TM} = A VERIFICA SE w

1. M COPIA SUL NASTRO TUTTO L'INPUT

1.a ESEGUE N SU INPUT X

1.b SE X CONTIENE ALMENO UN CAR. GRAWLIX, CONTINUA FINCHÉ NON CORRONI W

$\rightarrow N$ — LOOP FINCHÉ NON HAI W $\rightarrow x \subseteq W$ $\quad$ X $\quad$ $[\,x\ \dots\,]\,W$ VOLGARE

2.. RITORNA $\langle M_w \rangle \longrightarrow W \subseteq x_1 \dots x_m$

SE M ACCETTA, ACCETTA / ALTRIMENTI RIFIUTA

$\langle M, w \rangle \in A_{TM} \Rightarrow \langle M_w \rangle \in GROSS_{TM}$ $\quad A_{TM}$ ACCETTA

($\Rightarrow$) $\underline{w \in A_{TM}}$

$\Rightarrow x_1 \dots x_m =$ GRANLUX $\quad M_w =$ VOLGARE

($\Leftarrow$) $w \notin A_{TM} \rightarrow$ PAROLA NON VOLGARE $\rightarrow$ VA IN LOOP

RIFIUTA

PERSISTENT $\rightarrow G = CFG \rightarrow w$ $\quad G_{TM} = \emptyset$

G_{TM} PERSISTENT $\rightarrow \emptyset \rightarrow$ ACCETTA

$\forall w \rightarrow$ LOOP / RIFIUTA

---

2. (12 punti) Un linguaggio B è *emozionato* se ogni stringa in B assume la forma ww per qualche $w \in \{0,1\}^*$. Ad esempio, sia $\{00, 1111, 1010\}$ che $\emptyset$ sono linguaggi emozionati, mentre $\{00, 10\}$ non lo è. Considera il problema di determinare se il linguaggio di una TM M è emozionato.

(a) Formula questo problema come un linguaggio EX_{TM}.

(b) Dimostra che il linguaggio EX_{TM} è indecidibile.

VOLENDO PER RIDUZIONE

EQ_{TM} $\langle M_1, M_2 \rangle \Rightarrow \underline{M_1 = M_2}$

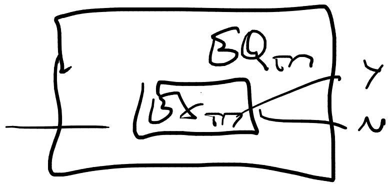
(a) $EX_{TM} = \{ \langle B, w \rangle \mid B$ è emozionato se $|W| = ww \}$

CARDINALITÀ

(b) $EX_{TM} = ww \rightarrow$ 00 / 11 / 1010

$$EQ_{TM} \le_m EX_{TM}$$

M_1 M_2



M_1 M_2 ? $M_1 = M_2$

| | = CARD. PARI

EQ_{TM} ACCETTA

F = SU INPUT $\langle M_1, M_2, w \rangle$:

TM

NASTRO 1 = M_1 / METÀ

NASTRO 2 = M_2 / METÀ

M_2

[#[10][10]#] → SIMULARE....

M_1 = METÀ

Y → EQ_{TM} → RIGHT!

[0 0 1 1 | 0 0 1 1

• = PALLINI PER CONTARE...

→ ACCETTA / RIFIUTA

→ [00 11 | 00 11] $\frac{\text{PARI}}{\text{NO}}$

w

1

GUARDO COPPIE

M_1 / M_2 = EQ_{TM}

$L^* = \{ .... \}$

M_1 M_2 M_1 M_2 M_2

{ 00 / 11 / 1010 }

ww ww ww

M_1

STOP ALLA SINGOLA METÀ = $\left[\frac{w}{2}\right]$

00 < 0 0

11 < 1 1

1010 < 10 10

ww ww

00 / 1010

M_1 M_2

[----](•)[----]

F → INPUT $\langle M_1, M_2, w \rangle$

REDUCTION FUNCTION

→ COPIA SUL NASTRO w E [DIVIDE A METÀ] → $\left|\frac{w}{2}\right|$

→ $M_1 \rightarrow [0; \frac{w}{2})$ } YES → ACCETTA → EQ_{TM}

→ $M_2 \rightarrow [\frac{w}{2}, w]$

RIFIUTA / VA IN LOOP

1° NASTRO

2° NASTRO → RITORNO $\langle M_1, M_2, w \rangle$