3. **(12 punti)** Date due stringhe w e t, diciamo che t è una *permutazione* di w se t e ha gli stessi simboli di w con ugual numero di occorrenze, ma eventualmente in un ordine diverso. Per esempio, le stringhe 01011,e 00111 sono entrambe permutazioni di 11001.

Dimostra che se $B \subseteq \{0,1\}^*$ è un linguaggio regolare, allora il linguaggio

$$\text{SCRAMBLE}(B) = \{t \in \{0,1\}^* \mid t \text{ è una permutazione di qualche } w \in B\}$$

è un linguaggio context-free.

→ PDA / CFG

SE CFG → G CHE RICONOSCE

t = PERMUTAZIONE DI W

W = PAROLA $\in G$

2. **(8 punti)** *Per ogni linguaggio L, sia* $\text{prefix}(L) = \{u \mid uv \in L$ *per qualche stringa* $v\}$. *Dimostra che se L è un linguaggio context-free, allora anche* $\text{prefix}(L)$ *è un linguaggio context-free.*

Se L è un linguaggio context-free, allora esiste una grammatica G *in forma normale di Chomski* che lo genera. Possiamo costruire una grammatica G' che genera il linguaggio $\text{prefix}(L)$ in questo modo:

- per ogni variabile V di G, G' contiene sia la variabile V che una nuova variabile V'. La variabile V' viene usata per generare i prefissi delle parole che sono generate da V;
- tutte le regole di G sono anche regole di G';
- per ogni variabile V di G, le regole $V' \to V$ e $V' \to \varepsilon$ appartengono a G;
- per ogni regola $V \to AB$ di G, le regole $V' \to AB'$ e $V' \to A'$ appartengono a G';
- se S è la variabile iniziale di G, allora S' è la variabile iniziale di G'.

→ COSTRUIAMO G' ALTRA CFG (STESSO N. OCCORRENZE PER SIMBOLO)

$G' = (V', \Sigma', R', S') \iff G = (V, \Sigma, R, S)$

- $\forall$ SIMBOLO $\in L$, ESISTE UNA PERMUTAZIONE PER OGNI W

$(A \to 01B \quad B \to 10A)$ IDEA → $\Sigma = \{0,1\}^*$

$\Sigma' = \Sigma$

$$\begin{bmatrix} V \to AB \\ V \to v \end{bmatrix} \qquad \begin{bmatrix} V' \to BA \\ V' \to v \end{bmatrix}$$

CHOMSKY $\equiv$ CHOMSKY

- $S' = S$

$(\iff)$ SE $\exists\, t$ (PERMUTAZIONI) ALLORA SCRAMBLE è CFL

---

3. **(12 punti)** Date due stringhe w e t, diciamo che t è una *permutazione* di w se t e ha gli stessi simboli di w con ugual numero di occorrenze, ma eventualmente in un ordine diverso. Per esempio, le stringhe 01011,e 00111 sono entrambe permutazioni di 11001.

Dimostra che se $B \subseteq \{0,1\}^*$ è un linguaggio regolare, allora il linguaggio

$$\text{SCRAMBLE}(B) = \{t \in \{0,1\}^* \mid t \text{ è una permutazione di qualche } w \in B\}$$

è un linguaggio context-free.

∃ PDA (∀ SIMBOLO 0, 1 — ∀ POP → STESSO NUMERO DI PUSH
(∀ PUSH → STESSO NUMERO DI POP

— ∀ SIMBOLO ∈ Σ → PUSH DELLA PILA FINO A CONSUMARE TUTTA LA STRINGA W

P PDA = CFG G (SCAMBIO)

QUINDI

ESISTE P' EQUIVALENTE

∀ SIMBOLO ∈ Σ' → POP DELLA PILA FINO A CONSUMARE TUTTA LA STRINGA W

SAPENDO CHE L'OUTPUT T

TALI CHE [N. PUSH / N. POP] è T

UNA VOLTA CONSUMATO TUTTO L'INPUT, SI TROVANO ENTRAMBI A PILA VUOTA = ε

STATO PUSH POP → q, i → j

```
A[q,i,j] → 0 A[δ(q,0), i-1, j] se i > 0
 - A[q,i,j] → 1 A[δ(q,1), i, j-1] se j > 0
 - A[q,0,0] → ε se q è finale
```

→ FORMALE → P'

2. (12 punti) Una stringa w è *palindroma* se rimane uguale letta da sinistra a destra e da destra a sinistra, cioè se $w = w^R$. Un linguaggio $B \subseteq \{0,1\}^*$ è *quasi-palindromo* se contiene al più una stringa non palindroma. Ad esempio, sia $\{00, 11011, 1001\}$ che $\{00, 101\}$ sono linguaggi quasi-palindromi, mentre $\{00, 10, 100\}$ non lo è. Considera il problema di determinare se il linguaggio di una TM M è quasi-palindromo.

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

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

{00, 1001, 100} ANCHE QUASI-PAL.

(a) $QPAL_{TM}$ = { <B, W> | B è QUASI-PALINDROMO (SE ∃! W NON PALINDROMA ∈ B

∀ ⟨M, w⟩ | M è una TM, e
w è UNA STRINGA QUASI-PALINDROMA

`QPALTM = {⟨M⟩ | M è una TM e L(M) è quasi-palindromo}`

2. (12 punti) Una stringa w è *palindroma* se rimane uguale letta da sinistra a destra e da destra a sinistra, cioè se $w = w^R$. Un linguaggio $B \subseteq \{0,1\}^*$ è *quasi-palindromo* se contiene al più una stringa non palindroma. Ad esempio, sia $\{00, 11011, 1001\}$ che $\{00, 101\}$ sono linguaggi quasi-palindromi, mentre $\{00, 10, 100\}$ non lo è. Considera il problema di determinare se il linguaggio di una TM M è quasi-palindromo.

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

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

$A \leq_m B \Rightarrow B$ è $QPAL_{TM}$ sse $\exists!\, w$ QUASI PAL.

$\Sigma = \{0,1\}^*$

F = SU INPUT ⟨M, w⟩:

- IGNORA w
- ESEGUI M' SU x

CASI BASO: $x = \varepsilon$, $x = 0$, $x = 1$ (w)

- SE x CONTIENE UNA STRINGA PALINDROMA, M' ACCETTA, SE x NON CONTIENE (w), VA IN LOOP PER TROVARE UNA STRINGA PALINDROMA, RIFIUTA ALTRIMENTI

```
S = "Su input ⟨M, w⟩:
1. Costruisci M':
   M' = "Su input x:
   a) Se x = 0 o x = 1: accetta
   b) Se x = 01: simula M su w
      - Se M accetta: accetta
      - Altrimenti: rifiuta
   c) Se x = 10: accetta
   d) Altrimenti: rifiuta"

2. Esegui R su ⟨M'⟩
3. Se R accetta: accetta
4. Se R rifiuta: rifiuta"
```

- COSTRUISCI ⟨M, w⟩

$\boxed{\boxed{01}\ 10}$ w → M, x → M'

→ RONDORA GENERALE

M' su x → $\frac{w}{2}$

M su w → $\frac{w}{2}$

} w → YES, → NO

→ CASI BASO → ε / 1 CARATTERE

1. (12 punti) Una macchina di Turing con "copia e incolla" (CPTM) è una macchina di Turing deterministica a singolo nastro, che può copiare e incollare porzioni di nastro. Le operazioni che una CPTM può fare sono le seguenti:

- selezionare l'inizio della porzione di nastro da copiare;
- selezionare la fine della porzione di nastro da copiare;
- copiare la porzione di nastro selezionata, sovrascrivendo il contenuto della cella corrente e di tante celle a destra della cella corrente quante sono le celle necessarie per effettuare la copia;
- fare le normali operazioni di scrittura e spostamento a sinistra o a destra della testina.

S S

[00111] # 00111 COPIATA E INCOLLA

W ORIGINALE

Fare una operazione di copia senza che sia stata selezionata una porzione di nastro non ha effetto.

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

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

(c) CPTM = T ⟼ {L, R, SELECT, COPY, PASTE}

(b) ∃ S ⇒ SINGLE TAPE EQUIVALENTE CON I SEGUENTI PASSI DI SIMULAZIONE

- $\delta(q,a) = (r,b,L)$ / .... (r,b,R) = SOLITO

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

SELECT → SPOSTA TUTTO A DX L'INPUT SELEZIONATO

1 0 0 # 0 1 0

SIMBOLO PRIMA

SIMBOLO DOPO (1 A DX)

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

PARTE SELEZIONATA DA $\#_{SX}$ A $\#_{DX}$

START_ORIGIN

.... # 0 1 0 # 1 0 0 # — END_ORIGIN

→ SCAN A DX DI |W| POSIZIONI

FINE COPIA

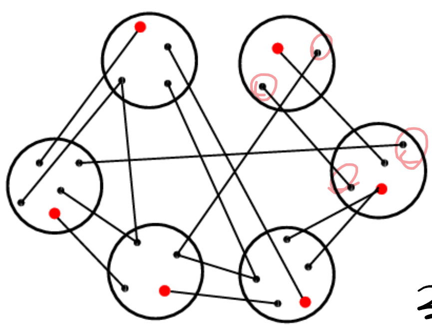
- RITORNA AL PASSO 2 (IN CASO DI NON FINE INPUT → W)

3. (12 punti) La Rettrice dell'Università di Padova vuole costituire una commissione selezionando un membro per ogni dipartimento dell'ateneo. Sappiamo che alcuni dei docenti si detestano a vicenda. Per evitare scontri, la Rettrice non vuole avere membri della commissione che si detestano tra di loro. Se ogni dipartimento è un insieme D_i di docenti, e se I è la relazione di inimicizia tra docenti, una *buona commissione* è un insieme C di docenti tali che:

- ogni dipartimento ha esattamente un rappresentante in commissione;
- non esistono coppie di docenti che si detestano.

La figura seguente mostra un esempio di istanza del problema dove i cerchi sono i dipartimenti, i punti sono i docenti e gli archi collegano docenti che si detestano. I docenti evidenziati in rosso sono i componenti di una buona commissione.

SE DIVERSI → PIÙ COMBO POSSIBILI A TRUE → 1



3-SAT

→ {0, 1, 0}

→ {~~1~~, 1, 1}

3-SAT → $(a \vee b \wedge c)$

Definiamo il linguaggio

$$COMMITTEE = \{\langle D_1, \ldots, D_m, I\rangle \mid \text{esiste una buona commissione } C\}.$$

(a) Dimostra che *COMMITTEE* è un problema NP.

(b) Dimostra che *COMMITTEE* è NP-hard, usando *3SAT* come problema NP-hard di riferimento.

XOR → 0 1 / 1 0 → 1 ALTRIMENTI NO

3. **(12 punti)** La Rettrice dell'Università di Padova vuole costituire una commissione selezionando un membro per ogni dipartimento dell'ateneo. Sappiamo che alcuni dei docenti si detestano a vicenda. Per evitare scontri, la Rettrice non vuole avere membri della commissione che si detestano tra di loro. Se ogni dipartimento è un insieme D_i di docenti, e se I è la relazione di inimicizia tra docenti, una *buona commissione* è un insieme C di docenti tali che:

- ogni dipartimento ha esattamente un rappresentante in commissione;
- non esistono coppie di docenti che si detestano.

La figura seguente mostra un esempio di istanza del problema dove i cerchi sono i dipartimenti, i punti sono i docenti e gli archi collegano docenti che si detestano. I docenti evidenziati in rosso sono i componenti di una buona commissione.

a) COMMITTEE

↓

$\langle D_1, \dots, D_m, I \rangle$

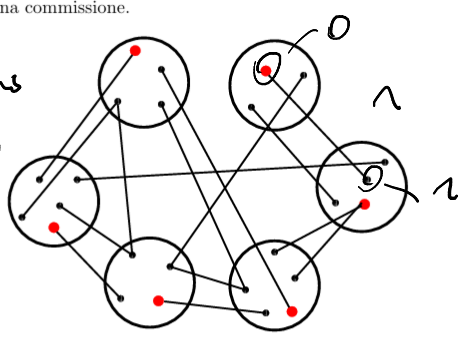
C ∃ UNA COMMISSIONE C

FORMATA DA DOCENTI TALI CHE

≥ 3 COMMISSIONI

↓

DIPARTIMENTO



DOCENTI

→ ∀ COPPIA $(D_1, D_2) \notin I$

Definiamo il linguaggio

$$COMMITTEE = \{\langle D_1, \dots, D_m, I\rangle \mid \text{esiste una buona commissione } C\}.$$

→ CONTROLLA CHE ∀ COPPIA

(D_2, D_3) --- $(D_{m-1}, D_m) \notin I$

(a) Dimostra che *COMMITTEE* è un problema NP.

(b) Dimostra che *COMMITTEE* è NP-hard, usando *3SAT* come problema NP-hard di riferimento.

ALMENO 2 COPPIE FORMANO UN DIPARTIMENTO

→ $(D_1, D_2) \vee (D_2, D_3) \wedge (D_3, D_4)$

↓

a, b, c

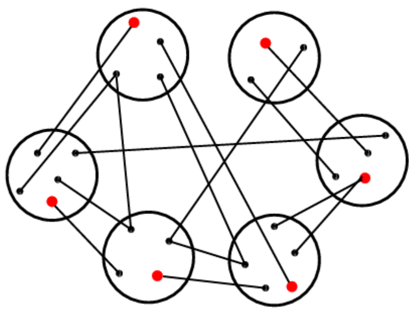
∃ DIP. CON BUONA COMMISSIONE C

SSE ∀ COPPIA, NO I

3. **(12 punti)** La Rettrice dell'Università di Padova vuole costituire una commissione selezionando un membro per ogni dipartimento dell'ateneo. Sappiamo che alcuni dei docenti si detestano a vicenda. Per evitare scontri, la Rettrice non vuole avere membri della commissione che si detestano tra di loro. Se ogni dipartimento è un insieme D_i di docenti, e se I è la relazione di inimicizia tra docenti, una *buona commissione* è un insieme C di docenti tali che:

- ogni dipartimento ha esattamente un rappresentante in commissione;
- non esistono coppie di docenti che si detestano.

La figura seguente mostra un esempio di istanza del problema dove i cerchi sono i dipartimenti, i punti sono i docenti e gli archi collegano docenti che si detestano. I docenti evidenziati in rosso sono i componenti di una buona commissione.



Definiamo il linguaggio

$$COMMITTEE = \{\langle D_1, \dots, D_m, I\rangle \mid \text{esiste una buona commissione } C\}.$$

(a) Dimostra che *COMMITTEE* è un problema NP.

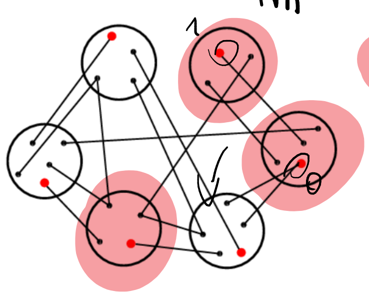
(b) Dimostra che *COMMITTEE* è NP-hard, usando *3SAT* come problema NP-hard di riferimento.

3-SAT $\leq_m$ COMMITTEE

$(a \wedge b \vee c)$ = 3-SAT

1 INSIEME ∧

2 INSIEMI ALLA VOLTA



↓

INIMICIZIA

$\{\langle D_1, \dots, D_m, I \rangle, C\}$

↑

COMMITTEE

$D_1 \to C_1$

$D_2 \to C_2$ $\} \notin I$?

$\underbrace{(D_1, D_2)}_{1} \wedge \underbrace{(D_3, D_4)}_{0} \vee \underbrace{(D_2, D_5)}_{1}$ TRUE / FALSE ?

NO INIMICIZIA

$\forall$ TRIPLETTA → RICORSIVAMENTE VALUTO PER RELAZIONE

$S_1 \wedge S_2 \neq S_2 \wedge S_3$

⇒ COMMITTEE → 3-SAT

$\langle D_1, \dots, D_m, C \rangle$

$\exists C \; \{ (D_1, D_2) \wedge (D_2 \wedge D_3) \vee (D_4 \wedge D_5)) \}$

$S_1 \wedge S_2 \wedge S_3 \neq F$

$1 \; \vdots \; m$

CONTROLLO TUTTO .... (CERTIFICATO C)

V = "Su input $\langle\langle D_1, \dots, D_m, I\rangle, C\rangle$:
1. Verifica $|C \cap D_i| = 1$ per ogni i
2. Verifica che non esistano $(d_1, d_2) \in I$ con $d_1, d_2 \in C$
3. Accetta se entrambe valgono"

NP

PUNTO A = CERTIFICATO

Riduzione da 3SAT:
Data formula $\phi = C_1 \wedge \dots \wedge C_m$, costruiamo:
Dipartimenti:

Per ogni variabile x_i: dipartimento $D_i = \{x_i, \neg x_i\}$
Per ogni clausola C_j: dipartimento $D'_j = \{c_{j,1}, c_{j,2}, \dots, c_{j,k}\}$ (gadget di scelta)

Inimicizie:

$(x_i, \neg x_i) \in I$ per ogni variabile
Collegamenti tra gadget clausola e variabili per forzare soddisfacimento

La riduzione preserva soddisfacibilità ⇔ esistenza commissione.

Un gadget è una "sottostruttura" che simula un elemento del problema di partenza nel problema di arrivo. È come un "pezzo di LEGO" che:

Ha un comportamento ben definito
Interagisce correttamente con altri gadget
Preserva le proprietà logiche del problema originale

I gadget sono "componenti" o "sottostrutture" che vengono usati nelle riduzioni per "simulare"
gli elementi del problema di partenza nel problema di arrivo. Nel caso specifico:

Gadget per variabili: simulano le variabili booleane e i loro possibili valori (true/false)
Gadget per clausole: simulano le clausole e assicurano che almeno un letterale sia soddisfatto

1. Gadget per Variabili

Per ogni variabile **x** in φ, creiamo:

```
Gadget variabile x:    x ⟷ ¬x
                       |     |
                   (due nodi collegati)
```

Significato:

- Una vertex cover deve scegliere **almeno uno** dei due nodi
- Scegliere **x** = assegnare **TRUE** alla variabile
- Scegliere **¬x** = assegnare **FALSE** alla variabile
- **Non possiamo scegliere entrambi** (sarebbe contraddittorio e costoso)

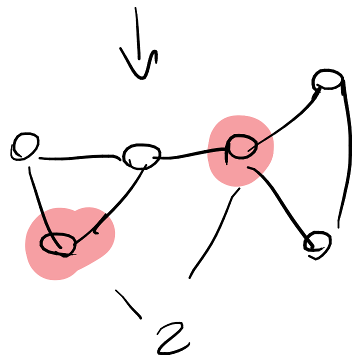
2. Gadget per Clausole

Per ogni clausola **(x ∨ y ∨ ¬z)**, creiamo:

```
Gadget clausola:          x
                         / \
                        /   \
                       y ── ¬z

                (triangolo con 3 letterali)
```

VERTEX COVER



VERTICI DIVERSI E SCOLLEGATI

$\rightarrow I \rightarrow X \neq \neg X$

$\rightarrow$ ALTERNATIVE

1. (12 punti) Data una stringa $w \in \Sigma^*$, definiamo una operazione che scambia di posizione i caratteri della stringa a due a due:

$$\text{SWAP}(w) = \begin{cases} \varepsilon & \text{se } w = \varepsilon \\ a & \text{se } w = a \text{ con } a \in \Sigma \\ a_1 a_0 \text{SWAP}(u) & \text{se } w = a_0 a_1 u \text{ con } a_0, a_1 \in \Sigma, u \in \Sigma^* \end{cases}$$

Per esempio, $\text{SWAP}(\texttt{ABCDE}) = \texttt{BADCE}$. $\rightarrow$ SCAMBIA COPPIA DI CARATTERI

Dimostra che se $L \subseteq \Sigma^*$ è un linguaggio regolare, allora anche il seguente linguaggio è regolare:

$$\text{SWAP}(L) = \{\text{SWAP}(w) \mid w \in L\}.$$

$L \rightarrow \text{SWAP}(L)$ è REGOLARE

└ ∃ DFA D CHE LO RICONOSCE $\rightarrow$ ∃ D' EQUIVALENTE

$D = (Q, \Sigma, q_0, \delta, F) \rightarrow D' = (Q', \Sigma', q_0', \delta', F')$

- $Q' = Q$
- $\Sigma' = \Sigma$

$$\text{SWAP}(w) = \begin{cases} \varepsilon & \text{se } w = \varepsilon \\ a & \text{se } w = a \text{ con } a \in \Sigma \\ a_1 a_0 \text{SWAP}(u) & \text{se } w = a_0 a_1 u \text{ con } a_0, a_1 \in \Sigma, u \in \Sigma^* \end{cases}$$

δ
- $\delta(q, \varepsilon) = \text{SWAP}(\varepsilon) = \varepsilon$
- $\delta(q, a) = \text{SWAP}(a) = a$
- $\delta(q, [a_1, a_0])$

$\delta(q, a_1) = \delta'(q, a_0)$

$\delta(q, a_0) = \delta'(q, a_1)$

$\delta'(q_0', a) = \{(\delta(q_0, a), \text{first})\}$ per ogni $a \in \Sigma$ $\rightarrow \varepsilon$

$\delta'((q, \text{first}), a) = \{(\delta(q, a), \text{even})\}$ per ogni $q \in Q$, $a \in \Sigma$ $\rightarrow a$

$\delta'((q, \text{even}), a) = \{(\delta(q, b), \text{odd}) \mid b \in \Sigma\}$ per ogni $q \in Q$, $a \in \Sigma$

$\delta'((q, \text{odd}), b) = \{(\delta(\delta(q, a), b), \text{even}) \mid \delta'((p, \text{even}), a) = \{(\delta(p, c), \text{odd})\}$ e p tale che $\delta(p, c) = q\}$

$$\delta'((q, 1), 0) = \delta(q, 0) . 1$$