2. (12 punti) Considera il linguaggio

$$L_2 = \{w \in \{0,1\}^* \mid w \text{ non è palindroma}\}.$$

Una parola è *palindroma* se rimane uguale letta da sinistra a destra e da destra a sinistra. Dimostra che L_2 non è regolare.

$$w = xy^iz = 1^k 0^p 1^q$$

$$Con\ k = q \to 1^k 0^p 1^k = 1^{2k} 0^p$$

Parola palindroma: 1001 → Siamo partiti da parola NON palindroma

Qui otterremmo una parola palindroma rispettando le condizioni del linguaggio → L non regolare

2. (12 punti) Considera il linguaggio

$$L_2 = \{uvvu \mid u, v \in \{0,1\}^*\}.$$

Dimostra che L_2 non è regolare.

Esempio di parola "falla" → 1001

$$w = xy^iz = xy^0z$$

$$x = 1^p 0^q 1^k, \qquad p, q, k \geq 0$$

$$xy^0z = 1^p 1^k$$

N. di 0 maggiore rispetto al n. di 1 → L non regolare.

3. (12 punti) Dimostra che se $L \subseteq \Sigma^*$ è un linguaggio context-free allora anche il seguente linguaggio è context-free:

$$dehash(L) = \{dehash(w) \mid w \in L\},$$

dove $dehash(w)$ è la stringa che si ottiene cancellando ogni $\#$ da w.

Costruiremo una grammatica G' che genera dehash(L) basandosi sulla grammatica originale G che genera L.

Dato:

- L è un linguaggio privo di contesto
- dehash(L) = {dehash(w) | w ∈ L}
- dehash(w) è la stringa ottenuta rimuovendo tutti i simboli '#' da w

Prova:

Poiché L è libera dal contesto, esiste una grammatica libera dal contesto G in Forma Normale di Chomsky che genera L.

Costruiamo una nuova grammatica G' che genera dehash(L) come segue:

a) G' contiene tutte le variabili di G.

b) Il simbolo di inizio di G' è uguale al simbolo di inizio di G.

c) Per ogni regola di G, creiamo le regole corrispondenti in G':

Se la regola è della forma A → BC, aggiungiamo A → BC a G'.

Se la regola è della forma A → a (dove a è un terminale), aggiungiamo:

A → a se a ≠ '#'

A → ε (epsilon) se a = '#'

Aggiungiamo anche A → A per ogni variabile A per permettere di saltare '#' nelle derivazioni più lunghe.

Prova che G' genera dehash(L):

- Le regole copiate da G permettono a G' di generare la struttura delle parole in L.
- La modifica delle regole terminali rimuove tutti i simboli '#'.
- Le regole A → A permettono a G' di “saltare” un numero qualsiasi di simboli '#' nella derivazione originale.

Correttezza:

- Qualsiasi parola generata da G' corrisponde a una parola in L con tutti i simboli '#' rimossi.
- Qualsiasi parola in dehash(L) può essere generata da G' seguendo una derivazione simile a quella di G, ma omettendo i passaggi che produrrebbero '#'.

Conclusione:

- Poiché abbiamo costruito una grammatica libera da contesto G' che genera dehash(L), abbiamo dimostrato che dehash(L) è libera da contesto quando L è libera da contesto.
- Questa costruzione garantisce che G' generi esattamente il linguaggio dehash(L), preservando la natura context-free del linguaggio originale L e rimuovendo tutti i simboli '#'.

4. **(9 punti)** Considera il seguente problema: data una TM M a nastro semi-infinito, determinare se esiste un input w su cui M sposta la testina a sinistra partendo dalla cella numero 2023 (ossia se in qualche momento durante la computazione la testina si muove dalla cella 2023 alla cella 2022).

   (a) Formula questo problema come un linguaggio 2023_{TM}.

   (b) Dimostra che il linguaggio 2023_{TM} è indecidibile.

$$A_{TM} \leq_m 2023_{TM}$$

Usiamo 2023_{TM} come sottoinput di A_{TM} → Se A è indecidibile allora B è indecidibile

Vale anche l’opposto! Quindi, se B (sottoinput) è decidibile, A è decidibile (superinput)

La funzione di riduzione F prende sempre in input M (TM) e w (stringa di input). Esempio classico:

$F =$ ”su input $\langle M, w\rangle$, dove M è una TM e w una stringa:
1. Costruisci la seguente macchina M':
   $M' =$ “su input x:
   1. Se $x \neq w$, vai in loop.
   2. Se $x = w$, esegue M su input w.
   3. Se M accetta, *accetta*.
   4. Se M rifiuta, vai in loop.”
2. Restituisci $\langle M'\rangle$.”

F = su input $\langle M, w \rangle$, dove M è una TM, e w è una stringa:

- Costruiamo M' (risolve A_{TM})
    - M' = su input x
        - Copia l'input su tutto il nastro
        - Se trova la cella 2023
        - Esegui M (che permette di risolvere A_{TM} usando 2023_{TM} = sottoinput)
        - Se M accetta, allora $x = w$
        - Se M rifiuta, rifiuta
    - Ritorna M'

Se $\langle M, w \rangle$ appartiene a 2023, la TM si ferma accettando la stringa qualora la cella 2023 compaia, così accetta.

Se non appartiene, non abbiamo trovato la cella 2023 e la macchina rifiuta