1. (12 punti) La *traslitterazione* è un tipo di conversione di un testo da una scrittura a un'altra che prevede la sostituzione di lettere secondo modalità prevedibili. Per esempio, il sistema di traslitterazione Hepburn permette di convertire la scrittura Kana giapponese nell'alfabeto latino usando la seguente tabella:

あ	ア	か	が	カ	ガ	さ	ざ	サ	ザ	た	だ	タ	ダ	な	ナ	は	ば	ぱ	ハ	バ	パ	ま	マ	や	ヤ	ら	ラ	わ	ワ
a	*a*	ka	ga	*ka*	*ga*	sa	za	*sa*	*za*	ta	da	*ta*	*da*	na	*na*	ha	ba	pa	*ha*	*ba*	*pa*	ma	*ma*	ya	*ya*	ra	*ra*	wa	*wa*
い	イ	き	ぎ	キ	ギ	し	じ	シ	ジ	ち	ぢ	チ	ヂ	に	ニ	ひ	び	ぴ	ヒ	ビ	ピ	み	ミ			り	リ		
i	*i*	ki	gi	*ki*	*gi*	shi	ji	*shi*	*ji*	chi	ji	*chi*	*ji*	ni	*ni*	hi	bi	pi	*hi*	*bi*	*pi*	mi	*mi*			ri	*ri*		
う	ウ	く	ぐ	ク	グ	す	ず	ス	ズ	つ	づ	ツ	ヅ	ぬ	ヌ	ふ	ぶ	ぷ	フ	ブ	プ	む	ム	ゆ	ユ	る	ル		
u	*u*	ku	gu	*ku*	*gu*	su	zu	*su*	*zu*	tsu	zu	*tsu*	*zu*	nu	*nu*	fu	bu	pu	*fu*	*bu*	*pu*	mu	*mu*	yu	*yu*	ru	*ru*		
え	エ	け	げ	ケ	ゲ	せ	ぜ	セ	ゼ	て	で	テ	デ	ね	ネ	へ	べ	ぺ	ヘ	ベ	ペ	め	メ			れ	レ		
e	*e*	ke	ge	*ke*	*ge*	se	ze	*se*	*ze*	te	de	*te*	*de*	ne	*ne*	he	be	pe	*he*	*be*	*pe*	me	*me*			re	*re*		
お	オ	こ	ご	コ	ゴ	そ	ぞ	ソ	ゾ	と	ど	ト	ド	の	ノ	ほ	ぼ	ぽ	ホ	ボ	ポ	も	モ	よ	ヨ	ろ	ロ	を	ヲ
o	*o*	ko	go	*ko*	*go*	so	zo	*so*	*zo*	to	do	*to*	*do*	no	*no*	ho	bo	po	*ho*	*bo*	*po*	mo	*mo*	yo	*yo*	ro	*ro*	o	*o*
														ん	ン														
														n	*n*														

Dati due alfabeti Σ e Γ, possiamo definire formalmente una traslitterazione come una funzione $T : \Sigma \mapsto \Gamma^*$ che mappa ogni simbolo di Σ in una stringa di simboli in Γ.

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

$$T(L) = \{w \in \Gamma^* \mid w = T(a_0)T(a_1)\dots T(a_n) \text{ per qualche } a_0a_1\dots a_n \in L\}.$$

1) DATO DFA CHE RICONOSCE (L) $\to (Q, \Sigma, \delta, q_0, F) \to M$

, ALLORA $\exists$ DFA M' $\to (Q, \Gamma, \delta, q_0, F) \to M'$

{$\exists$ NFA CHE RICONOSCE $T(L)$} = OPERAZIONI

$$\underbrace{T : \Sigma}_{\text{INPUT}} \to \underbrace{\Gamma^*}_{\text{OUTPUT}}$$

(δ = TRAN SEGUE
IN $\to$ OUT)

$(Q, \Sigma, \delta, q_0, F) \mapsto (Q', \Gamma, \delta', q_0', F)$

- $Q' = Q$
- Γ = ALFABETO = $T(a_0 \dots) \dots T(a_n)$

= $q_0' = q_0$ $F' = F$

$\forall q \in Q, \quad \delta'(q, T(a)) = \delta(q, b)$

$a \in \Sigma, \quad b \in T$

TRANSIZIONI UGUALI $\rightarrow \delta'(q', b) = \delta(q, a)$

$\rightarrow$ SCAMBIO LATINO $\underset{T(a)}{}$ CON TRASLITT. $\underset{T(a)'}{}$ HEBURN LA TRANS. δ'

CI PORTA, PARTENDO DALLO STESSO STATO INIZIALE

ALL'ALF. DI OUTPUT (TRASLITT.)

(IN $\mapsto$ OUT)

$a \mapsto T(a)$

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.

$G = (V, \Sigma, R, S) \mapsto G' = (V', \Sigma', R', S') \rightarrow$ CHOMSKY

$V = V', \quad \Sigma = \Sigma'$

⑤ → 0/1 INDECIDIBILE...

R → ∃ PDA RICONOSCE L(R) (PDA ⟷ CFG)

↓

1 REGOLA → PIÙ REGOLE (EQUIVALENTE)

∃ ∀ UNA DERIVAZIONE CHOMSKY

$$\left\{ \underbrace{0\,11\,0\,1}_{IN}^{A} \longrightarrow \underbrace{111\,00}_{OUT}^{B} \right\} \Rightarrow \Sigma = \Sigma^*$$

$$\left\{ \begin{array}{l} S \to S_{PERM} \\ S_{PERM} \to GEN\ A_1 \mid GEN\ A_2 \ldots \mid GEN\ A_n \\ GEN \to 0\ PERM\_B \mid 1\ PERM\_B \mid \varepsilon \end{array} \right\}$$

PERM_B ⊇ R(B)

$$\underbrace{\overset{A}{CF}}\ \cup\ \underbrace{\text{LING. REGOLARI}}_{\text{REG}} \Rightarrow CF$$

```
S → S_PERM
S_PERM → G_A1 | G_A2 | … | G_An    (per ogni stato di accettazione)

G_Ai → T_0 R_Ai | T_1 R_Ai | R_Ai    (genera 0, 1 o simula)
T_0 → 0
T_1 → 1

R_Ai → [transizioni che simulano il DFA per B partendo da Ai]
```

3. (12 punti) Dato un linguaggio $L \subseteq \Sigma^*$, definiamo il linguaggio

$$insert\#(L) = \{x\#y \mid xy \in L\}.$$

Dimostra che la classe dei linguaggi context free è chiusa per l'operazione $insert\#$.

$$\left[\begin{array}{l} A \to xB \\ B \to \#yC \\ C \to DE \end{array} \qquad \begin{array}{l} D \to x \\ E \to y \mid \varepsilon \end{array}\right]$$

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

- $\Sigma = \Sigma'$
- $V' = V$

$R \to R'$

$\cancel{xy}$

Δ

$\cancel{x\#y}$

$\left[\begin{array}{l} A \to BC \\ A \to c \end{array}\right]$ FNC

$S \to A$

$A \to x$

$B \to \#yC$

OGNI REGOLA INTERMEDIA CHE PRENDE $B \to \#$ VA SOSTITUITA CON $B \to \varepsilon$

$\underline{C \to DE} \to (xy) = D \,/\, \underline{E \to \varepsilon}$

FINISCE SICURAMENTE

LOGICA PER PARTE FINALE

Soluzione: Se L è un linguaggio context-free, allora esiste una grammatica $G = (V, \Sigma, R, S)$ che lo genera. Possiamo assumere che questa grammatica sia in forma normale di Chomsky. Per dimostrare che $dehash(L)$ è context-free, dobbiamo essere in grado di definire una grammatica che possa generarlo. Questa grammatica è una quadrupla $G' = (V', \Sigma', R', S')$ definita come segue.

- L'alfabeto tutti i simboli di Σ tranne $\#$: $\Sigma' = \Sigma \setminus \{\#\}$.
- L'insieme di variabili è lo stesso della grammatica G: $V' = V$.
- Il nuovo insieme di regole R' è ottenuto rimpiazzando ogni regola nella forma $A \to \#$ con la regola $A \to \varepsilon$, e lasciando invariate le regole nella forma ~~$A \to BC$~~, le regole nella forma $A \to b$ quando $b \neq \#$, e la regola $S \to \varepsilon$ (se presente).
- La variabile iniziale rimane la stessa: $S' = S$.

$A \cup B$

$b \notin \# \to \overline{xy}$

2. (12 punti) Considera il linguaggio

$$L_2 = \{0^{3n^2+2} \mid n \geq 0\}$$

Dimostra che L_2 non è regolare.

$n=0 \rightarrow 0^{(0+2)} = 0^2$

$n=1 \rightarrow 0^{(3(1)+2)} = 0^5$

$n=2 \rightarrow 0^{(3(2)+2)} = 0^7$

$n=3 \rightarrow 0^{(3(3)+2)} = 0^{11}$

INDUTTIVAMENTE ...

xy^iz, $i \geq 0$

$i > 0 \rightarrow z^{k-p+q}$

$i = 0 \rightarrow z^{k-p-q}$

$w = xy^iz$, $x \neq \varepsilon$

0^{3n^2+2}

$x = 0^{2} = p$

$y = z \rightarrow 0^{3} = q$

$z = 0^{k-(2)-3}$ (p, q)

$|k| \geq |p \cdot q|$

$p > 0$, $q > 0$

$\frac{k}{q} \leq p$, $\frac{k}{p} \leq q$

NON REGOLARE

1. (9 punti) Considera il linguaggio

$$L = \{1^n 0^{2^n} \mid n \leq 0\}.$$

Dimostra che L non è regolare.

$n = -2$

$1^{-2} 0^{2^{-2}}$

$= 1^{-2} 0^{-4} = \left(\frac{1}{2}\right)$

NO SE PUEDE!

$L = \{1^n 0^{2^n} \mid n \geq 0\}$

$n = 0 \rightarrow 10$

$n = 1 \rightarrow 100$

$n = 2 \quad 110000$

$w = x y^i z, \quad x \neq \varepsilon, \quad i \geq 0$

$x = 1^p 0^{2p}$

$n = p$

$y = 0^q$

$z = 0^{k-2p-q}$

$x y^2 z = 1^p 0^{2p} 0^{k-2p-q}$

$= 1^p 0^{k-q}$

L NON REGOLARE! $\leftarrow$ $2 \cdot |p| = |k - q|$

CONDIZIONE SU INDICI

---

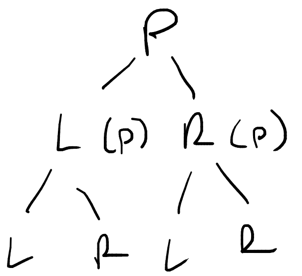
1. (12 punti) Una *macchina di Turing ad albero binario* usa un albero binario infinito come nastro, dove ogni cella nel nastro ha un figlio sinistro e un figlio destro. Ad ogni transizione, la testina si sposta dalla cella corrente al padre, al figlio sinistro oppure al figlio destro della cella corrente. Pertanto, la funzione di transizione di una tale macchina ha la forma

$$\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{P, L, R\},$$

dove P indica lo spostamento verso il padre, L verso il figlio sinistro e R verso il figlio destro. La stringa di input viene fornita lungo il ramo sinistro dell'albero.

Mostra che qualsiasi macchina di Turing ad albero binario può essere simulata da una macchina di Turing standard.

$\Gamma \Rightarrow \{ \underline{P, L, R} / P, L, R / P, L, R \}$



COMPLETAMENTO ...

TM STANDARD

TM NASTRO SINGOLO

MULTINASTRO

(NO FINE DX) SEMI-INFINITO

[SINGLE] $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R\}$

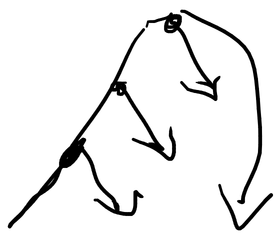
$\Downarrow \quad \subseteq$

[BIN-TREE] $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{P, L, R\}$

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

(INPUT = Γ)

M = COPIA W SUL NASTRO Γ'
FINCHÉ CI SONO INPUT



LOOP UNTIL $\underline{[L, P, R]}$

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

MARCA L'ULTIMO FIGLIO SX, LA CELLA SOTTO LA TESTINA È PADRE = P (SCRIVE A DX)

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

$\underline{[L, P, R]}\ ;\ \underline{[L, P, R]}$

L, P, R → SCRIVE b SUL NASTRO ANDANDO A DX