1. **(12 punti)** Se L è un linguaggio sull'alfabeto $\{0,1\}$, la *rotazione a destra* di L è l'insieme delle stringhe

$$\mathrm{ROR}(L) = \{aw \mid wa \in L, w \in \{0,1\}^*, a \in \{0,1\}\}.$$

Per esempio, se $L = \{0, 001, 10010\}$, allora $\mathrm{ROR}(L) = \{0, 100, 01001\}$. Dimostra che se L è regolare allora anche $\mathrm{ROR}(L)$ è regolare.

Per dimostrare che se L è regolare, allora anche ROR(L) è regolare, utilizzeremo il fatto che i linguaggi regolari sono chiusi rispetto all'operazione di concatenazione, all'operazione di unione e all'operazione di stella di Kleene.

Sia L un linguaggio regolare sull'alfabeto {0, 1}. Costruiamo un automa a stati finiti M che riconosce ROR(L) come segue:

1. Aggiungiamo uno stato iniziale q0 a M e per ogni stato q di L, aggiungiamo uno stato q' a M.

2. Per ogni transizione (q, a, r) in L, aggiungiamo una transizione (q', a, r') in M.

3. Aggiungiamo una transizione (q, a, q) per ogni a ∈ {0, 1}.

4. Aggiungiamo una transizione (q', a, q') per ogni a ∈ {0, 1}.

5. Aggiungiamo una transizione (q', a, q) per ogni a ∈ {0, 1}.

6. Aggiungiamo una transizione (q, λ, q').

7. Aggiungiamo uno stato finale q' a M.

In altre parole, l'automa M è costituito dalla copia dell'automa che riconosce il linguaggio L, con l'aggiunta di una transizione per ogni stato di L che porta ad uno stato corrispondente, la possibilità di tornare allo stato iniziale da ogni stato di L, e uno stato finale che corrisponde a tutti gli stati di L.

Mostriamo che M riconosce ROR(L). Sia w una stringa in ROR(L). Ci sono due casi da considerare:

- Se w = a, dove a ∈ {0, 1}, allora w appartiene a ROR(L) perché a ∈ L e λ ∈ {0, 1}*.

- Se w = aw', dove wa ∈ L e w' ∈ {0, 1}*, allora w' può essere visto come la rotazione a destra di w, cioè w' ∈ ROR(L). Inoltre, esiste uno stato q di L tale che wa può essere letto a partire dallo stato iniziale di q. Quindi, w può essere letto a partire dallo stato iniziale di q' in M, seguito da una transizione per leggere a, poi da una transizione per leggere λ e finalmente da una sequenza di transizioni per leggere w' che portano allo stato finale q'.

Pertanto, M riconosce ROR(L) e quindi ROR(L) è regolare, poiché l'automa M è un automa a stati finiti, cioè un tipo di automa che riconosce soltanto linguaggi regolari.

2. (12 punti) Considera l'alfabeto $\Sigma = \{0, 1\}$, e sia L_2 l'insieme di tutte le stringhe che contengono almeno un 1 nella loro seconda metà:

$$L_2 = \{uv \mid u \in \Sigma^*, v \in \Sigma^*1\Sigma^* \text{ e } |u| \geq |v|\}.$$

Dimostra che L_2 non è regolare.

Per dimostrare che L_2 non è regolare, utilizzeremo il pumping lemma per i linguaggi regolari.

Supponiamo per assurdo che L_2 sia regolare. Allora, esiste un intero positivo p tale che qualsiasi stringa w di L_2 con |w| >= p può essere scomposta in tre parti, w = xyz, con le seguenti proprietà:

- |xy| <= p,
- |y| > 0, e
- xy^iz appartiene a L_2 per ogni i ≥ 0.

Consideriamo la stringa $w = 0^p\ 1\ 0^{p-1}\ 1\ 0^{p-2}\ 1 \ldots 01$, che appartiene a L_2 poiché contiene almeno un 1 nella sua seconda metà. Poiché |w| = 2p, per il pumping lemma esiste una scomposizione $w = xyz$ tale che |xy| <= p e |y| > 0. Ci sono tre casi da considerare:

- La sottostringa y contiene solo 0: in tal caso, la stringa xy^0z non può contenere un 1 nella sua seconda metà, perché la parte sinistra di xy^0z contiene solo 0, che è più corta della seconda metà della stringa.
- La sottostringa y contiene solo 1: in tal caso, la stringa xy^2z contiene più di un 1 nella sua seconda metà, perché la parte destra di xy^2z contiene solo 0, che è più corta della seconda metà della stringa.
- La sottostringa y contiene sia 0 che 1: in tal caso, la stringa xy^2z contiene più di un 1 nella sua seconda metà, perché la sottostringa y contiene almeno un 1 e c'è spazio nella parte destra della stringa per aggiungere altri 1 dopo l'aggiunta di y.

In tutti e tre i casi, abbiamo ottenuto una contraddizione con la proprietà 3 del pumping lemma. Pertanto, a L_2 non è un linguaggio regolare.

Concludiamo che a L_2 non può essere riconosciuto da un automa a stati finiti e quindi non è regolare.

3. (12 punti) Mostra che per ogni PDA P esiste un PDA P_2 con due soli simboli di stack tale che $L(P_2) = L(P)$. *Suggerimento:* dai una codifica binaria all'alfabeto di stack di P.

Sia P = (Q, Σ, Γ, δ, q0, Z0, F) un PDA arbitrario, dove Q è l'insieme degli stati, Σ è l'alfabeto di input, Γ è l'alfabeto di stack, δ è la funzione di transizione, q0 è lo stato iniziale, Z0 è il simbolo iniziale della pila e F è l'insieme degli stati finali.

Costruiamo un PDA P2 con due soli simboli di stack, che riconosce lo stesso linguaggio di P, come segue:

1. L'alfabeto di stack di P2 è {0, 1}, cioè ogni simbolo di stack in Γ è codificato come una coppia di simboli binari.

2. Quando P legge un simbolo di input a in uno stato q, effettua la seguente sequenza di operazioni in P2:

- Per ogni transizione (q, b, γ) in P, aggiungi le seguenti transizioni in P2:
  - (q, a/b, γ/00)
  - (q, a/b, γ/01)
- Per ogni stato finale f in F, aggiungi la transizione (f, λ, 00) in P2.

3. Quando P legge il simbolo di input λ in uno stato q, effettua la seguente sequenza di operazioni in P2:

- Per ogni transizione (q, λ, γ) in P, aggiungi le seguenti transizioni in P2:
  - (q, λ/λ, γ/00)
  - (q, λ/λ, γ/01)
- Per ogni stato finale f in F, aggiungi la transizione (f, λ, 00) in P2.

4. Quando P legge un simbolo di stack γ in uno stato q, effettua la seguente sequenza di operazioni in P2:

- Per ogni transizione (q, λ, γ) in P, aggiungi le seguenti transizioni in P2:
  - (q, λ/λ, 00/γ)
  - (q, λ/λ, 01/γ)

5. P2 inizia in uno stato q0 e con il simbolo di stack 00.

Mostriamo che P2 riconosce lo stesso linguaggio di P. Sia w una stringa in L(P). Allora, esiste una sequenza di transizioni del PDA P che legge w e lo accetta. Osserviamo che ogni simbolo di stack in P può essere codificato come una coppia di simboli binari in P2, come descritto sopra. Quindi, esiste una sequenza di transizioni del PDA P2 che legge la stessa stringa w con i simboli di stack codificati come coppie di simboli binari, e lo accetta. Inoltre, le transizioni aggiuntive in P2 garantiscono che P2 accetti w solo se P accetta w.

Pertanto, P2 riconosce lo stesso linguaggio di P e ha solo due simboli di stack.