1. Una *macchina di Turing con reset a sinistra* è una variante delle comuni macchine di Turing, dove la funzione di transizione ha la forma:

$$\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{R, RESET\}.$$

Se $\delta(q, a) = (r, b, RESET)$, quando la macchina si trova nello stato q e legge a, la testina scrive b sul nastro, salta all'estremità sinistra del nastro ed entra nello stato r. Per sapere su quale cella saltare la macchina usa il simbolo speciale $\triangleright$ per identificare l'estremità di sinistra del nastro. Questo simbolo si può trovare solo in una cella del nastro, e non può essere sovrascritto o cancellato. La computazione di una macchina di Turing con reset a sinistra sulla parola w inizia con $\triangleright w$ sul nastro. Si noti che queste macchine non hanno la solità capacità di muovere la testina di una cella a sinistra.

Mostrare che le macchine di Turing con reset a sinistra riconoscono la classe dei linguaggi Turing-riconoscibili.

1. Per risolvere l'esercizio dobbiamo dimostrare che *(a)* ogni linguaggio riconosciuto da una TM con reset a sinistra è Turing-riconoscibile e *(b)* ogni linguaggio Turing-riconoscibile è riconosciuto da una TM con reset a sinistra.

(a) Mostriamo come convertire una TM con reset a sinistra M in una TM standard S equivalente. S simula il comportamento di M nel modo seguente. Se la mossa da simulare prevede uno spostamento a destra, allora S esegue direttamente la mossa. Se la mossa prevede un $RESET$, allora S scrive il nuovo simbolo sul nastro, poi scorre il nastro a sinistra finché non trova il simbolo $\triangleright$, e riprende la simulazione dall'inizio del nastro. Per ogni stato q di M, S possiede uno stato q_{RESET} che serve per simulare il reset e riprendere la simulazione dallo stato corretto.

S = "Su input w:

1. scrive il simbolo $\triangleright$ subito prima dell'input, in modo che il nastro contenga $\triangleright w$.
2. Se la mossa da simulare è $\delta(q, a) = (r, b, R)$, allora S la esegue direttamente: scrive b sul nastro, muove la testina a destra e va nello stato r.
3. Se la mossa da simulare è $\delta(q, a) = (r, b, RESET)$, allora S esegue le seguenti operazioni: scrive b sul nastro, poi muove la testina a sinistra e va nello stato r_{RESET}. La macchina rimane nello stato r_{RESET} e continua a muovere la testina a sinistra finché non trova il simbolo $\triangleright$. A quel punto la macchina sposta la testina un'ultima volta a sinistra, poi di una cella a destra per tornare sopra al simbolo di fine nastro. La computazione riprende dallo stato r.
4. Se non sei nello stato di accettazione o di rifiuto, ripeti da 2."

(b) Mostriamo come convertire una TM standard S in una TM con reset a sinistra M equivalente. M simula il comportamento di S nel modo seguente. Se la mossa da simulare prevede uno spostamento a destra, allora M può eseguire direttamente la mossa. Se la mossa da simulare prevede uno spostamento a sinistra, allora M simula la mossa come descritto dall'algoritmo seguente. L'algoritmo usa un nuovo simbolo $\triangleleft$ per identificare la fine della porzione di nastro usata fino a quel momento, e può marcare le celle del nastro ponendo un punto al di sopra di un simbolo.

M = "Su input w:

1. Scrive il simbolo $\triangleleft$ subito dopo l'input, per marcare la fine della porzione di nastro utilizzata. Il nastro contiene $\triangleright w \triangleleft$.
2. Simula il comportamento di S. Se la mossa da simulare è $\delta(q, a) = (r, b, R)$, allora M la esegue direttamente: scrive b sul nastro, muove la testina a destra e va nello stato r. Se muovendosi a destra la macchina si sposta sulla cella che contiene $\triangleleft$, allora questo significa che S ha spostato la testina sulla parte di nastro vuota non usata in precedenza. Quindi M scrive un simbolo blank marcato su questa cella, sposta $\triangleleft$ di una cella a destra, e fa un reset a sinistra. Dopo il reset si muove a destra fino al blank marcato, e prosegue con la simulazione mossa successiva.
3. Se la mossa da simulare è $\delta(q, a) = (r, b, L)$, allora S esegue le seguenti operazioni:
   - 3.1 scrive b sul nastro, marcandolo con un punto, poi fa un reset a sinistra
   - 3.2 Se il simbolo subito dopo $\triangleright$ è già marcato, allora vuol dire che S ha spostato la testina sulla parte vuota di sinistra del nastro. Quindi M scrive un blank e sposta il contenuto del nastro di una cella a destra finché non trova il simbolo di fine nastro $\triangleleft$. Fa un reset a sinistra e prosegue con la simulazione della prossima mossa dal nuovo blank posto subito dopo l'inizio del nastro. Se il simbolo subito dopo $\triangleright$ non è marcato, lo marca, resetta a sinistra e prosegue con i passi successivi.

3.3 Si muove a destra fino al primo simbolo marcato, e poi a destra di nuovo.
3.4 se la cella in cui si trova è marcata, allora è la cella da cui è partita la simulazione. Toglie la marcatura e resetta. Si muove a destra finché non trova una cella marcata. Questa cella è quella immediatamente precedente la cella di partenza, e la simulazione della mossa è terminata
3.5 se la cella in cui si trova non è marcata, la marca, resetta, si muove a destra finché non trova una marcatura, cancella la marcatura e riprende da 3.3.

4. Se non sei nello stato di accettazione o di rifiuto, ripeti da 2."

1. Una *tag-Turing machine* è una macchina di Turing con un singolo nastro e due testine: una testina può solo leggere, l'altra può solo scrivere. All'inizio della computazione la testina di lettura si trova sopra il primo simbolo dell'input e la testina di scrittura si trova sopra la cella vuota posta immediatamente dopo la stringa di input. Ad ogni transizione, la testina di lettura può spostarsi di una cella a destra o rimanere ferma, mentre la testina di scrittura deve scrivere un simbolo nella cella corrente e spostarsi di una cella a destra. Nessuna delle due testine può spostarsi a sinistra.

Dimostra che che le tag-Turing machine riconoscono la classe dei linguaggi Turing-riconoscibili.

1. Per risolvere l'esercizio dobbiamo dimostrare che *(a)* ogni linguaggio riconosciuto da una tag-Turing machine è Turing-riconoscibile e *(b)* ogni linguaggio Turing-riconoscibile è riconosciuto da una tag-Turing machine.

(a) Mostriamo come convertire una tag-Turing machine M in una TM deterministica a nastro singolo S equivalente. S simula il comportamento di M tenendo traccia delle posizioni delle due testine marcando la cella dove si trova la testina di lettura con un pallino sopra il simbolo, e marcando la cella la cella dove si trova la testina di scrittura con un pallino sotto il simbolo. Per simulare il comportamento di M la TM S scorre il nastro e aggiorna le posizioni delle testine ed il contenuto delle celle come indicato dalla funzione di transizione di M.

$S =$ "Su input $w = w_1 w_2 \dots w_n$:

1. Scrivi un pallino sopra il primo simbolo di w e un pallino sotto la prima cella vuota dopo l'input, in modo che il nastro contenga

$$\overset{\bullet}{w}_1 w_2 \dots w_n \underset{\bullet}{\sqcup}.$$

2. Per simulare una transizione, S scorre il nastro per trovare la posizione della testina di lettura e determinare il simbolo letto da M. Se la funzione di transizione stabilisce che la testina di lettura deve spostarsi a destra, allora S sposta il pallino nella cella immediatamente a destra, altrimenti lo lascia dov'è. Successivamente S si sposta verso destra finché non trova la cella dove si trova la testina di scrittura, scrive il simbolo stabilito dalla funzione di transizione nella cella e sposta la marcatura nella cella immediatamente a destra.
3. Se in qualsiasi momento la simulazione raggiunge lo stato di accettazione di M, allora *accetta*; se la simulazione raggiunge lo stato di rifiuto di M allora *rifiuta*; altrimenti prosegue con la simulazione dal punto 2."

(b) Mostriamo come convertire una TM deterministica a nastro singolo S in una tag-Turing machi M equivalente. M simula il comportamento di S memorizzando sul nastro una sequenza configurazioni di S separate dal simbolo #. All'interno di ogni configurazione un pallino mar il simbolo sotto la testina di S. Per simulare il comportamento di S la tag-Turing machine scorre la configurazione corrente e scrivendo man mano la prossima configurazione sul nastro.

M = "Su input $w = w_1 w_2 \dots w_n$:

1. Scrive il simbolo # subito dopo l'input, seguito dalla configurazione iniziale, in modo che il nastro contenga
$$w_1 w_2 \dots w_n \# \dot{w}_1 w_2 \dots w_n \textvisiblespace,$$
che la testina di lettura si trovi in corrispondenza del $\dot{w}_1$ e quella di lettura in corrispondenza del blank dopo la configurazione iniziale. Imposta lo stato corrente della simulazione st allo stato iniziale di S e memorizza l'ultimo simbolo letto $prec = \#$. L'informazione sui valori di st e $prec$ sono codificate all'interno degli stato di M.
2. Finché il simbolo sotto la testina di lettura non è marcato, scrive il simbolo precedente $prec$ e muove a destra. Aggiorna il valore di $prec$ con il simbolo letto.
3. Quando si trova un simbolo marcato $\dot{a}$ e $\delta(st, a) = (q, b, R)$:
   - aggiorna lo stato della simulazione $st = q$;
   - scrive $prec$ seguito da b, poi muove la testina di lettura a destra;
   - scrive il simbolo sotto la testina marcandolo con un pallino.
4. Quando si trova un simbolo marcato $\dot{a}$ e $\delta(st, a) = (q, b, L)$:
   - aggiorna lo stato della simulazione $st = q$;
   - scrive $\dot{prec}$; se $prec = \#$ scrive $\#\dot{\textvisiblespace}$;
   - scrive b.
5. Copia il resto della configurazione fino al # escluso. Al termine della copia la testina di lettura di trova in corrispondenza della prima cella nella configurazione corrente, e quella di lettura sulla cella vuota dopo la configurazione.
6. Se in qualsiasi momento la simulazione raggiunge lo stato di accettazione di S, allora *accetta*; se la simulazione raggiunge lo stato di rifiuto di M allora *rifiuta*; altrimenti prosegue con la simulazione dal punto 2."

3. Fornisci una descrizione a livello implementativo di una TM deterministica a nastro singolo che decide il linguaggio
$$L_3 = \{ww \mid w \in \{0,1\}^*\}$$
Una descrizione a livello implementativo descrive a parole il movimento della testina e la scrittura sul nastro, senza dare il dettaglio degli stati.

M = "su input x, dove x è una stringa:

1. Scorre la stringa x per controllare se il numero di simboli è pari o dispari. Se è dispari, rifiuta.
2. Divide la stringa x in due parti uguali. Per farlo marca il primo simbolo di x con un pallino, poi va alla fine della stringa e marca l'ultimo simbolo con un pallino. Continua a marcare un carattere all'inizio e uno alla fine della stringa muovendosi a zig zag.
3. L'ultimo carattere marcato è la posizione di inizio della seconda metà della stringa. Toglie tutte le altre marcature e ritorna all'inizio della stringa.
4. sostituisce il simbolo all'inizio della stringa con #, poi procede a destra e controlla se il simbolo marcato con un pallino è uguale al primo simbolo. Se sono diversi rifiuta.
5. Se i due simboli sono uguali, sostiuisce il simbolo marcato con # e sposta la marcatura al simbolo successivo.
6. Torna all'inizio della stringa: se ci sono ancora simboli diversi da #, ripeti da 4, altrimenti accetta."

3. Fornisci una descrizione a livello implementativo di una TM deterministica a nastro singolo che decide il linguaggio
$$L_3 = \{u\#w_1\# \dots \#w_n \mid u, w_i \in \{0,1\}^* \text{ ed esiste } w_j \text{ tale che } u = w_j\}$$
Una descrizione a livello implementativo descrive a parole il movimento della testina e la scrittura sul nastro, senza dare il dettaglio degli stati.

M = "Su input $u\#w_1\#\ldots\#w_n$:

1. Marca il simbolo più a sinistra dell'input. Se il simbolo è #, controlla che a destra ci sia un blank: se c'è *accetta*, altrimenti *rifiuta*.
2. Scorre a destra fino al primo # non marcato e marca il simbolo posto immediatamente a destra. Se non viene trovato nessun # non marcato prima di un blank, allora u è diverso da tutti i w_i, quindi *rifiuta*.
3. Procede a zig-zag confrontando i simboli di u con i simboli della stringa a destra del primo # non marcato. Se le due stringhe sono uguali, *accetta*.
4. Se le due stringhe sono diverse, marca il # e smarca tutti i simboli di u tranne il primo, poi ripete da 2."

1. *Una* macchina di Turing bidimensionale *utilizza una griglia bidimensionale infinita di celle come nastro. Ad ogni transizione, la testina può spostarsi dalla cella corrente ad una qualsiasi delle quattro celle adiacenti. La funzione di transizione di tale macchina ha la forma*

$$\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{\uparrow, \downarrow, \rightarrow, \leftarrow\},$$

dove le frecce indicano in quale direzione si muove la testina dopo aver scritto il simbolo sulla cella corrente.

Dimostra che ogni macchina di Turing bidimensionale può essere simulata da una macchina di Turing deterministica a nastro singolo.

Mostriamo come simulare una TM bidimensionale B con una TM deterministica a nastro singolo S. S memorizza il contenuto della griglia bidimensionale sul nastro come una sequenza di stringhe separate da #, ognuna delle quali rappresenta una riga della griglia. Due cancelletti consecutivi ## segnano l'inzio e la fine della rappresentazione della griglia. La posizione della testina di B viene indicata marcando la cella con $\widehat{\ }$. Nelle altre righe, un pallino $^{\bullet}$ indica che la testina si trova su quella colonna della griglia, ma in una riga diversa. La TM a nastro singolo S funziona come segue:

S = "su input w:

1. Sostituisce w con la configurazione iniziale $\#\#w\#\#$ e marca con $\widehat{\ }$ il primo simbolo di w.
2. Scorre il nastro finché non trova la cella marcata con $\widehat{\ }$.
3. Aggiorna il nastro in accordo con la funzione di transizione di B:
   - Se $\delta(r, a) = (s, b, \rightarrow)$, scrive b non marcato sulla cella corrente, sposta $\widehat{\ }$ sulla cella immediatamente a destra. Poi sposta di una cella a destra tutte le marcature con un pallino. Se in qualsiasi momento S sposta una marcatura sopra un #, S scrive un blank marcato al posto del # e sposta il contenuto del nastro da questa cella fino al ## finale, di una cella più a destra.
   - Se $\delta(r, a) = (s, b, \leftarrow)$, scrive b non marcato sulla cella corrente, sposta $\widehat{\ }$ sulla cella immediatamente a sinistra. Poi sposta di una cella a sinistra tutte le marcature con un pallino. Se in qualsiasi momento S sposta una marcatura sopra un #, S scrive un blank marcato al posto del # e sposta il contenuto del nastro da questa cella fino al ## iniziale, di una cella più a sinistra.
   - Se $\delta(r, a) = (s, b, \uparrow)$, scrive b marcato con un pallino nella cella corrente, e sposta $\widehat{\ }$ sulla prima cella marcata con un pallino posta a sinistra della cella corrente. Se questa cella marcata non esiste, aggiunge una nuova riga composta solo da blank all'inizio della configurazione.
   - Se $\delta(r, a) = (s, b, \downarrow)$, scrive b marcato con un pallino nella cella corrente, e sposta $\widehat{\ }$ sulla prima cella marcata con un pallino posta a destra della cella corrente. Se questa cella non esiste, aggiunge una nuova riga composta solo da blank alla fine della configurazione.
4. Se in qualsiasi momento la simulazione raggiunge lo stato di accettazione di B, allora *accetta*; se la simulazione raggiunge lo stato di rifiuto di B allora *rifiuta*; altrimenti prosegue con la simulazione dal punto 2."

3. (8 punti) *Una* Turing machine con alfabeto binario *è una macchina di Turing deterministica a singolo nastro dove l'alfabeto di input è* $\Sigma = \{0, 1\}$ *e l'alfabeto del nastro è* $\Gamma = \{0, 1, \sqcup\}$. *Questo significa che la macchina può scrivere sul nastro solo i simboli* $0, 1$ *e blank: non può usare altri simboli né marcare i simboli sul nastro.*

Dimostra che che le Turing machine con alfabeto binario machine riconoscono tutti e soli i linguaggi Turing-riconoscibili sull'alfabeto $\{0, 1\}$.

Per risolvere l'esercizio dobbiamo dimostrare che *(a)* ogni linguaggio riconosciuto da una Turing machine con alfabeto binario è Turing-riconoscibile e *(b)* ogni linguaggio Turing-riconoscibile sull'alfabeto $\{0, 1\}$ è riconosciuto da una Turing machine con alfabeto binario.

(a) Questo caso è semplice: una Turing machine con alfabeto binario è un caso speciale di Turing machine deterministica a nastro singolo. Quindi ogni linguaggio riconosciuto da una Turing machine con alfabeto binario è anche Turing-riconoscibile.

(b) Per dimostrare questo caso, consideriamo un linguaggio L Turing-riconoscibile, e sia M una Turing machine deterministica a nastro singolo che lo riconosce. Questa TM potrebbe avere un alfabeto del nastro Γ che contiene altri simboli oltre a $0, 1$ e blank. Per esempio potrebbe contenere simboli marcati o separatori.
Per costruire una TM con alfabeto binario B che simula il comportamento di M dobbiamo come prima cosa stabilire una *codifica binaria* dei simboli nell'alfabeto del nastro Γ di M. Questa codifica è una funzione C che assegna ad ogni simbolo $a \in \Gamma$ una sequenza di k cifre binarie, dove k è un valore scelto in modo tale che ad ogni simbolo corrisponda una codifica diversa. Per esempio, se Γ contiene 4 simboli, allora $k = 2$, perché con 2 bit si rappresentano 4 valori diversi. Se Γ contiene 8 simboli, allora $k = 3$, e così via.

La TM con alfabeto binario B che simula M è definita in questo modo:

$B =$ "su input w:

1. Sostituisce $w = w_1 w_2 \dots w_n$ con la codifica binaria $C(w_1)C(w_2)\dots C(w_n)$, e riporta la testina sul primo simbolo di $C(w_1)$.
2. Scorre il nastro verso destra per leggere k cifre binarie: in questo modo la macchina stabilisce qual è il simbolo a presente sul nastro di M. Va a sinistra di k celle.
3. Aggiorna il nastro in accordo con la funzione di transizione di M:
   - Se $\delta(r, a) = (s, b, R)$, scrive la codifica binaria di b sul nastro.
   - Se $\delta(r, a) = (s, b, L)$, scrive la codifica binaria di b sul nastro e sposta la testina a sinistra di $2k$ celle.
4. Se in qualsiasi momento la simulazione raggiunge lo stato di accettazione di M, allora *accetta*; se la simulazione raggiunge lo stato di rifiuto di M allora *rifiuta*; altrimenti prosegue con la simulazione dal punto 2."

3. (8 punti) Una *Turing machine con alfabeto ternario* è una macchina di Turing deterministica a singolo nastro dove l'alfabeto di input è $\Sigma = \{0, 1, 2\}$ e l'alfabeto del nastro è $\Gamma = \{0, 1, 2, \sqcup\}$. Questo significa che la macchina può scrivere sul nastro solo i simboli $0, 1$ e blank: non può usare altri simboli né marcare i simboli sul nastro.

Dimostra che ogni linguaggio Turing-riconoscibile sull'alfabeto $\{0, 1, 2\}$ può essere riconosciuto da una Turing machine con alfabeto ternario.

3. Dimostriamo che ogni linguaggio Turing-riconoscibile sull'alfabeto {0, 1, 2} può essere riconosciuto da una Turing machine con alfabeto ternario:

Sia M una TM che riconosce un linguaggio L sull'alfabeto {0, 1, 2}. Costruiamo una TM M' con alfabeto ternario {0, 1, 2, □} che simula M:

1. M' usa una codifica delle configurazioni di M sul suo nastro.
2. Per simulare una mossa di M, M' scansiona l'intera configurazione codificata, aggiornandola secondo la funzione di transizione di M.
3. M' accetta se e solo se M accetta.

La codifica delle configurazioni può essere fatta usando solo 0, 1, 2, □, ad esempio separando gli stati e i simboli con sequenze specifiche di questi simboli.

Questa costruzione mostra che M' può simulare M usando solo l'alfabeto ternario, quindi ogni linguaggio Turing-riconoscibile su {0, 1, 2} può essere riconosciuto da una TM con alfabeto ternario.

3. (8 punti) Un *automa a coda* è simile ad un automa a pila con la differenza che la pila viene sostituita da una coda. Una *coda* è un nastro che permette di scrivere solo all'estremità sinistra del nastro e di leggere solo all'estremità destra. Ogni operazione di scrittura (*push*) aggiunge un simbolo all'estremità sinistra della coda e ogni operazione di lettura (*pull*) legge e rimuove un simbolo all'estremità destra. Come per un PDA, l'input è posizionato su un nastro a sola lettura separato, e la testina sul nastro di lettura può muoversi solo da sinistra a destra. Il nastro di input contiene una cella con un blank che segue l'input, in modo da poter rilevare la fine dell'input. Un automa a coda accetta l'input entrando in un particolare stato di accettazione in qualsiasi momento. Mostra che ogni linguaggio Turing-riconoscibile può essere riconosciuto da un automa deterministico a coda.

Sia M una TM che riconosce un linguaggio L. Costruiamo un automa deterministico a coda D che simula M:

1. D usa la coda per simulare il nastro di M. Inizialmente, D copia l'input nella coda.
2. D mantiene nella sua memoria finita lo stato corrente di M e la posizione della testina.
3. Per simulare una mossa di M: a. D scorre la coda fino alla posizione corrente della testina. b. Legge il simbolo, lo rimuove dalla coda e lo riscrive alla fine della coda. c. Aggiorna lo stato e la posizione della testina secondo la funzione di transizione di M.
4. Se M accetta, D accetta. Se M rifiuta, D rifiuta.

Questa costruzione mostra che D può simulare M, quindi ogni linguaggio Turing-riconoscibile può essere riconosciuto da un automa deterministico a coda.

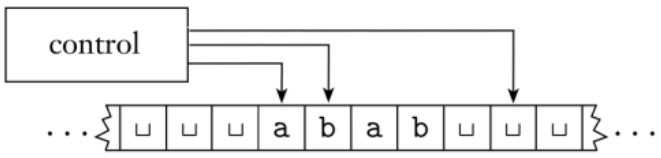
1. Una macchina di Turing a testine multiple è una macchina di Turing con un solo nastro ma con varie testine. Inizialmente, tutte le testine si trovano sopra alla prima cella dell'input. La funzione di transizione viene modificata per consentire la lettura, la scrittura e lo spostamento delle testine. Formalmente,

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

dove k è il numero delle testine. L'espressione

$$\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, L, R, \ldots, L)$$

significa che, se la macchina si trova nello stato q_i e le testine da 1 a k leggono i simboli $a_1, \ldots, a_k$ allora la macchina va nello stato q_j, scrive i simboli $b_1, \ldots, b_k$ e muove le testine a destra e a sinistra come specificato.



Un esempio di TM con tre testine.

Dimostra che qualsiasi macchina di Turing a testine multiple può essere simulata da una macchina di Turing deterministica a nastro singolo.

Dimostriamo che qualsiasi macchina di Turing a testine multiple può essere simulata da una macchina di Turing deterministica a nastro singolo:

Sia M una TM a k testine. Costruiamo una TM S a singola testina che simula M:

1. S usa un nastro diviso in k+1 tracce:
    - La prima traccia contiene l'input originale.
    - Le altre k tracce contengono un marcatore per la posizione di ciascuna testina di M.
2. Per simulare una mossa di M: a. S scansiona l'intero nastro da sinistra a destra, memorizzando i k simboli letti dalle posizioni marcate. b. S calcola la nuova configurazione di M (nuovo stato, simboli da scrivere, movimenti delle testine). c. S scansiona di nuovo il nastro, aggiornando i simboli e le posizioni dei marcatori.
3. S accetta se e solo se M accetta.
4. Questa costruzione mostra che S può simulare M, quindi ogni TM a testine multiple può essere simulata da una TM a nastro singolo.

1. (12 punti) Una *Macchina di Turing con inserimento* è una macchina di Turing deterministica a nastro singolo che può inserire nuove celle nel nastro. Formalmente la funzione di transizione è definita come

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

dove L, R indicano gli spostamenti a sinistra e a destra della testina, e I indica l'inserimento di una nuova cella nella posizione corrente della testina. Dopo una operazione di inserimento, la cella inserita contiene il simbolo blank, mentre la cella che si trovava sotto la testina si trova immediatamente a destra della nuova cella.

Dimostra che qualsiasi macchina di Turing con inserimento può essere simulata da una macchina di Turing deterministica a nastro singolo.

Soluzione. Mostriamo come convertire una macchina di Turing con Inserimento M in una TM deterministica a nastro singolo S equivalente.

$S =$ "Su input w:

1. Inizialmente S mette il suo nastro in un formato che gli consente di implementare l'operazione di inserimento di una cella, segnando con il simbolo speciale # la fine della porzione di nastro usata dalla macchina. Se w è l'input della TM, la configurazione iniziale del nastro è $w\#$.
2. La simulazione delle mosse del tipo $\delta(q, a) = (r, b, L)$ procede come nella TM standard: S scrive b sul nastro e muove la testina di una cella a sinistra.
3. La simulazione delle mosse del tipo $\delta(q, a) = (r, b, R)$ procede come nella TM standard: S scrive b sul nastro e muove la testina di una cella a destra. Se lo spostamento a destra porta la testina sopra il # che marca la fine del nastro, S scrive un blank al posto del #, e scrive un # nella cella immediatamente più a destra. La simulazione continua con la testina in corrispondenza del blank.
4. Per simulare una mossa del tipo $\delta(q, a) = (r, b, I)$ la TM S scrive un blank marcato nella cella corrente e sposta il contenuto del nastro, dalla cella corrente fino al # di fine nastro, di una cella più a destra. Quindi riporta la testina in corrispondenza del blank marcato, toglie la marcatura e scrive b nella cella immediatamente più a destra. La simulazione continua con la testina in corrispondenza della cella inserita.
5. Se in qualsiasi momento la simulazione raggiunge lo stato di accettazione di M, allora S termina con accettazione. Se in qualsiasi momento la simulazione raggiunge lo stato di rifiuto di M, allora S termina con rifiuto. Negli altri casi continua la simulazione dal punto 2."

1. **(12 punti)** Una *Macchina di Turing con eliminazione* è una macchina di Turing deterministica a nastro singolo che può eliminare celle dal nastro. Formalmente la funzione di transizione è definita come

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

dove L, R indicano i normali spostamenti a sinistra e a destra della testina, e D indica l'eliminazione della cella sotto la posizione corrente della testina. Dopo una operazione di eliminazione, la testina si muove nella cella che si trovava immediatamente a destra della cella eliminata.

Dimostra che qualsiasi macchina di Turing con eliminazione può essere simulata da una macchina di Turing deterministica a nastro singolo.

Soluzione. Mostriamo come convertire una macchina di Turing con Inserimento M in una TM deterministica a nastro singolo S equivalente. La simulazione usa il simbolo speciale # per segnare le celle che vengono eliminate.

$S =$ "Su input w:

1. La simulazione delle mosse del tipo $\delta(q, a) = (r, b, L)$ procede come nella TM standard: S scrive b sul nastro e muove la testina di una cella a sinistra. Se lo spostamento a sinistra porta la testina sopra un #, allora continua a spostare la testina a sinistra finché non si arriva in una cella che contiene un simbolo diverso dal #.
2. La simulazione delle mosse del tipo $\delta(q, a) = (r, b, R)$ procede come nella TM standard: S scrive b sul nastro e muove la testina di una cella a destra. Se lo spostamento a destra porta la testina sopra un #, allora continua a spostare la testina a destra finché non si arriva in una cella che contiene un simbolo diverso dal #.
3. Per simulare una mossa del tipo $\delta(q, a) = (r, b, D)$ la TM S scrive # nella cella corrente e e muove la testina di una cella a destra. Se lo spostamento a destra porta la testina sopra un #, allora continua a spostare la testina a destra finché non si arriva in una cella che contiene un simbolo diverso dal #.
4. Se in qualsiasi momento la simulazione raggiunge lo stato di accettazione di M, allora S termina con accettazione. Se in qualsiasi momento la simulazione raggiunge lo stato di rifiuto di M, allora S termina con rifiuto. Negli altri casi continua la simulazione dal punto 2."

1. **(12 punti)** Una *Macchina di Turing con stack di nastri* possiede due azioni aggiuntive che modificano il suo nastro di lavoro, oltre alle normali operazioni di scrittura di singole celle e spostamento a destra o a sinistra della testina: può salvare l'intero nastro inserendolo in uno stack (operazione di Push) e può ripristinare l'intero nastro estraendolo dallo stack (operazione di Pop). Il ripristino del nastro riporta il contenuto di ogni cella al suo contenuto quando il nastro è stato salvato. Il salvataggio e il ripristino del nastro non modificano lo stato della macchina o la posizione della sua testina. Se la macchina tenta di "ripristinare" il nastro quando lo stack è vuoto, la macchina va nello stato di rifiuto. Se ci sono più copie del nastro salvate nello stack, la macchina ripristina l'ultima copia inserita nello stack, che viene quindi rimossa dallo stack.

   Mostra che qualsiasi macchina di Turing con stack di nastri può essere simulata da una macchina di Turing standard. *Suggerimento:* usa una macchina multinastro per la simulazione.

Idea: Useremo una macchina di Turing multinastro per la simulazione.

Sia M la macchina con stack di nastri. Costruiamo una macchina di Turing standard S con 3 nastri:

- Nastro 1: simula il nastro di lavoro di M
- Nastro 2: simula lo stack di M
- Nastro 3: tiene traccia del contenuto del nastro di M quando viene salvato nello stack

S simula M come segue: a) Per le operazioni normali, S usa il nastro 1 come M usa il suo nastro di lavoro. b) Per l'operazione Push:

- S copia il contenuto del nastro 1 sul nastro 3
- S sposta la testina del nastro 2 all'inizio e scrive un delimitatore
- S copia il contenuto del nastro 3 sul nastro 2 dopo il delimitatore c) Per l'operazione Pop:
- S verifica se il nastro 2 è vuoto. Se lo è, va nello stato di rifiuto.
- Altrimenti, S cerca l'ultimo delimitatore sul nastro 2

- S copia il contenuto dopo questo delimitatore sul nastro 1, sovrascrivendo il contenuto precedente
- S cancella questa porzione dal nastro 2

Poiché le macchine di Turing multinastro possono essere simulate da macchine di Turing a singolo nastro, questo dimostra che una macchina di Turing con stack di nastri può essere simulata da una macchina di Turing standard.

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.

Sia M = (Q, Γ, δ, q0, qaccept, qreject) una macchina di Turing ad albero binario. Costruiamo una macchina di Turing standard S che simula M:

S utilizza tre nastri:

- Nastro 1: contiene la codifica dell'albero binario
- Nastro 2: memorizza il percorso corrente nell'albero
- Nastro 3: simula il nastro di lavoro di M

Codifica dell'albero:

- Ogni nodo è rappresentato da una sequenza di L (sinistra) e R (destra) che indica il percorso dalla radice
- I nodi sono separati da #

Simulazione:

1. S inizializza il nastro 1 con l'input lungo il ramo sinistro
2. S inizializza il nastro 2 con ε (radice)
3. Per ogni passo di M: a. S legge il simbolo corrente dal nastro 1 usando il percorso nel nastro 2 b. S simula la transizione di M c. Se M si sposta a P (padre), S rimuove l'ultimo simbolo dal nastro 2 d. Se M si sposta a L o R, S aggiunge L o R al nastro 2 e. Se necessario, S espande l'albero sul nastro 1
4. S accetta se M accetta, rifiuta se M rifiuta

Questa costruzione mostra che ogni macchina di Turing ad albero binario può essere simulata da una macchina di Turing standard.

1. **(12 punti)** Una macchina di Turing salva-nastro è simile a una normale macchina di Turing deterministica a nastro singolo semi-infinito, ma può spostare la testina al centro della parte non vuota del nastro. In particolare, se le prime s celle del nastro non sono vuote, allora la testina può spostarsi nella cella numero $\lfloor s/2 \rfloor$. A ogni passo, la testina della TM salva-nastro può spostarsi a sinistra di una cella (L), a destra di una cella (R) o al centro della parte non vuota del nastro (J).

   (a) Dai una definizione formale della funzione di transizione di una TM salva-nastro.

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

Soluzione.

(a) $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R, J\}$

(b) Per dimostrare che TM salva-nastro riconoscono la classe dei linguaggi Turing-riconoscibili dobbiamo dimostrare due cose: che ogni linguaggio Turing-riconoscibile è riconosciuto da una TM salva-nastro, e che ogni linguaggio riconosciuto da una TM salva-nastro è Turing-riconoscibile.

La prima dimostrazione è banale: le TM deterministiche a singolo nastro sono un caso particolare di TM salva-nastro che non effettuano mai la mossa J per saltare al centro della parte non vuota del nastro. Di conseguenza, ogni linguaggio Turing-riconoscibile è riconosciuto da una TM salva-nastro.

Per dimostrare che ogni linguaggio riconosciuto da una TM salva-nastro è Turing-riconoscibile, mostriamo come convertire una macchina di Turing salva-nastro M in una TM deterministica a nastro singolo S equivalente.

S = "Su input w:

1. Inizialmente S mette il suo nastro in un formato che gli consente di implementare l'operazione di salto al centro della parte non vuota del nastro, usando il simbolo speciale # per marcare l'inizio del nastro. Se w è l'input della TM, la configurazione iniziale del nastro è $\#w$.
2. La simulazione delle mosse del tipo $\delta(q, a) = (r, b, L)$ procede come nella TM standard: S scrive b sul nastro e muove la testina di una cella a sinistra. Se lo spostamento a sinistra porta la testina sopra il # che marca l'inizio del nastro, S si muove immediatamente di una cella a destra, lasciando inalterato il #. La simulazione continua con la testina in corrispondenza del simbolo subito dopo il #.
3. La simulazione delle mosse del tipo $\delta(q, a) = (r, b, R)$ procede come nella TM standard: S scrive b sul nastro e muove la testina di una cella a destra.
4. Per simulare una mossa del tipo $\delta(q, a) = (r, b, J)$ la TM S scrive b nella cella corrente, e poi sposta la testina a sinistra fino a ritornare in corrispondenza del # che marca l'inizio del nastro. A questo punto si sposta a destra: se il simbolo dopo il # è un blank, la simulazione continua con la testina in corrispondenza del blank. Se il simbolo dopo il # è diverso dal blank, la TM lo marca con un pallino, poi si sposta a destra fino ad arrivare al primo blank. Dopodiché marca l'ultimo cella non vuota prima del blank e procede a zig-zag, marcando via via una cella all'inizio e una alla fine della porzione di nastro non vuota. Quando la prossima cella da marcare è una cella che è già stata marcata, allora la TM si sposta a sinistra, e marca la cella con un simbolo diverso, come una barra. Poi scorre il nastro per togliere tutti i pallini, e riprende la simulazione con la testina in corrispondenza della cella marcata con la barra.
5. Se in qualsiasi momento la simulazione raggiunge lo stato di accettazione di M, allora S termina con accettazione. Se in qualsiasi momento la simulazione raggiunge lo stato di rifiuto di M, allora S termina con rifiuto. Negli altri casi continua la simulazione dal punto 2."

1. **(12 punti)** Una macchina di Turing "ecologica" (ETM) è uguale a una normale macchina di Turing deterministica a singolo nastro, ma può leggere e scrivere su entrambi i lati di ogni cella del nastro: fronte e retro. La testina di una TM ecologica può spostarsi a sinistra (L), a destra (R) o passare all'altro lato del nastro (F).

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

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

Soluzione.

(a) $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R, F\}$

(b) Per dimostrare che TM ecologiche riconoscono la classe dei linguaggi Turing-riconoscibili dobbiamo dimostrare due cose: che ogni linguaggio Turing-riconoscibile è riconosciuto da una ETM, e che ogni linguaggio riconosciuto da una ETM è Turing-riconoscibile.

La prima dimostrazione è banale: le TM deterministiche a singolo nastro sono un caso particolare di ETM che usano solamente il lato di fronte del nastro e non effettuano mai la mossa F per passare dall'altro lato del nastro. Di conseguenza, ogni linguaggio Turing-riconoscibile è riconosciuto da una ETM.

Per dimostrare che ogni linguaggio riconosciuto da una ETM è Turing-riconoscibile, mostriamo come convertire una macchina di Turing ecologica M in una TM deterministica a due nastri S equivalente. Il primo nastro di S rappresenta il lato di fronte del nastro di M, il secondo nastro rappresenta il lato di retro.

$S =$ "Su input w:

1. S usa lo stato per memorizzare lo stato di M ed il lato dove si trova la testina di M. All'inizio il lato corrente è "fronte" e lo stato di M è quello iniziale.
2. Se il lato corrente è "fronte": leggi il simbolo sotto la testina del primo nastro per stabilire la mossa da fare. Se il lato corrente è "retro", leggi il simbolo sotto la testina del secondo nastro per stabilire la mossa da fare.
3. La simulazione delle mosse del tipo $\delta(q, a) = (r, b, L)$ scrive b sul primo nastro se il lato corrente è "fronte", e scrive b sul secondo nastro se il nastro corrente è "retro". Poi sposta entrambe le testine di una cella a sinistra.
4. La simulazione delle mosse del tipo $\delta(q, a) = (r, b, R)$ scrive b sul primo nastro se il lato corrente è "fronte", e scrive b sul secondo nastro se il nastro corrente è "retro". Poi sposta entrambe le testine di una cella a destra.
5. Per simulare una mossa del tipo $\delta(q, a) = (r, b, F)$ la TM S scrive b sul primo nastro se il lato corrente è "fronte", poi cambia il valore del lato corrente in "retro". Se il lato corrente è "retro", scrive b sul secondo nastro, poi cambia il valore del lato corrente in "fronte". Sposta entrambe le testine di una cella a destra e poi una cella a sinistra, in modo da ritornare nella cella di partenza.
6. r diventa il nuovo stato corrente della simulazione. Se r è lo stato di accettazione di M, allora S termina con accettazione. Se r è lo stato di rifiuto di M, allora S termina con rifiuto. Negli altri casi continua la simulazione dal punto 2."

Per concludere, siccome sappiamo che le TM multinastro riconoscono i linguaggi Turing-riconoscibili, allora abbiamo dimostrato che ogni linguaggio riconosciuto da una ETM è Turing-riconoscibile.

1. **(12 punti)** Una *R2-L3 Turing Machine* è una macchina di Turing deterministica a nastro semi-infinito che può effettuare solo due mosse: spostarsi a destra di due celle (R2), oppure spostarsi a sinistra di tre celle (L3). Se in uno spostamento a sinistra la macchina tenta di spostare la testina a sinistra dell'inizio del nastro, allora lo spostamento termina con la testina nella prima cella del nastro.

   (a) Dai una definizione formale della funzione di transizione di una R2-L3 Turing Machine.

   (b) Dimostra che le R2-L3 Turing Machine riconoscono la classe dei linguaggi Turing-riconoscibili. Usa una descrizione a livello implementativo per definire le macchine di Turing.

   Soluzione.

   (a) $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{L3, R2\}$

   (b) Per dimostrare che le R2-L3 Turing Machine riconoscono la classe dei linguaggi Turing-riconoscibili dobbiamo dimostrare due cose: che ogni linguaggio Turing-riconoscibile è riconosciuto da una R2-L3 Turing Machine, e che ogni linguaggio riconosciuto da una R2-L3 Turing Machine è Turing-riconoscibile.

   Per la prima dimostrazione, mostriamo come convertire una macchina di Turing deterministica a nastro semi-infinito M in una R2-L3 Turing Machine S equivalente.

   S = "Su input w:

   1. Per simulare una mossa del tipo $\delta(q, a) = (r, b, L)$, S scrive b sul nastro e muove la testina di due celle a destra, e subito dopo di tre celle a sinistra. Se lo spostamento a sinistra porta la testina oltre l'inizio del nastro, allora vuol dire che la simulazione era partita dalla prima cella del nastro. In questo caso la simulazione riprende con la testina nella prima cella del nastro, come per le TM standard. Negli altri casi, la simulazione continua con la testina in corrispondenza della cella immediatamente a sinistra di quella di partenza.
   2. Per simulare una mossa del tipo $\delta(q, a) = (r, b, R)$, S scrive b sul nastro e muove la testina di due celle a destra, di nuovo di due celle a destra, e subito dopo di tre celle a sinistra. La simulazione continua con la testina in corrispondenza della cella immediatamente a destra di quella di partenza.
   3. Se in qualsiasi momento la simulazione raggiunge lo stato di accettazione di M, allora S termina con accettazione. Se in qualsiasi momento la simulazione raggiunge lo stato di rifiuto di M, allora S termina con rifiuto. Negli altri casi continua la simulazione dal punto 2."

   Per dimostrare che ogni linguaggio riconosciuto da una R2-L3 Turing Machine è Turing-riconoscibile, mostriamo come convertire una R2-L3 Turing Machine S in una TM deterministica a nastro semi-infinito M equivalente.

   M = "Su input w:

   1. Per simulare una mossa del tipo $\delta(q, a) = (r, b, L3)$, M scrive b sul nastro e muove la testina di tre celle a sinistra. Se lo spostamento a sinistra porta la testina oltre l'inizio del nastro, allora lo sposta meno a sinistra si ferma con la testina nella prima cella del nastro, come per le R2-L3 Turing Machine.
   2. Per simulare una mossa del tipo $\delta(q, a) = (r, b, R2)$, M scrive b sul nastro e muove la testina di due celle a destra.
   3. Se in qualsiasi momento la simulazione raggiunge lo stato di accettazione di S, allora M termina con accettazione. Se in qualsiasi momento la simulazione raggiunge lo stato di rifiuto di S, allora M termina con rifiuto. Negli altri casi continua la simulazione dal punto 2."

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.

   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.

(a) Definizione formale della funzione di transizione di una CPTM: Una CPTM può essere definita come una tupla (Q, Σ, Γ, δ, q0, qaccept, qreject), dove:

- Q è l'insieme degli stati
- Σ è l'alfabeto di input
- Γ è l'alfabeto del nastro (include Σ e il simbolo blank)
- δ: Q × Γ → Q × Γ × {L, R, S, C, V, P} è la funzione di transizione, dove: L: muovi a sinistra R: muovi a destra S: rimani fermo C: seleziona l'inizio della porzione da copiare V: seleziona la fine della porzione da copiare P: incolla il contenuto copiato
- q0 è lo stato iniziale
- qaccept è lo stato di accettazione
- qreject è lo stato di rifiuto

(b) Dimostrazione che le CPTM riconoscono la classe dei linguaggi Turing-riconoscibili:

1. Ogni TM standard può essere simulata da una CPTM ignorando le operazioni C, V, P.
2. Per dimostrare che una CPTM può essere simulata da una TM standard:
    - Usa due nastri aggiuntivi: uno per memorizzare l'inizio della selezione, uno per il contenuto copiato.
    - Simula C memorizzando la posizione corrente sul primo nastro aggiuntivo.
    - Simula V copiando il contenuto selezionato sul secondo nastro aggiuntivo.
    - Simula P copiando il contenuto dal secondo nastro aggiuntivo al nastro principale. Questa simulazione mostra che le CPTM non sono più potenti delle TM standard.

1. (12 punti) Le macchine di Turing con un solo stato sono definite "stateless". Queste macchine rimangono nello stesso stato per tutta la durata della computazione. L'unico modo in cui possono ricordare qualcosa è scriverlo sul nastro. Consideriamo una variante di Turing Machine, chiamata JTM, che è stateless, deterministica e a nastro singolo. Le JTM differiscono dalle TM convenzionali nel modo seguente:

- la testina si estende su tre celle consecutive del nastro, e può leggere/scrivere una stringa di tre simboli del nastro tutti insieme;
- il movimento della testina non è in termini di "blocchi di tre celle": ad ogni transizione la testina non salta al blocco di tre celle adiacenti, ma si sposta solo di una cella a destra o a sinistra;
- se la macchina scrive la stringa di tre simboli $\mathbb{YEA}$ sul nastro la computazione termina con accettazione;
- se la macchina scrive la stringa di tre simboli $\mathbb{NAY}$ sul nastro la computazione termina con rifiuto;
- i simboli $\mathbb{A}, \mathbb{E}, \mathbb{N}, \mathbb{Y}$ sono simboli speciali sempre inclusi nell'alfabeto del nastro;
- la macchina ha un solo stato e rimane in quello stato per sempre. Ciò significa che la nozione di "cambio di stato" (e di conseguenza la nozione stessa di "stati") diventa inutile nel contesto di un JTM.

(a) Dai una definizione formale della funzione di transizione di una JTM.
(b) Dimostra che le JTM riconoscono la classe dei linguaggi Turing-riconoscibili. Usa una descrizione a livello implementativo per definire le macchine di Turing.

(a) Definizione formale della funzione di transizione di una JTM: Una JTM è definita come una tupla (Γ, Σ, δ, q, B, L, R, Y, N), dove:

- Γ è l'alfabeto del nastro (include Σ e i simboli speciali B, L, R, Y, N)
- Σ è l'alfabeto di input

- $\delta : \Gamma \to \Gamma \times \{L, R\}$ è la funzione di transizione
- q è l'unico stato della macchina
- B è il simbolo blank
- L, R sono i simboli per muovere la testina a sinistra o destra
- Y, N sono i simboli per accettare o rifiutare

(b) Dimostrazione che le JTM riconoscono la classe dei linguaggi Turing-riconoscibili:

1. Ogni JTM può essere simulata da una TM standard:
    - Usa un secondo nastro per memorizzare il simbolo che sarebbe stato scritto da una JTM
    - Simula ogni mossa della JTM usando questo nastro ausiliario
2. Per dimostrare che una TM standard può essere simulata da una JTM:
    - Codifica gli stati della TM sul nastro della JTM
    - Usa una codifica del nastro che alterna lo stato corrente e il contenuto del nastro
    - Implementa la funzione di transizione della TM usando la funzione di transizione della JTM

Questa simulazione mostra che le JTM sono equivalenti alle TM standard in termini di potenza computazionale.

1. **(12 punti)** Una macchina di Turing con "undo" (UTM) è una macchina di Turing deterministica a singolo nastro, che può annullare (undo) le operazioni che ha eseguito in precedenza. Oltre alle normali operazioni di spostamento a sinistra e a destra, una UTM può effettuare l'operazione di UNDO, che riporta la macchina alla configurazione immediatamente precedente nella computazione. Se la UTM esegue due UNDO in sequenza, allora ritorna indietro di due configurazioni, e così via per sequenze di UNDO più lunghe.
   (a) Dai una definizione formale della funzione di transizione di una UTM.
   (b) Dimostra che le UTM riconoscono la classe dei linguaggi Turing-riconoscibili. Usa una descrizione a livello implementativo per definire le macchine di Turing.

1. (a) Una UTM è una quintupla $(Q, \Sigma, \Gamma, \delta, q_0)$ dove:

- Q è un insieme finito di stati
- Σ è l'alfabeto di input che non contiene il simbolo blank _
- Γ è l'alfabeto del nastro che contiene _ e Σ
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, UNDO\}$ è la funzione di transizione, dove L, R indicano lo spostamento a sinistra/destra della testina, e UNDO l'annullamento dell'ultima operazione
- $q_0 \in Q$ è lo stato iniziale

(b) Per dimostrare che le UTM riconoscono la classe dei linguaggi Turing-riconoscibili dobbiamo dimostrare due cose:

1. ogni linguaggio Turing-riconoscibile è riconosciuto da una UTM
2. ogni linguaggio riconosciuto da una UTM è Turing-riconoscibile

3. Banale: le TM a singolo nastro sono un caso particolare di UTM che non effettuano mai la mossa UNDO. Quindi ogni linguaggio Turing-riconoscibile è riconosciuto da una UTM.

4. Mostriamo come convertire una UTM M in una TM standard S equivalente a due nastri. S = "Su input w:

1. Simula M scrivendo ogni configurazione in sequenza sul nastro 1.

2. In caso di UNDO, torna indietro di due configurazioni sul nastro 1.

3. Quando M accetta/rifiuta, S accetta/rifiuta."

Siccome S riconosce lo stesso linguaggio di M e le TM a due nastri riconoscono i linguaggi Turing-riconoscibili, abbiamo dimostrato che ogni linguaggio riconosciuto da una UTM è Turing-riconoscibile.

1. **(12 punti)** Una macchina di Turing con "save e restore" (SRTM) è una macchina di Turing deterministica a singolo nastro, che può salvare la configurazione corrente per poi ripristinarla in un momento successivo. Oltre alle normali operazioni di spostamento a sinistra e a destra, una SRTM può effettuare l'operazione di SAVE, che salva la configurazione corrente, e l'operazione di RESTORE che ripristina la configurazione salvata. L'operazione di SAVE sovrascrive una eventuale configurazione salvata in precedenza. Fare il RESTORE in assenza di configurazione salvata non ha effetto: si mantiene inalterata la configurazione corrente.

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

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

1. (a) Una SRTM è una sestupla (Q, Σ, Γ, δ, q_0, q_save) dove:

- Q è un insieme finito di stati
- Σ è l'alfabeto di input che non contiene il simbolo blank _
- Γ è l'alfabeto del nastro che contiene _ e Σ
- δ : Q × Γ → Q × Γ × {L, R, SAVE, RESTORE} è la funzione di transizione, dove L, R indicano lo spostamento a sinistra/destra della testina, SAVE salva la configurazione corrente, e RESTORE ripristina una configurazione salvata in precedenza
- $q_0 \in Q$ è lo stato iniziale
- q_save ∈ Q è lo stato "save e restore"

(b) Per dimostrare che le SRTM riconoscono la classe dei linguaggi Turing-riconoscibili usiamo lo stesso approccio dell'esercizio 1:

1. Banale: le TM a singolo nastro sono un caso particolare di SRTM che non effettuano mai le mosse SAVE e RESTORE. Quindi ogni linguaggio Turing-riconoscibile è riconosciuto da una SRTM.

2. Mostriamo come convertire una SRTM M in una TM standard S equivalente a più nastri. S = "Su input w:

1. Simula M sul nastro di lavoro

2. In caso di SAVE, copia la configurazione corrente su un nuovo nastro

3. In caso di RESTORE, cerca l'ultima configurazione salvata e ripristinala sul nastro di lavoro

4. Quando M accetta/rifiuta, S accetta/rifiuta."

Siccome S riconosce lo stesso linguaggio di M e le TM multi-nastro riconoscono i linguaggi Turing-riconoscibili, abbiamo dimostrato che ogni linguaggio riconosciuto da una SRTM è Turing-riconoscibile.