[A]**3.10** Say that a ***write-once Turing machine*** is a single-tape TM that can alter each tape square at most once (including the input portion of the tape). Show that this variant Turing machine model is equivalent to the ordinary Turing machine model. (Hint: As a first step, consider the case whereby the Turing machine may alter each tape square at most twice. Use lots of tape.)

Mostriamo come una TM a sola scrittura B sia simulabile con una TM deterministica a nastro singolo S.
S memorizza il contenuto di B all'interno di un singolo nastro memorizzando con un simbolo extra la singola scrittura che va a realizzare. Ciò realizza due scritture: la prima per registrare il simbolo da scrivere a la seconda per indicare che è stato copiato, magari con un simbolo con un pallino.
La TM a nastro singolo S funziona come segue:

1) Sostituisce w con la configurazione iniziale del nastro
2) Copia secondo la funzione di transizione di B tutto sul nastro e segue i movimenti a destra e a sinistra di questo
3) Una volta che individua il punto in cui scrivere, aggiorna il nastro scrivendo e, una volta scorso tutto, aggiunge il simbolo delimitatore per indicare di aver raggiunto la fine del nastro
4) Se la macchina raggiunge uno stato di accettazione allora *accetta*, altrimenti *rifiuta*

3.11 A ***Turing machine with doubly infinite tape*** is similar to an ordinary Turing machine, but its tape is infinite to the left as well as to the right. The tape is initially filled with blanks except for the portion that contains the input. Computation is defined as usual except that the head never encounters an end to the tape as it moves leftward. Show that this type of Turing machine recognizes the class of Turing-recognizable languages.

Per risolvere l'esercizio dobbiamo mostrare che:

a) Ogni linguaggio riconosciuto da una TM a doppio nastro infinito è Turing-riconoscibile
b) Ogni linguaggio Turing-riconoscibile dell'alfabeto di TM I è riconosciuto dalla TM a doppio nastro infinito I

a) Il linguaggio di I è Turing-riconoscibile: infatti, considerato un generico alfabeto dato in pasto alla macchina, essa continuerà a scorrere indefinitamente a destra o a sinistra fintanto che non esisterà un modo per farla accettare; essendoci solo una porzione di input, la TM lo scorre tutto andando poi all'estrema sinistra del nastro ad inserire un simbolo, dopo averlo letto tutto, che indica la porzione letta (e che indica la prima lettura); essendo infinito anche a destra, userà dei mark per indicare la fine della lettura e rimpiazza i blank qualora si tratti di fare una scrittura, potenzialmente andando anche in loop. Ad ogni modo, l'infinità del nastro assicura che l'input venga scansionato a prescindere dalla dimensione e di non "scivolare" fuori dalla sua dimensione. In qualsiasi caso, il linguaggio viene riconosciuto.
   Quindi simuliamo M (normale) con I (doppio nastro infinito):
   1) sostituisce la stringa di input con una codifica corrispondente alla macchina di riferimento
   2) segue la funzione di transizione di I
   3) dato che la macchina I ha un doppio nastro infinito, semplicemente, porremo scorrendo tutto l'input un simbolo delimitatore a sinistra, per fare in modo la parte destra del nastro sia come d'abitudine
   4) se raggiunge uno stato d'accettazione, *accetta*, altrimenti riprova dal punto 2

b) Ora simuliamo I con M. Avremo bisogno di una macchina M a doppio nastro finito.
   La simulazione esegue questi passi:
   1) Contiene in un nastro tutta la porzione degli input, che viene copiato nel secondo nastro inizialmente contenente tutti blank
   2) A seconda della lunghezza dell'input esegue la copia della sola stringa w in tutta la sua lunghezza lasciando i blank intorno alla stringa

3) Una volta coperto tutto l'input, se raggiunge uno stato di accettazione, *accetta*, altrimenti va in loop

oppure
Per dimostrare che questa macchina a doppio nastro infinito sia simile ad una Turing Machine ordinaria, allora deve valere che la macchina ordinaria M sia simulabile dalla macchina I e viceversa.

1) Simulazione di M per I

Essendo questa macchina finita, allora basterà marcare la fine a sinistra per D e poi semplicemente continuare a muoversi a destra, prevenendo che la testina si muova a sinistra.

2) Simulazione di I per M

Qua dobbiamo simulare la situazione del nastro finito da una parte con la macchina infinita da entrambi i lati. Quindi immaginiamo una macchina a 2 nastri dove, partendo dalla stringa di input:

5) da una parte avremo la fine del nastro e ciò si simula con una serie di simboli blank
6) dall'altra avremo tutto l'input della macchina, ottenuto con i corrispettivi input presenti

Perciò la macchina è equivalente alla TM ordinaria.

Answer: A TM with doubly infinite tape can simulate an ordinary TM. It marks the left-hand end of the input to detect and prevent the head from moving off of that end.

To simulate the doubly infinite tape TM by an ordinary TM, we show how to simulate it with a 2-tape TM, which was already shown to be equivalent in power to an ordinary TM. The first tape of the 2-tape TM is written with the input string, and the second tape is blank. We cut the tape of the doubly infinite tape TM into two parts, at the starting cell of the input string. The portion with the input string and all the blank spaces to its right appears on the first tape of the 2-tape TM. The portion to the left of the input string appears on the second tape, in reverse order.

3.12 A ***Turing machine with left reset*** is similar to an ordinary Turing machine, but the transition function has the form

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

If $\delta(q, a) = (r, b, \mathrm{RESET})$, when the machine is in state q reading an a, the machine's head jumps to the left-hand end of the tape after it writes b on the tape and enters state r. Note that these machines do not have the usual ability to move the head one symbol left. Show that Turing machines with left reset recognize the class of Turing-recognizable languages.

Dobbiamo mostrare che per riconoscere la classe dei linguaggi Turing-riconoscibili, il linguaggio deve essere nella forma:

$$L = L(M) = \{w \in \Sigma^* : q_0 w \overset{*}{\Rightarrow} u q_{\text{accept}} v, \text{ for some } u, v \in \Gamma^* \}$$

cioè aggiungendo la transizione di *left-reset*, si riconosce lo stesso linguaggio.
In particolare, la macchina TM M è ordinaria, mentre M' implementa left-reset; quando M si muove a destra, M' farà lo stesso. Sappiamo invece che la macchina M' qualora si muova a sinistra, sposta il contenuto del proprio nastro una posizione più a destra per ogni simbolo. Dato che deve essere riconoscibile da una TM normale, quando M si sposta a sinistra, M' marca il simbolo con un mark, per ricordare di dove rispostare il nastro, facendo un nuovo reset per riportare tutti i simboli allo stesso stato di M, così entrambi hanno fatto la stessa cosa.
Idea completa:

Let M be an ordinary TM; let M' be the "left-reset" machine that simulates M as follows: if M makes a right-transition, M' does the same. If M makes a left-transition, M' acts as follows: it marks the current position of the head. To place this mark, if the current square contains a $b \in \Gamma$, M' replaces it with $\hat{b}$. The understanding is that $\Gamma_{M'} = \Gamma_M \cup \hat{\Gamma}_M$, where $\hat{\Gamma}$ consists of the "hatted" versions of all the symbols of Γ. Then M' does a left-reset, and moves the contents of each square one position to the right, except for the hatted square, i.e., the "hat" remains in the same tape position. Once all the squares have been shifted one position to the right, M' does a second left-reset, and travels with right-transitions until it reaches the hatted square, where it now does whatever M would have done.

Facendolo alla Breso:

1) Simuliamo il comportamento della TM con reset a sinistra R con una equivalente standard M.
   Essa ragiona in base alla mossa da eseguire, che può prevedere un eventuale reset, quindi la testina "salta" a sinistra. Quando ciò avviene, ha bisogno di un simbolo per ricordarsi dove saltare esattamente e deve essere un punto oltre la posizione iniziale dell'input, in particolare in un punto precisamente indicato.
   La simulazione di M, dunque, è composta da:
   - la scrittura di un particolare simbolo che faccia ricordare il punto prima dell'input
   - dipende tutto dalla funzione di transizione, in quanto se ci dobbiamo muovere a destra, scriviamo il simbolo sul nastro e si va verso lo stato r
   - se ci dobbiamo muovere a sinistra, eseguiamo la transizione reset, scrivendo il simbolo sul nastro, muovendo la testina a sinistra e andando nello stato di reset. Quando ciò avviene, la testina si sposta ancora a sinistra di una cella e poi ritorna nel vecchio simbolo iniziale
   - se arriva ad uno stato di accettazione o di rifiuto, allora esegue una delle due cose, altrimenti torna al secondo passaggio

2) Simuliamo il comportamento della TM standard M con una a reset a sinistra R.
   In questo caso abbiamo la transizione di reset qualora ci si debba muovere a sinistra ma preservando il comportamento standard.
   La simulazione si compone di:
   1) Scrivere il simbolo iniziale sul nastro e porre, alla fine della ricopiatura dell'input, un particolare simbolo a scelta, mettiamo *
   2) A seconda della funzione di transizione, se mi devo muovere a sinistra, allora la macchina esegue la mossa di reset; dato però che siamo in una macchina con computazione finita, essa deve andare oltre l'ultimo reset, simulando l'andamento a sinistra. Nel qual caso, marca l'ultimo reset avvenuto una volta compiuto e ci spostiamo gradualmente a sinistra, scrivendo i simboli sul nastro e andando verso lo stato *reset* successivo; se non fossero stati marcati reset, avrebbe eseguito comunque la mossa e lo resetta nuovamente
   3) Se ci muoviamo a destra, allora la macchina simula l'andamento a destra, scrivendo il simbolo sul nastro e spostandosi allo stato *r*
   4) Se non si è nello stato di accettazione o rifiuto, si torna al secondo passo

3.13 A ***Turing machine with stay put instead of left*** is similar to an ordinary Turing machine, but the transition function has the form

$$\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\mathrm{R}, \mathrm{S}\}.$$

At each point, the machine can move its head right or let it stay in the same position. Show that this Turing machine variant is *not* equivalent to the usual version. What class of languages do these machines recognize?

Il linguaggio è Turing-riconoscibile, perché ad un certo punto la computazione potrebbe andare in loop (dato che al posto di andare a sx si va in stay). Dobbiamo dunque mostrare che esiste una TM che riconosce questo linguaggio, ma potrebbe essere sottoposto a loop.

Semplicemente si usa una TM M che ha le stesse transizioni e continua la sua computazione leggendo a destra fino alla fine dell'input; nel qual caso accetta. Si muove solo a destra come un DFA e ogniqualvolta si ha una transizione *stay*, il nastro simula l'idea del blank, ma estende sempre un nuovo simbolo alla funzione di transizione, in maniera tale che continui a computare il linguaggio.
La funzione di transizione è descritta da:

$$\delta'(q,\sigma) = \begin{cases} q, & \text{if } q \in \{q_{\texttt{accept}}, q_{\texttt{reject}}\} \\ q_{\texttt{reject}}, & \text{if } M \text{ starting at state } q \text{ and reading } \sigma \text{ keeps staying put.} \\ q', & \text{where } q' \text{ is the state the } M \text{ enters when it first moves right} \\ & \quad \text{upon starting at state } q \text{ and reading } \sigma. \end{cases}$$

Siccome potenzialmente potrebbe sempre andare in loop, non è come le altre macchine Turing-riconoscibili; o va in loop e sta in fase stay oppure va avanti.

2. (**Eco-friendly TM**) An *eco-friendly* Turing machine (ETM) is the same as an ordinary (deterministic) one-tape Turing machine, but it can read and write on both sides of each tape square: front and back.

   At the end of each computation step, the head of the eco-friendly TM can move left (L), move right (R), or flip to the other side of the tape (F).

Per mostrare che una eco-friendly TM riconosce la classe dei linguaggi Turing-riconoscibili, dobbiamo dimostrare:

1) Il linguaggio di una TM Eco-friendly è Turing-riconoscibile
2) Una TM a nastro singolo simula correttamente TM Eco-friendly

Show that eco-friendly TMs recognize the class of Turing-recognizable languages. That is, use a simulation argument to show that they have exactly the same power as ordinary TMs.

1) Descriviamo una TM S a nastro singolo che descrive il comportamento di una TM eco-friendly. Dato che la testina può spostarsi liberamente a sinistra, destra e da tutt'altro lato del nastro, deve tenere traccia delle due corrispondenti posizioni più a sinistra e più destra del nastro, marcando le due corrispondenti posizioni. Seguendo la funzione di transizione di M la macchina si sposta.
   S = "Su input w1, w2, … wn":
   a. Sostituisce la configurazione iniziale del nastro con i simboli corrispondenti della TM eco-friendly
   b. Scorre il nastro dall'inizio dell'input (individuando una cella blank prima di esso) e marca il simbolo più a sinistra e scorre tutto l'input arrivando alla parte più a destra, sempre marcando la cella più a destra
   c. A seconda della funzione di transizione, ragiona in questo modo:
      1) Se δ(r, a) = (s, b, L), la macchina scrive b non marcato sulla cella corrente, si sposta a sinistra e va verso lo stato L
      2) Se δ(r, a) = (s, b, R), la macchina scrive b non marcato sulla cella corrente, si sposta a sinistra e va verso lo stato R
      3) Se δ(r, a) = (s, b, F), la macchina scrive b non marcato sulla cella corrente, poi la macchina si sposta a sinistra cercando il simbolo marcato. Essendo che si deve muovere a destra o a sinistra "saltandovi direttamente", allora poco importa se in questa transizione, la macchina vada a destra o sinistra. La macchina, una volta trovato il simbolo più a sinistra marcato, va nello stato F spostando la

testina dall'altra cella marcata dall'altra parte scorrendo tutto il nastro a destra.

4) Se non siamo nello stato di accettazione o rifiuto, riprendi dal passo 2

2) Ora convertiamo una TM standard in una TM Eco-friendly. Come prima, se devo spostarmi a sinistra o destra, poco cambia, la cosa importante è descrivere il funzionamento per eseguire lo shift dall'altra parte.
   M = "Su input w: "
   a. Prima dell'input, controlla che il nastro sia blank e pone sulla prima stringa dell'input un mark per ricordarsi della transizione. Scorre tutto l'input e pone il mark alla fine di questo.
   b. Simula il comportamento di S, infatti se la mossa da simulare è $\delta(r, a) = (s, b, L)$ si sposta a sinistra scrivendo b sul nastro e va allo stato s, se $\delta(r, a) = (s, b, R)$, si sposta a destra scrivendo b sul nastro e va allo stato s, mentre per la transizione F, scrive b sul nastro e si sposta a sinistra fino a trovare il primo simbolo più a sinistra con un mark. Una volta trovato, lo pone a blank e pone il mark sul simbolo a destra di quest'ultimo che dovrà essere non vuoto. Se ciò accade, l'input esiste ancora. A questo punto saltiamo alla parte destra del nastro e qui similmente, spostiamo il mark a sinistra e poniamo blank sulla cella attuale. La cosa continua ricorsivamente finché non esaurisce tutta la computazione.
   c. Se non siamo nello stato di accettazione o rifiuto, riprendi dal passo 2