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.

Per risolvere l'esercizio dobbiamo dimostrare che
a) ogni linguaggio riconosciuto da una Turing machine con alfabeto ternario è Turing-riconoscibile
b) ogni linguaggio Turing-riconoscibile sull'alfabeto {0, 1, 2} è riconosciuto da una Turing machine con alfabeto binario

Quindi:

a) Questo caso è semplice: una Turing machine con alfabeto ternario 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, 2 e blank. Per esempio, potrebbe contenere simboli marcati o separatori. Per costruire una TM con alfabeto ternario B che simula il comportamento di M dobbiamo come prima cosa stabilire una codifica ternario 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 9 simboli, allora k = 2, perché con 3 bit si rappresentano 9 valori diversi.

La TM con alfabeto ternario B che simula M è definita in questo modo:
B = Su input w:

- Sostituisce $w = w_1 w_2 \ldots w_n$ con la codifica ternaria $C(w_1)C(w_2)\ldots C(w_n)$, e riporta la testina sul primo simbolo di $C(w_1)$.
- 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.
- Aggiorna il nastro in accordo con la funzione di transizione di M:
    a. Se δ(r, a) = (s, b, R), scrive la codifica ternaria di b sul nastro.
    b. Se δ(r, a) = (s, b, L), scrive la codifica ternaria di b sul nastro e sposta la testina a sinistra di 3k celle.
- 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.

4. (8 punti) "Colorare" i vertici di un grafo significa assegnare etichette, tradizionalmente chiamate "colori", ai vertici del grafo in modo tale che nessuna coppia di vertici adiacenti condivida lo stesso colore. Considera la seguente variante del problema 4-COLOR. Oltre al grafo G, l'input del problema comprende anche un *colore proibito* f_v per ogni vertice v del grafo. Per esempio, il vertice 1 non può essere rosso, il vertice 2 non può essere verde, e così via. Il problema che dobbiamo risolvere è stabilire se possiamo colorare il grafo G con 4 colori in modo che nessun vertice sia colorato con il colore proibito.

$$\text{CONSTRAINED-4-COLOR} = \{\langle G, f_1, \ldots, f_n\rangle \mid \text{esiste una colorazione } c_1, \ldots, c_n \text{ degli } n \text{ vertici tale che } c_v \neq f_v \text{ per ogni vertice } v\}$$

(a) Dimostra che CONSTRAINED-4-COLOR è un problema NP.
(b) Dimostra che CONSTRAINED-4-COLOR è NP-hard, usando k-COLOR come problema NP-hard di riferimento, per un opportuno valore di k.

a) Per dimostrare che CONSTRAINED-4-COLOR/C4C è in NP deve esistere un verificatore in tempo polinomiale. Tale verificatore, prendendo in input il grafo G, l'insieme dei colori proibiti *f* e l'insieme di tutti i colori *c* verifica che:
- ogni vertice contenga un colore tra i 4 *f*
- verifica che ogni vertice sia collegato ad un vertice con un colore che non sia proibito per lui (come si vede, ai 4 colori accettati, corrispondo 4 colori proibiti), quindi controllando non stia in *c*
- controlla che tutti i vertici siano adiacenti ai colori permessi *(f)* e che le coppie di vertici adiacenti non contengano lo stesso colore (dunque a due a due non stiano in *c*); se tutti i test sono superati accetta, altrimenti rifiuta. Essendo una verifica fatta in tempo polinomiale, C4C sta in NP.

b) Per dimostrare che C4C è NP-Hard, dimostriamo che si può usare C4C come istanza per risolvere k-Color/KC.
La riduzione deve prendere l'insieme dei colori buoni *f* che sta in C4C ed usarli per risolvere KC.
Dimostriamo quindi che:

- sia S un'istanza buona di KC. Allora è possibile associare ad ogni vertice un colore compreso in *f* in tempo polinomiale (in quanto i colori *f* fanno parte di *k*). Per fare ciò consideriamo un grafo G che duplica ogni singolo vertice e forziamo tutti i vertici duplicati ad un colore fisso (che sono i 4 colori di *f,* quindi per k=4). Aggiungendo ogni colore tra i 4 di *f* ad un vertice, otteniamo per certo un grafo in KC e con colori che vanno bene a C4C.
- sia S' un'istanza buon di C4C. Tra tutti i colori di k, sappiamo che esiste il sottoinsieme che va bene (C_1, … C_n) e l'insieme che non va bene (f_1, … , f_m). Dobbiamo verificare a coppie che i vertici adiacenti siano diversi dall'insieme *f*. Dato che ogni coppia di vertici ha colore diverso, verifichiamo che ogni coppia abbia un colore *c* diverso dal corrispondente colore proibito in *f*; siccome abbiamo usato 4 colori, alla fine si riduce ad un controllo polinomiale facendo in modo di verificare che ogni singolo vertice contenga esattamente 4 colori; se ciò avviene, abbiamo un'istanza corretta di C4C.

Il se e solo se si ha dato che, usando 4 colori, di sicuro fanno parte dei k e similmente, partendo da k colori, verifico che siano tutti diversi e ciò avviene avendo usato effettivamente 4 colori diversi in partenza.
Pertanto, la riduzione è corretta ed è stata eseguita correttamente in tempo polinomiale.