15. 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 un linguaggio può essere riconosciuto da un automa deterministico a coda se e solo se è Turing-riconoscibile.

L'equivalenza tra un automa a coda ed una TM deve essere mostrata; quindi, che il linguaggio può essere riconosciuto dall'automa e dunque dalla TM. Si dimostra creando una simulazione che fa capire che l'automa Q si comporti esattamente come la TM M.

La simulazione di Q come M funziona nel modo seguente:

si consideri l'intero nastro come una coda. Ciascun simbolo viene modificato e il movimento del nastro avviene a destra. Quando più di un numero viene pushato nella coda, si shiftano tutti i valori a destra. Se si raggiunge la fine del nastro, allora il simbolo più a sinistra viene considerato e si capisce di essere arrivati alla fine.

La simulazione di M come Q funziona nel modo seguente:

l'alfabeto della macchina M viene espanso aggiungendo un simbolo extra e un simbolo marcatore a sinistra viene inserito nella coda. I simboli sono pushati a sinistra e letti (pop) a destra.

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.