TestiGara - QuartaEdizione

Esercizi di Gara della IV edizione

Se non specificato altrimenti negli esercizi, si assume che il nastro iniziale sia *non vuoto* ovvero contenga almeno un simbolo.

1. Programmare una macchina di Turing che, dato un nastro iniziale contenente una sequenza arbitraria di 0 e 1, termina la sua esecuzione lasciando sul nastro il bit di parità. Tale bit è 1 se e solo se vi sono un numero dispari di 1 nella sequenza iniziale; altrimenti vale 0.
Esempi:

nastro iniziale	*nastro finale*
1011100010	1
0001100101	0
0000	0

2. Programmare una macchina di Turing che, dato un nastro iniziale contenente una sequenza arbitraria di 0, 1 e 2, termina la sua esecuzione lasciando sul nastro un bit uguale a 1 se e solo se la sequenza è del tipo $0^n\,1^n\,0^n$, ovvero ci sono n simboli 0, poi n simboli 1 e infine n simboli 0, per un qualche intero $n > 0$. Altrimenti, il bit lasciato sul nastro è 0.
Esempi:

nastro iniziale	*nastro finale*
000111000	1
0011100	0
00100	0

3. Programmare una macchina di Turing che, dato un nastro iniziale contenente una sequenza arbitraria di 0 e 1, termina la sua esecuzione lasciando sul nastro una sequenza finale ottenuta da quella iniziale raddoppiando gli 1.
Esempi:

nastro iniziale	*nastro finale*
00101101010	0011011110110110
1001	110011
0000	0000

4. Sia fissata a priori la sequenza binaria $P = 10110$. Programmare una macchina di Turing che, dato un nastro iniziale contenente una sequenza arbitraria di 0 e 1, termina la sua esecuzione lasciando sul nastro un bit uguale a 1 se e solo se la sequenza iniziale contiene P. Altrimenti, il bit lasciato sul nastro è 0.
Esempi:

nastro iniziale	*nastro finale*
0100101101101	1
100010100	0
0100101101101010	1

5. Programmare una macchina di Turing che, dato un nastro iniziale contenente una sequenza di simboli 0, 1, 2 e 3, termina la sua esecuzione lasciando sul nastro la sequenza finale contenente gli stessi elementi ordinati in maniera non decrescente.
Esempi:

nastro iniziale	*nastro finale*
20230321020320002301	00000000112222223333
001223	001223
322100	001223

6. Programmare una macchina di Turing che, dato un nastro iniziale contenente un intero X rappresentato con cifre decimali, termina la sua esecuzione lasciando sul nastro il risultato della moltiplicazione di X per 11 (in cifre decimali).
Esempi:

nastro iniziale	*nastro finale*
48273	531003
0	0
12	132

7. Programmare una macchina di Turing che, dato un nastro iniziale contenente una sequenza arbitraria di 0 e 1, termina la sua esecuzione lasciando sul nastro un bit uguale a 1 se e solo se la sequenza iniziale è della forma xx, ovvero è una sequenza ripetuta due volte. Altrimenti, il bit lasciato sul nastro è 0.
Esempi:

nastro iniziale	*nastro finale*
01110111	1
11011011	0
101101	1

8. Programmare una macchina di Turing che, dato un nastro iniziale contenente una sequenza del tipo $xSyRz$ (in cui x, y e z sono sequenze di simboli 0, 1 e 2, con x e y di uguale lunghezza), termina la sua esecuzione lasciando sul nastro la sola sequenza z in cui ciascun carattere di x è stato sostituito, ordinatamente, con il corrispondente carattere di y.
Esempi:

nastro iniziale	*nastro finale*
1S0R12212	02202

02S10R02201	10011
10S02R10012	22222

). Programmare una macchina di Turing che, data una sequenza del tipo $n_1 x n_2 y n_3 z$ ... con ciascun numero $n_i > 0$ rappresentato con cifre decimali, termini lasciando sul nastro una sequenza di n_1 simboli x seguiti da n_2 simboli y, da n_3 simboli z ecc. Si assuma che i simboli x y e z siano A, B o C.
Esempi:

nastro iniziale	*nastro finale*
3A2B1A	AAABBA
1C2B1C3A	CBBCAAA
10C	CCCCCCCCCC

). Programmare una macchina di Turing che simuli il comportamento di una versione semplificata di macchina di Turing rappresentata come segue. La macchina si programma con quintuple, gli stati sono rappresentati da sequenze di 'S', mentre i simboli accettati sono 1, 0 e N (che rappresenta il blank o spazio). Una quintupla descritta da
`<stato><car><stato><car><dir>`

viene *codificata su nastro* come segue: `<stato>` è una sequenza lunga a piacere di 'S'; `<car>` è uno dei simboli '0', '1' e 'N'; infine,`<dir>` è 0 (sinistra), 1 (destra) oppure N (stai fermo). Seguono alcuni esempi di quintuple e della loro codifica corrispondente:

quintupla	*codifica su nastro*
`(sss, 0, s, 1, >)`	SSS0S11
`(sss, -, ss, 0, -)`	SSSNSS0N
`(s, -, ss, -, <)`	SNSSN0

Indicata con `<quintupla>` la codifica di una quintupla nel formato appena descritto, un programma sarà espresso come segue:

`B<quintupla><quintupla>...<quintupla>I`

Ad esempio il programma che scambia 0 con 1 e viceversa può essere realizzato tramite due quintuple:

quintupla	*codifica su nastro*
`(s, 0, s, 1, >)`	S0S11
`(s, 1, s, 0, >)`	S1S01

Viene quindi codificato come: BS0S11S1S01I

L'input della macchina di Turing così codificata si trova subito dopo la 'I' e può essere composto solo da '0', '1' e 'N'. La testina, indicata col simbolo 'T', precede il carattere che indica.

Un nastro contenente una macchina di Turing e il suo input potrebbe essere il seguente:
0001101N0T10010NN10

Si assuma che l'input della macchina simulata possa espandersi solo a destra. Quando l'input ha la testina come ultimo carattere, viene aggiunta una 'N' in fondo:

01010100010010T
→
01010100010010TN

La macchina inizialmente si trova nello stato 'S' e la testina segue il simbolo 'I'. La computazione deve terminare sempre con la testina sul simbolo 'I'. Un esempio di input è quindi il seguente:

BS0S11S1S01IT0100010100101

Si scriva un programma che interpreti una tale descrizione di macchina di Turing, rispettando l'input proposto. Inoltre seguendo l'algoritmo codificato dalle quintuple sul nastro, esegua il programma sul nastro di input.

Suggerimento: si segua il seguente algoritmo.

Lo stato corrente della macchina viene messo prima della `B` che indica l'inizio del programma. Subito prima dello stato si trova il carattere indicato dalla testina 'T'. Il nastro a un certo punto del calcolo potrebbe essere il seguente:

0SBS0S11S1S01IT0100010100101

Lo stato attuale della macchina simulata è 'S' e 0 è il simbolo che segue 'T'. L'interprete funziona come segue:

1. Scrive lo stato iniziale 'S' prima del simbolo 'B'.
2. Legge il simbolo corrente dopo la 'T' e scrive nel primo carattere bianco a sinistra dello stato corrente.
3. Posiziona la testina del simulatore sulla prima regola da controllare.
4. Confronta lo stato corrente con quello della regola corrente:
   - Lo stato coincide. Verifica che il carattere dopo lo stato sia uguale a quello corrente:
     - Coincide anche il simbolo. Applica la regola sostituendo allo stato corrente quello indicato dalla regola e andando a scrivere il nuovo simbolo dopo il carattere T. Sposta la testina come indicato nella regola. Legge il nuovo simbolo corrente e si ricomincia col passo 1.
     - Il simbolo non coincide. Va al passo 4.
   - Lo stato non coincide. Esamina la prossima regola. Va al passo 4.
5. Si trova l'inizio della prossima regola oppure il simbolo 'I'. Nel primo caso si torna al passo 3. Nel secondo caso, la computazione termina.

Esempi:

nastro iniziale	*nastro finale*	
BSNSS11SSNSSS01SSSNSSSS0NITN	BSNSS11SSNSSS01SSSNSSSS0NI10T0	Scrive 100
BS0S11S1S01IT0100010100101	NSBS0S11S1S01I1011101011010TN	scambia 1 con 0 e viceversa

Soluzione:

```
# Realizza la Macchina Universale

# Stato iniziale
(0, B, init, B, <)
(init, -, start, S, >)

# Va a leggere i simboli
(start, B, gord, B, >)
```

```
(gord, S, gord, S, >)
(gord, 0, gord, 0, >)
(gord, 1, gord, 1, >)
(gord, N, gord, N, >)
(gord, I, gord, I, >)
(gord, T, rd, T, >)

# Legge i simboli
(rd, 0, rd0, 0, <)
(rd, 1, rd1, 1, <)
(rd, N, rdN, N, <)
(rd, -, rd, N, -)

# Memorizza il simbolo all'estremita'
# sinistra della stringa
(rd0, T, rd0, T, <)
(rd0, 1, rd0, 1, <)
(rd0, 0, rd0, 0, <)
(rd0, I, rd0, I, <)
(rd0, S, rd0, S, <)
(rd0, N, rd0, N, <)
(rd0, B, rd0, B, <)
(rd0, -, gom, 0, >)

(rd1, T, rd1, T, <)
(rd1, 1, rd1, 1, <)
(rd1, 0, rd1, 0, <)
(rd1, I, rd1, I, <)
(rd1, S, rd1, S, <)
(rd1, N, rd1, N, <)
(rd1, B, rd1, B, <)
(rd1, -, gom, 1, >)

(rdN, T, rdN, T, <)
(rdN, 1, rdN, 1, <)
(rdN, 0, rdN, 0, <)
(rdN, I, rdN, I, <)
(rdN, S, rdN, S, <)
(rdN, N, rdN, N, <)
(rdN, B, rdN, B, <)
(rdN, -, gom, N, >)

# Inizia a fare il match dei simboli
(gom, S, gom, S, >)
(gom, B, gomS, B, >)
(gomS, S, msa, A, <)

# Stati di match
(msa, S, msa, S, <)
(msa, N, msa, N, <)
(msa, 1, msa, 1, <)
(msa, 0, msa, 0, <)
(msa, B, cs, B, <)
(cs, S, cs, S, <)
(cs, X, mark, X, >)
```

(cs, 0, mark, 0, >)
(cs, 1, mark, 1, >)
(cs, N, mark, N, >)

(mark, S, marked, X, >)
(mark, B, fvfy, B, <)
(marked, S, marked, S, >)
(marked, N, marked, N, >)
(marked, 1, marked, 1, >)
(marked, 0, marked, 0, >)
(marked, B, marked, B, >)
(marked, A, next, S, >)

Prossimo match
(next, S, msa, A, <)

(next, 0, vrfy0, Z, <)
(next, 1, vrfy1, U, <)
(next, N, vrfyN, M, <)

(vrfy0, S, vrfy0, S, <)
(vrfy0, 1, vrfy0, 1, <)
(vrfy0, 0, vrfy0, 0, <)
(vrfy0, N, vrfy0, N, <)
(vrfy0, B, vfy0, B, <)

(vrfy1, S, vrfy1, S, <)
(vrfy1, 1, vrfy1, 1, <)
(vrfy1, 0, vrfy1, 0, <)
(vrfy1, N, vrfy1, N, <)
(vrfy1, B, vfy1, B, <)

(vrfyN, S, vrfyN, S, <)
(vrfyN, 1, vrfyN, 1, <)
(vrfyN, 0, vrfyN, 0, <)
(vrfyN, N, vrfyN, N, <)
(vrfyN, B, vfyN, B, <)

(vfy0, X, vfy0, X, <)
(vfy0, S, fvfy, S, <)
(vfy0, 0, apply, -, >)
(vfy0, 1, fail, 1, >)
(vfy0, N, fail, N, >)

(vfy1, X, vfy1, X, <)
(vfy1, S, fvfy, S, <)
(vfy1, 0, fail, 0, >)
(vfy1, 1, apply, -, >)
(vfy1, N, fail, N, >)

(vfyN, X, vfyN, X, <)
(vfyN, S, fvfy, S, <)
(vfyN, 0, fail, 0, >)
(vfyN, 1, fail, 1, >)
(vfyN, N, apply, -, >)

Fallisce
(fvfy, S, fvfy, S, <)
(fvfy, X, fvfy, S, <)
(fvfy, 1, fail, 1, >)
(fvfy, 0, fail, 0, >)
(fvfy, N, fail, N, >)

(fail, X, fail, S, >)
(fail, S, fail, S, >)
(fail, B, fail, B, >)
(fail, 0, fail, 0, >)
(fail, 1, fail, 1, >)
(fail, N, fail, N, >)

(fail, U, nrule, 1, >)
(fail, Z, nrule, 0, >)
(fail, M, nrule, N, >)
(fail, A, failS, S, >)

(failS, S, failS, S, >)
(failS, 0, nrule, 0, >)
(failS, 1, nrule, 1, >)
(failS, N, nrule, N, >)

Prossima tupla
(nrule, S, nrule, S, >)
(nrule, N, nrules, N, >)
(nrule, 0, nrules, 0, >)
(nrule, 1, nrules, 1, >)
(nrules, 0, nrules, 0, >)
(nrules, 1, nrules, 1, >)
(nrules, N, nrules, N, >)
(nrules, S, msa, A, <)
(nrules, I, STOP, I, -)

Applica la tupla
(apply, X, apply, -, >)
(apply, B, apply, B, >)
(apply, S, apply, S, >)
(apply, 0, apply, 0, >)
(apply, 1, apply, 1, >)
(apply, N, apply, N, >)

(apply, U, sets, 1, >)
(apply, Z, sets, 0, >)
(apply, M, sets, N, >)

Prossimo stato
(sets, S, gow, X, <)
(gow, S, gow, S, <)
(gow, N, gow, N, <)
(gow, B, gow, B, <)
(gow, 0, gow, 0, <)
(gow, 1, gow, 1, <)

(gow, -, gowb, S, >)

(gowb, S, gowb, S, >)
(gowb, N, gowb, N, >)
(gowb, B, gowb, B, >)
(gowb, 0, gowb, 0, >)
(gowb, 1, gowb, 1, >)
(gowb, X, sets, S, >)

(sets, 0, s0d?, 0, >)
(sets, 1, s1d?, 1, >)
(sets, N, sNd?, N, >)

(s0d?, 0, s0d0, 0, >)
(s0d?, 1, s0d1, 1, >)
(s0d?, N, s0dN, N, >)
(s1d?, 0, s1d0, 0, >)
(s1d?, 1, s1d1, 1, >)
(s1d?, N, s1dN, N, >)
(sNd?, 0, sNd0, 0, >)
(sNd?, 1, sNd1, 1, >)
(sNd?, N, sNdN, N, >)

Aggiorna il nastro
(s0d0, S, s0d0, S, >)
(s0d0, N, s0d0, N, >)
(s0d0, I, s0d0, I, >)
(s0d0, 0, s0d0, 0, >)
(s0d0, 1, s0d0, 1, >)
(s0d0, T, ws0d0, T, >)
(ws0d0, 0, gd0, 0, <)
(ws0d0, 1, gd0, 0, <)
(ws0d0, N, gd0, 0, <)

(s0d1, S, s0d1, S, >)
(s0d1, N, s0d1, N, >)
(s0d1, I, s0d1, I, >)
(s0d1, 0, s0d1, 0, >)
(s0d1, 1, s0d1, 1, >)
(s0d1, T, gd1, 0, >)

(s0dN, S, s0dN, S, >)
(s0dN, N, s0dN, N, >)
(s0dN, I, s0dN, I, >)
(s0dN, 0, s0dN, 0, >)
(s0dN, 1, s0dN, 1, >)
(s0dN, T, ws0dN, T, >)
(ws0dN, 0, rd0, 0, <)
(ws0dN, 1, rd0, 0, <)
(ws0dN, N, rd0, 0, <)

(s1d0, S, s1d0, S, >)
(s1d0, N, s1d0, N, >)
(s1d0, I, s1d0, I, >)
(s1d0, 0, s1d0, 0, >)

(s1d0, 1, s1d0, 1, >)
(s1d0, T, ws1d0, T, >)
(ws1d0, 0, gd0, 1, <)
(ws1d0, 1, gd0, 1, <)
(ws1d0, N, gd0, 1, <)

(s1d1, S, s1d1, S, >)
(s1d1, N, s1d1, N, >)
(s1d1, I, s1d1, I, >)
(s1d1, 0, s1d1, 0, >)
(s1d1, 1, s1d1, 1, >)
(s1d1, T, gd1, 1, >)

(s1dN, S, s1dN, S, >)
(s1dN, N, s1dN, N, >)
(s1dN, I, s1dN, I, >)
(s1dN, 0, s1dN, 0, >)
(s1dN, 1, s1dN, 1, >)
(s1dN, T, ws1dN, T, >)
(ws1dN, 0, rd1, 1, <)
(ws1dN, 1, rd1, 1, <)
(ws1dN, N, rd1, 1, <)

(sNd0, S, s1d0, S, >)
(sNd0, N, s1d0, N, >)
(sNd0, I, s1d0, I, >)
(sNd0, 0, s1d0, 0, >)
(sNd0, 1, s1d0, 1, >)
(sNd0, T, ws1d0, T, >)
(wN1d0, 0, gd0, N, <)
(wN1d0, 1, gd0, N, <)
(wN1d0, N, gd0, N, <)

(sNd1, S, s1d1, S, >)
(sNd1, N, s1d1, N, >)
(sNd1, I, s1d1, I, >)
(sNd1, 0, s1d1, 0, >)
(sNd1, 1, s1d1, 1, >)
(sNd1, T, gd1, N, >)

(sNdN, S, s1dN, S, >)
(sNdN, N, s1dN, N, >)
(sNdN, I, s1dN, I, >)
(sNdN, 0, s1dN, 0, >)
(sNdN, 1, s1dN, 1, >)
(sNdN, T, wsNdN, T, >)
(wsNdN, 0, rdN, N, <)
(wsNdN, 1, rdN, N, <)
(wsNdN, N, rdN, N, <)

Sposta a sinistra
(gd0, T, gd0, T, <)
(gd0, 0, w0, T, >)
(gd0, 1, w1, T, >)
(gd0, N, wN, T, >)

```
(w0, T, rd0, 0, <)
(w1, T, rd1, 1, <)
(wN, T, rdN, N, <)

# Sposta a destra
(gd1, 0, rd, T, >)
(gd1, 1, rd, T, >)
(gd1, N, rd, T, >)

# Fine programma
```