2. (12 punti) I *grawlix* sono sequenze di simboli senza senso che sostituiscono le parolacce nei fumetti.

Un linguaggio è *volgare* se contiene almeno un grawlix. Considera il problema di determinare se il linguaggio di una TM è volgare.

(a) Formula questo problema come un linguaggio $GROSS_{TM}$.
(b) Dimostra che il linguaggio $GROSS_{TM}$ è indecidibile.

(a) $GROSS_{TM} = \{ \langle L, w \rangle \mid L$ è volgare se $\exists\, w = $ GRAWLIX $\}$

↓ M è una TM e $w \in L$

2. (12 punti) I *grawlix* sono sequenze di simboli senza senso che sostituiscono le parolacce nei fumetti.

Un linguaggio è *volgare* se contiene almeno un grawlix. Considera il problema di determinare se il linguaggio di una TM è volgare.

(a) Formula questo problema come un linguaggio $GROSS_{TM}$.
(b) Dimostra che il linguaggio $GROSS_{TM}$ è indecidibile.

$A \leq_m B$

$\boxed{\langle M', x \rangle}$ ↙ M, w

$\boxed{A_{TM} \quad w \in L}$ $\boxed{GROSS_{TM}}$

[L = VOLGARE SSE W = GRAWLIX]

F = SU INPUT $\langle M, w \rangle$:

→ COPIA SU M' TUTTO L'INPUT

→ SE X CONTIENE ALMENO UN GRAWLIX

→ ESEGUE M SU W

- SE M ACCETTA, $w \in L$ = VOLGARE
- SE M RIFIUTA, RIFIUTA

→ RITORNA $\langle M', x \rangle$

```
M'(input x):
1. Ignora l'input x
2. Simula M su w
3. Se M accetta w:
   - Scrivi e accetta il grawlix "*#@%!"
```

ALTERNATIVA

$$x \in A \iff f(x) \in B$$

$x \in A_{TM}$ SSE $\langle M, w \rangle \in CROSS_{TM}$

⇒: Se ⟨M, w⟩ ∈ A_TM, allora M accetta w.
Quindi M' termina se solo se accettà "*#@%!" ∈ GRAWLIX.
Pertanto L(M') ∩ GRAWLIX ≠ ∅, quindi ⟨M'⟩ ∈ GROSS_TM.

⇐: Se ⟨M'⟩ ∈ GROSS_TM, allora L(M') ∩ GRAWLIX ≠ ∅.
L'unico grawlix che M' può accettare è "*#@%!", e questo accade solo se M accetta w.
Quindi ⟨M, w⟩ ∈ A_TM.

2. (9 punti) La *traslitterazione* è un tipo di conversione di un testo da una scrittura a un'altra che prevede la sostituzione di lettere secondo modalità prevedibili. La tabella seguente mostra il sistema di traslitterazione che permette di convertire la scrittura Cherokee nell'alfabeto latino:

Ꭰ	a	Ꭱ	e	Ꭲ	i	Ꭳ	o	Ꭴ	u	Ꭵ	v	Ꭶ	ga	Ꭷ	ka	Ꭸ	ge	Ꭹ	gi	Ꭺ	go
Ꭻ	gu	Ꭼ	gv	Ꭽ	ha	Ꭾ	he	Ꭿ	hi	Ꮀ	ho	Ꮁ	hu	Ꮂ	hv	Ꮃ	la	Ꮄ	le	Ꮅ	li
Ꮆ	lo	Ꮇ	lu	Ꮈ	lv	Ꮉ	ma	Ꮊ	me	Ꮋ	mi	Ꮌ	mo	Ꮍ	mu	Ꮎ	na	Ꮏ	hna	Ꮐ	nah
Ꮑ	ne	Ꮒ	ni	Ꮓ	no	Ꮔ	nu	Ꮕ	nv	Ꮖ	qua	Ꮗ	que	Ꮘ	qui	Ꮙ	quo	Ꮚ	quu	Ꮛ	quv
Ꮝ	s	Ꮜ	sa	Ꮞ	se	Ꮟ	si	Ꮠ	so	Ꮡ	su	Ꮢ	sv	Ꮣ	da	Ꮤ	ta	Ꮥ	de	Ꮦ	te
Ꮧ	di	Ꮨ	ti	Ꮩ	do	Ꮪ	du	Ꮫ	dv	Ꮬ	dla	Ꮭ	tla	Ꮮ	tle	Ꮯ	tli	Ꮰ	tlo	Ꮱ	tlu
Ꮲ	tlv	Ꮳ	tsa	Ꮴ	tse	Ꮵ	tsi	Ꮶ	tso	Ꮷ	tsu	Ꮸ	tsv	Ꮹ	wa	Ꮺ	we	Ꮻ	wi	Ꮼ	wo
Ꮽ	wu	Ꮾ	wv	Ꮿ	ya	Ᏸ	ye	Ᏹ	yi	Ᏺ	yo	Ᏻ	yu	Ᏼ	yv						

Dati due alfabeti Σ e Γ, possiamo definire formalmente una traslitterazione come una funzione $T : \Sigma \mapsto \Gamma^*$ che mappa ogni simbolo di Σ in una stringa di simboli in Γ.

Dimostra che se $L \subseteq \Sigma^*$ è un linguaggio *context-free* e T è una traslitterazione, allora anche il seguente linguaggio è *context-free*:

$$T(L) = \{w \in \Gamma^* \mid w = T(a_0)T(a_1)\dots T(a_n) \text{ per qualche } a_0a_1\dots a_n \in L\}.$$

SSE L è CFG

ALTERNATIVE

∃ G è CFG (CHOMSKY)

PDA P

CHEROKEE -> LATINO

$S \to GA$

T

$\varepsilon, S \to GA$

O

$G \to G'$

✓ $G \to P$ (PDA)

∀ SIMBOLO G ∈ L, ESISTE UNA TRASLITTERAZIONE FINITA.

CHE NOI POSSIAMO DLTO STRANO CON UN PDA

PER OGNI SIMBOLO CHEROKEE:

- POP PER OGNI CHEROKEE
- PUSH DELL'INSIEME DEI CARATTERI LATINI CORRISPONDENTI

$\forall \alpha \in \Sigma, \quad T(\alpha) = T(a_1)\, T(a_2) \ldots T(a_k)$

∃ PDA → ∃ CFG EQUIVALENTE

Sia G = (V, Σ, R, S) una CFG per L. Costruiamo G' = (V, Γ, R', S) dove:
R' è ottenuta da R sostituendo ogni produzione A → α con A → T(α), dove:

Se $\alpha = a \in \Sigma$, allora $T(\alpha) = T(a)$
Se $\alpha = A_1A_2 \ldots A_k$ (variabili), allora $T(\alpha) = A_1A_2 \ldots A_k$
Se $\alpha = a_1A_1a_2A_2 \ldots a_kA_ka_{k+1}$, allora $T(\alpha) = T(a_1)A_1T(a_2)A_2 \ldots T(a_k)A_kT(a_{k+1})$

Correttezza:

T(L) ⊆ L(G'):

Se w ∈ T(L), allora w = T(u) per qualche u ∈ L.
Poiché u ∈ L(G), esiste una derivazione S ⇒* u in G.
Esiste un PDA che per ogni transizione, realizza un push e un pop con i corrispettivi caratteri.
La stessa sequenza di applicazioni di produzioni in G' produce S ⇒* T(u) = w.

L(G') ⊆ T(L):

Se w ∈ L(G'), esiste una derivazione S ⇒* w in G'.
Questa derivazione corrisponde a una derivazione S ⇒* u in G per qualche u ∈ Σ*, e w = T(u). Esiste un output latino il quale è completamente corrisposto nei Cherokee. Quindi w ∈ T(L).

3. (9 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 $0, 1, 2$ 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.

∃ UNA TM S A NASTRO SINGOLO EQUIVALENTE

S = SU INPUT w:

$\Sigma = \{0,1,2\}$

$\Gamma = \{0,1,2,\sqcup\}$

- $\delta(q,a) = (r,b,L)$

SCRIVIAMO "b" SPOSTANDOCI "L" SU L NASTRO ANDANDO NELLO STATO "r", ESISTE UNA CODIFICA $P \in \{0,1,2,\sqcup\}$ TALE CHE SPOSTANDOCI A SX SCRIVIAMO TUTTI I SIMBOLI
$a \to 00$, $b \to 01$, ...
QUANDO $\exists \sqcup \Rightarrow$ SPOSTO TUTTO A SX DI UNA CELLA NEL CASO IN CUI L'INPUT $\notin \{0,1,2\}$

- $\delta(q,a) = (r,b,R)$

SCRIVIAMO "b" SPOSTANDOCI "R" SU L NASTRO ANDANDO NELLO STATO "r", ESISTE UNA CODIFICA $P \in \{0,1,2,\sqcup\}$
QUANDO $\exists \sqcup \Rightarrow$ SPOSTO TUTTO A DX DI UNA CELLA NEL CASO IN CUI L'INPUT $\notin \{0,1,2\}$

Sia L un linguaggio Turing-riconoscibile su {0,1,2}.
Allora esiste una TM standard $M = (Q, \{0,1,2\}, \Gamma', \delta, q_0, q_{acc}, q_{rej})$ che riconosce L, dove Γ' può contenere simboli arbitrari.
Costruiamo una TM con alfabeto ternario $M' = (Q', \{0,1,2\}, \{0,1,2,\sqcup\}, \delta', q_0', q_{acc}', q_{rej}')$ equivalente a M.
Strategia: Simulare M usando solo i simboli $\{0,1,2,\sqcup\}$.
Codifica dei simboli:
Per ogni simbolo $s \in \Gamma'$, definiamo una codifica $encode(s) \in \{0,1,2\}^+$ tale che:

I simboli distinti hanno codifiche distinte
Le codifiche hanno tutte la stessa lunghezza $k = \lceil \log_3 |\Gamma'| \rceil$

Esempio di codifica:
Se $\Gamma' = \{0,1,2,\sqcup,a,b,c\}$, allora k = 2 e possiamo usare:

$0 \to 00$, $1 \to 01$, $2 \to 02$, $\sqcup \to 10$
$a \to 11$, $b \to 12$, $c \to 20$

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.

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

(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) VERIFICATORE V CHE, DATO UN CERTIFICATO C COMPOSTO DA $\langle G, f_1 .. f_n \rangle$

(1) ESISTE UN VERTICE "a" = CON UN COLORE PROIBITO

(2) ESISTE UN ARCO $e = (a, b)$ → COLLEGA "a" CON "b"

(3) PERCORRO TUTTI I VERTICI
$\exists$ COLORAZIONE $C_1 = V_1, C_2 = V_2 \ldots C_n = V_n$

(4) $\forall$ COPPIA, $C_n \neq f_v$

(5) RITORNA SI e NO BREAK

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.

$A \leq_m B$

4 - COLOR $\le_m$ CONSTRAINED - 4 - COLOR

$\langle C, M, Y, K \rangle$ R

BLACK = PROIBITO

$\Rightarrow$ SE $\exists$ UN'ISTANZA DI 4 - COLOR $\langle G, v_1 \dots v_m \rangle$, $\exists$ G' EQUIVALENTE A G TALE CHE PER OGNI COPPIA DI VERTICI ESISTE UN ARCO CHE LI COLLEGA

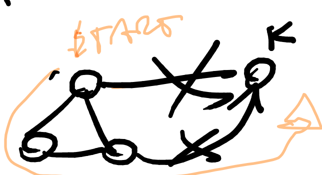
$[C \ne Y]$ $(M \ne K)$

CONTINUA A PERCORRERE IL GRAFO SEGUENDO LA 4 - COLORAZIONE

$\forall v_i, \exists c_i \Rightarrow$ COLORAZIONE $\in$ 4 - COLOR.

PER AVERE UN'ISTANZA VALIDA E CONSTRAINED, $\forall$ COLORAZIONE, SCONNETTIAMO GLI ARCHI CHE HANNO IL COLORE PROIBITO

$c_v \ne f_v$



$\leftarrow$ VERTICI ISOLATI = PROIBITO $\underline{K}$

Dato: Un grafo $G = (V, E)$ con $V = \{v_1, v_2, \dots, v_n\}$
Costruzione dell'istanza Constrained-4-Color:

Grafo $G' = G$ (stesso grafo)
Colori proibiti: f_i = giallo per ogni $v_i \in V$

Spiegazione: Proibiamo il quarto colore (giallo) a tutti i vertici,
effettivamente riducendo il problema a una 3-colorazione.

Correttezza della riduzione:
$\Rightarrow$: Se $G \in$ 3-Color, allora G può essere colorato con {rosso, blu, verde}.
Questa stessa colorazione è valida per Constrained-4-Color perché:

Non usa il colore giallo (rispetta i vincoli f_i = giallo)
Rispetta i vincoli di adiacenza (era una 3-colorazione valida)

Quindi $\langle G', f_1, \dots, f_n\rangle \in$ Constrained-4-Color.
$\Leftarrow$: Se $\langle G', f_1, \dots, f_n\rangle \in$ Constrained-4-Color, allora esiste una colorazione valida c.
Poiché f_i = giallo per ogni vertice, la colorazione c usa solo {rosso, blu, verde}.

Poiché c rispetta i vincoli di adiacenza, è una 3-colorazione valida di G.

CONSTRAINED $\rightarrow$ NP-COMPLETO $\rightarrow$ NP (a)
$\rightarrow$ NP-HARD (b)

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."

4. (9 punti) Supponiamo di avere un sistema elettorale composto da n elettori, dove ogni elettore i ha un "peso" $W[i]$, corrispondente al numero di voti che rappresenta. Nel caso di una votazione a maggioranza semplice, una coalizione di elettori ha bisogno di un numero di voti strettamente superiore alla metà della somma dei pesi per vincere. L'elettore n è detto *pivot* se esiste una situazione in cui il voto dell'elettore "conta", ossia dove l'aggiunta dell'elettore n ai voti "sì" rende il "sì" la maggioranza, ma l'aggiunta ai voti "no" rende il "no" maggioranza. Formalmente, l'elettore n è un pivot se esiste una coalizione di elettori $C \subseteq \{1, \ldots, n-1\}$ tale che

$$\sum_{j \in C} W[j] < \frac{1}{2} \sum_{j=1}^{n} W[j] \qquad \text{(C senza i voti di n è minoranza)} \rightarrow S_1$$

$$\sum_{j \in C} W[j] + W[n] > \frac{1}{2} \sum_{j=1}^{n} W[j] \qquad \text{(C con i voti di n è maggioranza)} \rightarrow S_2$$

$\} S$

e possiamo rappresentare il problema dell'elettore pivot con il linguaggio

$$PIVOT = \big\{\langle n, W\rangle \mid \text{l'elettore } n \text{ è un pivot}\big\}.$$

(a) Dimostra che *PIVOT* è un problema NP.

(b) Sappiamo che il linguaggio *SET-PARTITION* $= \{\langle S\rangle \mid S$ insieme di naturali, ed esistono $S_1, S_2 \subseteq S$ tali che $S_1 \cup S_2 = S, S_1 \cap S_2 = \emptyset, \sum_{x \in S_1} x = \sum_{y \in S_2} y\}$ è NP-completo. Dimostra che *PIVOT* è NP-hard, usando *SET-PARTITION* come problema NP-hard di riferimento.

$7 > \frac{1}{2}\, 13$

$\underline{6.5}$

Esempio: Supponiamo di avere 5 elettori, con pesi 4, 3, 3, 2, 1. La somma totale dei pesi è 13, quindi la maggioranza si ottiene con 7 voti. Il quinto elettore, con peso 1, è un pivot in coalizione con gli elettori di peso 4 e 2. La coalizione perde senza l'elettore pivot ma vince con lui.

NON SO GLI ELEMENTI

$S_1 = \{4, 3, 2, 2, 1\}$ $\Big\} \; \underline{S} = \underline{20}$

$S_2 = \{3, 5, 7, 2, 2\}$

NON SO

SEMPRE OTTENIBILE

$S_1 = 10$ e $S_2 = 10$

$7 = S$

$2, 2, 3 \; S_1$ $\Big\} \; 7$

$4, 2, 1 \; S_2 \rightarrow$ PESO MAX

4. (9 punti) Supponiamo di avere un sistema elettorale composto da n elettori, dove ogni elettore i ha un "peso" $W[i]$, corrispondente al numero di voti che rappresenta. Nel caso di una votazione a maggioranza semplice, una coalizione di elettori ha bisogno di un numero di voti strettamente superiore alla metà della somma dei pesi per vincere. L'elettore n è detto *pivot* se esiste una situazione in cui il voto dell'elettore "conta", ossia dove l'aggiunta dell'elettore n ai voti "sì" rende il "sì" la maggioranza, ma l'aggiunta ai voti "no" rende il "no" maggioranza. Formalmente, l'elettore n è un pivot se esiste una coalizione di elettori $C \subseteq \{1, \ldots, n-1\}$ tale che

$$\sum_{j \in C} W[j] < \frac{1}{2} \sum_{j=1}^{n} W[j] \qquad \text{(C senza i voti di n è minoranza)}$$

$$\sum_{j \in C} W[j] + W[n] > \frac{1}{2} \sum_{j=1}^{n} W[j] \qquad \text{(C con i voti di n è maggioranza)}$$

e possiamo rappresentare il problema dell'elettore pivot con il linguaggio

$$PIVOT = \big\{\langle n, W\rangle \mid \text{l'elettore } n \text{ è un pivot}\big\}.$$

(a) Dimostra che *PIVOT* è un problema NP.

(b) Sappiamo che il linguaggio *SET-PARTITION* $= \{\langle S\rangle \mid S$ insieme di naturali, ed esistono $S_1, S_2 \subseteq S$ tali che $S_1 \cup S_2 = S, S_1 \cap S_2 = \emptyset, \sum_{x \in S_1} x = \sum_{y \in S_2} y\}$ è NP-completo. Dimostra che *PIVOT* è NP-hard, usando *SET-PARTITION* come problema NP-hard di riferimento.

Esempio: Supponiamo di avere 5 elettori, con pesi 4, 3, 3, 2, 1. La somma totale dei pesi è 13, quindi la maggioranza si ottiene con 7 voti. Il quinto elettore, con peso 1, è un pivot in coalizione con gli elettori di peso 4 e 2. La coalizione perde senza l'elettore pivot ma vince con lui.

(a) NP $\rightarrow$ PIVOT

$C \subseteq \{1, \ldots, n-1\}$

$W[i]$ = PESO

$\exists$ UN VERIFICATORE V TALE CHE:

- $C[1] \rightarrow W[1]$

...

$C[n] \rightarrow W[n]$

(MAPPI $\forall$ ELEMENTO UN PESO) $\forall\, C[i] \mapsto W[i]$

$- \exists\ W[n] \rightarrow$ PUO DIVIDERE IN 2 METÀ TALI CHE

$$\sum W[J] < \frac{1}{2} W[J]$$

```
Verifica(⟨n, W⟩, C):
1. Calcola S = Σⱼ∈C W[j]
2. Calcola T = Σⁿⱼ₌₁ W[j]
3. Verifica che S < T/2 < S + W[n]
4. Return true se entrambe le condizioni sono soddisfatte
```

CALCOLARE

T = PIVOT

SPEZZA IN DUE METÀ

TRUE / VERO

$$W[J] + W[n] > \frac{1}{2} W[J]$$

4. **(9 punti)** Supponiamo di avere un sistema elettorale composto da n elettori, dove ogni elettore i ha un "peso" $W[i]$, corrispondente al numero di voti che rappresenta. Nel caso di una votazione a maggioranza semplice, una coalizione di elettori ha bisogno di un numero di voti strettamente superiore alla metà della somma dei pesi per vincere. L'elettore n è detto *pivot* se esiste una situazione in cui il voto dell'elettore "conta", ossia dove l'aggiunta dell'elettore n ai voti "sì" rende il "sì" la maggioranza, ma l'aggiunta ai voti "no" rende il "no" maggioranza. Formalmente, l'elettore n è un pivot se esiste una coalizione di elettori $C \subseteq \{1, \dots, n-1\}$ tale che

$$\sum_{j \in C} W[j] < \frac{1}{2} \sum_{j=1}^{n} W[j] \qquad (C \text{ senza i voti di } n \text{ è minoranza})$$

$$\sum_{j \in C} W[j] + W[n] > \frac{1}{2} \sum_{j=1}^{n} W[j] \qquad (C \text{ con i voti di } n \text{ è maggioranza})$$

e possiamo rappresentare il problema dell'elettore pivot con il linguaggio

$$PIVOT = \{\langle n, W \rangle \mid \text{l'elettore } n \text{ è un pivot}\}.$$

(a) Dimostra che *PIVOT* è un problema NP.

(b) Sappiamo che il linguaggio $SET\text{-}PARTITION = \{\langle S \rangle \mid S$ insieme di naturali, ed esistono $S_1, S_2 \subseteq S$ tali che $S_1 \cup S_2 = S, S_1 \cap S_2 = \emptyset, \sum_{x \in S_1} x = \sum_{y \in S_2} y\}$ è NP-completo. Dimostra che *PIVOT* è NP-hard, usando *SET-PARTITION* come problema NP-hard di riferimento.

Esempio: Supponiamo di avere 5 elettori, con pesi 4, 3, 3, 2, 1. La somma totale dei pesi è 13, quindi la maggioranza si ottiene con 7 voti. Il quinto elettore, con peso 1, è un pivot in coalizione con gli elettori di peso 4 e 2. La coalizione perde senza l'elettore pivot ma vince con lui.

$SP \leq_m PIVOT$

$(A \leq_m B)$

$(\Rightarrow)$ SE $\exists$ UN' ISTANZA DI SP

$S_1, S_2 \subseteq S,\ S_1 \cup S_2 = S$

$S_1 \cap S_2 = \emptyset$ (STESSO N. DI ELEMENTI)

$$\sum_{S_1} x = \sum_{S_2} y = S$$

$\downarrow$

$$C \mid \begin{cases} \sum_W < \frac{1}{2} W \\ \sum_W + \underbrace{W[n]}_{\text{PIVOT}} > \frac{1}{2} W \end{cases}$$

$S_1 \cup S_2 = C \longrightarrow$ PIVOT!

$\sum_{S_1} x \to \frac{C}{2} = \{1 \dots < \frac{n}{2}\} \to$ MINORANZA

$\exists$? $W[n]$ = PIVOT

$\sum_{S_2} y \to \frac{C}{2} = \{1 \dots \geq \frac{n}{2}\} \to$ MAGGIORANZA

SE $\exists$ PIVOT $\to$ $\underline{S_1, S_2 = S}$

$S_1 \Rightarrow \sum_j W[j] < \frac{1}{2} \sum W[j]$

(MINORANZA)

$S_2 \Rightarrow \sum_j W[j] + W[n] > \frac{1}{2} \sum W[j]$

(MAGGIORANZA $\Rightarrow$ PIVOT)

Riduzione: SET-PARTITION $\leq_p$ PIVOT

Dato: Un'istanza ⟨S⟩ di SET-PARTITION dove $S = \{s_1, s_2, \dots, s_k\}$ e $\underline{T} = \sum_i s_i$.

TARGET = QUORUM (TUTTI + PIVOT)

Costruzione:

$n = k + 1$

$W[i] = s_i$ per $i = 1, \dots, k$

$W[n] = W[k+1] = T/2$ (assumiamo T pari; se dispari, SET-PARTITION ha risposta NO)

Correttezza:

⇒: Se SET-PARTITION ha risposta YES, esistono S_1, S_2 con $S_1 \cup S_2 = S$, $S_1 \cap S_2 = \emptyset$, $\sum_{x \in S_1} x = \sum_{y \in S_2} y = T/2$.

Sia $C = \{i \mid s_i \in S_1\}$. Allora:

$\sum_{j \in C} W[j] = T/2$

$\sum_{j=1}^{n} W[j] = T + T/2 = 3T/2$

$T/2 < 3T/4$ e $T/2 + T/2 = T > 3T/4$

Quindi l'elettore n è pivot.

NP-COMPLETO

($\exists$ PIVOT) $\to$ $\underline{\exists\, W[n]}$ PIVOT

⇐: Se l'elettore n è pivot, esiste C tale che:

$\sum_{j \in C} W[j] < 3T/4$

$\sum_{j \in C} W[j] + T/2 > 3T/4$

MINORANZA S_1 S_2 MAGGIORANZA

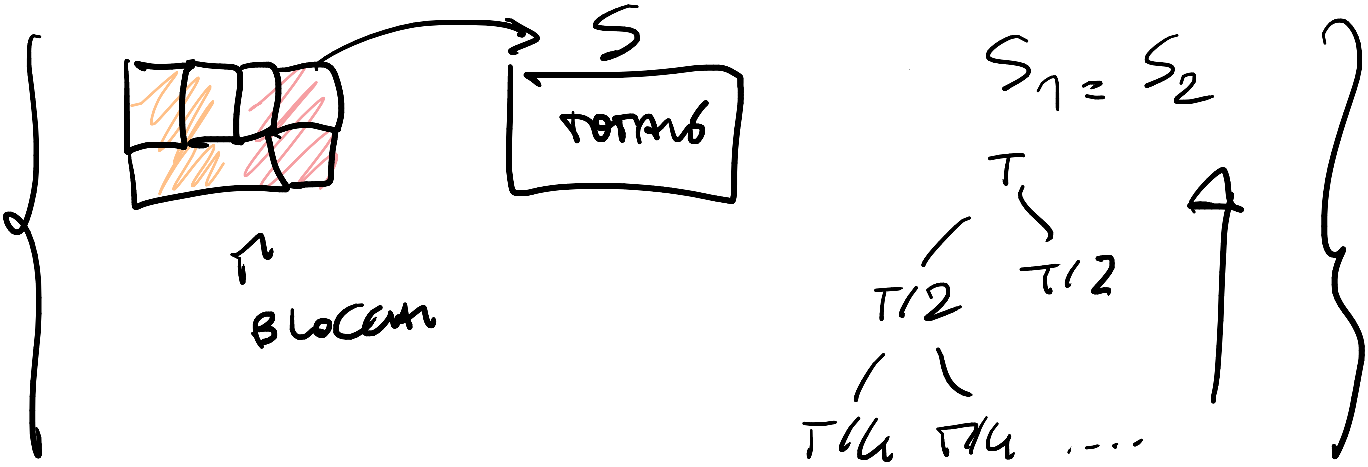
Dalla seconda: $\sum_{j \in C} W[j] > T/4$

Dalla prima: $\sum_{j \in C} W[j] < 3T/4$

Ma $\sum_{j \in C} W[j]$ è somma di elementi di S, e l'unico modo per avere una somma tra T/4 e 3T/4 che sia anche < 3T/4 è che sia esattamente T/2.

Quindi $\{s_i \mid i \in C\}$ è una partizione di S con somma T/2.

$S = C$



SPAZIO / ASSEGNAMENTO

1. **(12 punti)** Se L è un linguaggio sull'alfabeto $\{0,1\}$, la *rotazione a destra* di L è l'insieme delle stringhe

$$\mathrm{ROR}(L) = \{aw \mid wa \in L, w \in \{0,1\}^*, a \in \{0,1\}\}.$$

Per esempio, se $L = \{0, \underline{001}, 10010\}$, allora $\mathrm{ROR}(L) = \{0, 100, 01001\}$. Dimostra che se L è regolare allora anche $\mathrm{ROR}(L)$ è regolare.

001

$aw \mid wa$

$\rightarrow$ REGOLARE

100 $\rightarrow$ DESTRA

10010

01001

$\boxed{\text{DIM.}}$ SE L è REGOLARE $\rightarrow$ $\exists$ DFA D CHE LO RICONOSCE

$Q' = Q$

$\Sigma' = \Sigma$

$F' = F$

q_0 DI D = q_f DI D' (VICEVERSA)

$\delta' = \begin{cases} \delta_i : (\delta_F, a) = (a, \delta_F) \end{cases}$

INVERTE $\forall$ SIMBOLO $\rightarrow$

DFA $-\delta_F$ ALTRI TUTTI

L'ULTIMO DIVENTA IL PRIMO

δ TUTTI GLI STATI DOPO QUELLO INIZIALE VANNO AVANTI DI UNA TRANSIZIONE

$(\delta', (q,a)) = \delta'(q,b)$

$\rightarrow$ INVERTE SEGUENDO LA NOSTRA LOGICA

..... $\rightarrow$

$D \Rightarrow (Q, \Sigma, \delta, q_0, F)$

δ'((q,a), b) = {(δ(q,b), a)} per ogni q ∈ Q, a,b ∈ {0,1}