Corso: INFORMATICA TEORICA	**A. A.** 2024/25
Data esame: 27/05/2025	**Durata esame:** 3h 30min

Si ricorda che:
- Nome completo e matricola devono essere riportati su *tutti* i fogli che sono stati consegnati
- La grafia deve essere *facilmente* leggibile (testo in una grafia non facilmente leggibile sarà *ignorato*)
- Usare la *penna* per rispondere agli esercizi e alle domande (ciò che è scritto a matita sarà *ignorato*)
- Il numero dell'esercizio/domanda deve essere sempre riportato *prima* della risposta
- Consultare appunti, libri, slides, etc., *non* è consentito
- L'uso di computer, cellulari, tablet, etc., *non* è consentito
- *Questo è un test individuale; collaborazioni/interazioni con altri studenti non sono ammesse*

Domanda 1. [5 Punti]
Si definisca una macchina di Turing, eventualmente non-deterministica e/o multinastro, per il seguente linguaggio:

$$L = \{X^n \# A \# B \# C \mid n > 0 \wedge A, B, C \in (a|b|c|d)^+ \wedge \exists s \in (a|b|c|d)^+ \text{ t.c. } \|s\| = n \wedge s \subseteq A \wedge s \subseteq B \wedge s \not\subseteq C\}.$$

Ad esempio, la stringa $XXXXX\#d\underline{cabbc}b\#\underline{cabbc}dcbac\#dcabbdcad$, in cui $n = 5$ e $s = cabbc$, appartiene al linguaggio.

Domanda 2. [6 Punti]
(a) Com'è definita una riduzione?
(b) Intuitivamente, cosa sono i linguaggi NP-hard? Formalmente, come si definiscono?
(c) Cosa sono i problemi di decisione e di ricerca? Che relazione può esserci tra loro? Fornire un esempio.

Domanda 3. [4 Punti]
Dimostrare che P = NP se e solo se esiste un linguaggio L NP-complete tale che $L \in$ P.

Domanda 4. [4 Punti]
Dimostrare che $L_d \notin$ RE.

Domanda 5. [5 Punti]
Discutere la decidibilità del seguente linguaggio (in particolare: $L \in$ R? $L \in$ RE? $L \in$ RE $\setminus$ R? $L \notin$ RE?):

$L = \{\langle M, q\rangle \mid M$ è una Turing machine e q è uno stato di M,
e ad M capita di transire nell stato q per un qualche input $w\}$.

Domanda 6. [8 Punti]
Considerare il seguente problema LUDOTECA.

In una ludoteca si vuole decidere in quale stanze far giocare i bambini iscritti, in modo tale che tra di loro non ci siano "contese" per i giochi. In tutte le stanze è presente un esemplare di tutti i giochi di cui la ludoteca dispone. Ad esempio, se nella ludoteca si può giocare con un sonaglino, allora ogni stanza avrà un sonaglino. Però, i responsabili della ludoteca hanno osservato questo. Quando a due bambini nella stessa stanza piacciono giochi in comune, anche se c'è il modo affinché ognuno dei bimbi possa giocare con un gioco di proprio gradimento, i bimbi finiscono a contendersi i giochi. Ad esempio, se a due bimbi nella stessa stanza piacciono al primo una macchinina e un cubotto di peluche, e all'altro un trenino e il cubotto di peluche, sebbene possano giocare l'uno con la macchinina e l'altro con il trenino, finiranno per contendersi il cubutto di peluche. L'obiettivo dei responsabili della ludoteca è quello di riuscire a far giocare tutti i bambini iscritti nelle stanze di cui dispone la ludoteca, senza che ci possano essere conflitti sui giochi fra i pargoli.

Un'istanza $\mathcal{I} = \langle B, S, G, p\rangle$ di LUDOTECA è una quadrupla in cui B è l'insieme dei bambini iscritti alla ludoteca, S è l'insieme delle stanze della ludoteca, G è l'insieme dei giochi disponibili nella ludoteca, e p è una funzione $p\colon B \to 2^G$ che assegna ad ogni bambino i giochi che gli piacciono; la funzione p è reppresentata come un insieme di liste ℓ_b, una per ogni bambino $b \in B$, che elencano quali sono i giochi che piacciono a b. Un'istanza $\mathcal{I}$ di LUDOTECA è "sì" se e solo se è possibile assegnare *tutti* i bambini iscritti alla ludoteca alle stanze disponibili senza che ci siano conflitti sui giochi.

- Discutere la complessità del problema LUDOTECA (Suggerimento: riduzione da k-Colorabilità).
- Discutere la complessità del problema MIN-STANZE di *calcolare* il numero minimo di stanze della ludeoteca che è possibile utilizzare per sistemare i bimbi, evitando comunque contese sui giochi.

Possibile soluzione

Domanda 2

(a) Una riduzione da un linguaggio A a un linguaggio B è una funzione $f\colon \Sigma^* \to \Sigma^*$ che trasforma sringhe in stringhe tale che f è calcolabile e, per ogni stringa w, si ha che $w \in A \Leftrightarrow f(w) \in B$. Per f calcolabile si intende che esiste un trasduttore, cioè un macchina di Turing che produce un output, che per ogni stringa w in input lascia sul nastro di output in tempo finito, al termine della propria computazione su w, la stringa $f(w)$.

(b) Intuitivamente, i liguaggi NP-hard sono i linguaggi almeno difficili quanto tutti i linguaggi della classe NP. Più formalmente, un linguaggio L è NP-hard se per ogni linguaggio L' di NP esiste una riduzione polinomiale da L' a L. Con riduzione polinomiale si intende che la funzione di trsformazione della riduzione è calcolata in tempo polinomiale dal trasduttore che la valuta.

(c) I problemi di decisione sono quei problemi che per ogni istanza hanno associato come possibile risposta un Booleano, cioè o "sì" o "no". I problemi di ricerca sono quei problemi che per ogni istanza possono avere associata come risposta una stringa generica.

I problemi di ricerca ed i problemi di decisione sono collegati, poiché da un problema di ricerca possiamo sempre ricavarne una versione decisionale, al fine di semplificare l'analisi della complessità del problema. Ad esempio, si considerino i problemi Compute-HC, di calcolare per un grafo un ciclo Hamiltoniano (se esiste); ed il problema di decisione associato HC che chiede di decidere se in un grafo esiste un ciclo Hamiltoniano. La complessità di questi due problemi è legata. Infatti sapere che HC è un problema difficile implica che anche Compute-HC è un problema difficile. In quanto, se così non fosse, cioè se Compute-HC fosse risolvibile in poco tempo, allora anche HC lo sarebbe, in quanto sarebbe possibile rispondere ad HC lanciando prima un algoritmo per Compute-HC e poi rispondere "sì" o "no" in dipendenza del risultato di questo primo calcolo.

Domanda 3

($\Rightarrow$) Dimostriamo che se P = NP, allora esiste un linguaggio L NP-completo in P.

Considieriamo il linguaggio SAT, che sappiamo essere NP-completo. Dalla NP-completezza di SAT abbiamo che SAT appartiene a NP. Poiché P = NP per assunzione, allora abbiamo che SAT è in P.

($\Leftarrow$) Dimostriamo che se esiste un linguaggio L NP-completo che appartiene a P, allora P = NP.

Per dimostrare che P = NP, serve dimostrare che P $\subseteq$ NP e che NP $\subseteq$ P.

Che P $\subseteq$ NP lo si ha per la semplice definizione delle due classi.

Concentriamoci sul mostrare che NP $\subseteq$ P.

Sia L un linguaggio NP-completo che appartiene a P. Poiché $L \in$ P per assunzione, esiste una MdT *deterministica* N che in tempo polinomiale, diciamo $O(n^d)$, decide L.

Sia L' un generico linguaggio di NP. Poiché L è NP-completo, esiste una riduzione polinomiale f da L' a L. Per questa ragione, per ogni stringa w, abbiamo che $w \in L' \Leftrightarrow f(w) \in L$. Assumiamo che il calcolo di f avvenga in tempo $O(n^c)$ deterministico.

Consieriamo la macchina M che si comporta così: alla ricezione dell'input w, la macchina M esegue su w la trasformazione f per ottenere la stringa $w' = f(w)$. Fatto ciò, M si comporta come N sull'input w'. La macchina M così definita decide il linguaggio L'.

Notare che M è una macchina deterministica, perché f si calcola deterministicamente e N l'abbiamo assunta essere deterministica. Inoltre, il tempo di esecuzione di M su w è dato dal tempo di calcolo di f su w, più il tempo di esecuzione di N su w'. Il tempo di calcolo di $f(w)$ è $O(n^c)$ e quindi $O(\|w\|^c)$. Notare che durante il calcolo di $w' = f(w)$ non è possibile scrivere su nastro più simboli del numero di passi che la macchina impiega per calcolare il risultato. Quindi $\|w'\|$ è $O(\|w\|^c)$. Il tempo per eseguire N su w' è $O(n^d)$ e quindi $O(\|w'\|^d)$. Poiché la taglia di $\|w'\|$ è $O(\|w\|^c)$, abbiamo che N esegue in tempo $O((\|w\|^c)^d) = O(\|w\|^{c \cdot d})$. Quindi il tempo di esecuzione di M è $O(\|w\|^c + \|w\|^{c \cdot d})$ e quindi $O(\|w\|^{c \cdot d})$. Siccome c e d sono fissati, il tempo di esecuzione di M su w è polinomiale.

Da ciò segue che M decide in tempo polinomiale deterministico il linguaggio L', da cui $L' \in$ P. Siccome L' è un linguaggio generico, e questo discorso può essere fatto per un qualsiasi linguaggio di NP, allora tutti i linguaggi di NP sono in P, e quindi NP $\subseteq$ P. Da ciò segue che P = NP.

Domanda 4

Ogni macchina di Turing può essere codificata in una stringa binaria a partire dalla sua funzione di transione. Poiché le stringhe binarie possono essere enumerate, sia le stringhe binarie che le (codifiche di) macchine di Turing possono essere enumerate.

Il linguaggio L_d è l'insieme delle (codifiche di) MdT che non accettano in input la stringa della propria codifica, cioè $L_d = \{M_i \mid M_i \not\models w_i\}$, dove M_i e w_i sono l'i-ma macchina e stringa, rispettivamente.

Dimostriamo che $L_d \notin RE$.

Poiché tutte le MdT e le stringhe possono essere enumerate, possiamo costruire una tabella infinita in cui listiamo le MdT sulle righe e le stringhe sulle colonne. Nella cella (i, j) della tabella inseriamo 1 se la macchina i-ma accetta la stringa j-ma, altrimenti inseriamo 0. Una riga della tabella è una sequenza di simboli binari che indica quali stringhe siano accettate e quali no dalla macchina di quella riga. Tale sequenza prende il nome di vettore caratteristico del linguaggio (accettato dalla macchina).

Concentriamoci sulla diagonale della tabella, e prendiamo la sequenza D di simboli binari in essa contenuti; in particolare, l'i-mo simbolo di D è il simbolo nella cella (i, i) della tabella.

Prendiamo la sequenza D la complementiamo per ottenere $\overline{D}$. La sequenza $\overline{D}$ è una sequenza binaria, e quindi può essere interpretata come un vettore caratteristico di un linguaggio. Poiché in $\overline{D}$ l'i-mo simbolo vale 1 se e solo se $M_i \not\models w_i$, abbiamo che $\overline{D}$ è proprio il vettore caratteristico di L_d.

Quindi, poiché nella tabella sono elencate tutte le MdT con i loro vettori caratteristici, se mai ci fosse una MdT che accetta L_d questa dovrebbe apparire in una riga della tabella. Però, per definizione di $\overline{D}$, il vettore caratteristico della riga i-ma differisce in posizione i-ma rispetto al simbolo in $\overline{D}$. Da ciò, $\overline{D}$ non è il vettore caratteristico di nessuna delle macchine in tabella. E poiché le macchine in tabella sono tutte, vuole dire che non c'è alcuna macchina che accetti L_d.

Domanda 5

Il linguaggio L *non* è una proprietà di macchine, perché le proprietà di macchine sono insiemi di (codifiche di) macchine; invece L è costituito da coppie macchina-stato. Quindi, il teorema di Rice non si può applicare.

Dimostriamo che $L \in RE \setminus R$, cioè L appartiene ad RE ma non a R.

Dimostriamo prima che L appartiene ad RE. Per far ciò mostriamo che esista una MdT che accetta L (cioè è in grado di rispondere affermativamente in tempo finito sulle istanze "sì" di L).

Sia $\langle M, q\rangle$ un'istanza per L. Una MdT $\tilde{M}$ può verificare che M passi per lo stato q su una qualche stringa w in questo modo. All'inizio, $\tilde{M}$ guessa una stringa w. Quindi $\tilde{M}$ si comporta come la macchina universale e simula M su w. Durante la simulazione, $\tilde{M}$ controlla costantemente se M passi dallo stato q. Se questo accade, $\tilde{M}$ interrompe la simulazione e risponde "sì".

Dimostriamo ora che L non appertiene a R, e lo facciamo tramite una riduzione da L_{ne}.

Le istanze di L_{ne} sono macchine M, mentre le istanze di L sono coppie $\langle N, q\rangle$. Mostriamo come trasformare M in $\langle N, q\rangle$. Possiamo assumere senza perdita di generalità che M abbia un solo stato accettante q_{ACC} e che M passi per q_{ACC} solamente per accettare arrestandosi. Le riduzione genera $\langle N, q\rangle$ partendo da M così:

- $N := M$;
- $q := q_{ACC}$.

Dimostriamo che la riduzione sia corretta.

($\Rightarrow$) Assumiamo che M sia un'istanza "sì" di L_{ne}. Quindi esiste una stringa w tale che M accetta w. Se M accetta w, allora significa che M processando w passa per q_{ACC} e si arresta. Poiché N è la stessa di M, allora anche N passa per q che è q_{ACC}. Quindi $\langle N, q\rangle$ è un'istanza "sì" di L.

($\Leftarrow$) Assumiamo che M sia un'istanza "no" di L_{ne}. Da ciò deriva che *non* esiste alcuna stringa w accettata da M. Quindi, per assunzione sul funzionamento di M, M non passa mai per q_{ACC} su nessuna stringa. Poiché N è la stessa di M, allora anche per N non esiste alcuna stringa su cui N passi per q che eguaglia q_{ACC}. Quindi $\langle N, q\rangle$ è un'istanza "no" di L.

Domanda 6

Dimostriamo che LUDOTECA è un problema NP-completo. Per far questo dimostriamo prima che LUDOTECA sia in NP e poi che sia NP-hard.

LUDOTECA è in NP perché una MdT nondeterministica può deciderlo in tempo polinomiale in questo modo. Prima, guessiamo l'assegnamento dei bambini alle stanze (fattibile in tempo polinomiale). Poi verifichiamo che il guess assegna un bambino ad esattamente una stanza, e che ogni bambino sia assegnato ad almeno una stanza (fattibile in tempo polinomiale). Quindi verifichiamo per ogni stanza che non ci siano conflitti sui giochi; in

particolare, per ogni coppia di bambini assegnati ad una stessa stanza dal guess calcoliamo l'intersezione delle loro preferenze e verifichiamo che sia vuota. Il tutto è fattibile in tempo polinomiale.

Dimostriamo ora che LUDOTECA è NP-hard tramite una riduzione da k-COL.

Le istanze di k-Col sono coppie $\langle H, k\rangle$, dove $H = \langle V, E\rangle$ è un grafo non-orientato, e k è un intero positivo; $\langle H, k\rangle$ è un'istanza "sì" per k-COL se e solo se esiste un modo di colorare i nodi di H senza usare più di k colori distinti e perogni coppia di nodi collegata da un arco questi abbiano assegnati colori differenti.

Mostriamo come trasformare istanze $\langle H, k\rangle$ di k-COL in istanze $\langle B, S, G, p\rangle$ di LUDOTECA:

- Per ogni nodo $v_i \in V$ generiamo un bambino b_i; più precisamente, $B := \{b_i \mid v_i \in V\}$;
- Generiamo tante stanze quanti sono i colori; più precisamente, $S := \{S_1, \ldots, S_k\}$;
- Generiamo un gioco per ogni arco di E; più precisamente, $G := \{g_{i,j} \mid i < j \wedge (v_i, v_j) \in E\}$;[1]
- Le preferenze che i bambini hanno sui giochi si ottengono dall'incidenza degli archi sui nodi, ad esempio se l'arco e incide sul nodo v_i, allora il bambino b_i gradisce il gioco g_e; più precisamente, per ogni bambino $b_i \in B$, la lista delle sue preferenze è $\ell_{b_i} := \{g_{p,q} \mid i = p \vee i = q\}$.[2]

Dimostriamo ora che la riduzione sia corretta.

($\Rightarrow$) Supponiamo che $\mathcal{I} = \langle H, k\rangle$ sia un'istanza "sì" di k-COL. Questo significa che esiste una colorazione γ che assegna in modo congruo ad ogni nodo di H uno fra k colori. Mostriamo che l'istanza $\mathcal{J} = \langle B, S, G, p\rangle$ ottenuta da $\mathcal{I}$ nel modo descritto sopra sia un'istanza "sì" di LUDOTECA. Affinché $\mathcal{J}$ sia un'istanza sì di LUDOTECA, deve esistere un modo di allocare i bambini di G nelle stanze di S in modo che non ci siano conflitti sui giochi. Mostriamo che una tale allocazione esiste. Consideriamo la seguente allocazione λ_γ che alloca il bambino b_i nella stanza S_j se e solo se il nodo v_i è colorato con il colore j nella colorazione γ.

Affermiamo che λ_γ sia un allocazione congrua. Assumiamo per assurdo che ciò non sia vero, e che quindi λ_γ allochi due bambini $b_i \neq b_j$ con preferenze in comune in una stessa stanza S_c. Per il modo in cui λ_γ è stata ottenuta da γ avremmo che i nodi v_i e v_j sono colorati con lo stesso colore c in γ. Inoltre poiché b_i e b_j hanno preferenze in comune, assumiamo senza perdita di generalità che $i < j$, per il modo in cui la riduione genera i giochi deve essere per forza il gioco $g_{i,j}$. Ma il gioco $g_{i,j}$ è relativo all'arco (v_i, v_j). Da ciò i nodi v_i e v_j sono collegati, e colorati con lo stesso colore c in γ: contraddizione, perché γ deve essere una colorazione congrua. Perciò, λ_γ è un'allocazione corretta dei bambini nelle stanze, e quindi $\mathcal{J}$ è un'istanza "sì" di LUDOTECA.

($\Leftarrow$) Supponiamo che l'istanza $\mathcal{J} = \langle B, S, G, p\rangle$ ottenuta dalla riduzione sia un'istanza "sì" di LUDOTECA. Da ciò, esiste un'allocazione λ dei bambini nelle stanze in modo che non ci siano conflitti. Mostriamo ora che l'istanza $\mathcal{I} = \langle H, k\rangle$ di k-COL, da cui si è ottenuta $\mathcal{J}$ tramite la riduzione, è un'istanza "sì". Affinché $\mathcal{I}$ sia un'istanza "sì" di k-COL, deve esistere una colorazione congrua dei nodi di H. Consideriamo la colorazione γ_λ ottenuta a partire dall'allocazione λ in questo modo: se il bambino b_i è allocato alla stanza S_j, allora in γ_λ coloriamo il nodo v_i con il colore j.

Affermiamo che γ_λ sia una colorazione congrua. Assumiamo per assurdo che ciò non sia vero, e che quindi esistano due nodi $v_i \neq v_j$ di H connessi da un arco e che ricevono lo stesso colore c in γ_λ. Per come è stata costruita γ_λ a partire da λ, allora vuol dire che in λ ci sono i bambini b_i e b_j nella stessa stanza S_c che hanno la preferenza del gioco $g_{i,j}$ in comune (assumiamo senza perdita di generalità che $i < j$): contraddizione, perché λ è stata assunta essere un'allocazione congrua. Da ciò γ_λ è una colorazione congrua di H, e quindi $\mathcal{I}$ è un'istanza "sì" di k-COL.

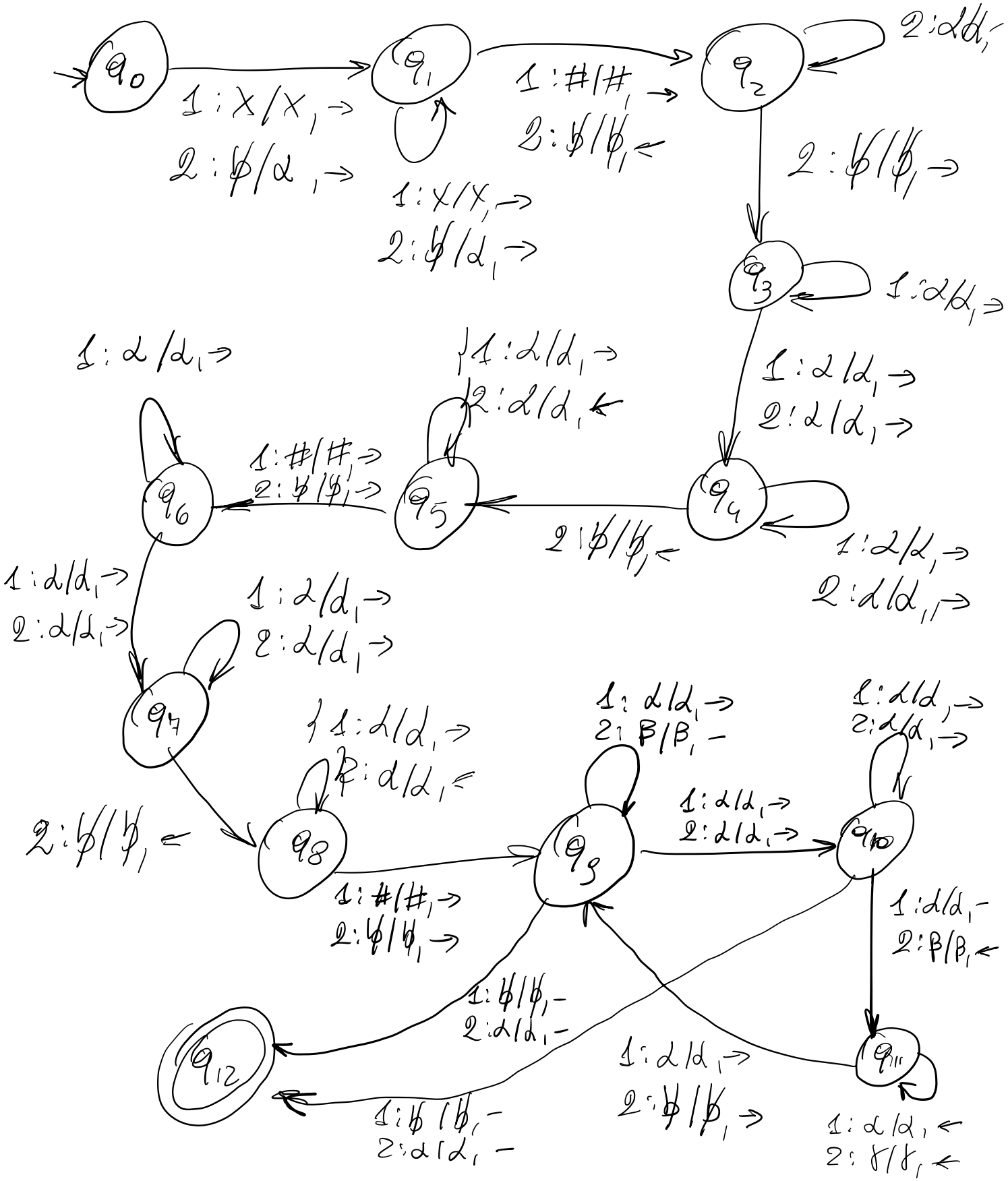
Per concludere, consideriamo la complessità del problema MIN-STANZE. Questo è un problema di ricerca. Lo possiamo risolvere tramite un trasduttore che chiama un oracolo. L'oracolo in questione è un decisore di LUDOTECA. Se vogliamo calcolare il numero minimo di stanze necessarie ad allocare i bambini senza conflitti, è sufficiente fare un ricerca binaria (deterministica) nel dominio $[1, |B|]$, e di volta in volta si chiede all'oracolo se sia possibile allocare i bambini in un certo numero di stanze. In base alla risposta dell'oracolo, facciamo la domanda successiva fino a quando questo numero minimo viene individuato.

Notare che questa ricerca binaria può essere eseguita in tempo polinomiale, e inoltre il numero di domande è logaritmico nella dimensione del dominio di ricerca. Poiché la dimensione del dominio di ricerca è polinomiale nella taglia dell'input (in quanto la lista dei bambini è rappresentata esplicitamente), il numero di domande all'oracolo è logaritmico nella taglia dell'input. Per concludere si noti che l'oracolo risponde ad una domanda NP. Per queste ragioni, possiamo affermare che il problema MIN-STANZE si colloca nella classe $\mathrm{FP}^{\mathrm{NP}[O(\log n)]}$.

[Punti totali: 32]

[1] Inserisco "$i < j$" perché, essendo H un grafo non-orientato, se c'è l'arco da v_i a v_j in H allora c'è anche l'arco da v_j a v_i in H; ma io voglio generare un solo gioco per quell'arco, non due. Ma questa è veramente una finezza.

[2] Definito così di nuovo perché H è non-orientato, e quindi i potrebbe essere l'indice più grande o più piccolo degli estremi dell'arco incidente in v_i.



$\alpha, \beta, \gamma \in \{a, b, c, d\}$

$\alpha \neq \beta$