Lezione di Informatica Teorica: Preparazione Esame

Appunti da Trascrizione Automatica

30 giugno 2025

Indice

1 Organizzazione Esame

Il voto massimo totale ottenibile dalla prova scritta è 32 punti (potrebbe passare a 31). La prova è lunga, si stima una durata di 3 ore e mezza.

1.1 Politica di Registrazione del Voto

- Se il punteggio allo scritto è ≥ 18: è possibile registrare il voto così com'è.
- Se il punteggio allo scritto è ≥ 29:
  - È possibile scegliere di **non** fare l'orale e registrare 28. (Il voto massimo senza orale è 28 per garantire flessibilità a chi ottiene voti molto alti).
  - È possibile sostenere l'orale: il voto dello scritto è perso, e il voto finale può salire o scendere. L'orale è l'unico modo per ottenere la lode.
- La decisione di accettare o rifiutare il voto dello scritto deve essere comunicata entro le date dell'orale. Il silenzio assenso implica l'accettazione del voto dello scritto.

1.2 Struttura dell'Orale

L'orale è riservato principalmente a chi ambisce ai voti più alti (≥ 29).

- Domande di teoria e dimostrazioni.
- Quesiti che collegano concetti diversi: "qual è la differenza tra X e Y?", "quale relazione c'è tra A e B?".
- Domande su argomenti non espressamente trattati a lezione, per valutare la capacità di ragionamento (es. Teorema di Cook, NP-completezza di problemi specifici come Hamilton Cycle o Tiling).

2 Struttura e Tipologia di Domande della Prova Scritta

La prova è divisa in diverse sezioni, con un totale di 32 punti.

- **Macchina di Turing (Domanda 1)**: circa 8 punti. Sarà un esercizio di complessità alta, simile a quelli visti alla fine del corso (e.g., controllo del non determinismo, manipolazione di stringhe complesse).
- **Domande Teoriche (Domanda 2)**: 3-4 domande da 2 punti ciascuna (totale 6-8 punti). Definizione formale di concetti chiave, domande sul "perché" certi aspetti della teoria esistono.
- **Dimostrazioni (Domande 3 e 4)**: 1-2 dimostrazioni su concetti visti a lezione (es. proprietà di linguaggi decidibili, transitività delle riduzioni polinomiali). Circa 2-3 punti ciascuna.
- **Esercizi Creativi (Domanda 5 e 6)**:
  - Un esercizio sull'indecidibilità (Domanda 5): dato un linguaggio, stabilirne la decidibilità/indecidibilità, eventualmente usando il Teorema di Rice o una riduzione diretta.
  - Un esercizio sull'NP-completezza (Domanda 6): dato un nuovo problema, dimostrarne l'NP-completezza tramite riduzione da un problema NP-completo noto (la sorgente della riduzione sarà suggerita).

2.1 Criteri di Valutazione per le Dimostrazioni e gli Esercizi Creativi

Sarà valutata l'abilità di argomentazione:

- **Chiarezza e Coerenza Logica**: le frasi devono essere collegate, il flusso di ragionamento deve essere guidato e comprensibile per il lettore.
- **Formalità**: riferimento corretto a oggetti e concetti definiti.
- **Correttezza**: anche se l'idea è corretta, una dimostrazione mal strutturata non otterrà il massimo dei punti.

3 Esercizi d'Esame: Esempio e Soluzioni

3.1 Domanda 1: Macchina di Turing

Si definisca una Macchina di Turing (eventualmente non-deterministica e/o multi-nastro) per il seguente linguaggio L:

$$L = \{A\#B\#W_1\#\ldots\#W_N \mid A, B, W_i \in \{a,b,c,d\}^+, N = |A|+|B|, \forall i \in [1,N] : W_i = R_i S_i R_i^R S_i, |R_i| \geq 1, |S_i| \geq 1\}$$

Esempio dato: $A = \texttt{bc}$, $B = \texttt{c}$. $|A| = 2$, $|B| = 1 \Rightarrow N = 3$. Quindi la stringa sarà della forma `bc#c#W1#W2#W3`. Un esempio di W_i è `ab c ba c`, dove $R_i = \texttt{ab}$, $S_i = \texttt{c}$. Allora $R_i^R = \texttt{ba}$. La struttura è $R_i S_i R_i^R S_i$.

Nota: Per la soluzione, qualsiasi approccio funzionante è accettato. È preferito un approccio non-deterministico se applicabile, per dimostrare la comprensione del non determinismo. Si richiede il grafo di transizione della macchina e un breve commento sul funzionamento (e.g., utilizzo dei nastri).

3.2 Domanda 2: Domande Teoriche

- **A. Perché ci focalizziamo sullo studio dei problemi di decisione? Risposta:** I problemi di decisione sono concettualmente più semplici da affrontare e da studiare formalmente. Un problema di ricerca o di ottimizzazione può spesso essere ricondotto a una serie di problemi di decisione.
- **B. Definizione formale di Big-O notation. Risposta:** Una funzione $f(n)$ è $O(g(n))$ se esistono costanti positive c e n_0 tali che $0 \leq f(n) \leq c \cdot g(n)$ per ogni $n \geq n_0$.
- **C. Cos'è un trasduttore? Risposta:** Un trasduttore è una macchina di Turing con almeno tre nastri: un nastro di input, un nastro di lavoro (work tape) e un nastro di output. Un trasduttore calcola una funzione f se, data una stringa w sul nastro di input, al termine della sua esecuzione, il nastro di output contiene $f(w)$.
- **D. Dimostrare che se un linguaggio L appartiene a R, allora il suo complemento $\bar{L}$ appartiene anch'esso a R. Risposta:** Se $L \in R$, esiste una MT M che decide L. M si ferma e accetta per $w \in L$ e si ferma e rifiuta per $w \notin L$. Per costruire una MT M' che decide $\bar{L}$, basta scambiare gli stati di accettazione e rifiuto di M. Se M accetta w, M' rifiuta w. Se M rifiuta w, M' accetta w. Poiché M si ferma sempre, anche M' si fermerà sempre. Quindi M' decide $\bar{L}$, e $\bar{L} \in R$.

3.3 Domanda 5: Decidibilità/Indecidibilità

3.3.1 Problema

Discutere la decidibilità del seguente linguaggio: L_{cambio} è l'insieme dei codici di Macchine di Turing M tali per cui M, quando computa su qualsiasi input, cambia stato ad ogni passo.

- **Passo 1: È una proprietà semantica?** Una proprietà è semantica se, date due macchine M_1 e M_2 che riconoscono lo stesso linguaggio ($L(M_1) = L(M_2)$), l'appartenenza di M_1 alla proprietà implica l'appartenenza di M_2 alla proprietà. Per dimostrare che L_{cambio} **non** è una proprietà semantica, forniamo un controesempio:

  Esempio 1 (Controesempio per proprietà semantica)**.** *Siano M_1 e M_2 due macchine di Turing.*

  - ***Macchina*** *M_1: Parte dallo stato q_0. Legge il primo simbolo α, lo riscrive α, mantiene la testina ferma (o si muove a destra/sinistra), e passa allo stato q_1. Poi si ferma. Questa macchina fa un solo passo e cambia stato ($q_0 \to q_1$). Il suo linguaggio è $\emptyset$ (non accetta nulla). $M_1 \in L_{cambio}$ (perché in ogni passo che fa, cambia stato).*
  - ***Macchina*** *M_2: Parte dallo stato q_0. Legge il primo simbolo α, lo riscrive α, mantiene la testina ferma, e rimane nello stato q_0. Non accetta mai. Questa macchina non cambia stato (rimane sempre in q_0). Il suo linguaggio è $\emptyset$ (non accetta nulla). $M_2 \notin L_{cambio}$.*

  *Poiché $L(M_1) = L(M_2) = \emptyset$, ma $M_1 \in L_{cambio}$ e $M_2 \notin L_{cambio}$, la proprietà L_{cambio} non è semantica. Di conseguenza, il Teorema di Rice **non può** essere applicato per determinarne la decidibilità.*

- **Passo 2: Collocazione in R/co-R**

  - **È** $L_{cambio} \in R$**?** Per $L_{cambio} \in R$, dovrebbe esistere una MT che per ogni input M decide se M cambia stato ad ogni passo su **tutti** gli input. Questo è difficile, perché simulare M su tutti gli input è impossibile.
  - **È** $L_{cambio} \in co-R$**?** Questo significa che il complemento $\overline{L_{cambio}}$ (macchine che **non** cambiano stato ad ogni passo, cioè esiste almeno un passo in cui rimangono nello stesso stato o almeno un input su cui si bloccano in uno stato) è in R. Per $\overline{L_{cambio}}$, possiamo costruire una macchina che risponde "sì" in tempo finito. Dato un codice M:
    - Iteriamo su tutte le possibili configurazioni raggiungibili da M (o simuliamo M su un insieme di input).
    - Per ogni configurazione, verifichiamo la transizione. Se in un qualsiasi momento M transisce da uno stato q a q (cioè $Q_{next} = Q_{current}$), allora possiamo dire "sì, questa macchina non cambia stato ad ogni passo".
    - Questo implica che $\overline{L_{cambio}} \in R$, quindi $L_{cambio} \in co-R$.

    Osservazione: Una semplice analisi sintattica della funzione di transizione (cercare loop $q \to q$) non è sufficiente, perché potrebbero esserci loop che la macchina non raggiunge mai, o transizioni che implicano un cambio di stato ma finiscono per essere bloccate in uno stato che non avanza.

- **Passo 3: Dimostrazione di indecidibilità (riduzione)** Per dimostrare che $L_{cambio} \notin R$, riduciamo un problema noto indecidibile a L_{cambio}. Useremo il linguaggio $L_u = \{\langle M, w\rangle \mid M \text{ accetta } w\}$, che è noto essere indecidibile. Costruiamo una riduzione f tale che $\langle M, w\rangle \in L_u \iff f(\langle M, w\rangle) \in L_{cambio}$. La funzione f prende $\langle M, w\rangle$ e costruisce una nuova macchina $N = f(\langle M, w\rangle)$ nel modo seguente:

- N ignora il proprio input.
- N scrive w su un nastro di lavoro.
- N simula la macchina M su w usando una versione modificata di M, che chiamiamo M'.

La macchina M' è costruita da M in modo che:

- Tutte le transizioni di M che portano ad un loop su se stesso (es. $q \xrightarrow{\alpha \to \beta, D} q$) vengono modificate. Invece di rimanere nello stato q, M' transisce a un nuovo stato q' e da q' transisce a q (creando un'oscillazione $q \leftrightarrow q'$). Questo garantisce che M' cambi sempre stato, simulando comunque il comportamento di M.
- Se M accetta w, M' entra in un ciclo infinito di oscillazione tra due stati distinti (es. $q_{acc} \leftrightarrow q'_{acc}$), garantendo un cambio di stato continuo.
- Se M rifiuta w o entra in un loop infinito (non accettando), M' entra in un unico stato di "blocco" (es. q_{loop}), dove rimane indefinitamente senza cambiare stato.

Analisi della riduzione:

- Se $\langle M, w \rangle \in L_u$ (cioè M accetta w): N (ignorando il suo input) simulerà M' su w. Poiché M' è progettata per oscillare continuamente se M accetta, N cambierà sempre stato durante la sua computazione su qualsiasi input. Quindi $N \in L_{cambio}$.
- Se $\langle M, w \rangle \notin L_u$ (cioè M non accetta w): N (ignorando il suo input) simulerà M' su w. Poiché M' è progettata per bloccarsi in un singolo stato se M non accetta, N a un certo punto non cambierà più stato durante la sua computazione. Quindi $N \notin L_{cambio}$.

La riduzione è polinomiale in tempo. Dato che L_u è indecidibile, e abbiamo una riduzione da L_u a L_{cambio}, allora L_{cambio} è indecidibile. In conclusione, L_{cambio} è un linguaggio indecidibile e appartiene a $co - R$ (ma non a R).

3.4 Domanda 6: NP-Completezza

3.4.1 Problema: Transazioni

In un sistema di basi di dati, si vuole gestire l'esecuzione di transazioni concorrenti. Si vuole accettare l'esecuzione del maggior numero possibile di transazioni, avendo cura di non accettare contemporaneamente due transazioni che chiedono di scrivere sullo stesso dato.

Definizione 1 (Istanza di Transazioni)**.** *Un'istanza I_{TD} è una quadrupla (T, D, A, K), dove:*

- $T = \{t_1, \ldots, t_m\}$ *è l'insieme delle transazioni.*
- $D = \{d_1, \ldots, d_p\}$ *è l'insieme dei dati.*
- *A è un insieme di triple (t_x, d_y, α), dove $\alpha \in \{R, W\}$, indicante che la transazione t_x accede al dato d_y in lettura (R) o in scrittura (W).*
- *K è un intero positivo.*

*Un'istanza I_{TD} è **SÌ** se e solo se è possibile accettare un insieme $X \subseteq T$ di transazioni tale che:*

- $|X| \geq K$.
- *Per ogni coppia di transazioni $t_i, t_j \in X$ ($t_i \neq t_j$), se t_i e t_j accedono allo stesso dato d_k, almeno una delle due non accede in scrittura (W). Formalmente: non esiste $d_k \in D$ tale che $(t_i, d_k, W) \in A$ e $(t_j, d_k, W) \in A$.*

3.4.2 Discutere la complessità del problema Transazioni

Suggerimento: Riduzione da Independent Set.

Passo 1: Membership in NP Per dimostrare che il problema delle Transazioni è in NP, dobbiamo mostrare che esiste una macchina non-deterministica che può verificarne una soluzione in tempo polinomiale.

1. **Guess (Fase non-deterministica):** La macchina non-deterministica "indovina" un sottoinsieme $X \subseteq T$ di transazioni. Questo guess è polinomiale poiché la cardinalità di T è finita e il numero di transazioni nel sottoinsieme non può superare $|T|$.

2. **Check (Fase deterministica polinomiale):** La macchina verifica i due vincoli:

   - **Cardinalità:** Controlla se $|X| \geq K$. Questo è un controllo in tempo polinomiale ($O(|X|)$).
   - **Conflitti di Scrittura:** Per ogni coppia di transazioni (t_i, t_j) in X:
     - Si scorre l'insieme A delle richieste.
     - Per ogni dato d_k, si verifica se sia t_i che t_j hanno una richiesta di scrittura su d_k.

     Questo controllo può essere fatto efficientemente. Per esempio, si possono creare liste di accesso per ogni dato: per ogni d_k, una lista di transazioni che scrivono su d_k. Poi si itera su ogni d_k. Se una lista contiene più di una transazione presente in X, e tutte queste transazioni scrivono su d_k, allora c'è un conflitto. Questo check è polinomiale, $O(|X|^2 \cdot |D|)$ o $O(|D| \cdot |T|^2)$ a seconda dell'implementazione.

   Poiché sia la fase di guess che quella di check sono polinomiali, il problema delle Transazioni appartiene a NP.

Passo 2: NP-Hardness (Riduzione da Independent Set) Dobbiamo mostrare una riduzione polinomiale da Independent Set (IS) al problema delle Transazioni.

Definizione 2 (Independent Set (IS))**.** *Un'istanza di IS è una coppia (G, k), dove $G = (V, E)$ è un grafo non orientato e k è un intero. L'istanza è **SÌ** se esiste un sottoinsieme $V' \subseteq V$ di nodi tale che $|V'| \geq k$ e nessun paio di nodi in V' sono collegati da un arco (cioè, V' è un insieme indipendente).*

Costruzione della Riduzione $f : (G, k) \to (T, D, A, L)$: Data un'istanza (G, k) del problema Independent Set, costruiamo un'istanza (T, D, A, L) per il problema delle Transazioni come segue:

1. **Insieme delle Transazioni T:** Per ogni nodo $v_i \in V$ del grafo G, creiamo una transazione $t_i \in T$. Quindi, $T = \{t_i \mid v_i \in V\}$. In questo modo, ogni transazione t_i corrisponde univocamente a un nodo v_i.

2. **Insieme dei Dati D:** Per ogni arco $e_j \in E$ del grafo G, creiamo un dato $d_j \in D$. Quindi, $D = \{d_j \mid e_j \in E\}$. In questo modo, ogni dato d_j corrisponde univocamente a un arco e_j.

3. **Insieme delle Richieste A:** Per ogni arco $e_j = (v_a, v_b) \in E$ in G:

   - Aggiungiamo la tripla (t_a, d_j, W) all'insieme A, indicando che la transazione t_a (corrispondente al nodo v_a) vuole scrivere sul dato d_j (corrispondente all'arco e_j).
   - Aggiungiamo la tripla (t_b, d_j, W) all'insieme A, indicando che la transazione t_b (corrispondente al nodo v_b) vuole scrivere sul dato d_j (corrispondente all'arco e_j).

   Non ci sono richieste di lettura o altre richieste di scrittura. L'obiettivo è che le transazioni corrispondenti a nodi collegati da un arco entrino in conflitto.

4. **Valore L:** Impostiamo $L = k$. Questo valore rappresenta la cardinalità minima richiesta per l'insieme indipendente, che ora si traduce nella cardinalità minima per l'insieme di transazioni accettate.

Correttezza della Riduzione: Dobbiamo dimostrare che (G, k) è un'istanza SÌ di IS se e solo se (T, D, A, L) è un'istanza SÌ di Transazioni.

Parte 1: Se (G, k) è SÌ $\implies$ (T, D, A, L) è SÌ Supponiamo che (G, k) sia un'istanza SÌ di IS. Questo significa che esiste un insieme indipendente $V' \subseteq V$ tale che $|V'| \geq k$ e per ogni $v_a, v_b \in V'$ $(v_a \neq v_b)$, $(v_a, v_b) \notin E$. Costruiamo un insieme di transazioni $X \subseteq T$ come segue: $X = \{t_i \mid v_i \in V'\}$.

- **Cardinalità:** Per costruzione, $|X| = |V'| \geq k$. Quindi il primo vincolo è soddisfatto.
- **Conflitti di Scrittura:** Supponiamo per assurdo che esistano due transazioni $t_a, t_b \in X$ $(t_a \neq t_b)$ che accedono in scrittura allo stesso dato d_j. Per costruzione della riduzione, se t_a e t_b scrivono su d_j, allora d_j deve corrispondere a un arco $e_j = (v_a, v_b) \in E$ che collega i nodi v_a e v_b (corrispondenti a t_a e t_b). Ma $v_a, v_b \in V'$, e V' è un insieme indipendente, il che implica che $(v_a, v_b) \notin E$. Questo è una contraddizione. Pertanto, non esistono conflitti di scrittura tra le transazioni in X.

Dato che entrambi i vincoli sono soddisfatti, (T, D, A, L) è un'istanza SÌ del problema delle Transazioni.

Parte 2: Se (T, D, A, L) è SÌ $\implies$ (G, k) è SÌ Supponiamo che (T, D, A, L) sia un'istanza SÌ di Transazioni. Questo significa che esiste un insieme $X \subseteq T$ di transazioni tale che $|X| \geq L$ (e $L = k$) e per ogni $t_a, t_b \in X$ $(t_a \neq t_b)$, non accedono in scrittura allo stesso dato. Costruiamo un sottoinsieme di nodi $V' \subseteq V$ come segue: $V' = \{v_i \mid t_i \in X\}$.

- **Cardinalità:** Per costruzione, $|V'| = |X| \geq L = k$. Quindi il primo vincolo è soddisfatto.
- **Assenza di Archi:** Supponiamo per assurdo che esistano due nodi $v_a, v_b \in V'$ $(v_a \neq v_b)$ collegati da un arco $(v_a, v_b) \in E$. Per costruzione della riduzione, se $(v_a, v_b) \in E$, allora esiste un dato d_j corrispondente a questo arco, e l'insieme A contiene le richieste (t_a, d_j, W) e (t_b, d_j, W). Ma $t_a, t_b \in X$, e per ipotesi X è un insieme di transazioni senza conflitti di scrittura. Questo è una contraddizione, poiché t_a e t_b entrerebbero in conflitto su d_j. Pertanto, nessun paio di nodi in V' è collegato da un arco, e V' è un insieme indipendente.

Dato che entrambi i vincoli sono soddisfatti, (G, k) è un'istanza SÌ del problema Independent Set.

Complessità della Riduzione: La costruzione di T, D, A è polinomiale. $|T| = |V|$, $|D| = |E|$. La costruzione di A richiede di iterare sugli archi di G, e per ogni arco si aggiungono due triple. Questo può essere fatto in tempo $O(|V| + |E|)$, che è polinomiale rispetto alla dimensione dell'input (G, k).

Conclusione: Poiché il problema Independent Set è NP-completo, e abbiamo dimostrato una riduzione polinomiale da Independent Set al problema delle Transazioni, allora il problema delle Transazioni è NP-Hard. Dato che è anche in NP, il problema delle Transazioni è **NP-Completo**.

3.4.3 Complessità del Problema di Ottimizzazione

Qual è la complessità di calcolare il numero più grande di transazioni che si possono eseguire contemporaneamente? Questo è un problema di ottimizzazione. Per risolverlo, possiamo utilizzare un oracolo per il problema decisionale NP-completo (Transazioni).

Algoritmo di Ricerca Binaria con Oracolo NP: Sia N_{max} la dimensione massima possibile dell'insieme di transazioni che possiamo accettare (che è al massimo $|T|$).

1. Inizializziamo un intervallo di ricerca: $min = 0$, $max = |T|$.
2. Finché $min \leq max$:
   - Calcoliamo $mid = (min + max)/2$.
   - Chiediamo all'oracolo per Transazioni se è possibile accettare *mid* transazioni (cioè, l'istanza (T, D, A, mid) è SÌ?).
   - Se l'oracolo risponde SÌ: significa che possiamo accettare almeno *mid* transazioni. Tentiamo valori più grandi: $best\_count = mid$, $min = mid + 1$.
   - Se l'oracolo risponde NO: significa che non possiamo accettare *mid* transazioni. Dobbiamo cercare valori più piccoli: $max = mid - 1$.
3. Il valore finale di *best_count* sarà il numero più grande di transazioni che si possono eseguire contemporaneamente.

Il numero di chiamate all'oracolo è $O(\log |T|)$. Poiché ogni chiamata all'oracolo risolve un problema in NP, la classe di complessità di questo problema di ottimizzazione è FP^{NP}, che denota le funzioni calcolabili in tempo polinomiale con accesso a un oracolo NP. Più precisamente, è FP^{NP}_{logn} o $FP^{NP}_{log|T|}$.