Lezione di Informatica Teorica: Nuove Definizioni di NP e NP-Completezza della Colorabilità dei Grafi

Appunti da Trascrizione Automatica

30 giugno 2025

Indice

1 Introduzione e Ripasso

Nella lezione precedente sono stati introdotti i concetti di classi di complessità P e NP, e le riduzioni polinomiali. Il professor Malizia ha ripassato la struttura delle dimostrazioni di riduzione:

- Si vuole dimostrare una riduzione da un problema A a un problema B ($A \leq_p B$).
- Si deve costruire una funzione f (algoritmo) che trasforma un'istanza w di A in un'istanza $f(w)$ di B.
- Questa funzione f deve essere calcolabile in tempo polinomiale.
- Si deve dimostrare l'equivalenza: $w \in A \iff f(w) \in B$. Questo richiede una dimostrazione in due direzioni (doppia implicazione logica).
- È fondamentale definire con precisione cosa sia un'istanza "sì" e un'istanza "no" per entrambi i problemi A e B. L'interpretazione vaga di questi termini può portare a confusioni.

2 Una Nuova Caratterizzazione della Classe NP

2.1 Critica alla Definizione Classica di NP

La definizione classica della classe NP è:

$$NP = \bigcup_{c \geq 1} NTIME(n^c)$$

dove $NTIME(n^c)$ è l'insieme dei linguaggi decidibili da Macchine di Turing non deterministiche in tempo $O(n^c)$.

La critica a questa definizione risiede nel fatto che essa si basa su un modello di calcolo, la Macchina di Turing non deterministica, che non è realizzabile. Inoltre, il non determinismo, in base a questa definizione, può essere utilizzato in qualsiasi punto della computazione, non necessariamente all'inizio. Tuttavia, negli esempi pratici di problemi in NP (es. SAT, Independent Set, ecc.), si osserva un pattern comune: si "indovina" una soluzione (un "certificato") e poi la si "verifica" deterministicamente in tempo polinomiale.

2.2 Il Concetto di Certificato

L'intuizione alla base della nuova caratterizzazione di NP è che i problemi in NP sono quelli per i quali una soluzione (se esiste) può essere verificata efficientemente. La "soluzione indovinata" viene chiamata certificato o testimone. Un certificato deve possedere due proprietà fondamentali:

- Conciso: la sua lunghezza non deve essere troppo grande rispetto all'input, tipicamente polinomiale nella lunghezza dell'input.
- Polinomialmente Verificabile: la sua correttezza può essere controllata da un algoritmo deterministico in tempo polinomiale.

2.3 Definizione di NP tramite Relazioni Binarie

Sia Σ^* l'insieme di tutte le stringhe sull'alfabeto Σ. Una relazione binaria R su Σ^* è semplicemente un sottoinsieme del prodotto cartesiano $\Sigma^* \times \Sigma^*$. Ovvero, R è un insieme di coppie (x, y) di stringhe.

Definizione 1 (Relazione Polinomialmente Bilanciata). Una relazione binaria $R \subseteq \Sigma^* \times \Sigma^*$ si dice polinomialmente bilanciata se esiste una costante $c > 0$ tale che per ogni $(x, y) \in R$, la lunghezza di y è limitata da un polinomio della lunghezza di x:

$$|y| \leq |x|^c$$

Questa proprietà assicura che il "certificato" y (la seconda componente della coppia) non sia eccessivamente lungo rispetto all'istanza x (la prima componente).

Definizione 2 (Relazione Polinomialmente Decidibile). Una relazione binaria $R \subseteq \Sigma^* \times \Sigma^*$ si dice polinomialmente decidibile se esiste una Macchina di Turing deterministica che, su input (x, y) (codificato opportunamente come una singola stringa), decide in tempo polinomiale se $(x, y) \in R$ oppure no. Questa proprietà assicura che la "verifica" del certificato sia efficiente.

Teorema 1 (Caratterizzazione di NP). Un linguaggio L appartiene alla classe NP se e solo se esiste una relazione binaria $R_L \subseteq \Sigma^* \times \Sigma^*$ che è sia polinomialmente bilanciata che polinomialmente decidibile, tale che:

$$L = \{x \in \Sigma^* \mid \exists y \in \Sigma^* \text{ tale che } (x, y) \in R_L\}$$

In altre parole, un linguaggio L è in NP se le sue istanze "sì" sono esattamente quelle x per cui esiste un "certificato" y (cioè la seconda componente della coppia (x, y) nella relazione R_L) che è conciso e verificabile deterministicamente in tempo polinomiale.

Dimostrazione. Parte 1: ($\Longleftarrow$) Se esiste R_L che è polinomialmente bilanciata e decidibile, allora $L \in NP$.

Sia $L = \{x \mid \exists y \text{ t.c. } (x, y) \in R_L\}$, dove R_L è polinomialmente bilanciata e polinomialmente decidibile. Dobbiamo mostrare che L può essere deciso da una Macchina di Turing non deterministica (NTM) M' in tempo polinomiale.

L'NTM M' per decidere L funziona come segue:

1. Su input x:
2. M' indovina non deterministicamente una stringa y. Poiché R_L è polinomialmente bilanciata, sappiamo che la lunghezza di y non eccede $|x|^c$ per una qualche costante c. Quindi, M' può "scrivere" (ovvero indovinare i bit e scriverli sul nastro) y in tempo polinomiale rispetto a $|x|$.
3. M' quindi controlla se la coppia (x, y) appartiene a R_L. Poiché R_L è polinomialmente decidibile, esiste una Macchina di Turing deterministica che esegue questo controllo in tempo polinomiale. M' simula questa MT deterministica.
4. Se il controllo restituisce "sì" (cioè $(x, y) \in R_L$), M' accetta l'input x. Altrimenti, M' rifiuta.

Poiché sia la fase di indovinamento (scrittura di y) che la fase di verifica (decisione di R_L) avvengono in tempo polinomiale, l'intera computazione di M' è in tempo polinomiale. Dunque, $L \in NP$.

Parte 2: ($\Longrightarrow$) Se $L \in NP$, allora esiste R_L che è polinomialmente bilanciata e decidibile.

Sia $L \in NP$. Per definizione, esiste una Macchina di Turing non deterministica M che decide L in tempo polinomiale. Sia $p(n)$ un polinomio tale che M decide L in tempo $p(n)$ (dove n è la lunghezza dell'input).

Definiamo la relazione R_L come segue:

$$R_L = \{(x, y) \mid y \text{ è una sequenza accettante di configurazioni di } M \text{ su input } x\}$$

In altre parole, $(x, y) \in R_L$ se x è l'input, e y è una stringa che codifica una sequenza valida di descrizioni istantanee (ID) di M che porta M ad accettare x. Una sequenza accettante di configurazioni è del tipo $ID_0 \to ID_1 \to \cdots \to ID_k$, dove ID_0 è la configurazione iniziale su x, ID_k è una configurazione accettante, e ogni ID_{i+1} è raggiungibile da ID_i con una mossa legale di M.

Dobbiamo dimostrare che R_L è polinomialmente bilanciata e polinomialmente decidibile.

- R_L è polinomialmente bilanciata: Poiché M lavora in tempo $p(n)$, il numero di passi in una computazione accettante è al massimo $p(n)$. Questo significa che la sequenza y contiene al massimo $p(n)$ descrizioni istantanee $(ID_0, \ldots, ID_{p(n)-1})$. Ogni descrizione istantanea (ID) ha una lunghezza che dipende dalla lunghezza del nastro della macchina. Poiché M esegue al più $p(n)$ passi, la sua testina può visitare al più $p(n)$ celle del nastro. Quindi, la lunghezza di ciascun ID_i è $O(p(n))$. La lunghezza totale di y (la sequenza di ID concatenati, opportunamente delimitati) sarà al più $p(n) \times O(p(n)) = O(p(n)^2)$. Poiché $p(n)$ è un polinomio, $p(n)^2$ è anch'esso un polinomio. Quindi, la lunghezza di y è polinomiale nella lunghezza di x. Dunque, R_L è polinomialmente bilanciata.

- R_L è polinomialmente decidibile: Per decidere se $(x, y) \in R_L$ con una Macchina di Turing deterministica, dobbiamo verificare la validità della sequenza di configurazioni y. Questo può essere fatto come segue:

  1. Verificare che ID_0 sia la configurazione iniziale corretta di M su input x. (Tempo polinomiale).
  2. Verificare che ID_k (l'ultima configurazione in y) sia uno stato accettante di M. (Tempo polinomiale).
  3. Per ogni i da 0 a $k-1$, verificare che ID_{i+1} sia raggiungibile da ID_i con una mossa legale di M. Questo implica controllare che la funzione di transizione di M permetta il passaggio da ID_i a ID_{i+1}. Questo controllo è una costante per ogni coppia di ID. Poiché ci sono al più $p(n)$ coppie di ID da controllare, il tempo totale per questa verifica è polinomiale.

  Tutte queste verifiche possono essere eseguite da una Macchina di Turing deterministica in tempo polinomiale. Dunque, R_L è polinomialmente decidibile.

Abbiamo dimostrato entrambi i versi del teorema. □

2.4 Implicazione della Nuova Caratterizzazione

Questo teorema fornisce una prospettiva alternativa e molto intuitiva sulla classe NP:

> I linguaggi in P sono quelli per cui una soluzione può essere calcolata in tempo polinomiale.
> I linguaggi in NP sono quelli per cui una soluzione può essere verificata in tempo polinomiale (una volta fornito un certificato).

La domanda P vs NP si riduce quindi a chiedersi se ogni problema la cui soluzione può essere facilmente verificata è anche un problema la cui soluzione può essere facilmente calcolata. La comune convinzione è che P $\neq$ NP.

3 Problema della Colorabilità dei Grafi

Il problema della colorabilità dei grafi è un altro esempio classico di problema NP-completo.

Definizione 3 (Colorazione di un Grafo). Una colorazione di un grafo $G = (V, E)$ è una funzione $c : V \to \{1, \ldots, k\}$ (dove k è il numero di colori) tale che per ogni arco $(u, v) \in E$, si ha $c(u) \neq c(v)$.

Definizione 4 (Problema della Colorabilità (GRAPH COLORING)). Il problema della Colorabilità dei Grafi prende come input una coppia (G, k), dove G è un grafo e k è un intero. La domanda è: "È possibile colorare i nodi di G con al più k colori tale che nessun nodo adiacente abbia lo stesso colore?"

3.1 Membership di GRAPH COLORING in NP

Per dimostrare che GRAPH COLORING $\in NP$:

- Certificato: Un certificato per un'istanza "sì" (G, k) è una colorazione valida $c : V \to \{1, \ldots, k\}$.
- Concisezza del Certificato: La colorazione è specificata assegnando un colore a ciascun nodo. Ci sono $|V|$ nodi, e ogni colore può essere rappresentato in $O(\log k)$ bit. La lunghezza del certificato è quindi $O(|V| \log k)$, che è polinomiale nella dimensione dell'input (che include $|V|$, $|E|$ e k).
- Verificabilità Polinomiale: Per verificare la correttezza di una data colorazione c:
  1. Controllare che tutti i colori usati siano $\leq k$.
  2. Per ogni arco $(u, v) \in E$, controllare che $c(u) \neq c(v)$.

  Entrambi i passi possono essere eseguiti in tempo polinomiale. Il secondo passo richiede di iterare su tutti gli archi, prendendo $O(|E|)$ tempo. Poiché $|E|$ è al più $O(|V|^2)$, la verifica è polinomiale.

Pertanto, GRAPH COLORING $\in NP$.

3.2 NP-Hardness di GRAPH COLORING (Riduzione da 3SAT)

Dimostriamo che GRAPH COLORING è NP-Hard riducendo 3SAT a GRAPH COLORING ($3SAT \leq_p$ *GRAPH COLORING*).

3.2.1 Definizione dei Problemi per la Riduzione

- Problema di Partenza (3SAT):
  - Input: Una formula booleana ϕ in 3-CNF (congiunzione di clausole, ogni clausola è una disgiunzione di esattamente 3 letterali).
  - Istanza "Sì": Esiste un assegnamento di verità alle variabili di ϕ che rende la formula vera (cioè ϕ è soddisfacibile).
  - Istanza "No": Nessun assegnamento di verità rende ϕ vera (cioè ϕ è insoddisfacibile).
- Problema di Arrivo (GRAPH COLORING):
  - Input: Una coppia (G, k) dove G è un grafo e k è un intero.

– Istanza "Sì": G può essere colorato con al più k colori in modo che nodi adiacenti abbiano colori diversi.
– Istanza "No": G non può essere colorato con al più k colori senza violare la condizione.

L'obiettivo è costruire una funzione polinomiale f che prende una formula ϕ in 3-CNF e produce una coppia $(G, 3)$ (il valore k sarà fissato a 3) tale che ϕ è soddisfacibile se e solo se G è 3-colorabile.

3.2.2 Costruzione del Grafo G

Sia $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ una formula con n variabili $x_1, \ldots, x_n$ e m clausole $C_1, \ldots, C_m$.

Il grafo G sarà costruito utilizzando diversi "gadget" (sottografi) che simulano la logica booleana. Il numero di colori k sarà fissato a 3. I colori saranno identificati come "Vero" (verde), "Falso" (rosso) e "Base/Neutro" (blu).

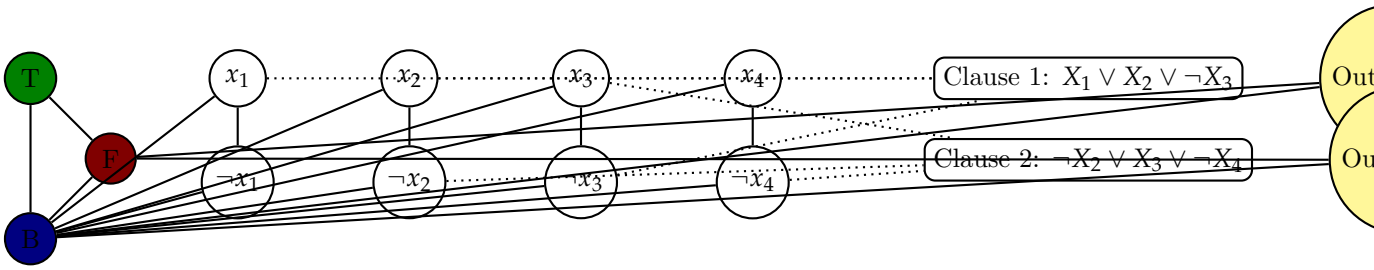

Figura 1: Struttura Generale del Grafo G per la Riduzione 3SAT $\rightarrow$ GRAPH COLORING

Il grafo G è composto da:

1. Truth Gadget: Una cricca di 3 nodi: T (True), F (False), B (Base/Neutral). Questi 3 nodi devono necessariamente avere colori distinti. Assegneremo a T il colore "Vero" (es. verde), a F il colore "Falso" (es. rosso), e a B il colore "Base" (es. blu). Questi colori serviranno da riferimento.

2. Variable Gadgets: Per ogni variabile booleana x_i nella formula ϕ, creiamo due nodi: x_i e $\neg x_i$.

   - Colleghiamo x_i e $\neg x_i$ con un arco. Questo assicura che debbano avere colori diversi.
   - Colleghiamo sia x_i che $\neg x_i$ al nodo B del Truth Gadget. Questo implica che x_i e $\neg x_i$ non possono avere il colore "Base". Quindi, possono solo assumere il colore "Vero" o "Falso".

   In questo modo, se x_i riceve il colore di T (Vero), $\neg x_i$ deve ricevere il colore di F (Falso), e viceversa. Questo simula un assegnamento di verità coerente.

3. Clause Gadgets: Per ogni clausola $C_j = (L_{j1} \vee L_{j2} \vee L_{j3})$ (dove L_{jk} è un letterale, x_i o $\neg x_i$), si costruisce un sottografo che simula la funzione logica OR.

   - Questo gadget riceve come "input" i nodi corrispondenti ai letterali L_{j1}, L_{j2}, L_{j3}.
   - Il gadget è progettato in modo che se tutti e tre i nodi input (L_{j1}, L_{j2}, L_{j3}) sono colorati con il colore "Falso" (rosso), allora un nodo "output" all'interno del gadget deve essere colorato con il colore "Falso" (rosso).
   - Se almeno uno dei nodi input è colorato con il colore "Vero" (verde), allora il nodo "output" può essere colorato con il colore "Vero" (verde).

- Il nodo "output" di ogni Clause Gadget è poi collegato ai nodi B (Base) e F (False) del Truth Gadget. Questo costringe il nodo "output" ad essere colorato con il colore "Vero" (verde), poiché non può essere "Base" (blu) né "Falso" (rosso) a causa dei suoi collegamenti.

L'intera costruzione del grafo G è polinomiale nella dimensione della formula ϕ. Il numero di nodi e archi è proporzionale al numero di variabili e clausole.

3.2.3 Dimostrazione dell'Equivalenza

Dobbiamo mostrare che ϕ è soddisfacibile $\iff$ G è 3-colorabile.

Parte 1: ($\implies$) Se ϕ è soddisfacibile, allora G è 3-colorabile.

Supponiamo che ϕ sia soddisfacibile. Sia σ un assegnamento di verità che soddisfa ϕ. Costruiamo una 3-colorazione C di G come segue:

1. Coloriamo il Truth Gadget: $C(T)$ = verde, $C(F)$ = rosso, $C(B)$ = blu. Questo è valido in quanto sono una cricca di 3 nodi e usiamo 3 colori distinti.

2. Coloriamo i Variable Gadgets: Per ogni variabile x_i:

   - Se $\sigma(x_i)$ = TRUE, allora $C(x_i)$ = verde e $C(\neg x_i)$ = rosso.
   - Se $\sigma(x_i)$ = FALSE, allora $C(x_i)$ = rosso e $C(\neg x_i)$ = verde.

   Questa colorazione è valida perché x_i e $\neg x_i$ sono collegati solo tra loro e a B. I colori "verde" e "rosso" sono distinti tra loro e distinti dal colore di B ("blu").

3. Coloriamo i Clause Gadgets: Per ogni clausola C_j: Poiché σ soddisfa ϕ, ogni clausola C_j è vera sotto σ. Ciò significa che almeno uno dei suoi letterali L_{j1}, L_{j2}, L_{j3} è vero. Per come abbiamo colorato i Variable Gadgets, questo implica che almeno uno dei nodi corrispondenti a L_{j1}, L_{j2}, L_{j3} è colorato "verde". La proprietà del Clause Gadget (OR) assicura che se almeno uno dei suoi input è "verde", il suo nodo "output" può essere colorato "verde". Siccome il nodo "output" di ogni clausola è collegato ai nodi B e F, ed è colorato "verde" (che è diverso da "blu" e "rosso"), la colorazione è valida.

Tutti i nodi sono colorati con 3 colori, e nessuna adiacenza ha lo stesso colore. Quindi, G è 3-colorabile.

Parte 2: ($\impliedby$) Se G è 3-colorabile, allora ϕ è soddisfacibile.

Supponiamo che G sia 3-colorabile. Sia C una 3-colorazione valida di G. Definiamo un assegnamento di verità σ per le variabili di ϕ basandoci sui colori in C:

1. Per ogni variabile x_i:

   - Se $C(x_i)$ è lo stesso colore di $C(T)$ (cioè "verde"), poniamo $\sigma(x_i)$ = TRUE.
   - Se $C(x_i)$ è lo stesso colore di $C(F)$ (cioè "rosso"), poniamo $\sigma(x_i)$ = FALSE.

   (Nota: Per costruzione, x_i e $\neg x_i$ sono collegati a B, quindi non possono avere il colore "blu". Essendo collegati tra loro, devono avere colori diversi, quindi se x_i è "verde", $\neg x_i$ deve essere "rosso" e viceversa, garantendo coerenza nell'assegnamento).

Dobbiamo dimostrare che questo assegnamento σ soddisfa la formula ϕ. Supponiamo per contraddizione che σ non soddisfi ϕ. Questo significa che esiste almeno una clausola C_j che è falsa sotto σ. Se $C_j = (L_{j1} \vee L_{j2} \vee L_{j3})$ è falsa sotto σ, allora tutti i suoi letterali L_{j1}, L_{j2}, L_{j3} sono falsi sotto σ. Per come abbiamo definito σ, questo implica che i nodi corrispondenti a L_{j1}, L_{j2}, L_{j3} nel grafo G sono tutti colorati con il colore di F (cioè "rosso"). Ma per la proprietà del Clause Gadget (OR): se tutti i suoi nodi input sono "rossi", allora il suo nodo "output" deve essere anch'esso "rosso". Tuttavia, il nodo "output" di C_j è collegato nel grafo ai nodi B e F. Poiché $C(F)$ è "rosso", se il nodo "output" è anch'esso "rosso", si avrebbe una violazione della colorazione (due nodi adiacenti con lo stesso colore). Questa è una contraddizione, poiché abbiamo assunto che C fosse una colorazione valida. Pertanto, la nostra supposizione iniziale (che σ non soddisfi ϕ) deve essere falsa. Dunque, σ soddisfa ϕ.

Poiché la trasformazione è polinomiale e l'equivalenza è dimostrata, $3SAT \leq_p GRAPH\ COLORING$. Dato che 3SAT è NP-completo, e GRAPH COLORING è in NP e NP-hard, concludiamo che GRAPH COLORING è NP-completo.