Esercitazione 5 - Complessità

10-05-2019

Antonio Cruciani
antonio.cruciani@alumni.uniroma2.eu

Esercizi a lezione

Esercizio 1:
Per ognuna delle seguenti affermazioni dimostrarne la veridicità o meno.
Si ricordi l'operazione di riduzione polinomiale $\leq_p$.

1) Essa gode della proprietà transitiva, ovvero siano A, B, C se $A \leq_p B$ e $B \leq_p C$ allora $A \leq_p C$

2) Essa gode della proprietà riflessiva, ovvero, sia A allora $A \leq_p A$.

3 Essa gode della proprietà commutativa, ovvero, siano A, B e $A \leq_p B$ allora $B \leq_p A$.

Esercizio 2:
Dato un grafo $G = (V, E)$, sia $\chi(G) = \langle M, P \rangle$ una sua codifica in cui M è la matrice di adiacenza di G e

$$P = \{\langle V_1, V_2, V_3 \rangle : V_1, V_2, V_3 \subseteq V \wedge V_1 \cup V_2 \cup V_3 = V \wedge V_1 \cap V_2 = \emptyset \wedge V_1 \cap V_3 = \emptyset \wedge V_2 \cap V_3 = \emptyset\}$$

Si consideri il seguente problema decisionale:
Dato un grafo non orientato $G = (V, E)$, esiste una partizione $\langle V_1, V_2, V_3 \rangle$ di V tale che per ogni $i = 1, 2, 3$ e per ogni $u, v \in V_i, (u, v) \notin E$?.

Dopo aver formalizzato la definizione del suddetto problema mediante $\langle I, S, \pi \rangle$ descrivere un algoritmo che presa in input $\chi(G)$ decide se $\langle G \rangle$ è un'istanza si del problema in tempo $\boldsymbol{Poly}(|\chi(G)|)$ (polinomiale nella dimensione della codifica).

Rispondere infine, alla seguente domanda:

"L'esistenza di tale algoritmo è sufficiente per dimostrare l'appartenenza del problema alla classe di complessità **P** ?" .

Esercizi per casa

Esercizio 1:
Siano $L \in \boldsymbol{NPC}$ ed $a \in L$.
Dimostrare che se $\boldsymbol{P} \neq \boldsymbol{NP} \Rightarrow L - \{a\} \notin \boldsymbol{P}$

Esercizio 2:
Siano L_1 e L_2 due linguaggi tali che:

1. $L_1 - L_2 = \{a_1, a_2, a_3\}$
2. $L_2 - L_1 = \{b_1, b_2\}$
3. $L_2 \in \boldsymbol{NPC}$

Dimostrare che in questa ipotesi $L_1 \in \boldsymbol{NPC}$

Soluzioni esercizi a lezione

Esercizio 1:
1)
Vero, la chiave di questa osservazione è la seguente: siano p,q due funzioni che hanno una crescita polinomiale allora la loro composizione p(q(n)) ha una crescita polinomiale.
Se f_1 è una riduzione polinomiale da A a B e f_2 è una riduzione da B a C allora il mapping $x \mapsto f_2(f_1(x))$ è una riduzione polinomiale da A a C poiché $f_2(f_1(x))$ richiede tempo polinomiale per calcolare x e $f_2(f_1(x)) \in C \iff x \in A$.

2)
Vero, sia I_A l'alfabeto utilizzato per definire tutte le istanze di A, sia T una TM che su input $x \in I_A$ scrive sul nastro di output x. Quindi se x era una istanza si di I_A allora anche $f_T(x)$ è una istanza si se x è una istanza no di I_A allora anche $f_T(x)$ è una istanza no. Allora T riduce A ad A.

3)
Falso.
Sia A un qualsiasi problema in **P** e sia B un problema **EXPC**. Poiché B è **EXPC** (EXP Completo) A è riducibile a B. Ora, se B fosse riducibile ad A avremmo quello che è, sostanzialmente, un algoritmo deterministico polinomiale per decidere B e (poiché $B \in$ **EXPC**) per decidere qualsiasi problema in **EXP** contraddicendo il teorema della gerarchia polinomiale, quindi tale algoritmo non può esistere e quindi l'operazione di riduzione polinomiale, in generale, non gode della commutatività.

Esercizio 2:
Il problema decisionale considerato, che chiameremo Γ, può essere formalizzato come segue:

$$I_\Gamma = \{\langle G = (V, E)\rangle : \text{ G È UN GRAFO NON ORIENTATO } \}$$
$$S_\Gamma(G) = \{\langle V_1, V_2, V_3\rangle : V_1, V_2, V_3 \subseteq V\}$$
$$\pi_\Gamma(G, S_\Gamma(G)) = \exists\langle V_1, V_2, V_3\rangle \in S_\Gamma(G) : V_1 \cup V_2 \cup V_3 = V \wedge V_1 \cap V_2 = \emptyset$$
$$\wedge V_1 \cap V_3 = \emptyset \wedge V_2 \cap V_3 = \emptyset \wedge \forall i = 1, 2, 3 \forall u, v \in V_i[(u, v) \notin E]$$

Osservazione: L'insieme P in $\chi(G)$ è un sottoinsieme di $S_\Gamma(G)$. In particolare il predicato π_Γ può essere espresso nel seguente modo:

$$\exists\langle V_1, V_2, V_3\rangle \in P : \forall i = 1, 2, 3 \forall u, v \in V_i[(u, v) \notin E]$$

Dopo questa breve osservazione forniamo l'algoritmo che prende in input la codifica $\chi(G)$ composta dalla matrice di adiacenza M e l'insieme P di tutte le partizioni di V e restituisce **True** se esiste una partizione di G in tre sottoinsiemi tali che inducono un'istanza si di Γ:

Algorithm 1

1: **Input:** $\chi(G)$
2: found $\leftarrow$ False
3: **while** $(P \neq \emptyset \wedge \text{found} = \text{False})$ **do Begin**
4:     Extract one element $\langle V_1, V_2, V_3 \rangle$ from P
5:     found $\leftarrow$ True
6:     **for** ($i \leftarrow 1$ ***to*** 3 ***step*** 1) **do**
7:         **for** $u \in V_i$ **do**
8:             **for** $v \in V_i$ **do**
9:                 **if** ($M[u,v] = 1$) **then**
10:                     found $\leftarrow$ False
return found

Analizziamo,ora, la complessità dell'algoritmo proposto.
L'accesso alla coordinata u, v della matrice di adiacenza M richiede tempo costante.
Per ogni $\langle V_1, V_2, V_3 \rangle \in P$ la cardinalità di V_1, V_2, V_3 è al più V, il doppio ciclo **for** alle linee 7 e 8 richiede tempo $\boldsymbol{O}(|V|^2)$.
Il numero di iterazioni del ciclo **for** alla riga 6 è costante, il numero di iterazioni del ciclo **while** è $|P|$, quindi la complessità computazionale dell'algoritmo è $\boldsymbol{O}(|P| \cdot \ |V|^2)$ è quindi è polinomiale nella dimensione dell'input, ovvero è $\mathbf{Poly}(|\chi(G)|)$.

Osserviamo esplicitamente che la codifica $\chi(G)$ non è una codifica ragionevole, in quanto $|P| = 3^{|V|}$ e quindi la codifica di G mediante la sola matrice di adiacenza(la quale codifica perfettamente tutte le informazioni necessarie ad individuare un grafo e che ha dimensione $|V|^2$) è esponenzialmente più corta di $\chi(G)$.
Possiamo, ora, rispondere all'ultima domanda:
Ricordiamo che un problema è in **P** se esiste un algoritmo deterministico che richiede tempo polinomiale nella dimensione di una <u>codifica ragionevole</u> delle sue istanze, l'algoritmo proposto non è sufficiente a dimostrate l'appartenenza a **P** del problema Γ.

Soluzioni esercizi per casa

Esercizio 1:
Assumiamo per assurdo che $L' = L - \{a\} \in \boldsymbol{P}$.
$\Rightarrow \exists T'$ (deterministica) $\wedge\, k \in \mathbb{N}[\forall y \in \Sigma^*, T'(y)$ termina in tempo $\boldsymbol{O}(|y|^k) \wedge O_{T'(y)} = q_a \iff y \in L']$. Allora, sfruttando T', possiamo definire una nuova macchina di Turing deterministica che decide L:

- Su input x

1) Se $x = a \Rightarrow$ **Accetta**

2) Altrimenti, simula $T'(x)$, se $O_{T'(x)} = q_a \Rightarrow$ **Accetta**

3) Altrimenti, **Rigetta**

Analizziamo la complessità computazionale di questa macchina di Turing T': Il passo 2) richiede tempo $\boldsymbol{O}(|x|^k)$ ovvero $\boldsymbol{Poly}(|x|) \Rightarrow L \in \boldsymbol{P}$. Ma $L \in \boldsymbol{NPC} \Rightarrow \boldsymbol{P} = \boldsymbol{NP}$ contraddicendo $\boldsymbol{P} \neq \boldsymbol{NP}$ allora, $L' \notin \boldsymbol{P}$.

Esercizio 2:
Per dimostrare che $L_1 \in \boldsymbol{NPC}$ dobbiamo:

1. Dimostrare che $L_1 \in \boldsymbol{NP}$

2. Dimostrare che L_1 è completo per **NP**

Per prima cosa, quindi, dimostriamo l'appartenenza di L_1 ad **NP**: poiché non abbiamo nessuna informazione circa la struttura di L_1, non possiamo progettare un algoritmo non deterministico per decidere L_1. Allora facciamo una riduzione polinomiale, riducendo L_1 a L_2 ($L_1 \leq_p L_2$). Assumiamo che $L_1, L_2 \subseteq \Sigma^*$ e consideriamo la seguente funzione di riduzione polinomiale:

$$\forall x \in \Sigma^*, f(x) = \begin{cases} x & \texttt{se } x \notin \{a_1, a_2, a_3, b_1, b_2\} \\ a_1 & \texttt{se } x \in \{b_1, b_2\} \\ b_1 & \texttt{se } x \in \{a_1, a_2, a_3\} \end{cases}$$

Chiaramente $f \in \boldsymbol{FP}$ poiché calcolare $f(x)$ richiede tempo costante.

Osserviamo che $x \in L_1 \iff f(x) \in L_2$, ovvero:

se $x \in L_1 \Rightarrow (x \in \{a_1, a_2, a_3\} \wedge f(x) = b_1 \in L_2) \vee (x \in L_1 - \{a_1, a_2, a_3\} \subsetneq L_2 \wedge f(x) = x \in L_2)$
se $x \notin L_1 \Rightarrow (x \in \{b_1, b_2\} \wedge f(x) = a_1 \notin L_2) \vee$
$(x \in \Sigma^* - (L_1 \cup \{b_1, b_2\}) \subsetneq \Sigma^* - L_2 \wedge f(x) = x \notin L_2)$

$\Rightarrow L_1 \leq_p L_2 \Rightarrow L_1 \in \boldsymbol{NP}$
Per dimostrare che $L_1 \in \boldsymbol{NPC}$ dobbiamo dimostrare che $L_2 \leq_p L_1$, poiché $L_1, L_2 \subseteq \Sigma^* \Rightarrow f(\cdot)$ per come è definita è anche una riduzione da L_2 a L_1.
Nella maniera analoga a quella sopra possiamo mostrare che

$$x \in L_2 \iff f(x) \in L_1 \Rightarrow L_2 \leq_p L_1 \Rightarrow L_1 \in \boldsymbol{NPC}$$

se $x \in L_2 \Rightarrow (x \in \{b_1, b_2\} \wedge f(x) = a_1 \in L_1) \vee (x \in L_2 - \{b_1, b_2\} \subsetneq L_1 \wedge f(x) = x \in L_2)$
se $x \notin L_2 \Rightarrow (x \in \{a_1, a_2, a_2\} \wedge f(x) = b_1 \notin L_1) \vee$
$(x \in \Sigma^* - (L_2 \cup \{a_1, a_2, a_3\}) \subsetneq \Sigma^* - L_1 \wedge f(x) = x \notin L_1)$

E quindi $L_2 \leq_p L_1 \Rightarrow L_1 \in \boldsymbol{NPC}$.