Complementi di Algoritmi

Massimo Perego

Indice

Capitolo 1

Complessità Computazionale

1.1 Riducibilità polinomiale tra problemi di decisione

Un **problema di decisione** X su $\{0,1\}$ è una coppia $(\mathcal{I}, q)$ dove

- $\mathcal{I} \subseteq \{0,1\}^*$ è l'insieme delle descrizioni (come stringhe binarie) delle **istanze valide del problema**
- $q : \mathcal{I} \to \{0,1\}$ è la **funzione di decisione**, che associa ogni istanza $I \in \mathcal{I}$ al suo valore, ovvero $q(I) = 1$ se positiva o $q(I) = 0$ se negativa

Un algoritmo A risolve un problema $X = (\mathcal{I}, q)$ se restituisce il valore corretto della funzione q su ogni istanza $I \in \mathcal{I}$.

Indichiamo con $|I|$ la **lunghezza della codifica** in bit di un'istanza $I \in \mathcal{I}$. In genere, non specifichiamo il modo con il quale un'istanza viene rappresentata come una stringa di bit, ma resta inteso che le istanze vengano rappresentate secondo rappresentazioni "ragionevoli" (e.g. lista di adiacenza per un grafo).

Se A è un algoritmo che risolve il problema $X = (\mathcal{I}, q)$, indichiamo con $T_A(I)$ il tempo di calcolo di A quando l'input è $I \in \mathcal{I}$. Definiamo ora

$$T_A(n) = \max \{T_A(I) \mid |I| \leq n\}$$

ovvero il massimo dei tempi di calcolo per istanze $I \in \mathcal{I}$ di lunghezza al più n. Diciamo che A risolve X in tempo polinomiale se esiste un intero k tale che $T_A(n) = \mathcal{O}\left(n^k\right)$.

Un algoritmo ha accesso a un **oracolo** per un problema $X = (\mathcal{I}, q)$ quando può ottenere in tempo unitario il valore $q(I)$ per istanze arbitrarie $I \in \mathcal{I}$.

Un problema Y è **polinomialmente riducibile** a un altro problema X, denotato con $Y \preceq_p X$ quando

1. Esiste un algoritmo che risolve Y in tempo polinomiale con accesso a un oracolo per X
2. L'oracolo viene interrogato una sola volta e l'algoritmo termina fornendo la risposta dell'oracolo

Questa nozione di riducibilità polinomiale è anche nota come riducibilità secondo Karp. Esiste anche una nozione più generale, nota come riducibilità secondo Turing, dove l'algoritmo può accedere all'oracolo un numero polinomiale di volte.

Equivalentemente, la riducibilità polinomiale (secondo Karp) può essere vista come "l'input di Y può essere trasformato in tempo polinomiale in input di X" (per poi risolvere il problema in tempo unitario grazie all'oracolo, coincide con la prima definizione).

Se $Y \preceq_p X$ ed esiste un algoritmo che risolve X in tempo polinomiale, allora esiste un algoritmo che risolve Y in tempo polinomiale (la polinomialità è robusta rispetto alla composizione). Viceversa, se $Y \preceq_p X$ e non esiste un algoritmo che risolve Y in tempo polinomiale, allora non esiste un algoritmo che risolve X in tempo polinomiale. Intuitivamente, $Y \preceq_p X$ significa che il problema Y non è *più difficile* del problema X. Se $X \preceq_p Y$ e $Y \preceq_p X$, ovvero X e Y sono polinomialmente riducibili l'uno con l'altro, allora scriviamo $X \equiv_p Y$.

Sia $G = (V, E)$ un grafo semplice (non diretto, non pesato, senza loop e archi multipli). Un **insieme indipendente** (independent set) in G è un sottoinsieme $S \subseteq V$ di vertici non adiacenti, ovvero $\forall i, j \in S$, $(i, j) \notin E$. Una **copertura** (vertex cover) di G è un sottoinsieme $S' \subseteq V$ di vertici tali che ogni arco di G ha almeno un estremo in S', ovvero $\forall (i, j) \in E$ esiste $k \in S'$ tale che $k = i$ o $k = j$.

Fatto 1.1.1. *Sia $G = (V, E)$ un grafo semplice. Allora S è un insieme indipendente se e solo se $S' = V \setminus S$ è una copertura.*

Dimostrazione. Sua S un insieme indipendente e sia $S' = V \setminus S$. Allora dato un qualsiasi $(i, j) \in E$, deve valere $i \notin S$ oppure $j \notin S$. Quindi $i \in S'$ oppure $j \in S'$, da cui ne segue che S' è una copertura di G.

Viceversa, sia S' una copertura di G e $S = V \setminus S'$. Allora dati $i, j \in S$ arbitrari $(i, j) \notin E$, altrimenti allora S' non sarebbe una copertura; da questo otteniamo che S è un independent set. □

Sia *Independent Set* il problema di decisione le cui istanze sono coppie $I = (G, k)$ dove G è un grafo semplice e k è un intero. La funzione di decisione q_{IS} è tale che $q_{IS}(I) = 1$ se e solo se G contiene un insieme indipendente di taglia almeno k. Le istanze del problema **Vertex Cover** sono le stesse di Independent Set, ma la funzione di decisione q_{VC} è tale che $q_{VC}(I) = 1$ se e solo se G contiene una copertura di taglia al più k. Conseguenza immediata del Fatto 1.1.1 è che Independent Set e Vertex Cover sono polinomialmente riducibili l'uno con l'altro.

Corollario 1.1.1. *Independent Set* $\equiv_p$ *Vertex Cover*

La relazione $\equiv_p$ gode di transitività.

Fatto 1.1.2. *Se* $Z \preceq_p Y$ *e* $Y \preceq_p X$*, allora* $Z \preceq_p X$*.*

Dimostrazione. Dato un oracolo per X possiamo risolvere un'istanza di Z nel modo seguente: eseguiamo l'algoritmo per risolvere Z usando l'oracolo per Y, ma ogni volta che dovrebbe essere invocato l'oracolo di Y, questo viene simulato eseguendo l'algoritmo per risolvere Y, il quale usa l'oracolo per X. □

Introduciamo ora il problema di decisione **Set Cover**: le istanze sono della forma $I = (U, \mathcal{S}, k)$ dove U è un insieme finito, $\mathcal{S} \equiv \{S_1, \ldots, S_m\}$ è una collezione di sottoinsiemi di U e k è un intero. Allora $q_{SC}(I) = 1$ se e solo se esistono al più k sottoinsiemi in $\mathcal{S}$ tali che la loro unione sia tutto U.

Fatto 1.1.3. *Vertex Cover* $\preceq_p$ *Set Cover*

Dimostrazione. Supponiamo di avere un oracolo per Set Cover e definiamo un algoritmo polinomiale per Vertex Cover.

Data un'istanza $I = (G, k)$ di Vertex Cover con $G = (V, E)$ costruiamo un'istanza I' di Set Cover dove $U \equiv E$ e $\mathcal{S} \equiv \{S_i \mid i \in V\}$ dove S_i è l'insieme degli archi di G incidenti su i (ottenibile in tempo lineare partendo dalla descrizione di G). Quindi ogni $u \in U$ è contenuto in esattamente due elementi di $\mathcal{S}$.

Verifichiamo ora che U è l'unione di al più k insiemi $S_1, \ldots, S_{|V|}$ se e solo se G ha una copertura con vertici di taglia al più k.

Supponiamo che l'unione di $S_{i_1}, \ldots, S_{i_r}$ sia U, con $r \leq k$. Allora, ogni arco in G è incidente a uno dei vertici $i_1, \ldots, i_r$. Quindi $\{i_1, \ldots, i_r\}$ è una copertura con vertici di taglia al più k. Viceversa, se $\{i_1, \ldots, i_r\}$ è una copertura con vertici di taglia al più k allora la gli insiemi corrispondenti $S_{i_1}, \ldots, S_{i_r}$ hanno U come unione.

Quindi possiamo definire un algoritmo che implementa la funzione di decisione q_{VC} per Vertex Cover nel modo seguente: data un'istanza di I Vertex Cover, l'algoritmo costruisce in tempo polinomiale un'istanza I' di Set Cover nel modo descritto sopra. Quindi chiama l'oracolo un'unica volta per ottenere $q_{SC}(I')$ che viene restituita in output. □

Così come possiamo vedere Set Cover come una generalizzazione di Vertex Cover, introduciamo ora Set Packing come generalizzazione di Independent Set. Le istanze di **Set Packing** sono le stesse di Set Cover, ovvero $I = (U, \mathcal{S}, k)$ dove U è un insieme finito, $\mathcal{S} \equiv \{S_1, \ldots, S_m\}$ è una collezione di sottoinsiemi di U e k è un intero. la funzione di decisione q_{SP} è tale che $q_{SP} = 1$ se e solo se esistono almeno k sottoinsiemi in $\mathcal{S}$ i quali sono a due a due disgiunti. Anche se Set Packing è apparentemente più generale di Independent Set, si può dimostrare che i due problemi sono equivalenti.

Fatto 1.1.4. *Independent Set* $\equiv_p$ *Set Packing.*

Dimostrazione. Data un'istanza $\mathcal{S} \equiv \{S_1, \ldots, S_m\}$ di Set Packing costruiamo in tempo polinomiale un grafo $G = (V, E)$ dove $V \equiv \{v_S \mid S \in \mathcal{S}\}$ e $(v_S, v_T) \in E$ se e solo se $S \cap T \not\equiv \emptyset$. Allora ogni insieme indipendente in G corrisponde a un packing della stessa taglia.

Viceversa, dato un grafo $G = (V, E)$ possiamo costruire in tempo polinomiale la collezione $\mathcal{S} \equiv \{S_i \mid i \in V\}$ dove S_i è l'insieme degli archi di G incidenti du i. Allora ogni packing in $\mathcal{S}$ corrisponde a un insieme indipendente in G della stessa taglia. □

In molte discipline bisogna spesso risolvere problemi di ottimizzazione combinatoria vincolata. In questi problemi si cerca un assegnamento di valori per un insieme di variabili discrete in modo da soddisfare un dato insieme di vincoli. In astratto, problemi di questo tipo sono formulati come problemi di soddisfacibilità su variabili booleane.

Sia $\mathcal{X}$ un insieme di variabili booleane $x_1, \ldots, x_n$. Un assegnamento di valori di verità a $\mathcal{X}$ è una funzione $\pi : \mathcal{X} \to \{0, 1\}$. Un letterale ℓ_i è la variabile x_i o la sua negazione $\bar{x}_i$. Una clausola $C = \ell_{i_1} \vee \cdots \vee \ell_{i_k}$ è una disgiunzione di letterali. Un assegnamento π soddisfa una clausola C se e solo se c'è almeno un letterale della forma $\ell_i = x_i$ e $\pi(x_i) = 1$ oppure $\ell_j = \bar{x}_j$ e $\pi(x_j) = 0$.

Sia **SAT** il problema di decisione le cui istanze I sono insiemi di clausole $\mathcal{C}$ su un insieme $\mathcal{X}$ di variabili booleane. Allora $q(I) = 1$ se e solo se esiste un assegnamento $\pi : \mathcal{X} \to \{0, 1\}$ tale che soddisfi tutte le clausole in $\mathcal{C}$.

Una versione ridotta di SAT è **3-SAT**, le cui istanze sono insiemi di clausole ciascuna contenente esattamente 3 letterali. Esempio di istanza di 3-SAT:

- $\mathcal{X} \equiv \{x_1, x_2, x_3, x_4\}$

- $\mathcal{C} \equiv \{(\bar{x}_1 \vee x_2 \vee \bar{x}_3), (\bar{x}_2 \vee x_3 \vee x_4), (x_1 \vee x_2 \vee \bar{x}_4)\}$

Un assegnamento che soddisfa tutte le clausole è $\pi(x_i) = 0$ per $i = 1, \ldots, 4$.

Teorema 1.1.1. *3-SAT $\preceq_p$ Independent Set.*

Dimostrazione. □