Algoritmi e Strutture Dati 2

Appunti ed Esercizi

Francesco Pasquale

5 ottobre 2020

Dato un insieme di variabili *booleane* $X = \{x_1, \dots, x_n\}$, una *clausola* è una disgiunzione di *letterali*, dove un *letterale* può essere una variabile x_i o una variabile negata $\overline{x_i}$. Per esempio,

$$x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4 \vee \overline{x_5} \tag{1}$$

è una clausola sulle variabili $\{x_1, x_2, x_3, x_4, x_5\}$. I letterali x_1 e x_4 si dicono *positivi*, mentre i letterali $\overline{x_2}, \overline{x_3}$ e $\overline{x_5}$, in cui le variabili compaiono negate, si dicono *negativi*.

Una *clausola di Horn* è una clausola in cui compare al più un letterale positivo. Una *formula di Horn* è una congiunzione di clausole di Horn. Per esempio,

$$(x_1) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_4) \wedge (\overline{x_3} \vee \overline{x_4} \vee \overline{x_5}) \wedge (x_5) \tag{2}$$

è una formula di Horn.

Una *assegnazione di verità* è una funzione $\sigma : X \to \{\texttt{True}, \texttt{False}\}$ che ad ogni variabile associa un valore di verità. Date una formula booleana $\mathcal{F}$ e un'assegnazione di verità σ sull'insieme di variabili X, si dice che σ *soddisfa* $\mathcal{F}$ se la formula $\mathcal{F}$ assume valore `True` quando i valori di verità delle variabili vengono assegnati secondo σ. Si dice che una formula $\mathcal{F}$ è *soddisfacibile* se esiste un'assegnazione di verità che la soddisfa. Per esempio, la seguente assegnazione di verità

x	x_1	x_2	x_3	x_4	x_5
$\sigma(x)$	`True`	`False`	`True`	`False`	`True`

non soddisfa la formula in (2) perché la seconda clausola, $\overline{x_1} \vee x_2 \vee \overline{x_3}$, non è soddisfatta. La formula in (2) è comunque soddisfacibile, infatti è soddisfatta dalla seguente assegnazione

x	x_1	x_2	x_3	x_4	x_5
$\sigma(x)$	`True`	`False`	`False`	`False`	`True`

Data una formula di Horn $\mathcal{F}$, dividiamo le clausole in due insiemi: chiamiamo IMP l'insieme delle clausole che contengono esattamente un letterale positivo e NEG l'insieme di quelle che contengono solo letterali negativi. Per esempio, nella formula in (2)

$$\begin{aligned} \textsc{imp} &= \{x_1,\ \overline{x_1} \vee x_2 \vee \overline{x_3},\ \overline{x_2} \vee x_4,\ x_5\} \\ \textsc{neg} &= \{\overline{x_3} \vee \overline{x_4} \vee \overline{x_5}\} \end{aligned}$$

Si consideri il seguente algoritmo che prende in input una formula di Horn su un insieme di variabili e restituisce un'assegnazione di verità delle variabili oppure restituisce `None`.

Algorithm 1 GreedyHorn

INPUT: Una formula di Horn $\mathcal{F}$ su un insieme di variabili X
OUTPUT: Un'assegnazione di verità σ oppure `None`

```
Imposta σ(x) = False per ogni variabile x ∈ X
while esiste una clausola C ∈ IMP non soddisfatta da σ do
    Sia C ∈ IMP una clausola non soddisfatta e sia x la
    variabile corrispondente all'unico letterale positivo di C
    Imposta σ(x) = True
if Esiste una clausola C ∈ NEG che non è soddisfatta da σ then
    Return None
else
    Return σ
```

Esercizio 1. Dimostrare che se l'algoritmo GreedyHorn restituisce `None` allora la formula di Horn in input non è soddisfacibile.
(Suggerimento: Supponi che la formula $\mathcal{F}$ in input sia soddisfacibile e sia τ un'assegnazione che la soddisfa. Per $i = 1, 2, \ldots$, sia y_i la variabile posta a `True` durante l'i-esima iterazione del ciclo `while`: mostra, per induzione su i, che $\tau(y_i)$ deve essere `True` per ogni i. Concludi che anche l'assegnazione σ costruita dall'algoritmo deve soddisfare $\mathcal{F}$ e quindi se la formula in input è soddisfacibile l'algoritmo non può restituire `None`).

Esercizio 2. Scaricate un software che implementi lo standard OpenPGP e generate una vostra coppia di chiavi. Inviatemi poi una mail a `pasquale@mat.uniroma2.it` cifrandola con la mia chiave pubblica (che si trova sulla mia pagina web) e firmandola con la vostra chiave. Nella mail indicate nome, cognome e numero di matricola e allegate la vostra chiave pubblica.

Esercizio 3. Dato in intero $n \in \mathbb{N}$ e un numero reale $p \in (0, 1)$, un *random graph* $G_{n,p}$ è un grafo aleatorio con n nodi in cui gli archi sono scelti "a caso" secondo questa legge: per ogni coppia di nodi $\{u, v\}$, la probabilità che $\{u, v\}$ sia un arco è uguale a p, indipendentemente dalle altre coppie di nodi.

Scrivere un programma che prenda in input n e p, generi un random graph $G_{n,p}$[1] e verifichi se il grafo ottenuto è connesso oppure no. Testate il programma con vari valori di n e p e osservate che per $p \ll (\log n)/n$ il grafo che ottenete è quasi sempre disconnesso mentre per $p \gg (\log n)/n$ è quasi sempre connesso.

[1] Per molti linguaggi di programmazione non dovrebbe essere difficile trovare delle librerie che implementano la generazione di questi grafi. Altrimenti potete riscrivervi un semplice programma che li genera.

Esercizio 4. Data una lista di n interi positivi $d_1, d_2, \ldots, d_n$, vogliamo determinare in modo efficiente se esiste un grafo $G = (V, E)$ in cui i gradi dei nodi sono esattamente $d_1, \ldots, d_n$. Ossia, se indichiamo con $V = \{v_1, \ldots, v_n\}$ l'insieme dei nodi, allora per ogni $i = 1, 2, \ldots, n$ il grado del nodo v_i deve essere esattamente d_i. Il grafo non deve contenere *self-loop* (archi che hanno entrambi gli estremi nello stesso nodo) o archi *multipli* fra la stessa coppia di nodi.

1. Dare un esempio di d_1, d_2, d_3, d_4 in cui $d_i \leqslant 3$ per ogni i e $d_1 + d_2 + d_3 + d_4$ è pari, ma non esiste nessun grafo con sequenza dei gradi (d_1, d_2, d_3, d_4);
2. Supponiamo che $d_1 \geqslant d_2 \geqslant \cdots \geqslant d_n$ e che esiste un grafo $G = (V, E)$ con sequenza di gradi $(d_1, \ldots, d_n)$. Dimostrare che deve esistere anche un grafo con questa sequenza di gradi e in cui in più i vicini del nodo v_1 sono esattamente $v_2, v_3, \ldots, v_{d_1+1}$;
3. Usando il punto precedente, progettare un algoritmo che dati $d_1, \ldots, d_n$ decide se esiste un grafo con questa sequenza di gradi. L'algoritmo deve avere tempo polinomiale in n.