# Algoritmi 2: esercizi svolti

*Irene Finocchi* [§]

April 18, 2006

## 1 Orientazione di grafi

**Correttezza.** Per dimostrare che l'algoritmo produce una orientazione, basta osservare che l'insieme di vertici del grafo viene progressivamente svuotato, e ogni volta che si rimuove un vertice viene assegnata una orientazione a tutti gli spigoli ad esso incidenti non ancora orientati. Quindi al termine dell'algoritmo tutti gli spigoli avranno una orientazione.

**Fattore di approssimazione.** Sia $\Delta$ il grado entrante massimo nell'orientazione prodotta dall'algoritmo. Sia inoltre $\Delta^*$ il grado entrante massimo in una orientazione ottima. Ovviamente deve essere

$$\Delta^* \leq \Delta$$

per definizione di ottimo in un problema di minimizzazione. Per dimostrare che il fattore di approssimazione dell'algoritmo e' 2, facciamo vedere che risulta anche

$$\Delta \leq 2\Delta^*$$

Sia $v$ il primo vertice con grado entrante $\Delta$ scelto dall'algoritmo. Siano $n'$ ed $m'$ il numero di nodi e di archi nel sottografo $G'$ che l'algoritmo deve ancora esaminare nel momento in cui viene scelto $v$. Poiche' $v$ ha grado minimo $\Delta$ in $G'$ e tutti gli spigoli incidenti a $v$ in $G'$ sono "assegnati" a $v$, deve essere:

$$m' \geq \frac{n' \cdot \Delta}{2}$$

essendo $G'$ non orientato. Una orientazione ottima di $G$ deve avere grado $\Delta^*$ maggiore o uguale al grado entrante di una orientazione ottima di $G'$. Inoltre, una orientazione ottima di $G'$ deve avere grado maggiore o uguale del grado medio di $G'$, poiche' ogni arco deve essere "assegnato" ad uno dei suoi estremi. Quindi deve risultare:

$$\Delta^* \geq \frac{m'}{n'} \geq \frac{\Delta}{2}$$

[§]Dipartimento di Informatica, Università degli Studi di Roma "La Sapienza", Via Salaria 113, 00198 Rome, Italy. E-mail: {finocchi}@di.uniroma1.it.

**Tightness.** Dimostriamo ora che l'analisi e' "stretta", ovvero che esiste un grafo per cui $\Delta = 2\Delta^*$. Basta considerare un ciclo: su tale grafo l'orientazione prodotta dall'algoritmo e' tale che $\Delta = 2$ (per via del primo nodo scelto). Una soluzione ottima si ottiene orientando tutti gli archi nella stessa direzione, e quindi e' tale che $\Delta^* = 1$.

# 2 Sottosequenza comune

**Correttezza.** Dimostriamo innanzitutto che l'algoritmo produce una lunghezza $L$ che corrisponde ad una sottosequenza comune. Osserviamo che $\min\{NX, NY\}$ e' la lunghezza di una sottosequenza comune costituita da soli 0, mentre $\min\{n - NX, m - NY\}$ e' la lunghezza di una sottosequenza comune costituita da soli 1. L'algoritmo restituisce il massimo di qieste due quantita', che quindi corrisponde ad una sottosequenza comune.

**Non ottimalita'.** Date due qualunque sequenze $X = Y$ che contengano almeno uno 0 ed almeno un 1, la soluzione ottima e' $|X|$, ma l'algoritmo restituisce $\max\{$numero di 0, numero di $1\} < |X|$.

**Fattore di approssimazione.** Una sottosequenza comune di lunghezza massima $L^*$ puo' contenere al massimo il massimo numero possibile di 0 che le sequenze hanno in comune ($= \min\{NX, NY\}$), piu' il massimo numero possibile di 1 che le sequenze hanno in comune ($= \min\{n - NX, m - NY\}$). Quindi

$$L^* \leq \min\{NX, NY\} + \min\{n - NX, m - NY\} \leq 2 \cdot L$$

e il fattore di approssimazione e' 2.

**Tightness.** Basta considerare sequenze uguali in cui il numero di 0 e' pari al numero di 1, ad esempio $0^x1^x$, con $x > 0$. Su tali sequenze $L^* = 2x$ e $L = x$.

# 3 Intervalli

Dato un vettore $V$ di $n$ numeri reali, calcolare il minimo numero di intervalli di lunghezza unitaria in grado di ricoprire tutti i valori di $V$.

L'algoritmo è dato in Figura 1.

- Soluzione parziale: insieme di intervalli
- Estensione locale: aggiunta di un intervallo
- Criterio di convenienza: l'intervallo inizia dal punto più piccolo tra quelli non ancora coperti.

**Lemma 1** *Per ogni $h = 0, 1, \ldots, n$ esiste una soluzione ottima $SOL^*$ che include la soluzione corrente dell'algoritmo $SOL_h$.*

```
   Algoritmo RicopriConIntervalli
1. Input Un vettore V di n numeri reali
2. Output Un insieme di intervalli di lunghezza unitaria in grado di ricoprire tutti i valori di V
3. begin
4.     Ordina i numeri in ordine crescente
5.     SOL ← ∅
6.     for i = 1 to n do
7.         if V[i] non e' gia' coperto
8.         then SOL ← SOL ∪ {[V[i], V[i] + 1]}        {inserisci un nuovo intervallo in SOL}
9. end
```

Figure 1: Algoritmo per il ricoprimento di valori reali

**Proof.** Per induzione su $h$. Passo base: per $h = 0$ si ha $SOL_0 = \emptyset$, e l'affermazione risulta banalmente vera.

Supponiamo che l'affermazione sia vera fino al passo $h$ incluso, con $h < n$. Sia $p_{h+1}$ l'$(h+1)$-esimo punto da coprire. Se $p_{h+1}$ è già coperto, allora $SOL_{h+1} = SOL_h \subseteq SOL^*$.

Se $p_{h+1}$ non è coperto, allora l'algoritmo aggiunge l'intervallo $I = [p_{h+1}, p_{h+1} + 1]$ e quindi $SOL_{h+1} = SOL_h \cup \{I\}$. Se $I \in SOL^*$, allora $SOL_{h+1} \subseteq SOL^*$. Assumiamo quindi che $I \notin SOL^*$.

Osserviamo che $p_{h+1}$ deve essere coperto in $SOL^*$ da un intervallo $I^*$ tale che $I^* \notin SOL_h$ (altrimenti $p_{h+1}$ sarebbe già stato coperto e l'algoritmo non avrebbe aggiunto l'intervallo $I$). Sia $\widehat{SOL^*} = SOL^* \setminus \{I^*\} \cup \{I\}$. Mostriamo che $\widehat{SOL^*}$ è ancora una soluzione ammissibile:

- Poiché i punti sono presi in ordine crescente, per ogni $p_i$, con $1 \leq i \leq h$, esiste un intervallo $I' \in SOL_h$ tale che $I'$ copre $p_i$: poiché $I^* \notin SOL_h$, deve essere $I' \neq I^*$. Inoltre $I' \in SOL^*$ dato che $SOL_h \subseteq SOL^*$ per ipotesi induttiva. Quindi $SOL_h \subseteq \widehat{SOL^*}$ e tutti i punti a sinistra di $p_{h+1}$ sono ancora coperti in $\widehat{SOL^*}$.

- Il punto $p_{h+1}$ è coperto da $I$.

- I punti a destra di $p_{h+1}$ rimangono coperti in $\widehat{SOL^*}$. Infatti sostituire $I^*$ con $I$ equivale a far slittare $I^*$ verso destra finché l'estremo sinistro non coincide con $p_{h+1}$: tale operazione non può far scoprire nessun punto a destra di $p_{h+1}$.

Abbiamo quindi trovato una soluzione ottima (e legale) $\widehat{SOL^*}$ tale che $SOL_{h+1} \subseteq \widehat{SOL^*}$, dimostrando l'affermazione. □

Poiché l'algoritmo copre tutti i punti in $V$, risulta $SOL_n = SOL^*$.

Esercizio: considerare la variante del problema in cui gli intervalli devono avere lunghezza unitaria, e gli estremi devono essere numeri interi.