Esercizi Capitolo 14 - Algoritmi Greedy

Alberto Montresor

19 Agosto, 2014

Alcuni degli esercizi che seguono sono associati alle rispettive soluzioni. Se il vostro lettore PDF lo consente, è possibile saltare alle rispettive soluzioni tramite collegamenti ipertestuali. Altrimenti, fate riferimento ai titoli degli esercizi. Ovviamente, si consiglia di provare a risolvere gli esercizi personalmente, prima di guardare la soluzione.
Per molti di questi esercizi l'ispirazione è stata presa dal web. In alcuni casi non è possibile risalire alla fonte originale. Gli autori originali possono richiedere la rimozione di un esercizio o l'aggiunta di una nota di riconoscimento scrivendo ad `alberto.montresor@unitn.it`.

1 Problemi

1.1 Algoritmo di Prim (Esercizio 14.3 del libro)

Qual è la complessità dell'algoritmo di Prim se invece di utilizzare una realizzazione della coda con priorità basata su heap binario, si utilizza un vettore? Quando è conveniente utilizzare questa versione?

1.2 Algoritmo di Prim (Esercizio 14.4 del libro)

Qual è la complessità dell'algoritmo di Prim se invece di utilizzare una realizzazione della coda con priorità basata su heap binario, si utilizza un heap di Fibonacci?

1.3 Batterie

State viaggiando con una modernissima auto elettrica su un'autostrada, entrando al km 0 con la batteria carica ed dovendo uscire al km N. L'autonomia della batteria è r km; esistono n aree di servizio, ai km $d[1 \ldots n]$, dove la vostra batteria può essere sostituita con una carica.
Descrivete un algoritmo greedy che minimizzi il numero di soste, dimostrandone la correttezza e discutendone la complessità.
Nota: Negli esercizi aggiuntivi del Capitolo 13 relativi alla programmazione dinamica si trova una formulazione diversa di questo problema.

Soluzione: Sezione 2.3

1.4 Meglio blu

Un grafo G ha archi colorati di rosso e blu. Scrivere un algoritmo che restituisca un albero di copertura per G con il numero minimo di archi rossi.

Soluzione: Sezione 2.4

1.5 Matrimoni stabili

Dati n uomini e n donne, supponiamo che ogni persona abbiamo prodotto una lista di tutti i membri del sesso opposto, ordinata per preferenza dal più preferito al meno preferito. L'obiettivo è sposare tutti gli uomini e donne in modo tale che nessuna coppia uomo-donna preferirebbe sposarsi fra di loro invece che i loro attuali partner. Se non ci sono tali coppie, il matrimonio viene detto stabile e non ci sono tradimenti.
Nota: questo problema non è così assurdo. Si vocifera che venga utilizzato dalle facoltà di medicina USA per assegnare le specializzazioni agli studenti.

Soluzione: Sezione 2.5

1.6 Sciatori e sci

Si consideri il problema seguente. Siano dati n sciatori di altezza $p_1, \ldots, p_n$, e n paia di sci di lunghezza $s_1, \ldots, s_n$. Il problema è assegnare ad ogni sciatore un paio di sci, in modo da minimizzare la differenza totale fra le altezza degli sciatori e la lunghezza degli sci; ovvero, se allo sciatore i è assegnato il paio di sci $h(i)$, minimizzare la seguente quantità:

$$\sum_{i=1}^{n} |p_i - s_{h(i)}|$$

Si consideri il seguente algoritmo greedy. Si individui la coppia (sciatore, sci) con la minima differenza. Si assegni allo sciatore questo paio di sci. Si ripete con gli sciatori restanti fino a quando non si è terminato. Provare la correttezza di questo algoritmo o trovare un controesempio.

Soluzione: Sezione 2.6

1.7 Archi minimi

Sia dato un grafo non orientato $G = (V, E)$ e una funzione di pesi $w : E \rightarrow \mathbb{R}$, con pesi sugli archi tutti distinti. Siano e_1, e_2 ed e_3 l'arco con peso minimo, l'arco con il secondo peso minimo e l'arco con il terzo peso minimo, rispettivamente. Confutare o provare le seguenti affermazioni:

- Tutti gli alberi di copertura di peso minimo del grafo G contengono l'arco e_1.
- Tutti gli alberi di copertura di peso minimo del grafo G contengono l'arco e_2.
- Tutti gli alberi di copertura di peso minimo del grafo G contengono l'arco e_3.

Soluzione: Sezione 2.7

1.8 Albero di copertura unico

Sia G un grafo non orientato, connesso con pesi sugli archi tutti distinti. Dimostrare che esiste un unico minimo albero di copertura per G.

Soluzione: Sezione 2.8

1.9 Minimum product spanning tree

Nel "minimum product spanning tree", il costo di un albero è dato dal prodotto di tutti i pesi dell'albero, invece della somma. Si assuma che tutti i vertici abbiano un peso positivo.
Scrivere un algoritmo che calcoli il minimum product spanning tree (suggerimento: pensate ai logaritmi). Discutere la correttezza e la complessità dell'algoritmo proposto.

Soluzione: Sezione 2.9

2 Soluzioni

2.1 Algoritmo di Prim (Esercizio 14.3 del libro)

Utilizzando un vettore al posto di un *min-heap* nella realizzazione della coda con priorità, l'operazione deleteMin() ha costo $O(n)$ mentre l'operazione decrease() ha costo $O(1)$. Il costo diventa $O(n^2)$, che è migliore di $O(m \log n)$ per grafi densi ($m = \Theta(n^2)$).

2.2 Algoritmo di Prim (Esercizio 14.4 del libro)

Utilizzando uno heap di Fibonacci al posto di un *min-heap* nella realizzazione della coda con priorità, l'operazione deleteMin() ha costo $O(\log n)$ mentre l'operazione decrease() ha costo $O(1)$. Il costo diventa $O(m + n \log n)$.

2.3 Batterie

Per minimizzare il numero di fermate, si prosegue fino all'ultima stazione di servizio possibile, prima di rimanere a secco. Per semplificare il codice, supponiamo che esista un 'ultima fermata $D[n+1]$, come sentinella. Inoltre, assumiamo che non esistano due stazioni di servizio distanti più di r km (altrimenti l'autostrada non può essere percorsa).

```
SET fermate(integer[] D, integer n)
    deadline ← r
    SET stops ← Set()
    for i ← 1 to n do
        if D[i] ≤ deadline and D[i + 1] > deadline then
            stops.insert(i)
            deadline ← D[i] + r
    return stops
```

Il costo di questo algoritmo è chiaramente $O(n)$. La sottostruttura ottima è dimostrabile semplicemente facendo notare che se viene effettuata una ricarica alla stazione i, ci si riduce al problema $D[i \dots n]$ con una lunghezza della strada pari a $N - D[i]$; qualunque soluzione ottima per questo problema fa parte della soluzione ottima del problema originale.
La scelta greedy può essere dimostrata nel modo seguente: assumiamo che esista una soluzione S che non includa la stazione k, ultima possibile a partire dall'inizio. Sia i la prima stazione utilizzata in S, con $i < k$; se sostituisco i con k, ottengo una soluzione $S' = S - \{i\} \cup \{k\}$ che è comunque una soluzione ottima (ha la stessa dimensione) e rispetta tutti i vincoli (k è raggiungibile dall'inizio e può raggiungere la stazione successiva in S, perchè questa era raggiungibile da i che era più indietro).

2.4 Meglio blu

È necessario costruire una funzione di peso che associa il peso 1 agli archi rossi, il peso 0 agli archi blu. Si applica poi un algoritmo per ottenere il minimo albero di copertura, che conterrà il numero minimo di archi rossi.

2.5 Matrimonio stabile

Nel 1962, Gale e Shapley dimostrarono (tramite un algoritmo) che per ogni quantità n di uomini e donne, è sempre possibile costruire un insieme di matrimoni stabili.
L'algoritmo Gale-Shapley è basato su un numero di "round" in cui ogni uomo non fidanzato m si "propone" alla donna preferita w a cui non si è già proposto. Se w è già fidanzata con un uomo che preferisce ad m,

rifiuta la proposta; se w è già fidanzata con un uomo m' "meno preferito" di m, "scarica" m' e si fidanza con m; se w non è fidanzata, si fidanza con m.
Si garantisce che:

- Al termine di tutti i round, tutti sono fidanzati e possono sposare il proprio partner. Infatti, non possono esistere un uomo m e una donna w non fidanzati alla fine: ad un certo punto m deve essersi proposto a w (gli uomini ci provano con tutte), ed essendo non fidanzata, lei deve aver accettato. Una volta fidanzata, una donna resterà sempre fidanzata con qualcuno.
- I matrimoni sono stabili. Sia m un uomo e w una donna. Supponiamo che non siano sposati assieme. Non è possibile che sia m che w si preferiscano l'uno con l'altra rispetto ai loro attuali partner w' e m'. Infatti, se m preferisce w rispetto a w', deve essersi proposto a w prima di proporsi a w'. Se w non è sposata con m alla fine, deve aver accettato la proposta di qualcuno preferito ad m; quindi w preferisce il suo partner attuale a m.

Siano:

- n il numero di uomini e donne;
- $RM[1\dots n][1\dots n]$ il ranking degli uomini; $RM[m,k] = w$ significa che la donna w è in posizione k nelle preferenze dell'uomo m;
- $RW^{-1}[1\dots n][1\dots n]$ la funzione (inversa) di ranking delle donne; $RW^{-1}[w,m] = k$ significa che l'uomo m è in posizione k nelle preferenze della donna w.
- $wife[1\dots n]$ l'associazione marito-moglie; $wife[m] = 0$ significa che m non è ancora fidanzato; $wife[m] = w$ significa che m è fidanzato con w;
- $husband[1\dots n]$ l'associazione moglie-marito; $husband[w] = 0$ significa che w non è ancora fidanzata; $husband[w] = m$ significa che w è fidanzata con m;
- $last[1\dots n]$ un vettore di contatori; $last[m]$ è il numero di approcci effettuati da u.

```
stableMarriage(integer n, integer[][] RM, integer[][] RW^-1)
    integer[] wife ← new integer[1...n]
    integer[] husband ← new integer[1...n]
    for integer i ← 1 to n do
        wife[i] ← 0
        husband[i] ← 0
        last[i] ← 0
    while ∃m : wife[m] = 0 and last[m] < n do
        last[m] ← last[m] + 1
        w ← RM[m, last[m]]
        if husband[w] = 0 then
            wife[m] ← w; husband[w] ← m;
        else
            m' ← husband[w]
            k ← RW^-1[w, m]
            k' ← RW^-1[w, m']
            if k < k' then
                wife[m] ← w; husband[w] ← m; wife[m'] ← 0;
```

Complessità, caso pessimo Il caso pessimo si ha quando tutti gli uomini condividono lo stesso ordine di preferenze; quindi tutti gli uomini si "propongono" prima alla stessa donna, la quale sceglie il preferito. Poi si passa alla seconda, e così via. Questo richiede una complessità $O(n^2)$.

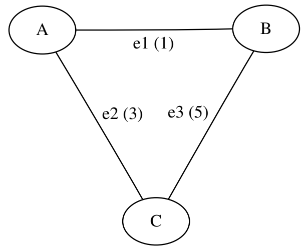

Figura 1: Controesempio per il problema 2.7

2.6 Sciatori e sci

Si consideri il seguente input: $p = \{5, 10\}$, $s = \{9, 14\}$. Secondo l'algoritmo, associamo $p[2]$ ad $s[1]$ e $p[1]$ ad $s[2]$ (ovvero $h(1) = 2$, $h(1) = 2$. La differenza totale è $|10 - 9| + |14 - 5| = 10$. Se l'associazione fosse $h(1) = 1$ e $h(2) = 2$, la differenza totale sarebbe: $|9 - 5| + |10 - 14| = 8$. Quindi l'algoritmo proposto non è corretto.

2.7 Archi minimi

La prima affermazione è vera. Per assurdo, sia T un albero ottimo che non contiene e_1. Si consideri il grafo $\{e_1\} \cup T$; per definizione di albero, in questo grafo è presente un ciclo. Si consideri un qualunque arco $e \neq e_1$ di questo ciclo composto da almeno tre archi, e lo si rimuova; poiché $w(e) > w(e_1)$, si è così ottenuto un albero di copertura di peso inferiore a quello di T, assurdo.
La seconda affermazione è vera. Per assurdo, sia T un albero ottimo che non contiene e_2. Sappiamo già che T deve contenere e_1. Si consideri il grafo $\{e_2\} \cup T$; come prima, in questo grafo è presente un ciclo. Questo ciclo deve essere composto da almeno tre archi, e quindi esiste in questo ciclo un arco e diverso sia da e_1 che da e_2; rimuovendo quest'arco, si ottiene un albero $T \cup \{e_2\} - \{e\}$ che è un albero di copertura di peso inferiore a T, assurdo.
La terza affermazione è falsa, dimostrabile facilmente con un controesempio. Si consideri il grafo di Figura 1, il cui arco di copertura di peso minimo contiene e_1 ed e_2 ma non e_3.

2.8 Albero di copertura unico

Si consideri uno albero di copertura di peso minimo $T \subseteq E$ ottenuto dall'algoritmo di Kruskal (gli archi vengono selezionati in ordine crescente di peso, evitando di costruire cicli). Per assurdo, si consideri uno albero di copertura $T' \subseteq E$ diverso da T che abbia lo stesso peso minimo di T.
Siano $e_1, e_2, \ldots, e_n$ gli archi di E ordinati per peso crescente. Si identifichi il più basso valore di j tale per cui $e_j \in T' - T$. Al momento in cui Kruskal ha valutato e_j, l'arco non è stato inserito perché si veniva a formare un ciclo. Si consideri un arco e_k di questo ciclo; per definizione di e_j, $k < j$; per costruzione dell'algoritmo di Kruskal, $w(e_k) < w(e_j)$. Quindi l'albero $T' - \{e_j\} \cup \{e_k\}$ è un albero di costo inferiore a T', il che è assurdo.

2.9 Minimum product spanning tree

È possibile notare che $\log(a \cdot b) = \log a + \log b$; quindi è sufficiente costruire una funzione di peso w' tale che $w'(u, v) = \log w(u, v)$ e trovare un albero di copertura minimo con questi pesi.

3 Problemi aperti

3.1 Matrice di incidenza (Esercizio 14.8 del libro)

Sia data la matrice di incidenza nodi-archi di un grafo non orientato. Si dimostri che un insieme di colonne della matrice è linearmente indipendente se e solo se gli archi corrispondenti a tali colonne non formano un circuito.

3.2 Cardinalità insiemi indipendenti massimali (Esercizio 14.9 del libro)

Si dimostri per assurdo che tutti gli insiemi indipendenti massimali di un matroide hanno la stessa cardinalità.

3.3 Massimizzazione e minimizzazione (Esercizio 14.10 del libro)

Si dimostri se i problemi MASSIMO ALBERO DI COPERTURA e CAMMINI MASSIMI sono equivalenti o meno ai problemi MINIMO ALBERO DI COPERTURA e CAMMINI MINIMI.

3.4 Monete in Strangeland

Si consideri il seguente insieme di monete: $1, 5, 10, 20, 25, 50$. L'algoritmo greedy per il resto restituisce la risposta ottima per questo insieme di monete?

3.5 Huffman può sbagliare?

Si consideri l'algoritmo di compressione di Huffman. È possibile progettare un testo la cui forma compressa sia più lunga della lunghezza originale?