Laboratorio di Algoritmi

Progetto "Tubazioni" (novembre 2021)

Nota: La scadenza del progetto è fissata per martedì 2 novembre **compreso**.

Nota: Si consiglia di consultare sulla pagina web il documento che riporta le avvertenze utili per lo svolgimento del progetto. Si consiglia anche di verificare di tanto in tanto gli aggiornamenti a questo documento, che potrà essere aggiornato con la correzione di eventuali errori e le risposte ai dubbi degli studenti.

Nota: Purtroppo, i problemi di ottimizzazione trattati nel progetto ammettono diverse soluzioni equivalenti. Ho cercato di descrivere gli algoritmi in modo da suggerire la via verso le soluzioni allegate negli esempi, ma non penalizzerò implementazioni che forniscano soluzioni diverse, purché queste siano corrette e quindi abbiano lo stesso costo.

Commento generale Contrariamente a quello di altri progetti, il testo di questo fornisce una descrizione degli algoritmi da realizzare. La descrizione è volutamente schematica per lasciare spazio nella relazione a una descrizione precisa e soprattutto ad un'analisi corretta e completa.

Il problema Una tecnica standard per tenere le reti idrauliche sotto controllo è di dividerle in sottoreti e localizzare ai confini delle sottoreti stesse dei sensori di pressione, portata, ecc... La definizione corretta delle sottoreti dipende da molti aspetti, ma qui ci concentriamo su uno di loro: affinché i sensori eseguano le misure più significative possibili, è bene che ogni sottorete abbia una quota uniforme, ovvero sia quanto più "piatta" possibile.

Per semplicità, descriveremo la rete idraulica come un insieme di tubazioni che si incrociano fra loro e misureremo le quote h_i solo in punti i specifici, che corrispondono ai punti di incrocio fra tubazioni, ai punti di terminazione (sorgenti e pozzi) e a punti di particolare interesse che spezzano una singola tubazione in due e che tratteremo come "incroci" da cui si dipartono due tubazioni. In questo modo, il problema si riconduce alla partizione di una rete in un dato numero p di sottoreti tra loro disgiunte. Ciascuna sottorete dovrà essere connessa, cioè dovrà essere possibile muoversi da un punto a un altro della stessa sottorete usando solo tubazioni appartenenti alla sottorete. Le tubazioni che collegano punti di sottoreti diverse non appartengono ad alcuna sottorete. Per semplicità, è ammesso che una sottorete si riduca a un singolo punto senza tubazioni (potrebbe essere un punto di particolare interesse). Per ogni sottorete $r \in \{1, \ldots, p\}$ definiremo il *dislivello* γ_r come differenza fra la quota massima e la quota minima dei punti della sottorete stessa; per le sottoreti con un solo punto, il dislivello è ovviamente nullo.

Siccome una buona partizione è caratterizzata dall'avere p sottoreti tutte di dislivello basso, si può valutare la qualità di una partizione in due modi:

1. problema *min-max*: come il massimo dislivello fra le p sottoreti $\max_{r=1,\ldots,p} \gamma_r$;
2. problema *min-sum*: come la somma dei dislivelli delle p sottoreti $\sum_{r=1}^{p} \gamma_r$ (che ovviamente equivale al dislivello medio).

È stato dimostrato, sia per il problema *min-max* sia per il problema *min-sum*[1] che l'esistenza di un algoritmo polinomiale implicherebbe $\mathcal{P} = \mathcal{NP}$, cioè l'esistenza di

[1] Rispettivamente, da me e da un collega.

algoritmi polinomiali per centinaia di altri problemi di importanza fondamentale in tutti i campi applicativi. Trovarne uno è quindi probabilmente impossibile, e comunque estremamente difficile. Ci sono però buone notizie nel caso in cui la rete idraulica abbia forme particolari, fra cui quelle di cui ci occupiamo qui, e precisamente:

- *cammini*: la rete è una semplice sequenza di tubazioni;
- *caterpillar* ("bruco"): la rete è una semplice sequenza di tubazioni con brevi diramazioni laterali (singoli punti di terminazione);

La Figura 1 illustra i due casi. Queste forme ammettono algoritmi polinomiali per ottimizzare entrambe le funzioni obiettivo e il progetto consiste nel realizzarli.

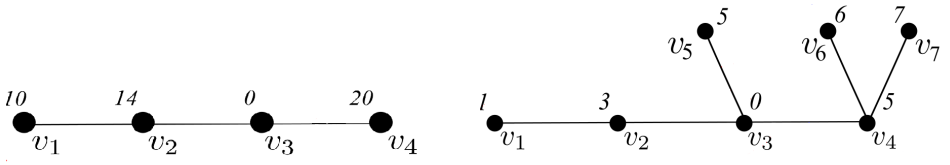

Figura 1: I due tipi di rete idraulica considerati (cammini e bruchi)

Il progetto Il programma da realizzare carica i dati da un file di testo il cui nome va fornito dall'utente nella linea di comando. Il file contiene la descrizione di una rete e il numero di sottoreti in cui va partizionata. In particolare, la prima riga del file contiene il numero di punti la cui quota è stata misurata, il numero di tubazioni che li collegano e il numero di sottoreti da ottenere, separati da spazi. Per esempio, la riga:

```
7 6 3
```

si riferisce al caterpillar a destra in Figura 1, che ha 7 punti e 6 tubazioni, e indica che esso va diviso in 3 sottoreti.

La riga seguente riporta le quote dei punti stessi, che sono tutti numeri interi separati da spazi, nell'ordine corrispondente agli indici numerici dei punti:

```
1 3 0 5 5 6 7
```

quindi il punto v_1 ha quota 1, il punto v_2 ha quota 3, ecc...

Il file prosegue con l'indicazione delle tubazioni, cioè delle coppie di punti direttamente collegati fra loro da una tubazione. Ogni coppia occupa una riga a parte, ed è identificata dagli indici numerici dei due punti estremi (prima il minore poi il maggiore), racchiusi fra parentesi tonde e separati da una virgola. Le coppie non seguono un ordine particolare. Considerando sempre l'esempio del caterpillar di Figura 1, si ha:

```
(1,2)
(3,5)
(4,6)
(2,3)
```

```
(4,7)
(3,4)
```

La prima elaborazione da compiere una volta caricata la rete è di riconoscerne la natura. Precisamente, bisogna indicare se si tratta di uno dei due tipi trattabili, stampando a video una riga che contenga la parola chiave corrispondente (`cammino` o `caterpillar`). Esistono anche tre tipi di reti non trattabili:

1. le reti *sconnesse* contengono coppie di punti non reciprocamente raggiungibili attraverso le tubazioni;
2. le reti *cicliche* contengono coppie di punti raggiungibili fra loro attraverso più cammini diversi;
3. le reti ad *albero* sono connesse e acicliche come i cammini e i caterpillar, ma non ricadono in tali categorie.

In corrispondenza, il programma dovrà stampare le parole chiave `sconnessa`, `ciclica` e `albero`. Siccome esistono anche reti che sono sia sconnesse sia cicliche, può capitare di dover stampare entrambe le parole chiave, cioè `sconnessa ciclica`. Per le reti non trattabili, il programma termina subito dopo il riconoscimento. Per le altre, si prosegue risolvendo prima il problema *min-max* poi il problema *min-sum*, e stampando le relative soluzioni.

Vale una proprietà molto utile: *se è possibile partizionare una rete connessa in p sottoreti con dislivello massimo (oppure con dislivello totale) $\leq \gamma$, è anche possibile partizionarla in p' sottoreti con dislivello massimo (oppure totale) $\leq \gamma$ per qualsiasi valore di p' compreso fra p e il numero totale di punti n.* Questo è garantito dal fatto che si può sempre staccare da una sottorete connessa un singolo punto "di frontiera", opportunamente scelto in modo da mantenere connessa la sottorete residua (nel caso di un cammino si stacca il primo o l'ultimo punto, nel caso di un caterpillar il primo o l'ultimo punto del cammino centrale oppure un punto esterno). In questo modo si ottengono due sottoreti, una delle quali (il singolo punto) ha dislivello nullo, mentre l'altra ha dislivello non superiore a quello della sottorete originale. In breve, staccare singoli punti aumenta il numero di sottoreti e non peggiora il valore dei due obiettivi.

Cammino *min-max* Fissiamo sul cammino un orientamento convenzionale, scegliendo come origine il punto estremo di indice minore e come destinazione l'altro punto estremo (nell'esempio di Figura 1 il cammino viene orientato da v_1 a v_4). Fissato un valore di soglia $\bar{\gamma}$, si può trovare la partizione della rete nel minimo numero di sottoreti il cui dislivello rispetti la soglia stessa con la procedura descritta più sotto. Si hanno tre casi possibili:

1. il numero minimo di sottoreti è p: la soluzione trovata è ammissibile;
2. il numero minimo di sottoreti è $< p$: si può costruire una soluzione con p sottoreti staccando via via un punto per volta dalle sottoreti ottenute (per convenzione, considerando le reti e staccando i punti secondo l'orientamento convenzionale del cammino);
3. il numero minimo di sottoreti è $> p$: non esistono soluzioni con un dislivello massimo $\leq \bar{\gamma}$.

Il valore ottimo del dislivello per il problema *min-max* è il valore minimo che consente ancora di costruire una partizione in p sottoreti. Si noti che la soglia $\bar{\gamma}$ ha un numero finito di valori possibili, dato che il dislivello di ogni sottorete è la differenza fra l'altitudine massima e minima dei suoi punti.

Per determinare la partizione di un cammino nel minimo numero di sottoreti che rispettino una soglia data, basta partire dal primo punto e scorrere i punti in sequenza aggiungendoli alla prima sottorete e aggiornandone via via il dislivello, per fermarsi all'ultimo punto nel quale la soglia rimane rispettata. Si riparte poi dal punto successivo con un'altra sottorete, e così via, sino al termine del cammino.

Cammino *min-sum* Per il problema *min-sum* si può realizzare un algoritmo di programmazione dinamica simile a quello descritto nel corso per il problema dello zaino. Per semplicità, nel seguito rinumeriamo i punti da 1 a n seguendo l'ordine convenzionale lungo il cammino. L'idea è costruire una matrice con p righe (una per sottorete) e n colonne (una per punto), associando alla generica cella (r, i) la partizione ottima (con il minimo dislivello totale) in r sottoreti dei punti $1, \ldots, i$. La cella (p, n) conterrà quindi la soluzione ottima dell'intero problema. Costruire la prima riga di questa matrice ($r = 1$) è banale, dato che la cella $(1, i)$ rappresenta un solo sottocammino dal punto 1 al punto i, con il suo dislivello. Nel problema dello zaino, data una generica cella, si passava alla riga seguente considerando le due scelte possibili: prendere o non prendere l'oggetto successivo. Ognuna delle due scelte dava luogo a una soluzione potenziale, che poteva aggiornare la cella corrispondente della matrice, se il premio ottenuto era maggiore di quello contenuto in essa. In questo problema, invece, data la generica cella (r, i), si passa alla riga seguente (cioè alla sottorete successiva) considerando tutte le scelte possibili, che sono tutte le sottoreti che partono dal punto $i + 1$ e terminano in ciascun punto j successivo. La sottorete aggiuntiva ha un dislivello che va a sommarsi a quello totale conservato nella cella (r, i), dando luogo a una potenziale soluzione per la cella $(r + 1, j)$, che va confrontata con quella attualmente conservata nella cella stessa per eventualmente aggiornarla.

Caterpillar *min-max* I caterpillar somigliano molto ai cammini, dato che consistono in un cammino centrale a cui sono appesi singoli punti esterni. La presenza dei punti esterni consente un altro modo di creare sottoreti, oltre a dividere il cammino centrale in sottocammini: si possono tagliare singoli punti esterni, creando delle sottoreti di dislivello nullo. Si può modificare l'algoritmo di programmazione dinamica per i cammini per tener conto di queste nuove opportunità. Si avrà una matrice con p righe associate alle sottoreti e n_c colonne associate ai punti del cammino centrale. La cella generica (r, i) indica la partizione ottima in r sottoreti dei punti $1, \ldots, i$ del cammino centrale, ma anche dei punti esterni adiacenti ad essi. Costruire la prima riga rimane banale, dato che la cella $(1, i)$ rappresenta una singola sottorete contenente tali punti, con il relativo dislivello; le righe seguenti, però possono rappresentare una sottorete contenente i punti $1, \ldots, i$ del cammino centrale e $r - 1$ sottoreti ottenute tagliando singoli punti esterni, ma anche più sottoreti che si dividono il cammino centrale, ciascuna con eventuali punti esterni tagliati. Ad ogni passo che si muove dalla situazione iniziale (nessuna sottorete e nessun punto del cammino centrale) oppure dalla soluzione parziale rappresentata dalla cella (r, i) si può aggiungere:

- una singola sottorete contenente i punti del cammino centrale da $i + 1$ a j e i punti esterni adiacenti ad essi, generando una soluzione potenziale per la cella $(r + 1, j)$;
- due sottoreti: una principale contenente i punti del cammino centrale da $i + 1$ a j e i punti esterni adiacenti tranne uno, e una secondaria costituita dal punto esterno tagliato; ogni coppia di sottoreti di questo tipo genera una soluzione potenziale per la cella $(r + 2, j)$;

- ...
- s sottoreti: una principale contenente i punti del cammino centrale da $i+1$ a j e i punti esterni adiacenti tranne $s-1$, e $s-1$ secondarie costituite dai punti esterni tagliati; ogni insieme di sottoreti di questo tipo genera una soluzione potenziale per la cella $(r+s, j)$.
- ...

Può sembrare una situazione esplosiva, ma non lo è, perché, fissati gli estremi $i+1$ e j della sottorete sul cammino centrale e il numero $s-1$ di punti esterni tagliati, si tratta solo di trovare il miglior insieme di sottoreti corrispondenti. Siccome i punti tagliati hanno dislivello nullo, si tratta solo di tagliare quelli che lasciano la sottorete principale con dislivello residuo minimo. Conviene risolvere a parte, preliminarmente, questo problema ausiliario nel modo descritto in seguito. Data la sua soluzione per ogni valore di i, j e s, si può costruire la matrice per la programmazione dinamica come prima, sommando il dislivello delle sottoreti aggiuntive a quello conservato nella cella (r, i), ottenendone una soluzione potenziale per la cella $(r+s, j)$, confrontandola con quella attualmente conservata nella cella stessa ed eventualmente aggiornandola.

Per determinare quali $s-1$ punti esterni tagliare da un tratto $(i+1, \ldots, j)$ del cammino centrale, ci si fa guidare dal valore del dislivello risultante. Si determina l'intervallo compreso fra la quota minima e massima del sottocammino centrale. Si ordinano i punti esterni adiacenti per quota crescente (a parità di quota, per indice crescente). Siccome conviene tagliare solo punti che riducono il dislivello complessivo, si tagliano quelli che stanno al principio o alla fine della sequenza ordinata. Dato il numero $s-1$ di punti da tagliare, si prova sistematicamente a tagliarne s' al principo e $s-1-s'$ alla fine con s' variabile da 0 a $s-1$ e si valuta il dislivello rimanente. Bisogna tener conto del fatto che tale dislivello deriva sia dai punti esterni rimasti sia dal sottocammino centrale, che non viene toccato. Consideriamo il solito esempio in Figura 1, e supponiamo di partire dalla cella $(r, i) = (1, 2)$, cioè di avere una prima sottorete composta dal sottocammino (v_1, v_2), con dislivello $3-1=2$. Supponiamo di voler arrivare al punto al punto v_4 (dunque $j=4$). Le quote estreme del sottocammino centrale (v_3, v_4) sono 0 e 5. Tre punti esterni sono adiacenti e le loro quote in ordine crescente sono 5, 6 e 7. Se si vuole ottenere una sola sottorete ($s=1$), non si taglia alcun punto esterno, il dislivello della sottorete è $7-0=7$ e si passa dalla cella $(r, i) = (1, 2)$ alla cella $(r+s, j) = (2, 4)$. Il dislivello complessivo ($\max(2, 7) = 7$) va confrontato con quello eventualmente già contenuto nella cella di destinazione, aggiornandolo se è migliore, cioè più piccolo. Se invece si vogliono ottenere due sottoreti aggiuntive ($s=2$), raggiungendo la cella $(r+s, j) = (3, 4)$, bisogna tagliare uno dei tre punti esterni adiacenti. Si possono tagliare $s'=0$ punti al principio e $s-1=1$ alla fine, cioèil punto v_7, che ha quota 7, ottenendo un dislivello pari a $6-0=6$. Oppure si possono tagliare $s'=1$ punti al principio (cioè v_5) e nessuno alla fine, ottenendo un dislivello pari a $7-0=7$, dato che il sottocammino centrale impone l'intervallo $[0; 5]$. Anche qui si confronta il dislivello complessivo con quello eventualmente già presente nella cella di destinazione. Infine, se si vogliono ottenere tre sottoreti aggiuntive ($s=3$), si possono tagliare due punti alla fine, oppure uno al principio e uno alla fine, oppure due al principio. Se $s=4$, si tagliano tutti e tre i punti esterni. Non è possibile tagliarne più di tre.

Caterpillar *min-sum* L'algoritmo per questo caso è praticamente identico a quello del problema *min-max*. L'unica cosa che cambia è che, quando si ipotizza di aggiungere s sottoreti passando dalla cella (r, i) alla cella $(r+s, j)$, invece di

considerare il massimo fra il dislivello della nuova sottorete e il valore riportato nella cella di partenza, se ne calcola la somma, dato che la funzione obiettivo per questo problema non è il massimo, ma la somma dei singoli dislivelli.

Stampa della soluzione Per ciascuno dei due problemi, il programma deve stampare a video la soluzione ottima, riportando nella prima riga il nome del problema (`min-max` oppure `min-sum`), seguito dal valore ottimo. Nelle p righe seguenti, il programma riporterà le sottoreti, ordinate lessicograficamente in base agli indici dei punti che vi appartengono. Ogni sottorete occupa una riga diversa, sulla quale si devono stampare in ordine crescente gli indici numerici dei punti che vi appartengono. Per ciascuna sottorete, dopo l'elenco dei punti il programma deve stampare una barra verticale ("`|`"), la quota massima, un meno ("`-`"), la quota minima un uguale ("`=`") e il dislivello.

Esempi Supponendo di dover dividere in $p = 2$ sottoreti la rete a sinistra nella Figura 1, si dovrà prima riconoscere che si tratta di un cammino, poi risolvere il problema *min-max*, ottenendo la sottorete (v_1, v_2, v_3) di dislivello 14 e la sottorete (v_4) di dislivello nullo, con un dislivello massimo pari a $\max(14, 0) = 14$. Infine, si dovrà risolvere il problema *min-sum*, ottenendo le stesse due sottoreti, con un dislivello totale pari a $14 + 0 = 14$. Di conseguenza:

```
cammino
min-max 14
1 2 3 | 14 - 0 = 14
4 | 20 - 20 = 0
min-sum 14
1 2 3 | 14 - 0 = 14
4 | 20 - 20 = 0
```

Nel caso della rete di destra nella Figura 1, che abbiamo già supposto doversi dividere in $p = 3$ sottoreti, si dovrà riconoscere che si tratta di un caterpillar, risolvere il problema *min-max*, ottenendo la sottorete (v_1, v_2, v_3) di dislivello 3, la sottorete (v_5) di dislivello nullo e la sottorete (v_4, v_6, v_7) di dislivello 2, con un dislivello massimo pari a $\max(3, 0, 2) = 3$. Infine, si dovrà risolvere il problema *min-sum*, ottenendo le stesse tre sottoreti, con un dislivello totale pari a $3 + 0 + 2 = 5$[2]. Di conseguenza:

```
caterpillar
min-max 3
1 2 3 | 3 - 0 = 3
4 6 7 | 7 - 5 = 2
5 | 5 - 5 = 0
min-sum 5
1 2 3 4 5 | 5 - 0 = 5
6 | 6 - 6 = 0
7 | 7 - 7 = 0
```

Chiarimenti

- *È possibile assumere che gli indici dei punti seguano un ordine particolare lungo i cammini, caterpillar o le altre reti?*

[2] È solo un caso che in questo esempio i due problemi abbiano soluzioni identiche.

No: il fatto che gli indici siano ordinati nella Figura 1 è un caso. Non la correggo perché dovrei riscrivere tutti gli esempi nel testo e non ho tempo di farlo, ma gli indici sono distribuiti casualmente nella rete. Si può assumere che siano tutti diversi e compresi fra 1 e il numero totale di punti.