Laboratorio di Algoritmi

Progetto "Rete" (febbraio 2026)

Nota: La scadenza del progetto è fissata per domenica 15 febbraio **compresa**.

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 potranno riportare risposte ai dubbi degli studenti e correzioni di eventuali errori.

Osservazioni introduttive Il testo di questo progetto, contrariamente a quanto avviene di solito, descrive gli algoritmi da realizzare. Lo fa in modo volutamente schematico affinché sia possibile valutare la completezza e correttezza della descrizione fornita dalla relazione e soprattutto dell'analisi di complessità.

Un'altra caratteristica insolita di questo progetto è che ammette in diverse parti molteplici soluzioni corrette. La descrizion degli algoritmi cerca di indirizzare verso le soluzioni allegate negli esempi, ma implementazioni che forniscano soluzioni equivalenti saranno considerate accettabili, purché siano corrette.

Il problema Nel tenere le reti idrauliche sotto controllo, è uso comune dividerle in sottoreti posizionando fra l'una e l'altra sottorete opportuni sensori di pressione e portata. Affinché i sensori forniscano misure significative, è bene che ogni sottorete abbia una quota quasi uniforme, ovvero sia quanto più "piatta" possibile.

Per semplicità, descriviamo una rete idraulica come un insieme di punti di particolare interesse nei quali sia possibile installare sensori e un insieme di tubazioni, ognuna delle quali collega due punti di interesse senza passare per altri. Per ogni punto di interesse i è nota la quota h_i. Il problema consiste allora nel suddividere la rete in un dato numero p di sottoreti che contengono punti diversi. Ogni sottorete contiene tutte le tubazioni fra i punti ad essa assegnati e deve essere connessa, cioè tale da potersi muovere da un punto a un altro usando solo le proprie tubazioni. Le tubazioni che collegano punti di sottoreti diverse non appartengono ad alcuna sottorete. Sono ammesse sottoreti ridotte a un singolo punto di interesse senza tubazioni. Per ogni sottorete $r \in \{1, \ldots, p\}$, il *dislivello* γ_r è la differenza fra la quota massima e la quota minima dei punti della sottorete stessa; per le sottoreti contenenti un solo punto, il dislivello è ovviamente nullo.

Una buona suddivisione è formata da p sottoreti di dislivello basso. Valuteremo la qualità di una suddivisione in due modi:

1. nel problema *min-max*, come il massimo dislivello fra le p sottoreti $\max_{r=1,\ldots,p} \gamma_r$;
2. nel problema *min-sum*, come la somma dei dislivelli delle p sottoreti $\sum_{r=1}^{p} \gamma_r$.

Entrambi i problemi sono computazionalmente difficili su reti generiche. Tuttavia, esistono algortimi polinomiali per reti con le seguenti forme particolari:

- *cammino*: la rete è una semplice sequenza di tubazioni (Figura 1 a sinistra);
- *caterpillar* ("bruco"): la rete è una sequenza principale di tubazioni con brevi diramazioni laterali formate da tubazioni singole (Figura 1 a destra).

Il progetto consiste nel realizzare questi algoritmi polinomiali per ottimizzare le due funzioni obiettivo.

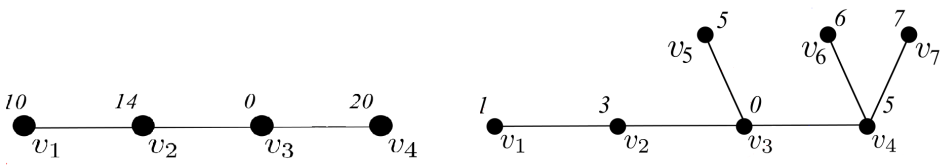

Figura 1: I due tipi di rete idraulica considerati (cammini e bruchi)

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. La prima riga di tale file contiene il numero n di punti di interesse, il numero m di tubazioni e il numero p di sottoreti da ottenere, separati da spazi. Per esempio, la riga:

```
7 6 3
```

descrive il 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...
Ciascuna delle righe seguenti contiene gli indici numerici dei due punti estremi di una tubazione. Gli indici sono racchiusi fra parentesi tonde, ordinati per valori crescenti e separati da una virgola. Le coppie non seguono un ordine particolare. Considerando sempre l'esempio del caterpillar a destra in Figura 1, si ha:

```
(1,2)
(3,5)
(4,6)
(2,3)
(4,7)
(3,4)
```

Si noti che i punti sono numerati da 1 a n, ma la numerazione non segue alcun particolare ordine legato alla struttura della rete. Qui si è assunto $v_i = i$ per fornire un esempio più leggibile, ma in generale non sarà così.

Per prima cosa, il programma deve riconoscere la natura della rete. Vi sono cinque tipi di reti:

1. i *cammini*, definiti sopra;
2. i *caterpillar*, definiti sopra;

3. le reti *sconnesse*, che contengono coppie di punti non reciprocamente raggiungibili attraverso le tubazioni;

4. le reti *cicliche*, che contengono coppie di punti raggiungibili fra loro attraverso più sequenze di tubazioni diverse;

5. le reti ad *albero*, nelle quali ogni punto è raggiungibile da ogni altro attraverso una e una sola sequenza di tubazioni, ma senza ricadere nel caso dei cammini e dei caterpillar (che pure godono della stessa proprietà)

Il programma deve indicare il tipo di rete, stampando a video una riga che contenga la parola chiave corrispondente: `cammino`, `caterpillar`, `sconnessa`, `ciclica` o `albero`. Per le reti che sono sia sconnesse sia cicliche, si stamperà `sconnessa ciclica`. Eventuali reti costituite da un solo punto sono considerate cammini. Per i cammini e i caterpillar, il programma prosegue risolvendo il problema *min-max* e il problema *min-sum* come descritto nel seguito. Per le altre, termina dopo il riconoscimento.

Gli algoritmi che seguono sfruttano una proprietà fondamentale valida per entrambi i problemi: staccare da una sottorete singoli punti aumenta il numero di sottoreti e non peggiora il valore dell'obiettivo. Più formalmente, *se si può suddividere una rete connessa in p sottoreti connesse con dislivello (massimo o totale) $\leq \gamma$, si può suddividerla anche in p' sottoreti connesse con dislivello (massimo o totale) $\leq \gamma$ per qualsiasi valore di p' compreso fra p e il numero totale di punti n.* Infatti, si può 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 della sequenza principale oppure un punto esterno). Questo produce due sottoreti, una delle quali ridotta a un singolo punto con dislivello nullo, mentre l'altra ha dislivello non superiore a quello della sottorete originale.

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). Il valore ottimo del dislivello per il problema *min-max* è il valore minimo di una soglia $\bar{\gamma}$ che consenta di suddividere la rete in p sottoreti di dislivello $\leq \bar{\gamma}$. Questa soglia ha un numero finito di valori possibili, dato che il dislivello di ogni sottorete è certamente compreso fra zero e la differenza fra la quota massima e la quota minima della rete.

Per determinare se è possibile suddividere la rete in p sottoreti rispettando la soglia $\bar{\gamma}$, basta trovare il minimo numero di sottoreti disgiunte che la rispettino e confrontarlo col numero p richiesto:

1. se il numero minimo di sottoreti è p, la soluzione trovata è ammissibile;

2. se il numero minimo di sottoreti è $< p$, si può costruire una soluzione con p sottoreti staccando un punto per volta dalle sottoreti ottenute; per convenzione, stacchiamo i punti scorrendo il cammino secondo l'orientamento convenzionale;

3. se il numero minimo di sottoreti è $> p$, non è possibile dividere la rete rispettando la soglia.

Per trovare il minimo numero di sottoreti disgiunte rispettino la soglia $\bar{\gamma}$, basta scorrere il cammino secondo l'orientamento convenzionale e aggiungere i punti visitati via via alla sottorete corrente, fermandosi e creando una nuova sottorete quando il dislivello violerebbe la soglia.

Cammino *min-sum* Il problema *min-sum* si risolve con un algoritmo di programmazione dinamica simile a quello descritto nel corso per il problema dello zaino. Costruiamo una matrice di p righe (associate alle sottoreti) e n colonne (associate ai punti, ordinati secondo l'orientamento convenzionale del cammino). La cella (r, i) della matrice rappresenta la suddivisione ottima (con il minimo dislivello totale) in r sottoreti dei punti nelle prime i posizioni. Quindi, la cella (p, n) rappresenta la soluzione ottima del problema. Costruire la prima riga della matrice è banale, dato che la cella $(1, i)$ rappresenta la sottorete dal primo all'i-esimo punto, con il relativo dislivello, facilmente calcolabile. Nel problema dello zaino, da una cella della riga i si passa a due possibili celle sulla riga $i + 1$, che rappresentano le due scelte possibili: prendere o non prendere l'oggetto $i + 1$-esimo. Ognuna delle due scelte produce una soluzione potenziale che aggiorna la cella corrispondente della matrice se ha un premio maggiore di quello contenuto in essa. In questo problema, dalla cella (r, i) si può passare alla riga seguente, cioè alla sottorete $r + 1$ considerando tutte le scelte possibili, cioè aggiungendo una nuova sottorete che parte dal punto $i + 1$ e termina in ciascun punto j successivo. Il dislivello della nuova sottorete si somma a quello totale conservato nella cella (r, i) e genera una soluzione potenziale per la cella $(r + 1, j)$, che va confrontata con quella attualmente conservata in essa per decidere se aggiornarla[1].

Caterpillar *min-max* I caterpillar consistono in una sequenza principale di tubazioni a cui sono appesi singoli punti esterni. Questi punti consentono di creare sottoreti non solo dividendo la sequenza centrale, ma anche staccando singoli punti esterni a formare sottoreti di dislivello nullo. Si può modificare l'algoritmo di programmazione dinamica usato per i cammini per tener conto delle nuove opportunità. Si avrà ancora una matrice con p righe associate alle sottoreti e n_c colonne associate ai punti della sequenza principale. La cella generica (r, i) indica la partizione ottima in r sottoreti dei punti $1, \ldots, i$ della sequenza principale, 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 $1, \ldots, i$ della sequenza principale 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 della sequenza principale) oppure dalla soluzione parziale rappresentata dalla cella (r, i) si può aggiungere:

- una singola sottorete contenente i punti della sequenza principale 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 della sequenza principale 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 della sequenza principale 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 $(r + s, j)$.

[1] Come si vede, l'algoritmo è diverso. L'accenno al problema dello zaino serve da un lato a facilitare per analogia la comprensione dell'algoritmo, dall'altro a istigare gli studenti a scrivere nella relazione bestialità riguardo il problema dello zaino.

- ...

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

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