Algoritmi e Strutture Dati

L'uncinetto di nonna Alberta (`nonna`)

Testo del problema

Slides originali su: judge.science.unitn.it/slides/asd23/prog2.pdf

La leggenda di Nonna A.

Ogni tanto, nelle nostre case, ci imbattiamo in quei delicati lavori all'uncinetto conosciuti come **centrini**. Alcuni sono semplici e lineari, altri incredibilmente intricati, bianchi o dai colori più vari. Sono ovunque, decorano i nostri mobili e aggiungono un tocco di eleganza e tradizione. Ma vi siete mai fermati a pensare da dove provengano? Come fanno a raggiungere le nostre case?
La risposta è avvolta nel mistero e nella magia. Esiste una leggenda, tramandata di bocca in bocca, che parla di un'anziana signora conosciuta come Nonna A. Nessuno sa il suo vero nome, né dove abiti, ma si dice che lei sia la creatrice instancabile di tutti questi centrini. Prima insegnava algoritmi, ora è in pensione ed ogni notte, mentre tutti dormono, Nonna A. lavora al suo uncinetto, creando senza sosta quei piccoli capolavori di cotone.

Una nonna meticolosa

Nonna A. lavora contemporaneamente a N_c centrini utilizzando N_g gomitoli. Ogni gomitolo può contribuire a uno o più centrini, e ogni centrino può essere realizzato con uno o più gomitoli.[1] Ci possono essere dei centrini già completati che non sono collegati a nessun gomitolo e dei gomitoli che non vengono utilizzati per produrre alcun centrino.

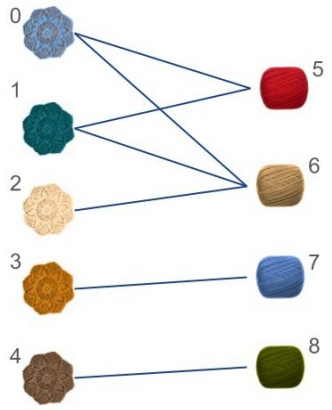
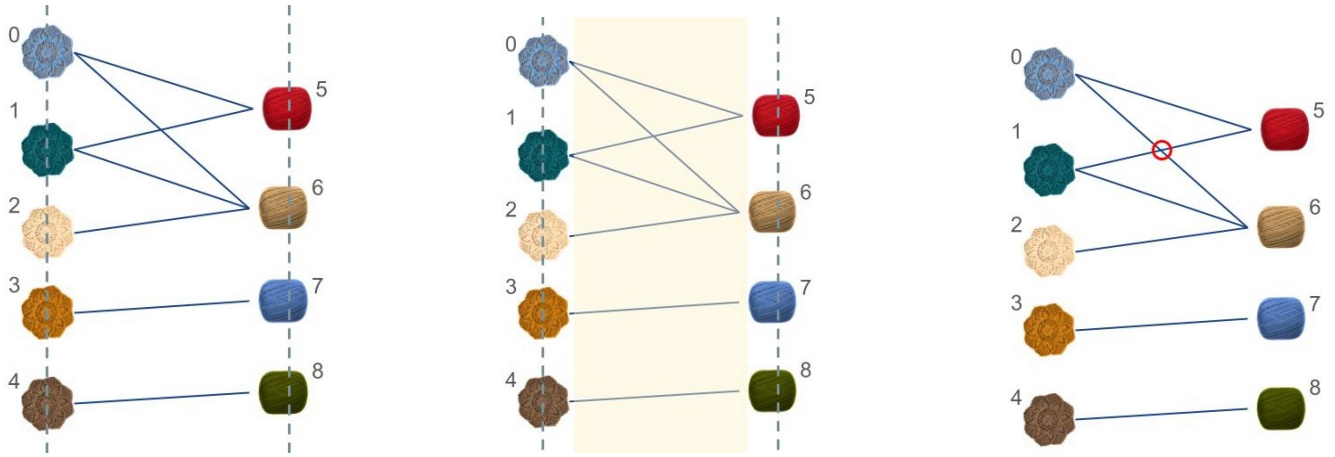


Figura 1: Esempio con 5 centrini (sinistra) e 4 gomitoli (destra), ogni gomitolo è legato da un filo ai centrini in cui viene usato. Se completi, i centrini non sono collegati a nessun gomitolo. Ci possono essere dei gomitoli inutilizzati.

Dovendo gestire questa mole enorme di lavoro e dovendo garantire determinati standard di qualità, Nonna A. non può che dover essere molto organizzata e precisa. Come avete già potuto notare, la nonna ha ordinato sulla sinistra tutti i centrini in corso d'opera (inclusi i centrini completati), e sulla destra tutti i gomitoli (inclusi quelli inutilizzati). Sia centrini che gomitoli sono **ordinati verticalmente su linee rette** (Figura 2a). Inoltre, i fili sono molto tesi **(sono linee rette)** e **passano solo e soltanto dentro l'area delimitata dalle linee verticali** che

[1] Per i più appassionati di uncinetto tra di voi che in questo momento si stanno arrabbiando: sono gomitoli magici.

creano i centrini e i gomitoli. In Figura 2b, quest'area è evidenziata in giallo. Ma non solo! Un occhio più attento potrà notare che la nonna ha posizionato centrini e gomitoli in modo da minimizzare il numero di incroci dei fili! **Un incrocio avviene quando due fili distinti si incontrano in un punto** (escludendo quando toccano gomitoli o centrini). In questo caso, per esempio, il numero di incroci è 1 (Figura 2c).



(a) Centrini e gomitoli sono disposti su due linee rette parallele.

(b) I fili possono incrociarsi sono all'interno dell'area in giallo tra le due rette.

(c) I cerchi rossi evidenziano gli incroci tra i fili.

Figura 2: Esempio di possibile disposizione dei centrini e dei gomitoli.

Lo scherzo dei nipoti

Nonna A. ha dei nipoti abbastanza vivaci e dispettosi. E non conoscono la leggenda di Nonna A. perché non si sono mai interessati ai centrini. Anzi, i centrini li hanno sempre abbastanza annoiati, li trovano passati di moda! Volendo fare uno scherzo a Nonna A., i nipotini entrano nel suo laboratorio, cambiando l'ordine di centrini e gomitoli, che potrebbe non essere più ottimo! Per esempio, ci sono 5 incroci nell'esempio in Figura 3.

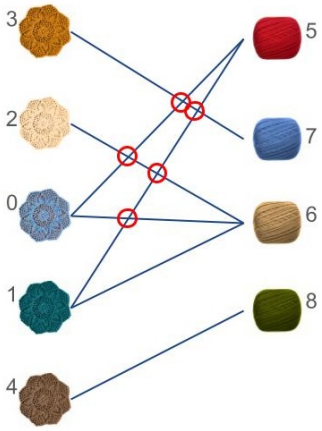


Figura 3: Situazione dopo lo scherzo dei nipotini.

Nella confusione, fanno cadere della marmellata sul pavimento, ed i **gomitoli** sono ora **bloccati per sempre nella loro posizione**. Per fortuna, i **centrini** si sono salvati, ed **è ancora possibile spostarli**! Ovviamente Nonna A.

si è arrabbiata, e vuole impartire una bella lezione ai suoi nipotini. E infatti li obbliga a riordinare tutto! Chissà che questo non li porti ad apprezzare un po' i centrini...

Domanda

Il vostro compito è molto semplice: partendo dalla situazione causata dai nipotini, trovate una strategia che vi permetta di avvicinarvi il più possibile al numero di incroci dell'organizzazione originale della nonna. In altre parole, dovete **minimizzare il numero di incroci** dei fili.
Ricordate che potete riordinare solo i centrini, i gomitoli sono bloccati per sempre nella loro posizione a causa della marmellata!

Esempi

Esempio I

- 5 centrini, 4 gomitoli e 7 fili.
- C'è un solo incrocio di fili, e questo numero non può essere ulteriormente ridotto.
- Un possibile ordinamento dei centrini che ottiene 1 incrocio è quello proposto in Figura 4: **la situazione in input era già ottima** dato l'ordinamento dei gomitoli.

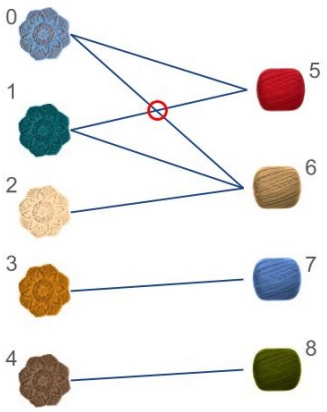


Figura 4: Soluzione per l'Esempio I: l'ordinamento in input è già ottimo.

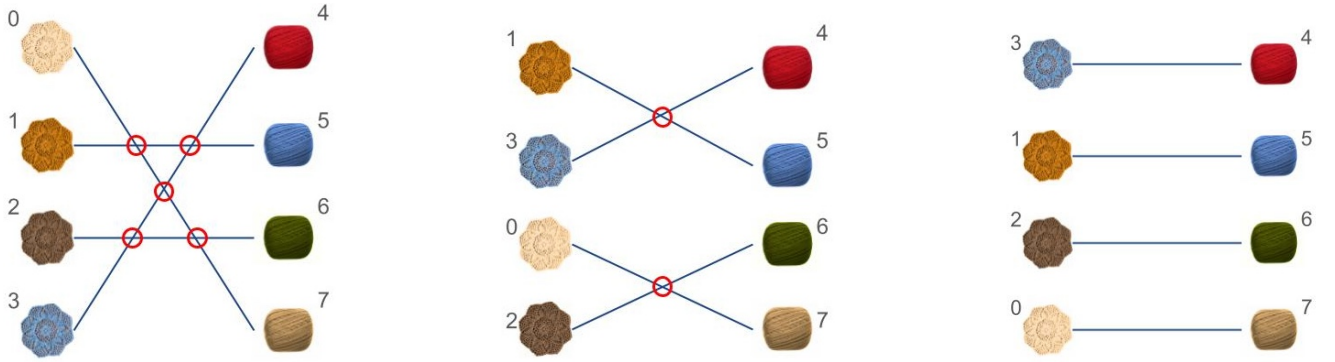
Esempio II

- 4 centrini, 4 gomitoli e 4 fili.
- Ci sono 5 incroci di fili.
- Possibili ordinamenti dei centrini sono quelli in Figura 5.

Input/Output

Input: Un file con $1 + M$ righe.

- La prima riga riporta 3 numeri interi: N_c (`int`), N_g (`int`) e M (`int`), rispettivamente il numero di centrini, il numero di gomitoli e il numero di fili che congiungono gomitoli e centrini.



(a) Situazione iniziale: 5 incroci.

(b) Soluzione intermedia: 2 incroci.

(c) Soluzione ottima: 0 incroci.

Figura 5: Possibili soluzioni per l'Esempio II.

- Le successive M righe descrivono i fili. Ogni riga i riporta 2 interi c_i (`int`) e g_i (`int`), rappresentanti gli identificativi del centrino e del gomitolo collegati dal filo i-esimo.

Questo file descrive la situazione *dopo* che i nipoti abbiano cambiato l'ordine di centrini e gomitoli, ovvero descrive la situazione su cui dovrete agire voi.

Output: Un file con le vostre proposte di soluzione, ogni proposta è composta da 3 righe:

- La prima riga deve riportare il valore di K (`int`) individuato, ovvero il numero di incroci;
- la successiva riga deve contenere N_c valori (`int`), ovvero gli identificativi dei centrini nell'ordine in cui realizzano il numero di incroci K individuato;
- l'ultima riga contiene tre asterischi `***`;

Limiti e assunzioni

- $1 < N_c < 5.000$.
- $1 < N_g < 5.000$.
- $1 < M < 100.000$.
- $0 \leq K \leq 4.999.950.000$.
- I centrini e i gomitoli sono identificati rispettivamente da interi da 0 a $N_c - 1$ (centrini) e da N_c a $N_c + N_g - 1$ (gomitoli). L'indice ne rappresenta anche la rispettiva posizione nell'ordinamento.
- I fili sono rappresentati da coppie (c_i, g_j) dove c_i è un centrino e g_j è un gomitolo. Non esiste un gomitolo g_i collegato ad un altro gomitolo g_j. Non esiste un centrino c_i collegato ad un altro centrino c_j
- Non ci sono fili doppi, ovvero esiste al massimo un filo descritto da una coppia (c, g) di centrini e gomitoli.
- Possono esserci dei centrini e dei gomitoli che non hanno collegamenti.

Casi di test

Ci sono 20 casi di test in totale:

- In 6 casi su 20, i centrini sono già ordinati in modo ottimale;
- Nei restanti casi non ci sono particolari limitazioni.

Limiti di tempo e memoria

I limiti di tempo e memoria sono:

- Tempo limite: 3 secondi (timeout: $3,3$ secondi).
- Memoria massima: 32 MB.

Punteggio

Ci sono 20 casi di test ed ogni caso di test assegna un punteggio massimo di 5 punti per un totale massimo teorico di 100 punti.

- Una soluzione è valida se rispetta tutte le richieste. Soluzioni non valide fanno **zero punti**. Per esempio, in tutti i casi seguenti la soluzione da **zero punti**:
  - il formato di output non è rispettato.
  - il numero dichiarato di incroci (`K`) non corrisponde a quello che si ottiene con l'ordinamento specificato.
  - l'ordinamento dei centrini dichiarato non è una permutazione corretta dei centrini (lunghezza errata, gomitoli ripetuti, ecc.).
- Il punteggio viene calcolato usando l'**ultima soluzione** terminata con tre asterischi `***`.

I parametri di valutazione sono i seguenti:

- `K`: il numero di incroci del vostro ordinamento in output;
- `initK`: il numero di incroci dell'ordinamento in input;
- `minK`: lower bound del minimo numero di incroci ottenibili.

Per ogni caso di test per cui la vostra soluzione fornisce un output valido entro i limiti di tempo e memoria otterrete il seguente punteggio $\mathcal{P}$:

$$\mathcal{P} = 5 \times \begin{cases} 0 & \text{se } \texttt{minK} \leq \texttt{initK} < \texttt{K} \\ 1 & \text{se } \texttt{minK} = \texttt{initK} = \texttt{K} \\ \dfrac{\texttt{initK} - \texttt{K}}{\texttt{initK} - \texttt{minK}} & \text{se } \texttt{minK} \leq \texttt{K} < \texttt{initK} \end{cases}$$

Bound sul numero di incroci

- `minK` è un *lower bound* al numero minimo di incroci ottenibile riordinando i centrini. Potrebbe non essere sembre possibile raggiungere questo bound, e quindi fare 100 punti. Tuttavia, nei casi per la sufficienza, questo bound è sempre raggiungibile.
- `initK` è il numero di incroci della soluzione banale che si ottiene lasciando i centrini nell'ordine in cui vi vengono dati. Sebbene sia possibile "fare di peggio", se la vostra soluzione ha un numero di incroci superiore, riceverete comunque **0 punti**. Di fatto è l'*upper bound* per il caso di test. Se la vostra soluzione ha un numero di incroci uguale a `initK`, riceverete **5 punti** se la soluzione iniziale era già ottima (`initK` = `minK`), oppure **0 punti** se è possibile fare di meglio (`initK` > `minK`).

Nota 1: dato che il numero di incroci è tale che $0 \leq K \leq 4.999.950.000$ potete usare una variabile di tipo `long int` per memorizzarlo.
Nota 2: non è necessario che calcoliate il punteggio della vostra soluzione – vi servirebbe il bound `minK` che non avete. Il vostro obiettivo è in ogni caso di **minimizzare il numero di incroci**.

Esempi (punteggio)

Nell'esempio in Figura 4, l'output

```
1
0 1 2 3 4
***
```

è valido e ottimo. In questo caso `minK` = `initK` = `K` = 1, quindi $\mathcal{P} = 5 \times 1 = 5.0$.
Invece, l'output

```
6
3 2 0 1 4
***
```

avrebbe un numero di incroci maggiore della situazione di partenza. Siamo nella situazione `minK` $\leq$ `initK` $<$ `K`, quindi $\mathcal{P} = 0$.
Consideriamo invece l'esempio in Figura 5, e l'output

```
2
1 3 0 2
***
```

con `minK` = 0, `initK` = 5 e `K` = 2. In questo caso siamo nella situazione `minK` $\leq$ `K` $<$ `initK` e $\mathcal{P} = 5 \times \frac{5-2}{5-0} = 5 \times 0.6 = 3.0$.

Valutazione

Per valutazione del progetto:

- Conta il punteggio dell'**ultimo sorgente** inviato al sistema;
- Il progetto è superato con un punteggio non inferiore a 30 punti;
- C'è un limite di 40 sottoposizioni per gruppo;

Dataset di esempio

Potete trovare un dataset equivalente con cui esercitarvi in locale:
judge.science.unitn.it/slides/asd23/dataset_nonna.zip.
Nota: per gli input forniti nel dataset di esempio non è stata calcolata una soluzione ottima. Nei file di output troverete il lower bound `minK` per ogni caso di test.

Istruzioni di compilazione

Di seguito riportiamo le istruzioni per testare i vostri programmi su vari sistemi. Si suppone che il sorgente con il vostro codice si chiami file `nonna.cpp`. I file `nonna.cpp`, `grader.cpp` e `nonna.h` devono stare nella stessa cartella.

Sistemi GNU/Linux

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -static -s -o nonna nonna.cpp grader.cpp
```

Sistemi Mac OS X

Su sistemi Mac OS X usate il seguente comando di compilazione:

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -o nonna nonna.cpp grader.cpp
```

Se ottente un errore del tipo: `use of undeclared identifier quick_exit`, sostituite in `grader.cpp` l'istruzione `quick_exit(EXIT_SUCCESS);` con `exit(EXIT_SUCCESS);`.

Sistemi Windows

Per il sistema Windows 10 potete installare il "Windows Subsystem for Linux' '[2]. Successivamente potete installare i tool necessari per usare Visual Studio Code[3] o Visual Studio 2017[4] seguendo le relative guide riportate nelle note. Usando questo sistema fate attenzione a dove salvate i file e a quale nome gli date in quanto potreste avere delle difficoltà con percorsi che contengano spazi e caratteri speciali.
In alternativa, o per sistemi precedenti a Windows 10 potete installare *Cygwin*[5], un ambiente completamente POSIX-compatibile per Windows. Anche in questo caso esistono guide per configurare i comuni editor disponibili su Windows di modo che utilizzino l'ambiente Cygwin, come per esempio Visual Studio[6].
Una volta installato Cygwin è possibile simulare quanto avviane su arena compilando il proprio sorgente senza includere l'header `nonna.h` e il grader `grader.cpp`:

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -static -s -o nonna nonna.cpp
```

e lanciare il comando come:

```
timeout.exe 3 ./nonna
```

`timeout.exe` arresterà il programma dopo 3 secondi.

[2] `https://docs.microsoft.com/en-us/windows/wsl/install-win10`
[3] `https://code.visualstudio.com/docs/cpp/config-wsl`
[4] `https://devblogs.microsoft.com/cppblog/targeting-windows-subsystem-for-linux-from-visual-studio/`
[5] `https://www.cygwin.com/`
[6] `https://devblogs.microsoft.com/cppblog/using-mingw-and-cygwin-with-visual-cpp-and-open-folder/`

Esempi di input/output

File `input.txt`	File `output.txt`

`3 5 10`
`0 5`
`0 6`
`0 4`
`0 3`
`1 3`
`1 5`
`2 7`
`2 5`
`2 4`
`2 6`
	`6`
	`1 0 2`
	`***`

File `input.txt`	File `output.txt`

`5 4 7`
`0 5`
`0 8`
`1 6`
`2 7`
`3 8`
`4 5`
`4 8`
	`5`
	`0 1 2 4 3`
	`***`

File `input.txt`	File `output.txt`

`5 6 10`
`3 9`
`0 8`
`0 10`
`3 6`
`3 7`
`4 7`
`1 5`
`1 9`
`2 9`
`0 5`
	`12`
	`1 3 4 0 2`
	`***`

File `input.txt`	File `output.txt`

`4 4 10`
`2 7`
`0 5`
`1 5`
`3 4`
`3 5`
`3 6`
`1 6`
`1 7`
`2 6`
`0 4`
	`3`
	`0 3 1 2`
	`***`