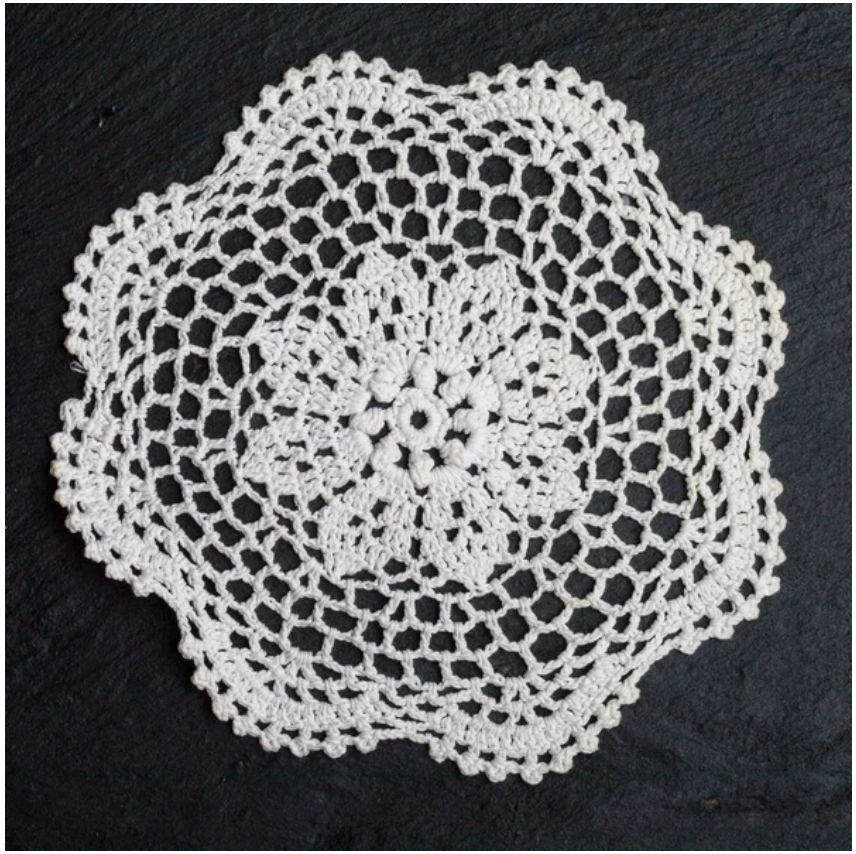
Secondo Progetto ASD 2023/2024

Secondo Progetto ASD 2023/2024

I "centrini della nonna"

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?



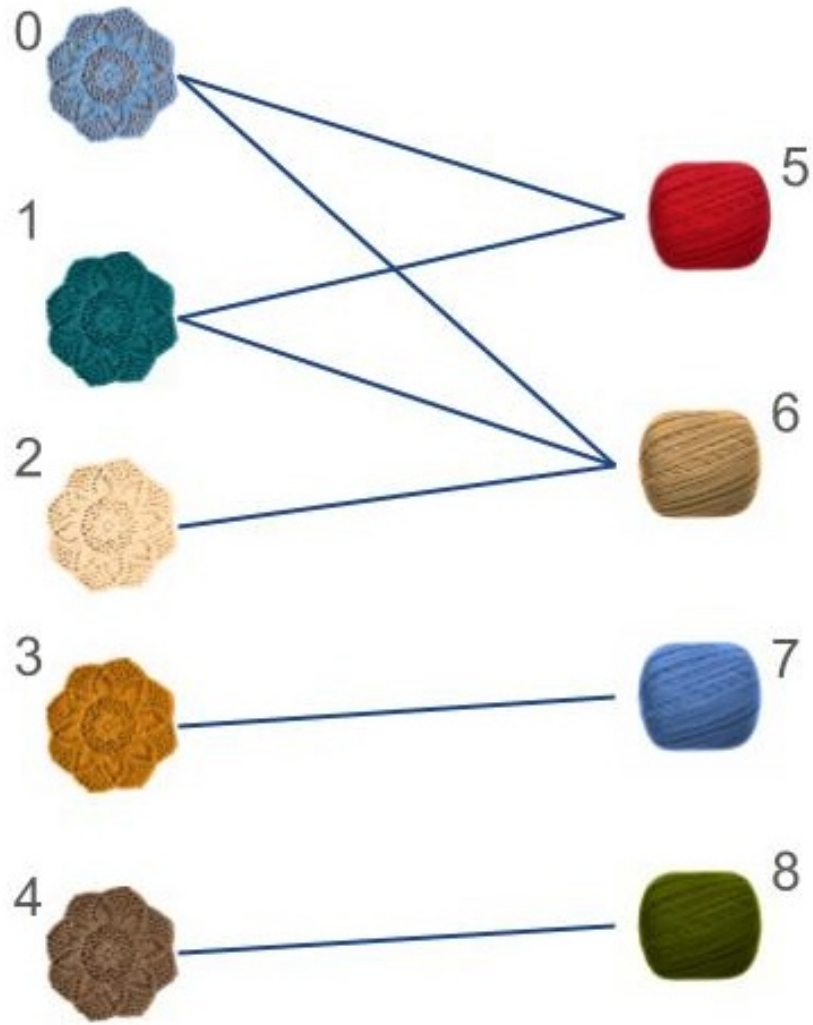
La leggenda di Nonna A.

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.[a] 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.

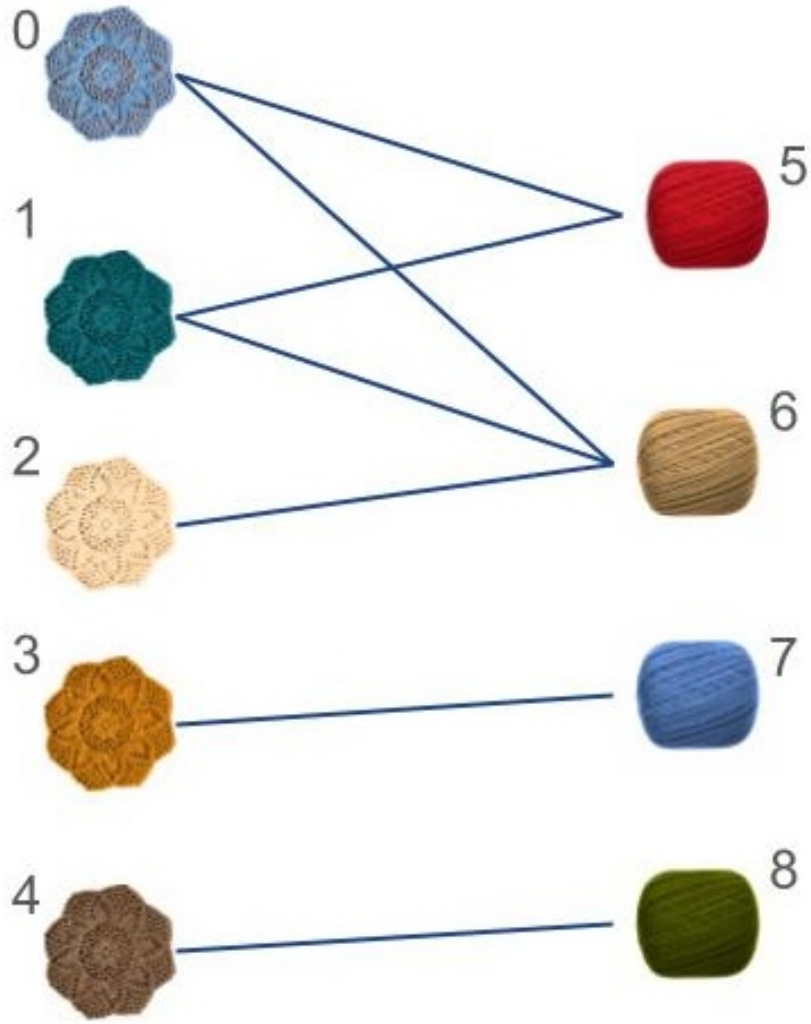


[a] Per i più appassionati di uncinetto tra di voi che in questo momento si stanno arrabbiando: sono gomitoli magici.

Una nonna meticolosa

Qui abbiamo 5 centrini e 4 gomitoli, ogni gomitolo è legato da un filo ai centrini in cui viene usato.

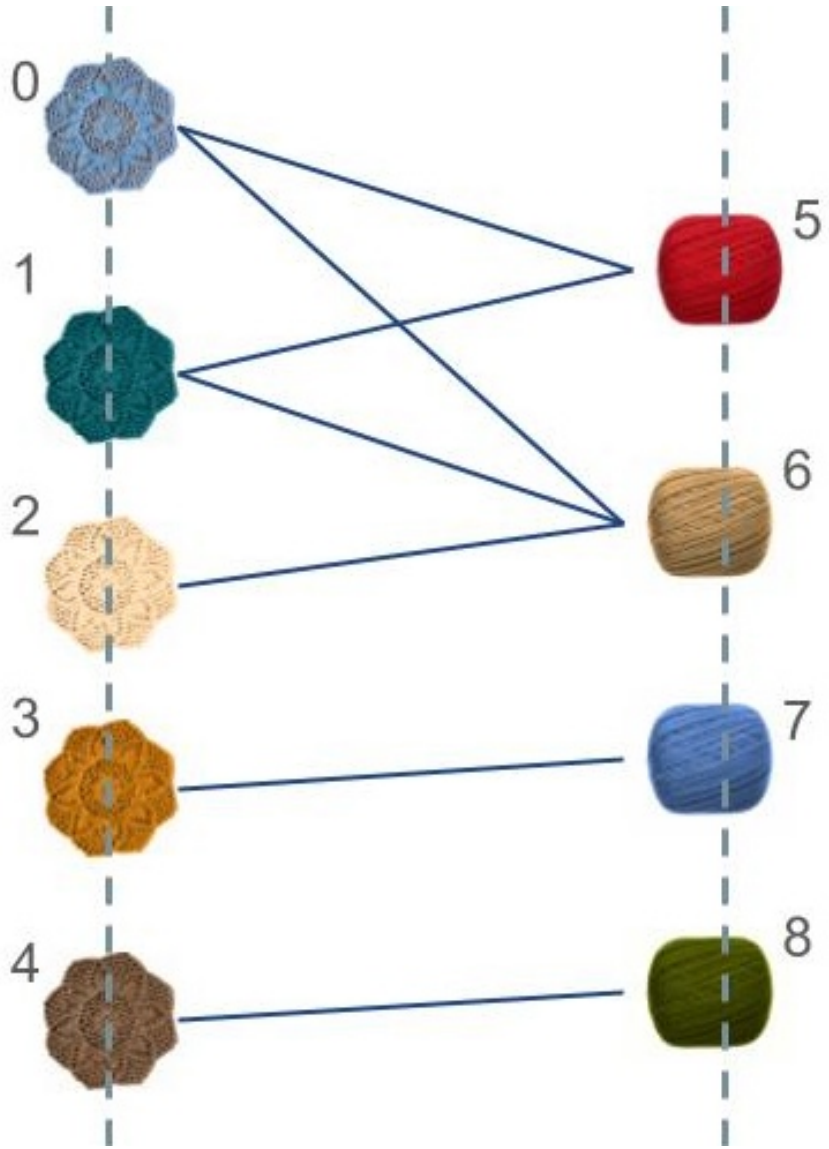
Dovendo gestire questa mole enorme di lavoro e dovendo garantire determinati standard di qualità, Nonna A. non può che dover essere molto organizzata e precisa.



Una nonna meticolosa

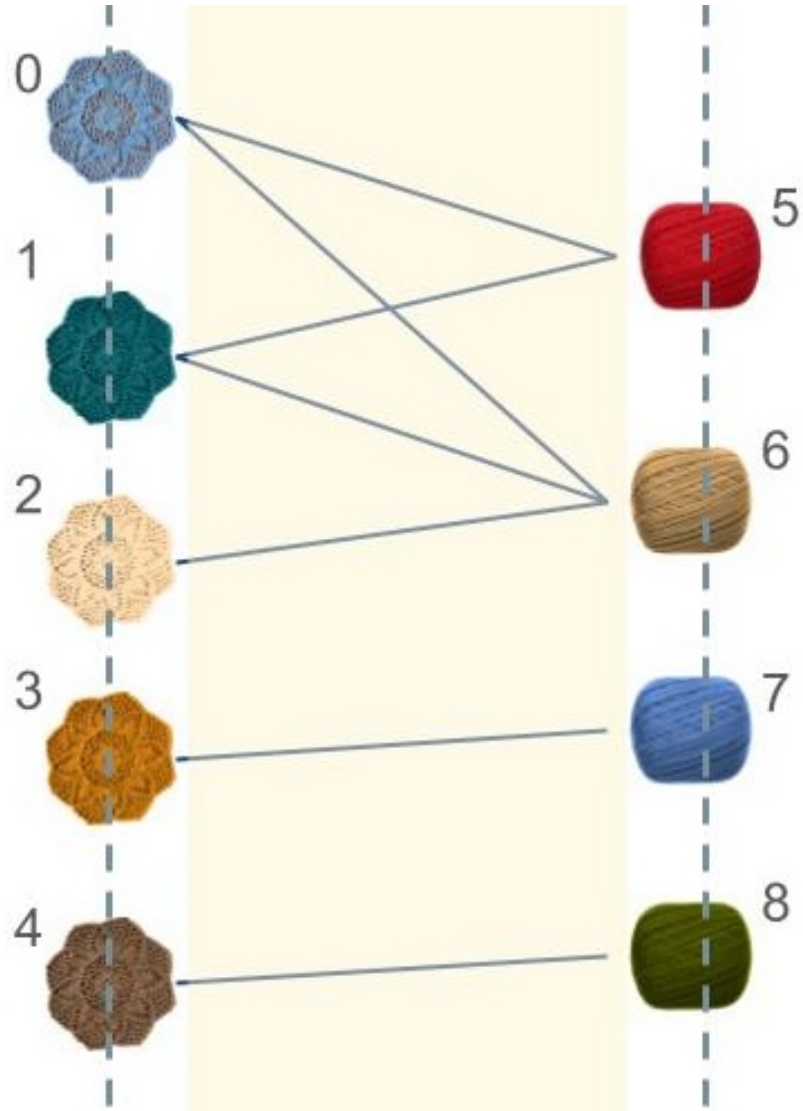
Come avete già potuto notare, la nonna ha ordinato sulla sinistra tutti i centrini in corso d'opera (inclusi quelli completati), e sulla destra tutti i gomitoli (inclusi quelli inutilizzati).

Sia centrini che gomitoli sono **ordinati verticalmente su linee rette**.



UNA NONNA METICOLOSA

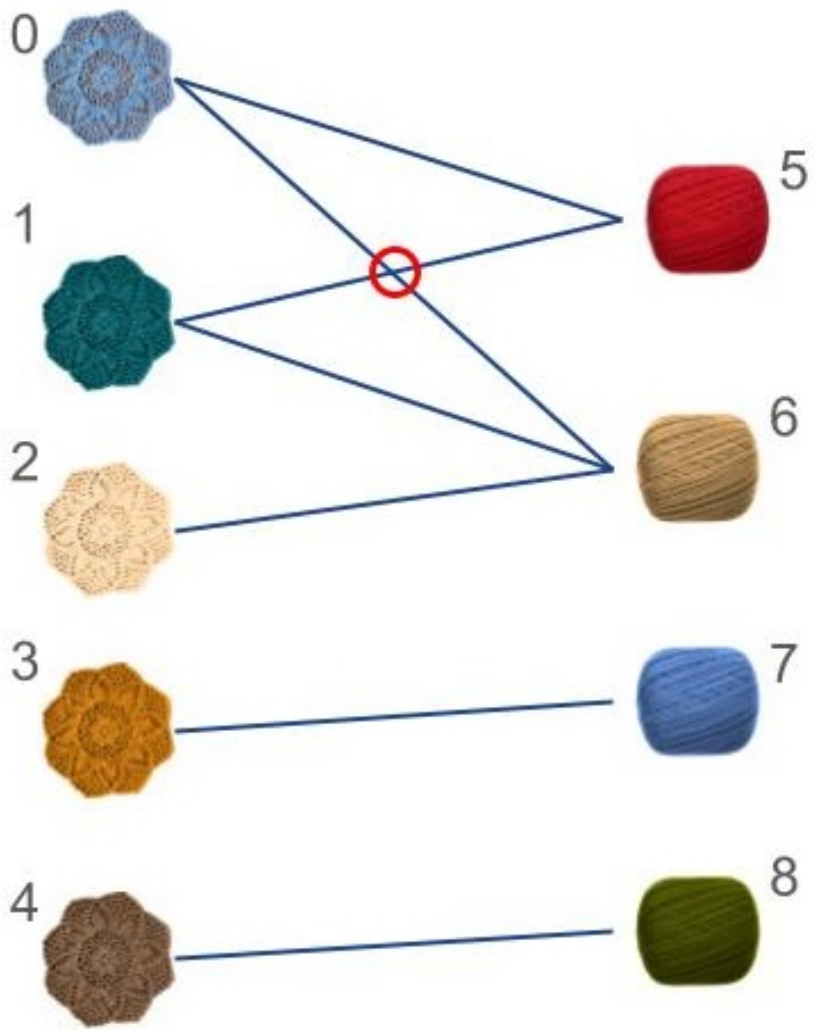
Inoltre, i fili sono molto tesi **(sono linee rette)** e **passano solo e soltanto dentro l'area delimitata dalle linee verticali** che creano i centrini e i gomitoli. Nella figura, quest'area è evidenziata in giallo.



Una nonna meticolosa

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.



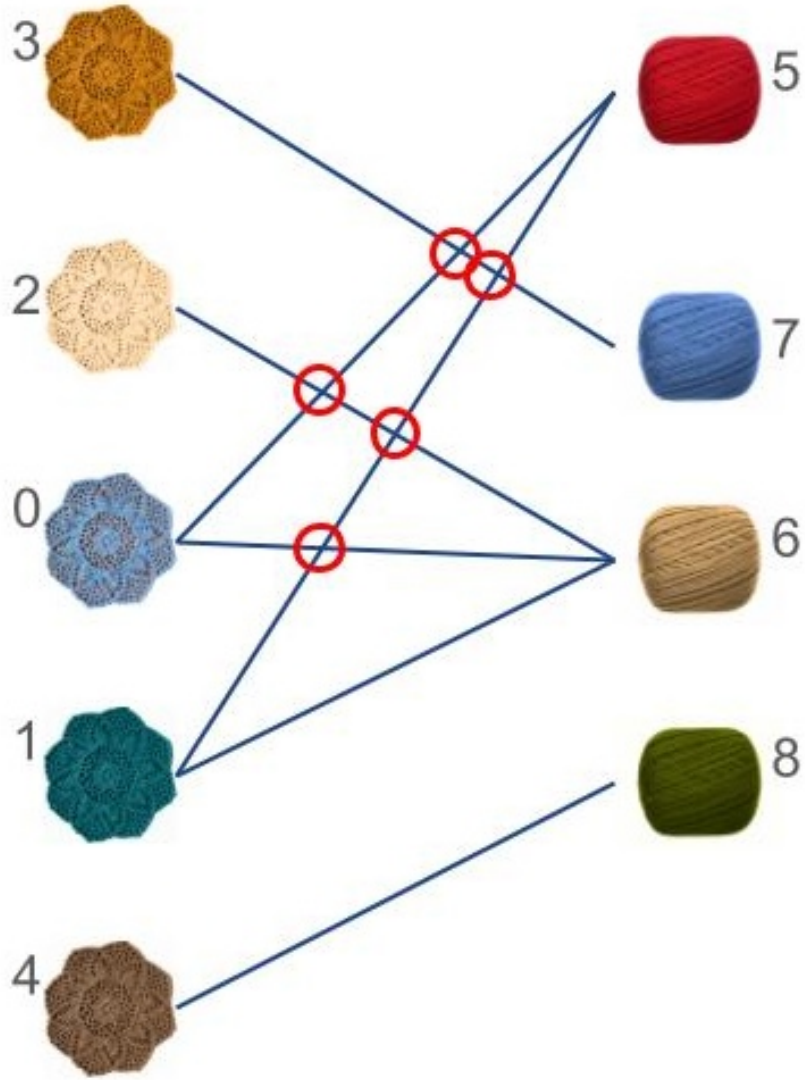
I NIPOTI

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!

Lo scherzone

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! (5 incroci nell'esempio)

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**!



Uno scherzo finito male

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...

E ora? Il vostro compito

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!

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.
- 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.

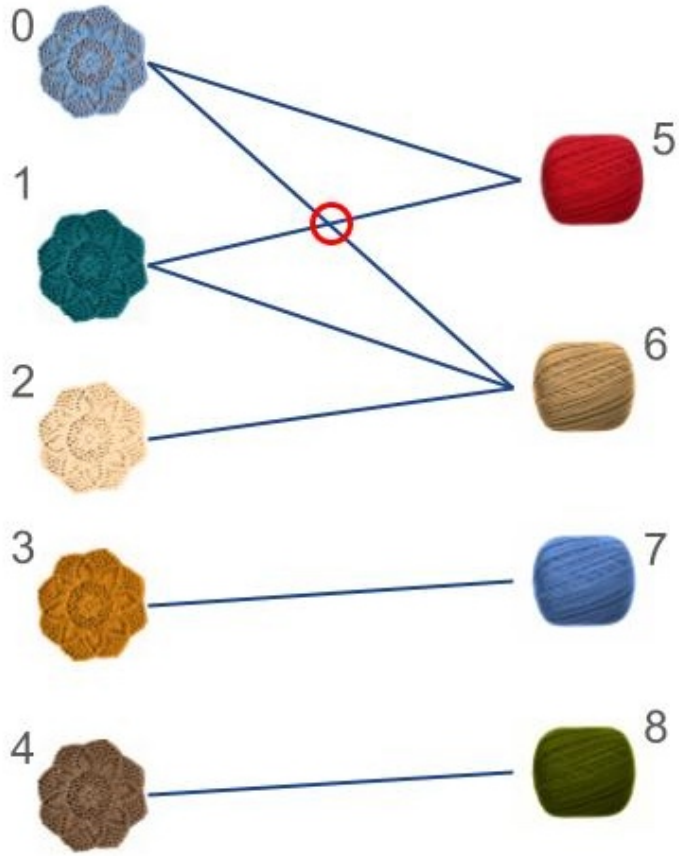
Output

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 `***`;

Esempio I - input ottimo

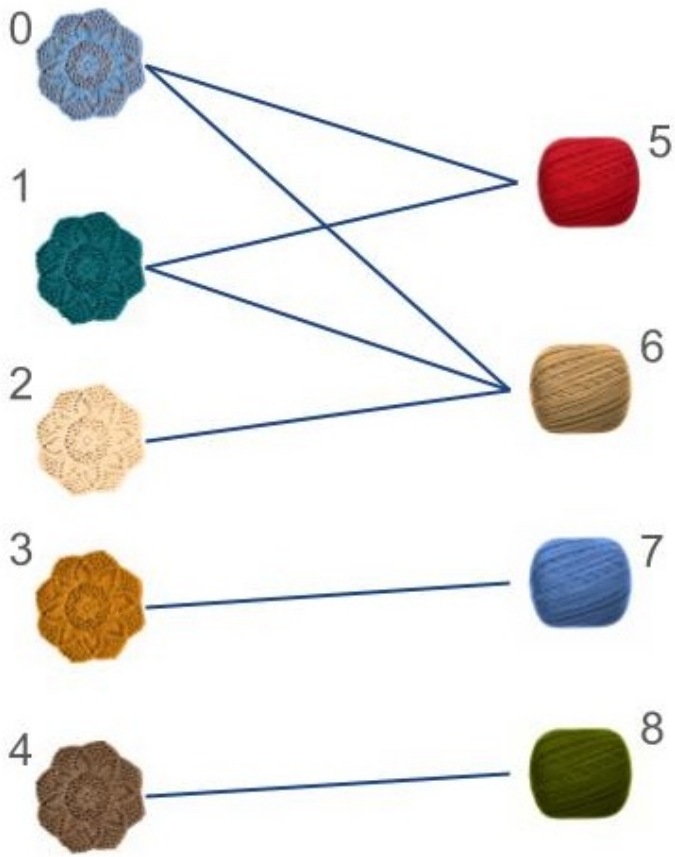
- 5 centrini, 4 gomitoli e 7 fili.
- C'è 1 incrocio di fili, e questo numero non può essere ulteriormente ridotto.
- Un possibile ordinamento dei centrini che ottiene 1 incrocio è quello proposto in figura: **la situazione in input era già ottima** dato l'ordinamento dei gomitoli.



Esempio I - input

Nell'esempio precedente, l'input che trovereste sarebbe il seguente:

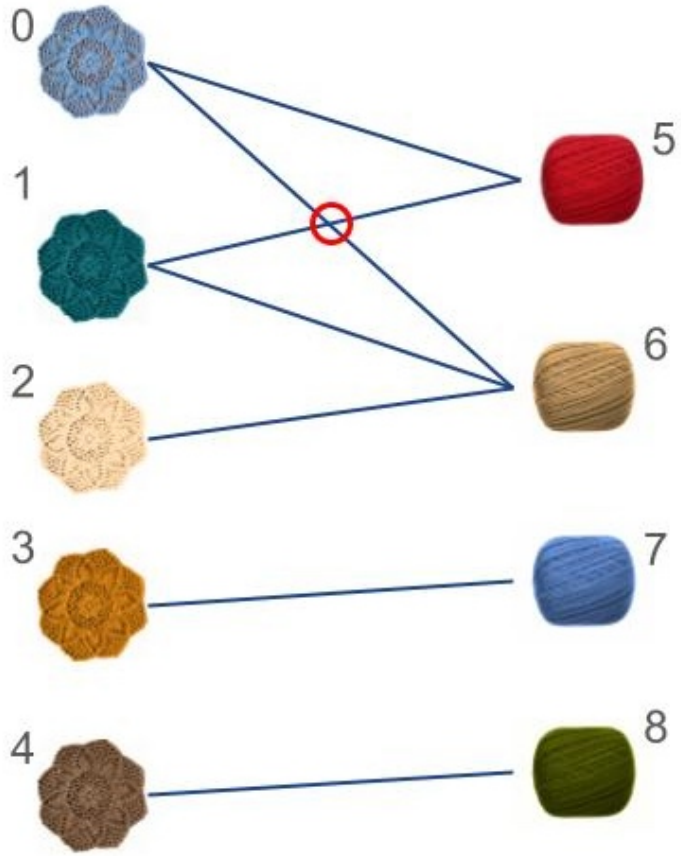
```
5 4 7
0 5
0 6
1 5
1 6
2 6
3 7
4 8
```



Esempio I - output valido e ottimo

Nell'esempio precedente, un output valido (mantiene banalmente l'ordine dei centrini dato in input) sarebbe il seguente:

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



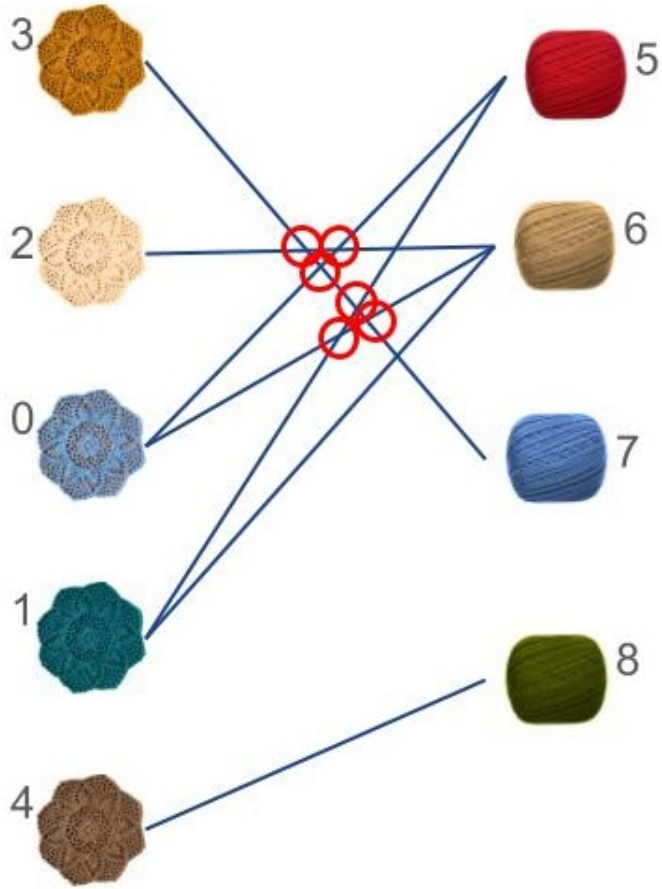
Che come abbiamo visto è anche ottimo.

Esempio I - output valido ma subottimo

Un altro output valido sarebbe il seguente:

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

Che è peggiore del precedente (e peggiore dell'ordine in input), ma comunque valido.

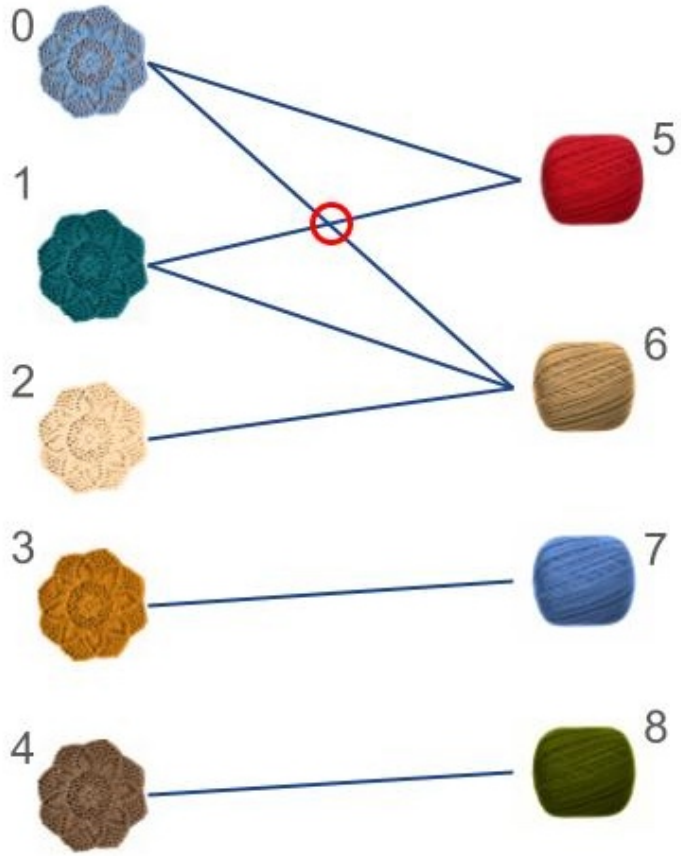


Esempio I - output errato

Nell'esempio precedente, un **output errato** sarebbe il seguente:

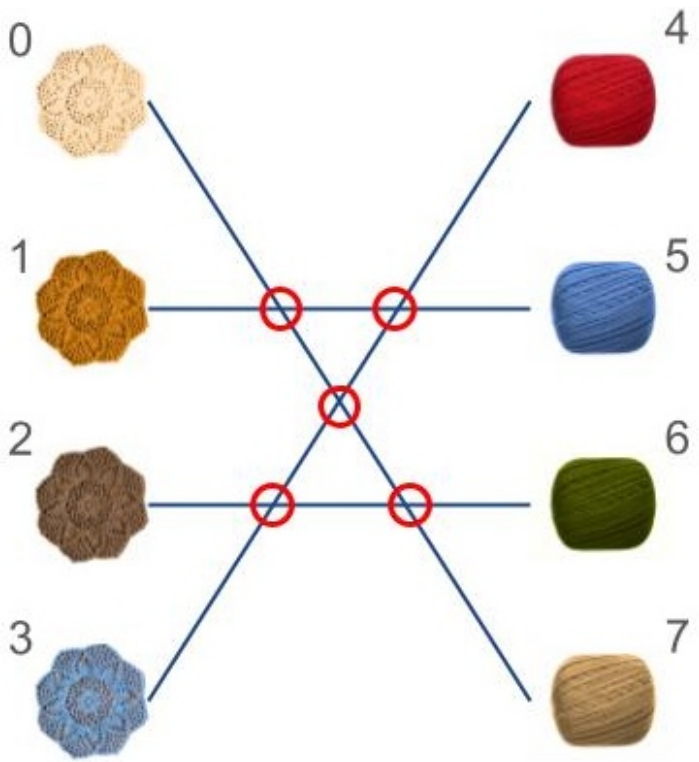
```
0
0 1 2 3 4
***
```

Infatti, l'ordine dei centrini sulla seconda riga prevede 1 incrocio.



Esempio II

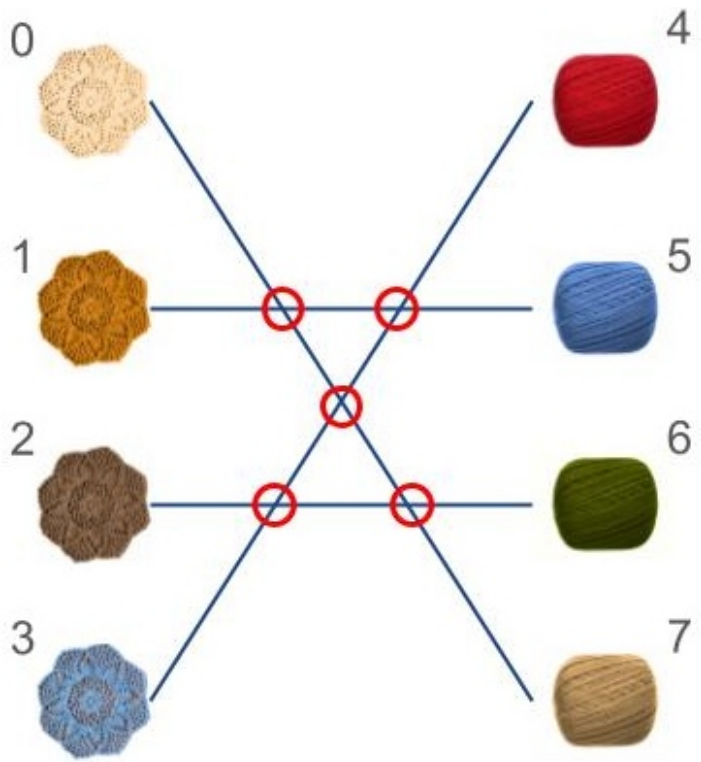
- 4 centrini, 4 gomitoli e 4 fili.
- Analizziamo i nodi ordinati secondo il loro ID.
- Secondo questo ordine, ci sono 5 incroci di fili.



Esempio II - input

L'input che trovereste sarebbe il seguente:

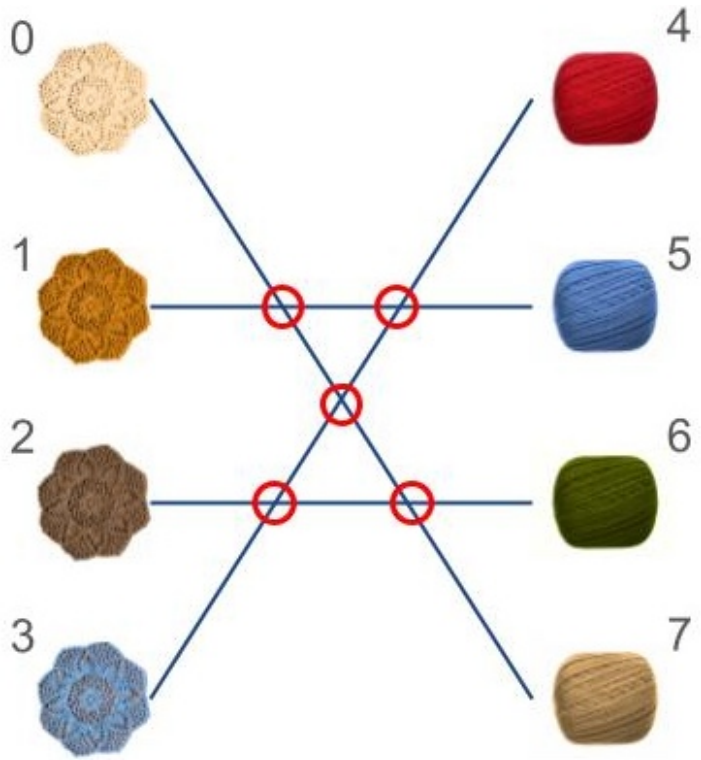
```
4 4 4
0 7
1 5
2 6
3 4
```



Esempio II - output subottimo

Se voleste dare in output questo ordinamento, l'output corretto sarebbe il seguente:

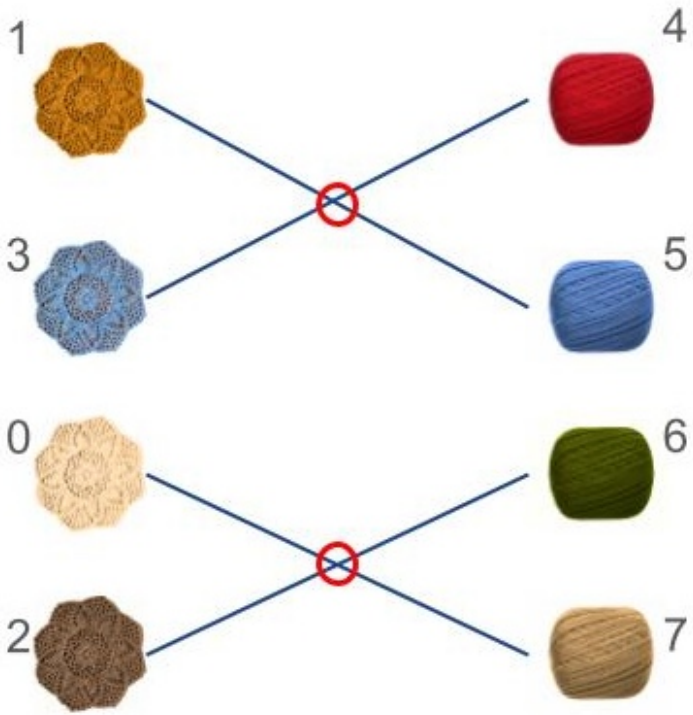
```
5
0 1 2 3
***
```



Esempio II - output subottimo

Si potrebbe fare un po' meglio di così. Cambiando ordinamento dei centrini, riusciamo ad ottenere anche 2 incroci. L'output sarebbe il seguente:

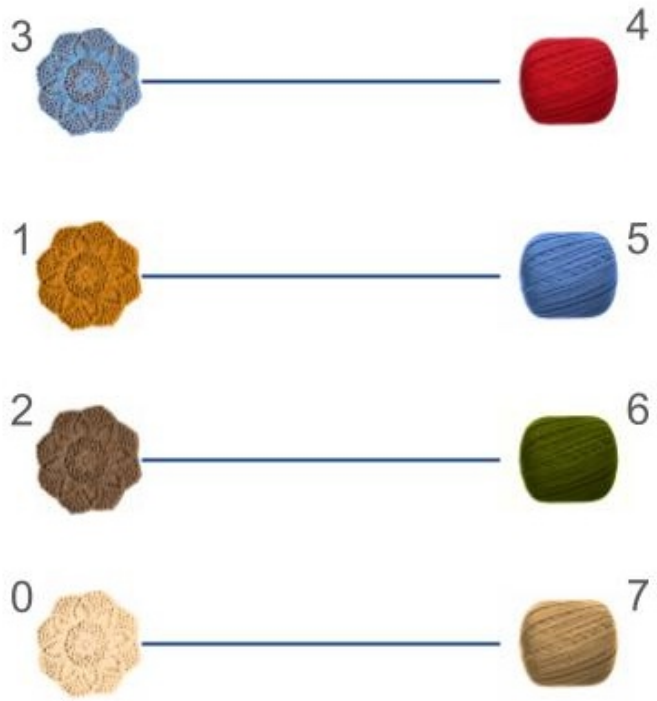
```
2
1 3 0 2
***
```



Esempio II - output ottimo

Infine, possiamo provare ancora un altro ordinamento dei centrini, ottenendo addirittura 0 incroci. L'output valido sarebbe il seguente:

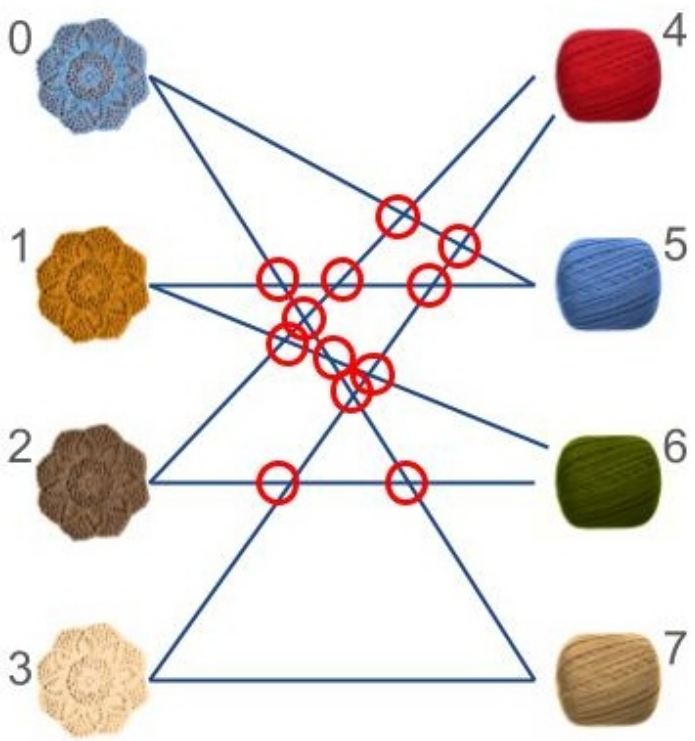
```
0
3 1 2 0
***
```



Ovviamente non si può fare meglio di così.

Esempio III

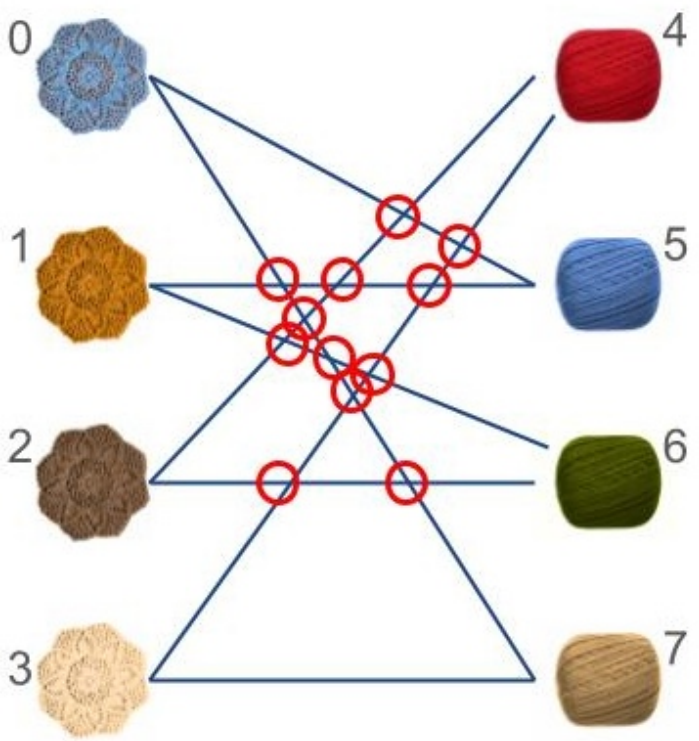
- Altro esempio: 4 centrini, 4 gomitoli e 8 fili.
- Analizziamo i nodi ordinati secondo il loro ID.
- Ci sono ben 12 incroci di fili, ordinando i centrini in questo modo.



Esempio III - output subottimo

Nell'esempio precedente, un output valido sarebbe il seguente:

```
12
0 1 2 3
***
```

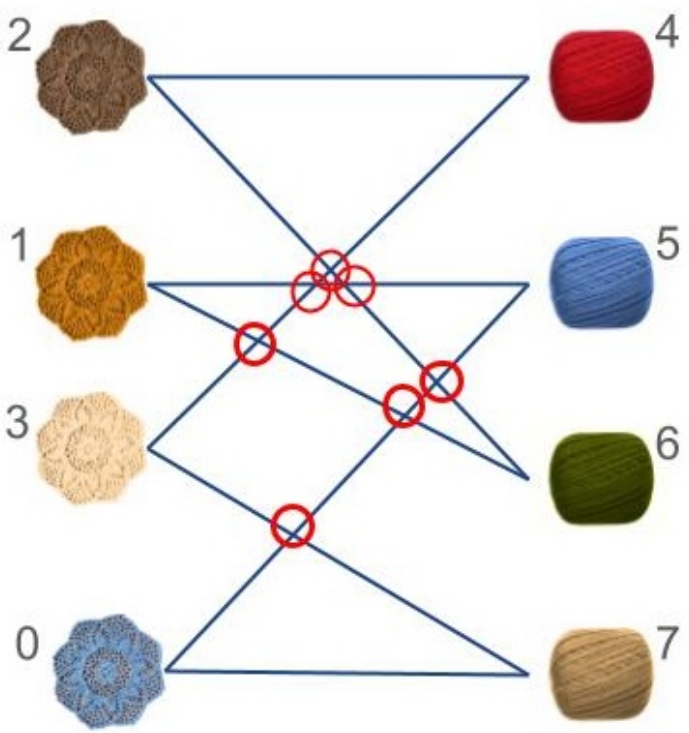


Non benissimo, ma comunque valido.

Esempio III - output ottimo

L'output migliore possibile sarebbe il seguente:

```
7
2 1 3 0
***
```

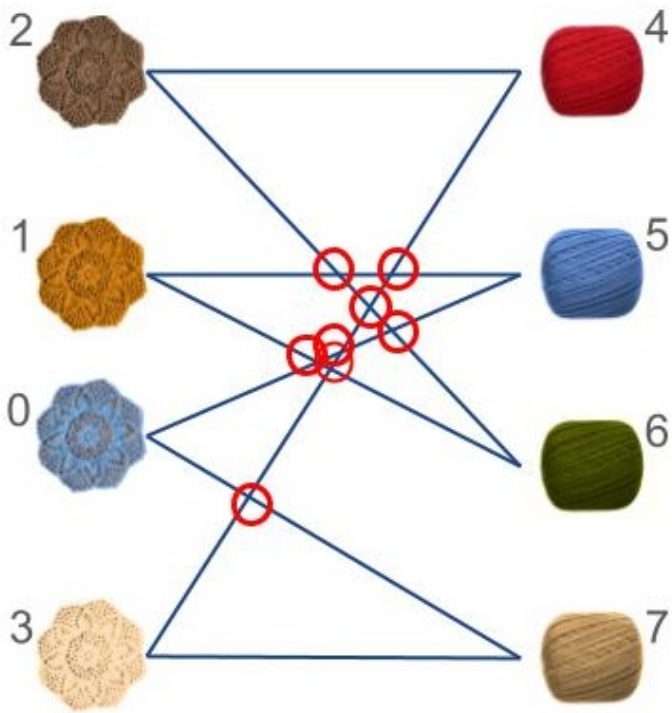


Avendo pochi centrini e gomitoli coinvolti, possiamo calcolare la soluzione ottima.

Esempio III - output subottimo

Un altro output possibile sarebbe il seguente:

```
8
2 1 0 3
***
```

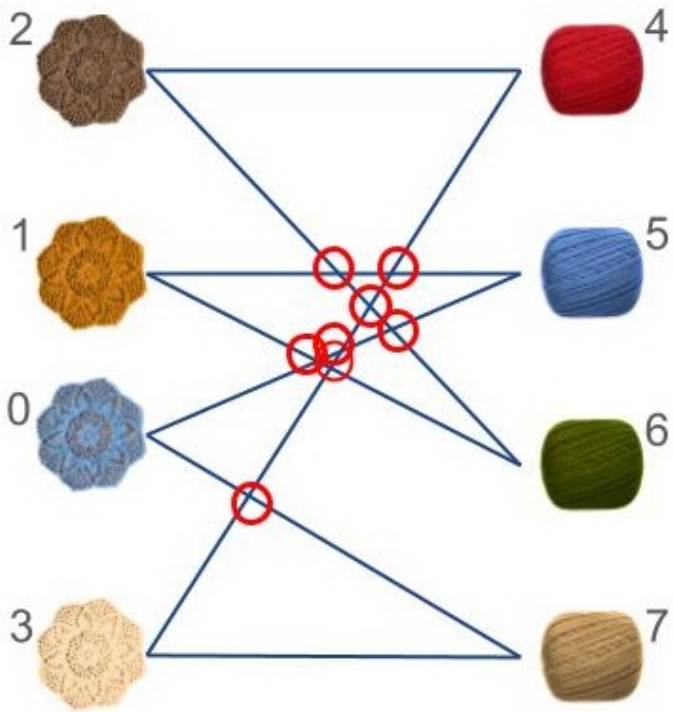


Soluzione non ottimale, ma comunque valida e vicina a quella ottima.

Esempio III - output non valido

Ovviamente un **output non valido** sarebbe il seguente:

```
7
2 1 0 3
***
```



Perché il numero di incroci non corrisponde a quelli ottenuti con questo ordinamento, ovvero 8 come abbiamo visto prima.

Assunzioni generali (I)

- $1 < N_c < 5\,000$.
- $1 < N_g < 5\,000$.
- $1 < M < 100\,000$.
- $1 \leq K \leq 4\,999\,950\,000$.

Assunzioni generali (II)

- I centrini e i gomitoli sono identificati rispettivamente da interi da 0 a $N_c - 1$ (centrini) e da 0 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 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;
- ☆☆ Nel restante dei casi non ci sono particolari limitazioni.

Per la sufficienza si possono risolvere i 6 casi di difficoltà ☆★.

Limiti di tempo e memoria

I limiti di tempo e memoria sono:

- Tempo limite massimo: 3 secondi (timeout: 3.3 secondi).
- Memoria massima: 32 MB.

$\Rightarrow$ Limite di 50 sottoposizioni per gruppo.

$\Rightarrow$ Potete provare con un dataset equivalente sulla vostra macchina

$\Rightarrow$ **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.

Output

Potete stampare le soluzioni in maniera incrementale:

- Importate `nonna.h` (scaricabile da judge).
- Man mano che migliorate la soluzione, scrivetela in output terminando la riga con `***`.
- La libreria arresterà il programma prima del timeout.

```
... include delle librerie di sistema ...
#include "nonna.h"
int main() {
  ...
}
```

Note:

- Il `main` va sempre dichiarato come `int main()` o `int main(void)`.
- Il correttore considererà l'**ultima soluzione** terminata da `***` quindi, anche se non stampate soluzioni multiple, terminate l'output con `***`.

Compilare in locale

Per testare le vostre soluzioni in locale (supponiamo che il vostro file si chiami `nonna.cpp`):

- Scaricate `grader.cpp`
- Il comando di compilazione è il seguente

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

I file `nonna.cpp`, `grader.cpp` e `nonna.h` devono essere nella stessa cartella.

Per sistemi Mac OS X e Windows vedere la nota nel testo.

Nota

Per questo esercizio è necessario usare `C++`, non è possibile usare `C`.

Punteggio

Ogni caso di test vale 5 punti. Il punteggio massimo è di 100 punti.

I parametri di valutazione sono i seguenti:

- `initK`: il numero di incroci dell'ordine in input;
- `K`: il numero di incroci del vostro ordine in output;
- `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:

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

Bounds

Lower bound

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**.

Upper bound

`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. Se la vostra soluzione ha un numero di incroci uguale a `initK`, riceverete 5 punti se la soluzione iniziale era già ottima ($\texttt{initK} = \texttt{minK}$), oppure 0 punti se è possibile fare di meglio ($\texttt{initK} > \texttt{minK}$).

Bounds

Numero di incroci

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.

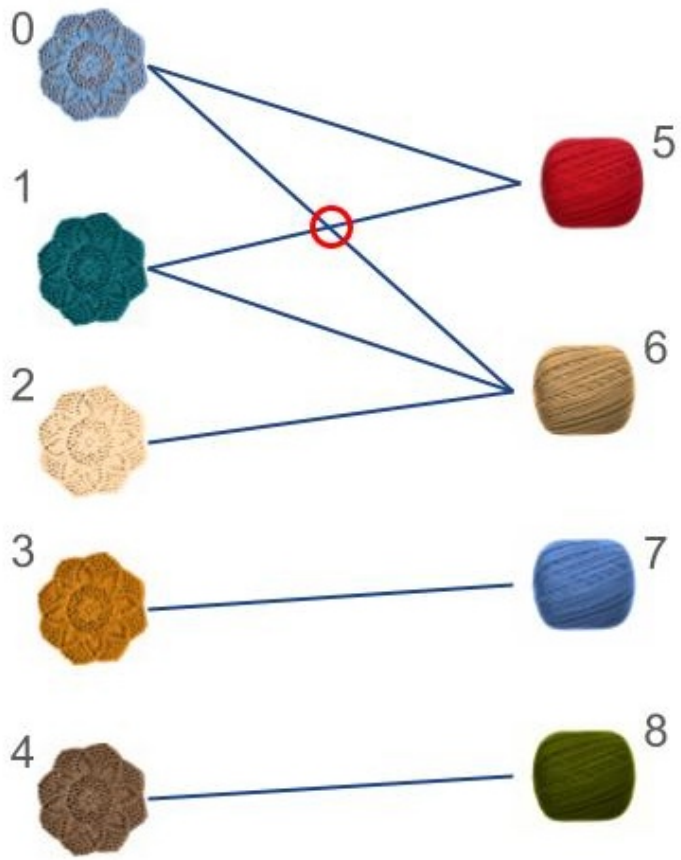
Calcolo del punteggio

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**.

Esempio calcolo punteggio I

In questo esempio avevamo come output:

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

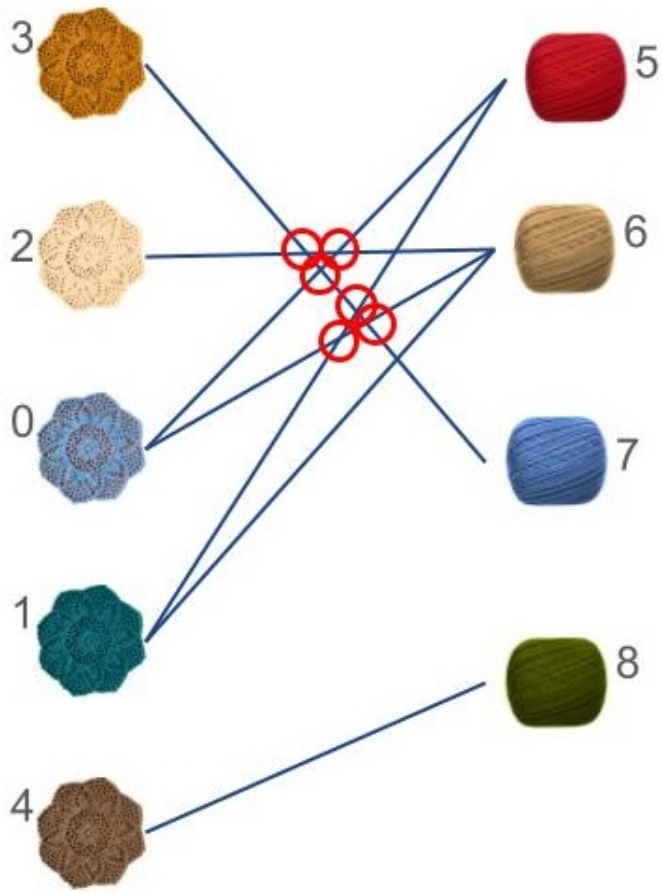


Che come abbiamo visto è valido e ottimo.
In questo caso
`minK` $=$ `initK` $=$ `K` $= 1$, quindi
$\mathcal{P} = 5 \times 1 = 5$

Esempio calcolo punteggio II

In questo output dallo stesso esempio precedente:

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

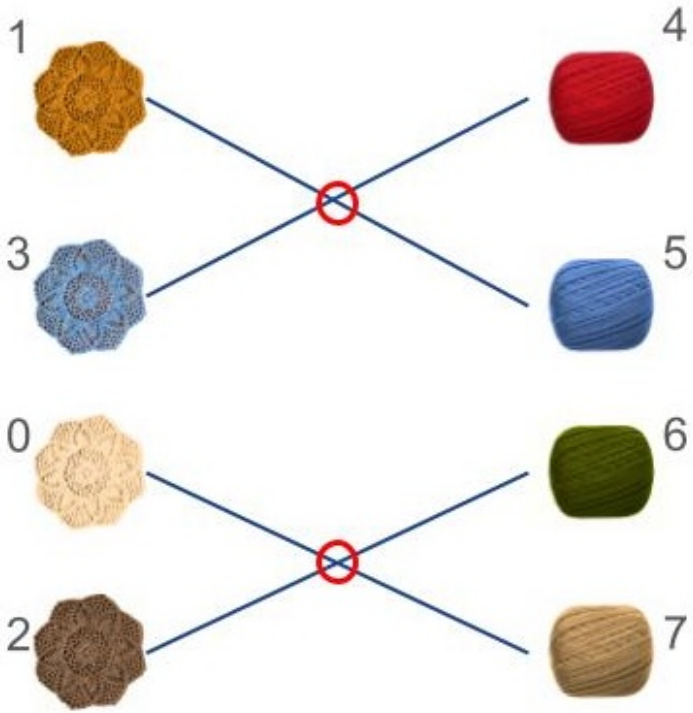


avevamo un numero di incroci maggiore della situazione di partenza. Siamo nella situazione $\texttt{minK} \leq \texttt{initK} < \texttt{K}$ dove $\texttt{minK} = \texttt{initK} = 1$ e $\texttt{K} = 6$, quindi $\mathcal{P} = 0$.

Esempio calcolo punteggio III

Qui abbiamo 5 incroci secondo l'ordine in input, mentre il valore ottimo è 0. Calcoliamo il punteggio per questa soluzione subottima, che è meglio dell'ordine in input:

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



qui abbiamo $\texttt{minK} = 0$, $\texttt{initK} = 5$ e $\texttt{K} = 2$. In questo caso siamo nella situazione $\texttt{minK} \leq \texttt{K} < \texttt{initK}$ e $\mathcal{P} = \frac{5-2}{5-0} \times 5 = 0.6 \times 5 = 3$.

Punteggio

⇒ La **sufficienza è posta a 30 punti**.

✗ se una qualsiasi informazione data in output non è corretta (ad esempio il numero di incroci dei fili dato il vostro ordinamento dei centrini), si ottengono 0 punti.

✗ se una vostra soluzione dovesse avere un numero di incroci superiore ad `initK`, verranno assegnati 0 punti per quel caso di test.

Punti bonus per l'esame

L'assegnazione punti avviene in maniera competitiva:

- **3 punti** ai gruppi nel primo terzile della classica (primo terzo della classifica);
- **2 punti** ai gruppi nel secondo terzile della classifica (secondo terzo della classifica);
- **1 punto** ai gruppi nel terzo terzile della classifica (ultimo terzo della classifica).

Vengono considerati nella classifica per l'assegnazione dei punti solamente i **gruppi che raggiungono la sufficienza** (punteggio maggiore o uguale a 30).

⇒ Classifica:
https://judge.science.unitn.it/arena/ranking/

Consegna

Consegna: venerdì 31 maggio 2024 ore 18:00

Per caricare il vostro codice, recatevi su
`https://judge.science.unitn.it/arena/`

Ricordiamo che nel calcolo del punteggio verrà considerata **l'ultima** soluzione consegnata.

Suggerimenti

Cominciate subito a lavorare al progetto per presentarvi al prossimo ricevimento (martedì 28 maggio) con tutte le domande che vorrete fare.
In ogni caso, sappiate che:

- potete venire a ricevimento
- rispondiamo su Telegram
- risponderemo alle vostre mail

Do's

È permesso:

- Discutere all'interno del gruppo
- Chiedere chiarimenti sul testo
- Chiedere opinioni su soluzioni
- Sfruttare codice fornito nei laboratori
- Utilizzare pseudocodice da libri o Wikipedia
- Richiedere aiuto (anche pesante) per la soluzione "minima"
- Venire a ricevimento
- Mandare meme (algoritmici) sul nostro canale Telegram: it's your time to shine!

Don'ts

È vietato:

- Discutere con altri gruppi
- Mettere il proprio codice su repository pubblici
- Utilizzare codice scritto da altri
- Condividere codice (abbiamo potenti mezzi!)
- Chiedere suggerimenti online (es: stackoverflow)
- Fare riferimenti a soluzioni del progetto nei meme

Ricevimento

Date e orari

- martedì 28 maggio 2024 dalle 13:30 alle 15:30 (A101);
- giovedì 30 maggio 2024 dalle 11:30 alle 15:30 (A101);
- ci saranno ulteriori ore di ricevimento online nei giorni in cui non c'è laboratorio in presenza, vi faremo sapere.

⇒ Negli orari di ricevimento saremo a disposizione sempre online, quando avrete bisogno di un aiuto scrivetelo sul gruppo telegram e risponderemo in chat privata o tramite una chiamata zoom.

⇒ Per qualsiasi domanda mandateci una mail a: `asd.disi@unitn.it` oppure contattateci su Telegram.