Via Aemilia (`aemilia`)

«Il famoso storico romano Tito Livio, nel suo libro *Ab Urbe Condita*, racconta che i Romani decisero di costruire una strada, chiamata *Via Aemilia*, che collegasse le città di *Ariminum* (l'odierna Rimini) a *Placentia* (l'odierna Piacenza), passando per *Regium Lepidi* (l'odierna Reggio Emilia)», spiega l'archeologo Indiana Julio ai suoi studenti del corso di Storiografia 1.

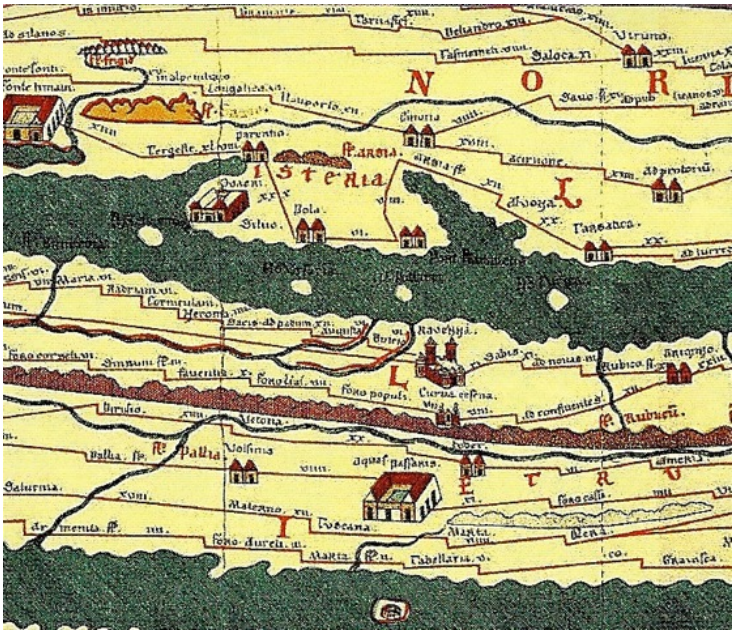


Figura 1: La *Via Aemilia* come appare nella *Tabula Peutingeriana*.

Tra *Ariminum* e *Placentia* ci sono esattamente N città, disposte lungo una retta e convenientemente numerate da 0 a $N-1$.

«La strada, commissionata dal console Cairo Massimo Flaminio, fu costruita in N anni», continua Julio:

- Nell'anno 0 fu scelta una città iniziale da cui far partire la strada.
- In ognuno degli $N-1$ anni seguenti la strada venne estesa di una singola città contigua alla strada già costruita, o verso sinistra oppure verso destra.

«Recentemente è stato ritrovato un antico documento contenente un array di interi T», prosegue Julio, «L'array T, indicizzato da 0 a $N-1$, riporta l'anno T_i entro cui la città i doveva essere collegata alla strada». Indiana Julio però sospetta che possa trattarsi di un falso!

Aiuta Indiana Julio a verificare la correttezza del documento: è possibile che i Romani abbiano costruito la strada rispettando i vincoli posti dall'array T? E se sì, in quale ordine sarebbero state collegate le città? In caso di soluzioni multiple, è sufficiente riportarne una qualsiasi.

✍ Questo problema ammette un **punteggio parziale** pari al 50% dei punti in caso si riconosca correttamente se la strada sia costruibile, ma venga riportato un ordine di costruzione errato.

Implementazione

Dovrai sottoporre un unico file, con estensione `.cpp`.

☞ Tra gli allegati a questo task troverai un template `aemilia.cpp` con un esempio di implementazione.

Dovrai implementare la seguente funzione:

C++
```
vector<int> verifica(int N, vector<int> T);
```

- L'intero N rappresenta il numero di città.
- L'array T, indicizzato da 0 a $N-1$, contiene l'anno entro cui vanno collegate le città.
- La funzione deve restituire:
  - un array vuoto se non è possibile costruire la strada rispettando i vincoli; altrimenti
  - un array P, indicizzato da 0 a $N-1$, tale che P_i è la città collegata nell'anno i.

Grader di prova

Negli allegati di questo problema è presente una versione semplificata del grader usato durante la correzione, che puoi usare per testare le tue soluzioni in locale.

Il grader allegato legge i dati da `stdin`, chiama la funzione `verifica` e scrive su `stdout`, secondo il seguente formato.

L'input è composto da 2 righe, contenenti:

- Riga 1: l'intero N.
- Riga 2: N interi $T_0, \ \ldots, \ T_{N-1}$.

L'output è composto da 2 righe, contenenti:

- Riga 1: la lunghezza L dell'array P restituito dalla funzione `verifica`.
- Riga 2: L interi $P_0, \ldots, P_{L-1}$.

Assunzioni

- $1 \leq N \leq 200\,000$.
- $0 \leq T_i < N$ per $i = 0, \ldots, N-1$.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Il punteggio relativo ad un subtask è uguale al peggiore dei punteggi ottenuti su uno dei suoi test case, moltiplicato per il valore del subtask.

- **Subtask 1** [**0 punti**]: Casi d'esempio.
- **Subtask 2** [**12 punti**]: $T_0 = 0$.
- **Subtask 3** [**22 punti**]: I valori T_i sono tutti distinti ($T_i \neq T_j$ per ogni $i \neq j$).
- **Subtask 4** [**12 punti**]: $N \leq 10$.
- **Subtask 5** [**14 punti**]: $N \leq 100$.
- **Subtask 6** [**14 punti**]: $N \leq 2000$.
- **Subtask 7** [**26 punti**]: Nessuna limitazione aggiuntiva.

In ogni subtask, otterai il 50% dei punti se determini solo se è possibile o meno costruire la strada, riportando un ordine di costruzione non valido. In particolare, i punti sono assegnati ad ogni test case come dettagliato nel seguito.

Nei casi in cui non è possibile costruire la strada, il tuo programma riceverà:

- 1 punto se restituisce un array vuoto;
- 0 punti altrimenti.

Nei casi in cui è possibile costruire la strada, il tuo programma riceverà:

- 0 punti se restituisce un array vuoto;
- 1 punto se restituisce un array che rappresenta un ordine valido per costruire la strada;
- 0.5 punti altrimenti.

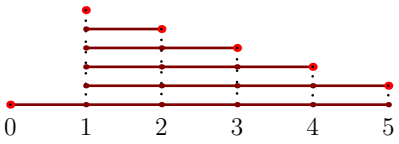
Esempi di input/output

stdin	stdout
6 5 2 2 2 4 4	6 1 2 3 4 5 0
6 4 5 3 0 3 4	0
9 7 5 3 1 0 2 4 6 8	9 4 3 5 2 6 1 7 0 8

Spiegazioni

Nel **primo caso di esempio** i Romani riescono a costruire una strada soddisfando tutti i vincoli:

- Nell'anno 0 scelgono di far partire la strada dalla città 1;
- Nell'anno 1 estendono la strada a destra, la città 2 diventa quindi raggiungibile;
- Nell'anno 2 estendono la strada a destra, la città 3 diventa quindi raggiungibile;
- Nell'anno 3 estendono la strada a destra, la città 4 diventa quindi raggiungibile;
- Nell'anno 4 estendono la strada a destra, la città 5 diventa quindi raggiungibile;
- Nell'anno 5 estendono la strada a sinistra, la città 0 diventa quindi raggiungibile.



Nel **secondo caso di esempio** non è possibile costruire la strada rispettando tutti i vincoli. Dato che $T_0 = T_5 = 4$, la strada deve aver connesso sia la prima che l'ultima città (e quindi essere completa) entro l'anno 4, ma per qualunque ordine di costruzione la strada è completa solo nell'anno 5.

Nel **terzo caso di esempio** i Romani riescono a costruire una strada soddisfando i vincoli come segue: