Road Reorganization (roads)

L'esercito Romano ha recentemente costruito nuove carrozze adatte al trasporto di un numero maggiore di persone. Le nuove carrozze sono più grandi delle precedenti e spesso si bloccano quando c'è traffico nella direzione opposta. Per evitare disagi, l'Imperatore decide di rendere ogni strada dell'Impero a senso unico.

Tuttavia, all'Imperatore piace tornare spesso alla sua città natale, e vuole assicurarsi che quando si trova lì, possa tornare rapidamente a Roma se sorge qualcosa di importante. Per garantire ciò, vuole assicurarsi che il percorso più breve dalla sua città natale a Roma non diventi più lungo di quanto non sia attualmente. L'Imperatore vuole inoltre assicurarsi di poter tornare nella sua città natale, ma non necessariamente nel modo più veloce possibile.

L'Impero Romano consta di N città collegate da M strade a doppio senso. La città natale dell'Imperatore è la città 0 e Roma è la città $N-1$. Formalmente, l'Imperatore vuole scegliere un verso per ogni strada in modo tale che il percorso più breve dalla città 0 alla città $N-1$ non sia più lungo di quanto non fosse prima che le strade diventassero a senso unico. L'imperatore vuole inoltre assicurarsi che dopo l'orientazione delle strade esista ancora un percorso dalla città $N-1$ alla città 0. Il tuo compito è orientare le strade secondo le richieste dell'Imperatore o fargli sapere che ciò è impossibile.

> ⇨ È garantito che ogni città dell'Impero Romano è raggiungibile da ogni altra città. Non è richiesto che ciò rimanga vero dopo l'orientazione le strade.

Nota che orientare le strade in modo da poter andare da 0 a $N-1$ e ritorno, senza necessariamente minimizzare la lunghezza del primo viaggio, vale comunque punti.

Implementazione

Dovrai inviare un singolo file sorgente `.cpp`.

> ⚠ Tra gli allegati di questo compito troverai un template `roads.cpp` con una implementazione di esempio.

Dovrai implementare la seguente funzione:

C++	`vector<pair<int, int>> orient_roads(int N, int M, vector<int> A, vector<int> B, vector<int> W)`

- L'intero N rappresenta il numero di città.
- L'intero M rappresenta il numero di strade.
- Gli array A, B e W descrivono le strade dell'Impero Romano: per ogni i da 0 a $M-1$, c'è una strada tra la città $A[i]$ e la città $B[i]$ di lunghezza $W[i]$.
- Se esiste un'orientazione delle strade che accontenta l'Imperatore, la funzione deve restituire un vettore di coppie di lunghezza M. Una coppia $\{F, T\}$ indica che c'è una strada orientata da F a T.
- Se non esiste un'orientazione delle strade che accontenta l'Imperatore, la funzione deve restituire un vettore vuoto.

È garantito che c'è al massimo una strada tra due città e che per tutti gli i si ha $A[i] \neq B[i]$. Se la funzione non restituisce un vettore di lunghezza 0 o M, la risposta si considera errata. Tutte le strade menzionate nell'input devono essere presenti nel vettore di output esattamente una volta e nessun'altra strada dovrebbe essere presente. L'ordine delle strade nel vettore di output è irrilevante.

Grader di prova

Tra gli allegati di questo problema troverai una versione semplificata del grader utilizzato durante la valutazione, che puoi usare per testare le tue soluzioni in locale. Il grader di esempio legge i dati da `stdin`, chiama la funzione `orient_roads` e scrive su `stdout` usando il seguente formato.

L'input è composto da $M + 1$ righe, contenenti:

- Riga 1: gli interi N e M.
- Riga $2 + i$ $(0 \leq i < M)$: gli interi A_i, B_i e W_i.

L'output è composto da 1 o $M + 1$ righe, contenenti:

- Riga 1: `Yes` o `No`, se è o meno possibile orientare le strade nel modo richiesto.
- Se la risposta è `Yes`: Riga $2 + i$ $(0 \leq i < M)$: F_i e T_i, a indicare che la strada tra queste due città deve essere orientata da F_i a T_i.

Nell'output, gli orientamenti delle strade sono stampati nell'ordine in cui vengono restituiti dalla funzione.

Assunzioni

- $3 \leq N \leq 100\,000$.
- $N - 1 \leq M \leq 100\,000$.
- $0 \leq A_i, B_i \leq N - 1$ e $A_i \neq B_i$.
- $1 \leq W_i \leq 1\,000\,000\,000$.
- È garantito che ogni città dell'Impero è raggiungibile da ogni altra città.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi casi di test raggruppati in subtask. Per ottenere il punteggio completo di un subtask, il tuo programma deve risolvere correttamente tutti i casi di test relativi a quel subtask. Se la soluzione è valida ma non ottimale, otterrai metà dei punti per il relativo caso di test.

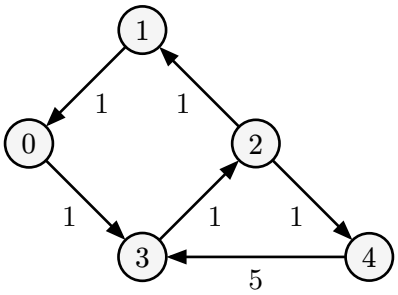
- **Subtask 0** [**0 punti**]: Casi di test di esempio.
- **Subtask 1** [**6 punti**]: $\max(N, M) \leq 20$.
- **Subtask 2** [**11 punti**]: $M \leq N$.
- **Subtask 3** [**17 punti**]: È garantito che c'è una strada diretta tra la città 0 e la città $N - 1$.
- **Subtask 4** [**18 punti**]: $\max(N, M) \leq 2000$.
- **Subtask 5** [**17 punti**]: $W_i = 1$.
- **Subtask 6** [**31 punti**]: Nessun vincolo aggiuntivo.

⇨ Anche nei casi in cui non esiste un modo di orientare le strade per soddisfare *pienamente* l'imperatore, puoi comunque ottenere metà punteggio restituendo un'orientazione che permetta di viaggiare da 0 a $N - 1$ e ritorno.

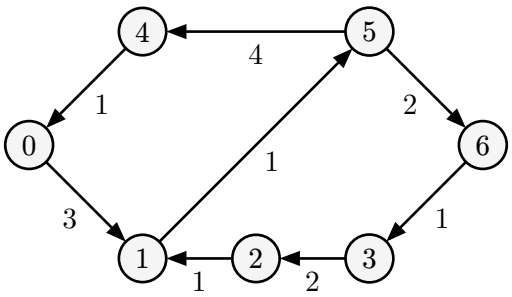
Esempi di input/output

stdin	stdout
5 6 0 1 1 1 2 1 0 3 1 3 2 1 2 4 1 3 4 5	Yes 1 0 2 1 0 3 3 2 2 4 4 3
7 8 0 1 3 1 2 1 2 3 2 3 6 1 0 4 1 4 5 4 5 6 2 1 5 1	Yes 0 1 2 1 3 2 6 3 4 0 5 4 5 6 1 5
8 9 0 1 3 1 2 4 2 4 2 0 3 6 3 4 1 4 5 3 5 6 1 6 7 2 5 7 3	No

Spiegazione



Nel primo esempio, si può andare dalla città 0 alla 4 tramite $0 \to 3 \to 2 \to 4$. Questo percorso ha lunghezza 3, che è anche la lunghezza minima di un percorso da 0 a 4 prima dell'orientazione. È possibile tornare tramite $4 \to 3 \to 2 \to 1 \to 0$ e non importa la lunghezza di questo percorso.



Nel secondo esempio, si può andare da 0 a 6 con il percorso $0 \to 1 \to 5 \to 6$ e tornare seguendo $6 \to 3 \to 2 \to 1 \to 5 \to 4 \to 0$.

Nel terzo esempio, si dimostra che non esiste un orientamento delle strade che renderebbe felice l'Imperatore.