Pedaggio autostradale

In Giappone le città sono collegate da una rete autostradale composta da N città e M autostrade. Ogni autostrada connette una coppia di città distinte, e nessuna coppia di città è connessa da più di una autostrada. Le città sono numerate da 0 a $N-1$ e le autostrade da 0 a $M-1$. Tutte le autostrade sono percorribili in entrambe le direzioni. È possibile viaggiare da ogni città ad ogni altra città usando le autostrade.

Per usare una certa autostrada è necessario pagare un pedaggio, che dipende dal traffico presente nella stessa. Il traffico può infatti essere **scorrevole** o **congestionato**. Quando è scorrevole, il pedaggio è di A¥, quando è congestionato è invece pari a B¥. È garantito che $A < B$. Conosci i valori di A e B.

Il capo ingegnere della società autostradale, pace all'anima sua, mise a punto un grosso marchingegno che (data la situazione del traffico per ogni autostrada) calcolava il minimo pedaggio totale da pagare per effettuare un viaggio tra due specifiche città, che indicheremo come S e T. Purtroppo, dopo l'incidente, la società si è ritrovata senza sapere quali fossero le due città che il capo ingegnere aveva scelto quando era ancora in vita. Sappiamo solo che erano due città distinte ($S \neq T$).

Come nuovo capo ingegnere, il tuo compito è ora quello di determinare i valori di S e T in modo da essere in grado di sfruttare al meglio questo marchingegno. Per far ciò, puoi descrivere diverse situazioni di traffico come input al marchingegno, ed usare le risposte ottenute per dedurre S e T. Poiché questo processo può essere pericoloso, come sa bene il tuo predecessore, non vuoi attivare il marchingegno troppe volte.

Dettagli implementativi

Devi implementare la seguente procedura:

```
find_pair(int N, int[] U, int[] V, int A, int B)
```

- `N`: il numero di città.
- `U` e `V`: vettori di lunghezza M, dove M è il numero di autostrade che connettono le città e l'i-esima autostrada (per ogni $0 \leq i \leq M-1$) connette le città `U[i]` e `V[i]`.
- `A`: il pedaggio per un'autostrada quando il traffico è scorrevole.
- `B`: il pedaggio per un'autostrada quando il traffico è congestionato.
- Questa procedura viene chiamata esattamente una volta per ogni caso di test.
- Notare che il valore di M può essere ottenuto tramite la lunghezza dei vettori.

La procedura `find_pair` può chiamare la seguente funzione:

```
int64 ask(int[] w)
```

- La lunghezza di `w` deve essere M. Il vettore `w` descrive la condizione del traffico.
- Per ogni i ($0 \leq i \leq M-1$), `w[i]` indica la condizione del traffico sull'autostrada i. Il valore di `w[i]` deve essere 0 o 1.
  - `w[i] = 0` indica che il traffico sull'autostrada i è scorrevole.
  - `w[i] = 1` indica che il traffico sull'autostrada i è congestionato.
- Questa funzione restituisce il minore totale di pedaggi da pagare per viaggiare dalla città S alla città T con le condizioni di traffico indicate da `w`.
- Questa funzione può essere chiamata al più 100 volte (per ogni caso di test).

`find_pair` deve chiamare la seguente procedura per dare la risposta:

```
answer(int s, int t)
```

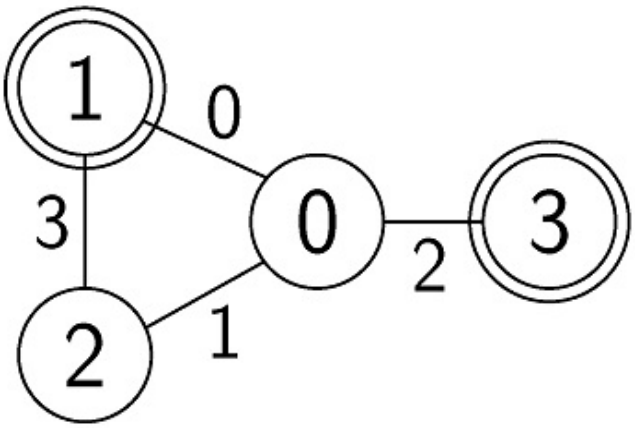
- `s` e `t` devono essere la coppia S e T (l'ordine è indifferente).
- Questa procedura deve essere chiamata esattamente una volta.

Se qualcuno dei vincoli sopracitati non viene soddisfatto, il tuo programma sarà giudicato come **`Wrong Answer`**. Altrimenti, il tuo programma sarà giudicato come **`Accepted`** e il tuo punteggio sarà calcolato in base al numero di chiamate ad `ask` (guardare i subtask).

Esempio

Siano $N = 4$, $M = 4$, $U = [0, 0, 0, 1]$, $V = [1, 2, 3, 2]$, $A = 1$, $B = 3$, $S = 1$, e $T = 3$.

Il grader chiama `find_pair(4, [0, 0, 0, 1], [1, 2, 3, 2], 1, 3)`.



Nella figura sopra, l'arco col numero i corrisponde all'autostrada i.

Alcune possibili chiamate ad `ask` e i corrispondenti valori restituiti sono:

Chiamata	Valore restituito
`ask([0, 0, 0, 0])`	2
`ask([0, 1, 1, 0])`	4
`ask([1, 0, 1, 0])`	5
`ask([1, 1, 1, 1])`	6

Per la chiamata `ask([0, 0, 0, 0])`, il traffico di ogni autostrada è scorrevole e il pedaggio quindi è 1.

La strada meno costosa da $S = 1$ a $T = 3$ è $1 \to 0 \to 3$. Il pedaggio totale di questo percorso è 2, e quindi la funzione restituisce 2.

A questo punto, la procedura `find_pair` deve chiamare `answer(1, 3)` o `answer(3, 1)` per essere considerata corretta.

Il file `sample-01-in.txt` nell'archivio compresso in allegato corrisponde a questo esempio. Altri esempi di input sono disponibili nell'archivio.

Assunzioni

- $2 \leq N \leq 90\,000$
- $1 \leq M \leq 130\,000$
- $1 \leq A < B \leq 1\,000\,000\,000$
- Per ogni $0 \leq i \leq M - 1$
  - $0 \leq U[i] \leq N - 1$
  - $0 \leq V[i] \leq N - 1$
  - $U[i] \neq V[i]$
- $(U[i], V[i]) \neq (U[j], V[j])$ e $(U[i], V[i]) \neq (V[j], U[j])$ $(0 \leq i < j \leq M - 1)$
- Puoi viaggiare da ogni città a qualunque altra città usando le autostrade.
- $0 \leq S \leq N - 1$
- $0 \leq T \leq N - 1$
- $S \neq T$

In questo problema il grader non è adattivo. Questo significa che S e T sono fissate all'inizio dell'esecuzione del grader e non dipendono dalle domande fatte dalla tua soluzione.

Subtask

1. (5 punti) uno tra S e T è 0, $N \leq 100$, $M = N - 1$
2. (7 punti) uno tra S e T è 0, $M = N - 1$
3. (6 punti) $M = N - 1$, $U[i] = i$, $V[i] = i + 1$ $(0 \leq i \leq M - 1)$
4. (33 punti) $M = N - 1$
5. (18 punti) $A = 1$, $B = 2$
6. (31 punti) Nessuna limitazione aggiuntiva

Assumendo che il tuo programma è giudicato **Accepted** e fa X chiamate ad `ask`, il tuo punteggio P per il caso di test, dipendentemente dal subtask, è calcolato nel seguente modo:

- Subtask 1. $P = 5$.
- Subtask 2. Se $X \leq 60$, $P = 7$. Altrimenti $P = 0$.
- Subtask 3. Se $X \leq 60$, $P = 6$. Altrimenti $P = 0$.
- Subtask 4. Se $X \leq 60$, $P = 33$. Altrimenti $P = 0$.
- Subtask 5. Se $X \leq 52$, $P = 18$. Altrimenti $P = 0$.
- Subtask 6.
  - Se $X \leq 50$, $P = 31$.
  - Se $51 \leq X \leq 52$, $P = 21$.
  - Se $53 \leq X$, $P = 0$.

Nota che il tuo punteggio per ogni subtask è il minimo dei punteggi per ogni caso di test nel subtask.

Grader di esempio

Il grader di esempio legge l'input nel seguente formato:

- riga 1: N M A B S T
- righe $2 + i$ $(0 \leq i \leq M - 1)$: $U[i]$ $V[i]$

Se il tuo programma voene giudicato come **Accepted**, il grader di esempio stampa `Accepted: q`, dove `q` è il numero di chiamate effettuate ad `ask`.

Se il tuo programma voene giudicato come **Wrong Answer**, stampa `Wrong Answer: MSG`, dove `MSG` è uno tra:

- `answered not exactly once`: La procedura `answer` non è stata chiamata esattamente una volta.
- `w is invalid`: La lunghezza di `w` dato a `ask` non è M o `w[i]` non è 0 o 1 per qualche i $(0 \leq i \leq M - 1)$.
- `more than 100 calls to ask`: La funzione `ask` è stata chiamata più di 100 volte.
- `{s, t} is wrong`: La procedura `answer` è stata chiamata con valori di `s` e `t` errati.