D. Censimento (census)

Limite di tempo: 1 secondi
Limite di memoria: 128 MiB

Un fatto poco noto su Cesenatico è che ospita una società segreta di N informatiche. Questa società è davvero molto segreta; nessun membro ne conosce un'altro. Ogni membro ha un ID unico: un intero non negativo I.

L'unica comunicazione tra i membri è indiretta, tramite numeri scritti col gesso in diversi luoghi in città. Ogni 100 anni, la società svolge un censimento per contare i propri membri. Al termine del censimento, ogni membro dovrebbe conoscere il numero totale di membri nella società.

Il censimento si svolge su più giorni. Ogni giorno, ogni membro ancora partecipe al processo sceglie ed esegue esattamente un'azione: **leggere**, **scrivere** o **smettere** di partecipare.

- Se un membro sceglie di **leggere**, sceglie un luogo P. Durante il giorno, visita il luogo P e legge il numero che vi è scritto.
- Se un membro sceglie di **scrivere**, sceglie un luogo P e un numero V. In tarda serata, visita il luogo P e cambia il numero scritto in precedenza con V. Dato che è già buio, non può leggere il vecchio numero prima di scrivere quello nuovo.
- Se un membro sceglie di **smettere**, non compie più alcuna azione nei giorni successivi.

Se un membro vedesse un'altra scrivere un numero, potrebbe riconoscerla. Pertanto, è severamente vietato che due o più membri scelgano di scrivere nello stesso luogo lo stesso giorno. (Non esiste tale restrizione per la lettura, poiché può essere fatta in modo discreto.)

Se una o più membri leggono da un luogo in cui un'altro membro vuole scrivere nello stesso giorno, tutte le letture avvengono prima della scrittura.

Come dovrebbe pianificare il censimento la società per minimizzare il numero di giorni necessari affinché tutte conoscano il conteggio corretto dei membri?

Implementazione

> ⇨ Questo è un problema interattivo, in cui verrà eseguito simultaneamente un numero sconosciuto di istanze ($1 \le N \le 100$) del tuo programma. Ogni istanza simula un membro della società.

Ci sono 10^{18} luoghi. Il numero P di un luogo deve soddisfare $0 \le P < 10^{18}$. Inizialmente, il valore scritto in tutti i luoghi è $V = 0$.

Il nuovo valore V scritto in un luogo deve sempre essere un intero tale che $0 \le V \le 10^9$. Nella maggior parte dei subtask, V può essere solo 0 o 1. Vedi la sezione Punteggio per ulteriori dettagli.

Quando un'istanza del tuo programma si avvia, dovrebbe prima leggere una riga con due interi, I e M ($0 \le I \le M - 1$): l'ID unico del membro della società rappresentata da questa istanza e il numero totale di ID possibili. All'interno di ogni caso di test, tutte le istanze riceveranno lo stesso valore M e valori I distinti. Nota che potrebbero esserci ID non assegnati a nessuna membro.

Poi, per ogni giorno del processo di censimento, il tuo programma dovrebbe scegliere l'azione che vuole eseguire e stampare una riga di conseguenza:

Azione	Significato
`r` P	**Leggi** il luogo P. Dopo aver stampato questa riga, il tuo programma dovrebbe leggere una riga con il valore attuale scritto in P.
`w` P V	**Scrivi** nel luogo P il nuovo valore V. Se più istanze scrivono nella stessa P lo stesso giorno, otterrai il verdetto *Non corretto*. Eccetto per gli esempi e il sottoproblema 3, devi scrivere $0 \leq V \leq 1$; vedi la sezione Punteggio.
`!` N	**Rispondi e smetti**: riporta che ci sono N membri e smetti di partecipare al censimento. Dopo aver risposto, **il tuo programma dovrebbe terminare normalmente**. (Nota che altre istanze del tuo programma potrebbero continuare a girare per giorni aggiuntivi prima di rispondere e terminare.)

Se un'istanza del tuo programma risponde con il valore di N errato, viola il protocollo, utilizza più di 500 giorni o supera il limite di tempo/memoria (per processo), la tua sottomissione sarà giudicata come *Non corretta* per il caso di test dato.

Altrimenti, il tuo programma sarà *(Parzialmente) Corretto* sul caso di test e valutato in base al valore D: il numero massimo di giorni impiegato da qualsiasi istanza per rispondere. Per il punteggio pieno, devi risolvere ogni caso di test con $D \leq 61$ e $V \leq 1$. Vedi la sezione Punteggio per i dettagli.

Flushing. Se non stai usando i template forniti, assicurati di effettuare il flush dello standard output dopo aver stampato ogni riga, altrimenti il tuo programma potrebbe essere giudicato *Non corretto*. In Python, questo succede automaticamente se usi `input()` per leggere le righe. In C++, `cout << endl;` effettua il flush oltre a stampare un newline; se usi printf, usa `fflush(stdout)`.

Assunzioni

- $1 \leq N \leq 100$.
- $1 \leq M \leq 100\,000$.
- Puoi usare al massimo 500 giorni.

Assegnazione del punteggio

Il tuo programma sarà testato su diversi casi di test raggruppati in subtask. Per ottenere il punteggio per un subtask, devi risolvere correttamente tutti i test che contiene.

- **Subtask 0 [0 punti]**: Esempi (puoi scrivere qualsiasi intero $0 \leq V \leq 1\,000\,000\,000$).
- **Subtask 1 [11 punti]**: $M \leq 100$, e le N membri hanno ID $0, 1, ..., N-1$.
- **Subtask 2 [12 punti]**: $1 \leq N \leq 2$.
- **Subtask 3 [22 punti]**: $M \leq 8000$, e puoi scrivere qualsiasi intero $0 \leq V \leq 1\,000\,000\,000$.
- **Subtask 4 [55 punti]**: Nessun vincolo aggiuntivo.

Nei subtask 1, 2 e 4, puoi scrivere solo $V = 0$ o $V = 1$ in ogni azione di Scrittura.

Sia X_s il punteggio massimo per il subtask s (mostrato sopra), e D_s il numero massimo di giorni che una qualsiasi delle tue istanze utilizza su un test nel subtask s. Allora:

$$\text{score}_s = \begin{cases} X_s & \text{se } D_s \leq 61 \\ X_s \cdot \left(0.2 + 0.8 \cdot 1.01^{(60-D_s)}\right) & \text{se } 61 < D_s \leq 500 \\ 0 & \text{se } 500 < D_s. \end{cases}$$

Il valore di score_s viene arrotondato all'intero più vicino per subtask, e il tuo punteggio totale è la somma di questi. Per ottenere il punteggio pieno per il problema, devi avere $D \leq 61$ e $V \leq 1$ su ogni caso di test.

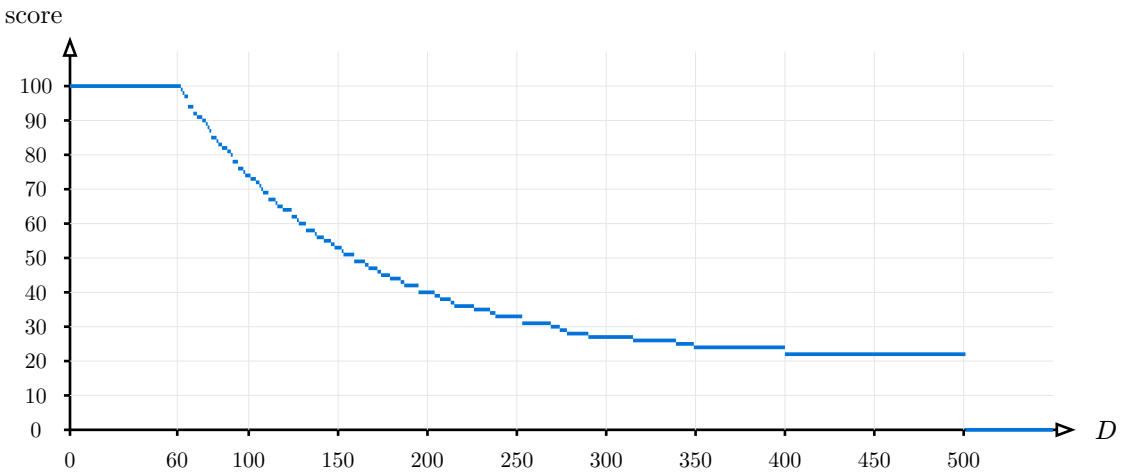

Figura 1: Punteggio totale, assumendo che ogni subtask sia risolto con lo stesso D massimo.

Esempi di input/output

Primo esempio. Ogni coppia di colonne mostra la comunicazione tra il grader e un'istanza.

Gra.	Ist. 0	Gra.	Ist. 1	Gra.	Ist. 2	Gra.	Ist. 3	Gra.	Ist. 4
0 100		1 100		2 100		3 100		4 100	
	w 12 1		w 50 1		w 99 0		w 7 1		r 5
								0	
	r 50		r 7		r 12		w 1 1		! 5
1		1		1					
	! 5		r 1		w 0 0		! 5		
		1							
			! 5		! 5				

Secondo esempio.

Grader	Istanza 0
0 8000	
	w 0 0
	w 1 1
	r 2
1	
	! 2

Grader	Istanza 1
3 8000	
	w 2 1
	r 1
0	
	r 2
1	
	r 1
1	
	! 2

Spiegazione

Primo Esempio. Abbiamo $N = 5$ membri con ID consecutivi $0, 1, 2, 3, 4$ e $M = 100$ (valido per i subtask 1, 3 e 4). L'istanza i corrisponde al membro con ID i. L'interazione sopra è solo una possibile sequenza legale di operazioni e **non** intende essere una strategia efficiente o sensata; è mostrata solo per illustrare come funziona il protocollo.

Secondo Esempio. Abbiamo $N = 2$ membri, con ID 0 e 3, e $M = 8000$ (valido per i subtask 2, 3 e 4). Il primo giorno, il membro con ID 0 scrive 0 nel luogo 0 (nessun cambiamento), e il membro con ID 3 scrive 1 nel luogo 2.

location	0	1	2	3	4	...
value	0	0	1	0	0	...

Il secondo giorno, l'ID 0 scrive 1 nel luogo 1, e l'ID 3 legge quello stesso luogo. Nota che la lettura avviene durante il giorno, prima della scrittura di sera. Pertanto, l'ID 3 vede ancora uno 0.

location	0	1	2	3	4	...
value	0	1	1	0	0	...

Il terzo giorno, entrambe leggono il luogo 2, dove è scritto un 1.

Il quarto giorno, l'ID 0 risponde che ci sono 2 membri (corretto), mentre l'ID 3 legge l'1 nel luogo 1. L'ID 0 termina immediatamente dopo questo e non partecipa ai giorni successivi.

Infine, al giorno $D = 5$, il membro rimanente risponde anch'esso correttamente $N = 2$.

Test

Per facilitare il test della tua soluzione, forniamo un semplice tool che puoi scaricare da CMS. L'uso del tool è facoltativo. Nota che il grader ufficiale su CMS è diverso dal tool di test.

Per usare il tool ti serve un file di input. Puoi usare gli esempi di input forniti `census.input0.txt` e `census.input1.txt`, oppure crearne uno tuo. Il file di input dovrebbe iniziare con il numero di membri N ed il numero di ID possibili M, seguito da una riga con N numeri che specificano gli ID dei membri della società.

Per i programmi Python, diciamo `census.py` (normalmente eseguito come `pypy3 census.py`) esegui il tool di test come segue:

```
python3 testing_tool.py pypy3 census.py < census.input0.txt
```

Per i programmi C++, prima compila la tua soluzione:

```
g++ -DEVAL -std=gnu++20 -O2 -pipe -static -s -o census census.cpp
```

e poi esegui il tool di test:

```
python3 testing_tool.py ./census < census.input0.txt
```

Nota che in questo problema lo standard output è usato per la comunicazione con il grader, quindi non dovrebbe essere usato per il debug. Invece, puoi usare lo standard error output (stderr). In C++ puoi usare `cerr << msg << endl;`. In Python puoi usare `print(msg, file=sys.stderr)`.

Il tool di test leggerà e presenterà questi messaggi stderr insieme alle query eseguite da tutte le istanze del tuo programma. Nota che per motivi tecnici potrebbero non essere sincronizzate l'una con l'altra.