D. Volkstelling (census)

Tijdslimiet: 1 seconden
Geheugenlimiet: 128 MiB

Een minder bekend feit over Cesenatico is dat het de thuisbasis is van een geheim genootschap van N vrouwelijke informatici. Dit genootschap is echt heel geheim; geen enkel lid kent iemand van de anderen. Elk lid heeft een uniek ID: een niet-negatief geheel getal I.

De enige communicatie tussen leden verloopt indirect, via getallen die met krijt zijn gekrabbeld op verschillende plaatsen in de stad. Elke 100 jaar houdt het genootschap een volkstelling om de leden te tellen. Nadat de volkstelling is voltooid, moet elk lid weten wat het totaal aantal leden van het genootschap is.

De volkstelling vindt plaats gedurende meerdere dagen. Elke dag kiest elk lid dat nog deelneemt aan het proces precies één actie en voert deze uit: **lezen**, **schrijven** of **stoppen** met deelname.

- Als een lid kiest voor **lezen**, kiest zij een locatie P. Overdag gaat ze naar locatie P en leest het getal dat daar geschreven staat.
- Als een lid kiest voor **schrijven**, kiest zij een locatie P en een getal V. 's Avonds laat gaat ze naar locatie P en verandert het getal dat daar geschreven staat in V. Omdat het al donker is, kan ze het oude getal niet lezen voordat ze het nieuwe schrijft.
- Als een lid kiest voor **stoppen**, onderneemt ze de volgende dagen geen acties meer.

Als een lid een ander lid een getal ziet schrijven, zou ze haar kunnen herkennen. Daarom is het strikt verboden dat twee of meer leden om op dezelfde dag op dezelfde locatie te schrijven. (Voor lezen is er niet zo'n beperking, omdat dat discreet kan gebeuren.)

Als één of meer leden lezen op een locatie waar een ander lid op dezelfde dag wil schrijven, vinden alle leesacties plaats vóór het schrijven.

Hoe moet het genootschap zijn volkstellingsproces plannen om het aantal dagen te minimaliseren totdat iedereen het juiste aantal leden kent?

Implementatie

⇨ Dit is een interactief probleem, waarbij een onbekend aantal instanties ($1 \leq N \leq 100$) van je programma tegelijkertijd zal worden uitgevoerd. Elke instantie simuleert één lid van het genootschap.

Er zijn 10^{18} locaties. Het nummer P van een locatie moet voldoen aan $0 \leq P < 10^{18}$. In het begin is de waarde geschreven op alle locaties $V = 0$.

De nieuwe waarde V die op een locatie wordt geschreven, moet altijd een geheel getal zijn dat voldoet aan $0 \leq V \leq 10^9$. In de meeste subtasks (deelopgaven) kan V alleen 0 of 1 zijn. Zie de sectie Scoring voor meer details.

Wanneer een instantie van je programma start, moet deze eerst een regel met twee gehele getallen I en M ($0 \leq I \leq M - 1$) lezen: het unieke ID van het lid van het genootschap vertegenwoordigd door deze instantie en het totaal aantal mogelijke ID's. Binnen elke testcase krijgen alle instanties dezelfde

waarde M en verschillende waarden I. Merk op dat er ID's kunnen zijn die aan geen enkel lid zijn toegewezen.

Voor elke dag in het volkstellingsproces, moet je programma dan de actie kiezen die het wil uitvoeren en een regel printen die daarbij hoort:

Actie	Betekenis
`r` P	**Lees** locatie P. Na het printen van deze regel moet je programma een regel lezen met de huidige waarde die op P geschreven staat.
`w` P V	**Schrijf** op locatie P de nieuwe waarde V. Als meerdere instanties op dezelfde dag op dezelfde P schrijven, krijg je het oordeel *Niet correct*. Met uitzondering van de voorbeelden en subtask 3, moet je $0 \leq V \leq 1$ schrijven; zie de sectie Scoring.
`!` N	**Antwoord en stop**: rapporteer dat er N leden zijn en stop met deelname aan de volkstelling. Na het antwoorden **moet je programma normaal afsluiten**. (Merk op dat andere instanties van je programma nog gedurende extra dagen kunnen blijven draaien voordat ze antwoorden en afsluiten.)

Als een instantie van je programma de verkeerde waarde voor N antwoordt, het protocol schendt, meer dan 500 dagen gebruikt, of de (per-instantie) tijd/geheugenlimiet overschrijdt, wordt je oplossing beoordeeld als *Niet correct* voor de betreffende testcase.

Anders zal je programma *(Gedeeltelijk) Correct* zijn op de testcase en worden gescoord op basis van de waarde D: het maximum aantal dagen dat de instanties van je programma nodig hadden om te antwoorden. Voor de volledige score moet je alle testcases oplossen met $D \leq 61$ en $V \leq 1$. Zie de sectie Scoring voor details.

Flushen.

Als je de meegeleverde niet templates gebruikt, zorg er dan voor dat je de standaarduitvoer flusht na het printen van elke regel, anders wordt je programma mogelijk beoordeeld als *Niet correct*. In Python gebeurt dit automatisch als je `input()` gebruikt om regels te lezen. In C++ zorgt `cout << endl;` voor flushen, en het print een nieuwe regel; als je printf gebruikt, gebruik dan `fflush(stdout)`.

Randvoorwaarden

- $1 \leq N \leq 100$.
- $1 \leq M \leq 100\,000$.
- Je mag maximaal 500 dagen gebruiken.

Scoring

Je programma wordt getest op verschillende testcases die zijn gegroepeerd in subtasks (deelopgaven). Om de punten voor een subtaak te verkrijgen, moet je alle tests die erin staan correct oplossen.

- **Subtask 0 [0 punten]**: Voorbeelden (je mag elk geheel getal $0 \leq V \leq 1\,000\,000\,000$ schrijven).
- **Subtask 1 [11 punten]**: $M \leq 100$, en de N leden hebben ID's $0, 1, ..., N-1$.
- **Subtask 2 [12 punten]**: $1 \leq N \leq 2$.
- **Subtask 3 [22 punten]**: $M \leq 8000$, en je mag elk geheel getal $0 \leq V \leq 1\,000\,000\,000$ schrijven.

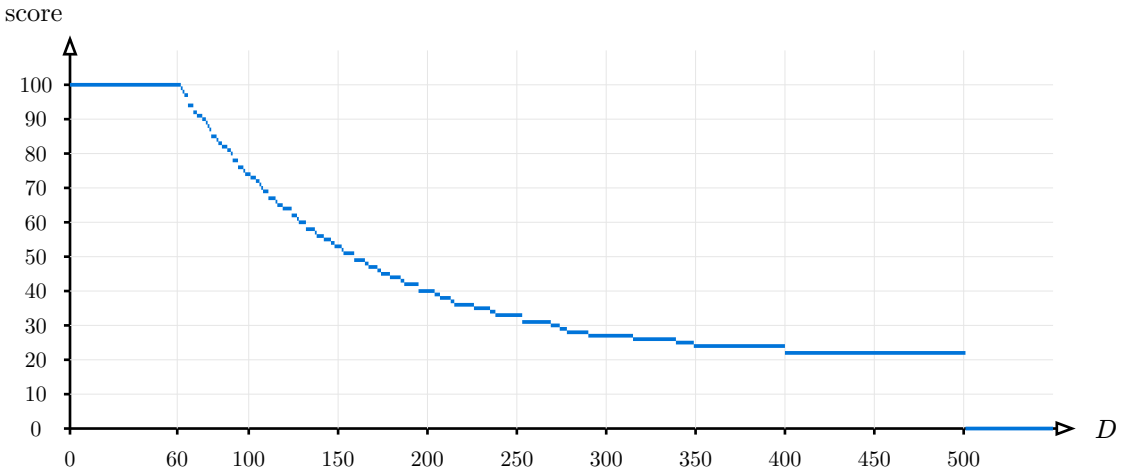
- **Subtask 4 [55 punten]**: Geen extra randvoorwaarden.

In subtasks 1, 2 en 4 mag je alleen $V = 0$ of $V = 1$ schrijven bij elke Schrijf-actie.

Als X_s de maximale score voor subtask s is (hierboven getoond), en D_s het grootste aantal dagen dat één van de instanties van je programma gebruikt bij een test in subtaak s, dan is je score:

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

De waarde van score_s wordt per subtask afgerond naar het dichtstbijzijnde gehele getal, en je totale score is de som hiervan. Om de volledige score voor de taak te krijgen, heb je $D \leq 61$ en $V \leq 1$ nodig bij elke testcase.



Figuur 1: Totale score, aannemende dat elke subtask wordt opgelost met dezelfde maximale D.

Voorbeelden

Eerste voorbeeld. Elk tweetal kolommen toont de communicatie tussen de grader en één instantie.

Gra.	Inst. 0	Gra.	Inst. 1	Gra.	Inst. 2	Gra.	Inst. 3	Gra.	Inst. 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
	r 50		r 7		r 12		w 1 1	0	
1		1		1					! 5
	! 5		r 1		w 0 0		! 5		
		1							
			! 5		! 5				

Tweede voorbeeld.

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

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

Uitleg

Eerste voorbeeld. We hebben $N = 5$ leden met opeenvolgende ID's $0, 1, 2, 3, 4$ en $M = 100$ (geldig voor subtasks 1, 3 en 4). Instantie i komt overeen met het lid met ID i. De interactie hierboven is alleen maar één mogelijke toegestane reeks operaties en is **niet** bedoeld als een efficiënte of verstandige strategie; het is alleen getoond om te laten zien hoe het protocol werkt.

Tweede voorbeeld. We hebben $N = 2$ leden, met ID's 0 en 3, en $M = 8000$ (geldig voor subtaken 2, 3 en 4). Op de eerste dag schrijft het lid met ID 0 een 0 op locatie 0 (geen verandering), en het lid met ID 3 schrijft een 1 op locatie 2.

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

Op de tweede dag schrijft ID 0 een 1 op locatie 1, en ID 3 leest diezelfde locatie. Merk op dat het lezen overdag gebeurt, vóór het schrijven 's avonds. Daarom ziet ID 3 nog steeds een 0.

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

Op de derde dag lezen ze beiden locatie 2, waar een 1 staat geschreven.

Op de vierde dag antwoordt ID 0 dat er 2 leden zijn (correct), terwijl ID 3 de 1 op locatie 1 leest. ID 0 nu sluit direct af en neemt niet deel aan de komende dagen.

Tenslotte, op dag $D = 5$, antwoordt het overgebleven lid ook correct dat $N = 2$.

Testen

Om het testen van je oplossing makkelijker te maken, geven we je een eenvoudige tool die je kunt downloaden van CMS. Gebruik van de tool is optioneel. Let op dat de officiële grader op CMS anders is dan de testing tool.

Om de tool te gebruiken heb je een invoerbestand nodig. Je kunt de meegeleverde voorbeeldinvoer `census.input0.txt` en `census.input1.txt` gebruiken, of je eigen invoer maken. Het invoerbestand moet beginnen met het aantal leden N en de mogelijke ID's M, gevolgd door een regel met N getallen die de ID's van de genootschapsleden specificeren.

Voor Python-programma's, bijvoorbeeld `census.py` (normaal uitgevoerd als `pypy3 census.py`) voer je de testing tool als volgt uit:

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

Voor C++-programma's, compileer eerst je oplossing:

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

en voer vervolgens de testing tool uit:

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

Let op dat in dit probleem de standaarduitvoer wordt gebruikt voor communicatie met de grader, dus deze mag niet worden gebruikt voor debuggen. Gebruik in plaats daarvan de standaard-error-uitvoer (stderr). In C++ kun je `cerr << msg << endl;` gebruiken. In Python kun je `print(msg, file=sys.stderr)` gebruiken.

De testing tool zal deze stderr-berichten lezen en laten zien samen met de queries uitgevoerd door al je programma-instanties. Merk op dat deze om technische redenen mogelijk iets out of sync kunnen verschijnen.