D. Volkszählung (census)

Zeitlimit: 1 Sekunden
Speicherlimit: 128 MiB

Eine eher unbekannte Tatsache über Cesenatico ist, dass es die Heimat eines Geheimbunds von N Informatikerinnen ist. Dieser Geheimbund ist wirklich sehr geheim; keine der Mitglieder kennt die anderen. Jedes Mitglied hat eine eindeutige ID: eine nicht-negative Ganzzahl I.

Die einzige Kommunikation zwischen den Mitgliedern läuft indirekt ab: über Zahlen, die mit Kreide an verschiedenen Orten in der Stadt notiert werden. Alle 100 Jahre führt der Bund eine Volkszählung durch, um die Mitglieder zu zählen. Nachdem die Volkszählung abgeschlossen ist, sollte jedes Mitglied die Gesamtzahl der Mitglieder im Bund kennen.

Die Volkszählung erstreckt sich über mehrere Tage. An jedem Tag wählt jedes Mitglied, das noch am Prozess teilnimmt, genau eine Aktion: **Lesen**, **Schreiben** oder **Aufhören**.

- Wenn ein Mitglied **lesen** will, wählt sie einen Ort P. Tagsüber besucht sie den Ort P und liest die dort notierte Zahl.
- Wenn ein Mitglied **schreiben** will, wählt sie einen Ort P und eine Zahl V. Am späten Abend besucht sie den Ort P und ändert die dort stehende Zahl auf V. Da es bereits dunkel ist, kann sie die alte Zahl vor dem Schreiben nicht lesen.
- Wenn ein Mitglied **aufhören** will, nimmt sie an den folgenden Tagen an keiner Aktion mehr teil.

Wenn ein Mitglied sieht, wie eine andere eine Zahl schreibt, könnte sie sie erkennen. Daher ist es strengstens verboten, dass zwei oder mehr Mitglieder am selben Tag am selben Ort schreiben. (Für das Lesen gibt es diese Einschränkung nicht, da dies diskret geschehen kann.)

Wenn ein oder mehrere Mitglieder an einem Ort lesen, an dem ein anderes Mitglied am selben Tag schreiben will, finden alle Lesevorgänge vor dem Schreibvorgang statt.

Wie sollte der Geheimbund seine Volkszählung planen, um die Anzahl der Tage zu minimieren, bis alle die korrekte Mitgliederzahl kennen?

Implementierung

⇨ Das ist ein interaktives Problem, bei dem eine unbekannte Anzahl von Instanzen ($1 \leq N \leq 100$) deines Programms gleichzeitig ausgeführt wird. Jede Instanz simuliert ein Mitglied des Geheimbunds.

Es gibt 10^{18} Orte. Die Nummer P eines Ortes muss $0 \leq P < 10^{18}$ erfüllen. Anfangs ist der Wert, der an allen Orten notiert ist, $V = 0$.

Der neue Wert V, der an einem Ort geschrieben wird, muss immer eine Ganzzahl sein, sodass $0 \leq V \leq 10^9$. In den meisten Teilaufgaben kann V nur 0 oder 1 sein. Siehe den Abschnitt „Bewertung“ für weitere Details.

Wenn eine Instanz deines Programms startet, sollte sie zuerst eine Zeile mit zwei Ganzzahlen I und M ($0 \leq I \leq M - 1$) lesen: die eindeutige ID des Mitglieds, das von dieser Instanz repräsentiert wird, und

die Gesamtzahl der möglichen IDs. Innerhalb eines Testfalls erhalten alle Instanzen denselben Wert M und unterschiedliche Werte I. Beachte, dass es IDs geben kann, die keinem Mitglied zugewiesen sind.

Dann sollte dein Programm für jeden Tag der Volkszählung die gewünschte Aktion wählen und entsprechend eine Zeile ausgeben:

Aktion	**Bedeutung**
`r` P	**Lesen** von Ort P. Nach dem Ausgeben dieser Zeile sollte dein Programm eine Zeile mit dem aktuellen Wert an Ort P lesen.
`w` P V	**Schreiben** des neuen Werts V an Ort P. Wenn mehrere Instanzen am selben Tag an demselben Ort P schreiben, erhältst du das Urteil *Nicht korrekt*. Außer bei den Beispielen und Teilaufgabe 3 musst du $0 \leq V \leq 1$ schreiben; siehe den Abschnitt „Bewertung“.
`!` N	**Antworten und aufhören**: Melde, dass es N Mitglieder gibt und nimm nicht mehr an der Volkszählung teil. Nach dem Antworten **sollte dein Programm normal beenden**. (Beachte, dass andere Instanzen deines Programms noch für weitere Tage laufen könnten, bevor sie antworten und beenden.)

Falls irgendeine Instanz deines Programms den falschen Wert für N antwortet, das Protokoll verletzt, mehr als 500 Tage benötigt oder das Zeit-/Speicherlimit (pro Prozess) überschreitet, wird deine Einreichung für den jeweiligen Testfall als *Nicht korrekt* gewertet.

Ansonsten wird dein Programm für den Testfall als *(Teilweise) Korrekt* gewertet und basierend auf dem Wert D bewertet: die maximale Anzahl an Tagen, die irgendeine Instanz bis zum Antworten benötigt hat. Für die volle Punktzahl musst du jeden Testfall mit $D \leq 61$ und $V \leq 1$ lösen. Siehe den Abschnitt „Bewertung“ für Details.

Leeren des Ausgabepuffers (Flushing). Falls du nicht die bereitgestellten Vorlagen verwendest, stelle sicher, dass du nach dem Drucken jeder Zeile den Standardausgabepuffer leerst (flushen), sonst könnte dein Programm als *Nicht korrekt* gewertet werden. In Python passiert das automatisch, wenn du `input()` zum Lesen von Zeilen verwendest. In C++ leert `cout << endl;` den Puffer zusätzlich zum Zeilenumbruch; falls du printf verwendest, nutze `fflush(stdout)`.

Einschränkungen

- $1 \leq N \leq 100$.
- $1 \leq M \leq 100\,000$.
- Du darfst maximal 500 Tage verwenden.

Punktevergabe

Dein Programm wird auf mehreren Testfällen getestet, die in Teilaufgaben gruppiert sind. Um die Punktzahl für eine Teilaufgabe zu erhalten, musst du alle enthaltenen Tests korrekt lösen.

- **Teilaufgabe 0 [0 Punkte]**: Beispiele (du kannst jede Ganzzahl $0 \leq V \leq 1\,000\,000\,000$ schreiben).
- **Teilaufgabe 1 [11 Punkte]**: $M \leq 100$, und die N Mitglieder haben die IDs $0, 1, ..., N-1$.
- **Teilaufgabe 2 [12 Punkte]**: $1 \leq N \leq 2$.
- **Teilaufgabe 3 [22 Punkte]**: $M \leq 8000$, und du kannst jede Ganzzahl $0 \leq V \leq 1\,000\,000\,000$ schreiben.

- **Teilaufgabe 4 [55 Punkte]**: Keine zusätzlichen Einschränkungen.

In den Teilaufgaben 1, 2 und 4 kannst du bei jeder Schreib-Aktion nur $V = 0$ oder $V = 1$ schreiben.

Sei X_s die maximale Punktzahl für Teilaufgabe s (oben angezeigt) und D_s die größte Anzahl an Tagen, die eines deiner Programme bei einem Test in Teilaufgabe s verwendet. Dann gilt:

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

Der Wert von score_s wird pro Teilaufgabe auf die nächste Ganzzahl gerundet, und deine Gesamtpunktzahl ist die Summe dieser. Um die volle Punktzahl für die Aufgabe zu erhalten, musst du $D \leq 61$ und $V \leq 1$ in jedem Testfall erreichen.

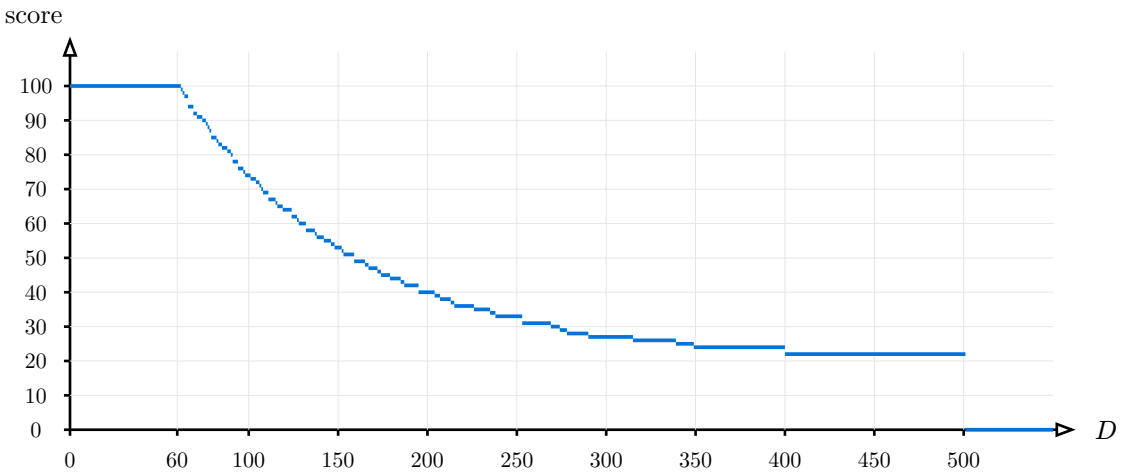


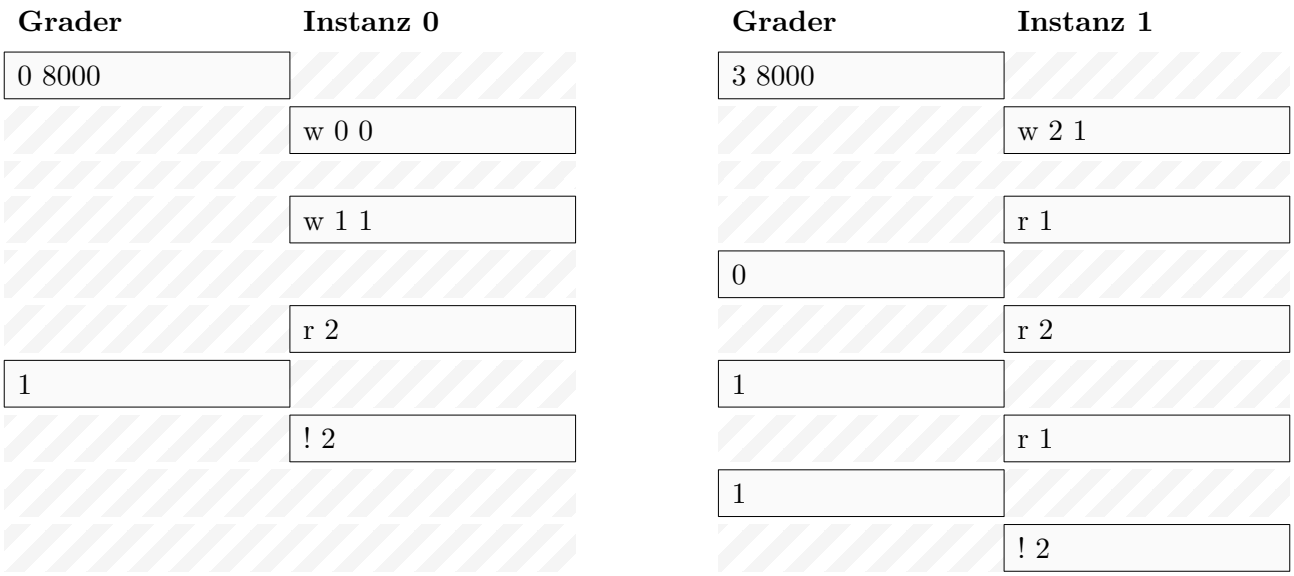
Abbildung 1: Gesamtpunktzahl, unter der Annahme, dass jede Teilaufgabe mit dem gleichen maximalen D gelöst wird.

Beispiele

Erstes Beispiel. Jedes Paar an Spalten zeigt die Kommunikation zwischen dem Grader und einer Instanz.

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
								0	
	r 50		r 7		r 12		w 1 1		! 5
1		1		1					
	! 5		r 1		w 0 0		! 5		
		1							
			! 5		! 5				

Zweites Beispiel.



Erklärung

Erstes Beispiel. Wir haben $N = 5$ Mitglieder mit aufeinanderfolgenden IDs $0, 1, 2, 3, 4$ und $M = 100$ (gültig für Teilaufgaben 1, 3 und 4). Instanz i entspricht dem Mitglied mit ID i. Die Interaktion oben ist nur eine mögliche legale Abfolge von Operationen und ist **nicht** als effiziente oder sinnvolle Strategie gedacht; sie dient nur dazu, die Funktionsweise des Protokolls zu illustrieren.

Zweites Beispiel. Wir haben $N = 2$ Mitglieder mit den IDs 0 und 3 und $M = 8000$ (gültig für Teilaufgaben 2, 3 und 4). Am ersten Tag schreibt das Mitglied mit ID 0 eine 0 an Ort 0 (keine Änderung), und das Mitglied mit ID 3 schreibt eine 1 an Ort 2.

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

Am zweiten Tag schreibt ID 0 eine 1 an Ort 1, und ID 3 liest denselben Ort. Beachte, dass das Lesen tagsüber passiert, vor dem Schreiben am Abend. Daher sieht ID 3 noch eine 0.

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

Am dritten Tag lesen beide Ort 2, an dem eine 1 notiert ist.

Am vierten Tag antwortet ID 0, dass es 2 Mitglieder gibt (korrekt), während ID 3 die 1 an Ort 1 liest. ID 0 beendet unmittelbar danach und nimmt an den kommenden Tagen nicht mehr teil.

Schließlich, an Tag $D = 5$, antwortet auch das verbleibende Mitglied korrekt $N = 2$.

Testen

Um das Testen deiner Lösung zu erleichtern, stellen wir ein einfaches Tool bereit, das du über CMS herunterladen kannst. Das Tool ist optional. Beachte, dass der offizielle Grader auf CMS ein anderer ist als das Test-Tool.

Um das Tool zu benutzen, brauchst du eine Eingabedatei. Du kannst die bereitgestellten Beispiel-Eingaben `census.input0.txt` und `census.input1.txt` verwenden oder eigene erstellen. Die Eingabedatei sollte mit der Anzahl der Mitglieder N und den möglichen IDs M beginnen, gefolgt von einer Zeile mit N Zahlen, die die IDs der Mitglieder des Geheimbunds angeben.

Für Python-Programme, sagen wir `census.py` (normalerweise ausgeführt als `pypy3 census.py`), führe das Test-Tool wie folgt aus:

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

Für C++-Programme kompiliere zuerst deine Lösung:

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

und führe dann das Test-Tool aus:

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

Beachte, dass bei diesem Problem die Standardausgabe für die Kommunikation mit dem Grader verwendet wird, sie sollte daher nicht zum Debuggen genutzt werden. Verwende stattdessen die Standardfehlerausgabe (stderr). In C++ kannst du `cerr << msg << endl;` nutzen. In Python kannst du `print(msg, file=sys.stderr)` verwenden.

Das Test-Tool liest diese stderr-Nachrichten und stellt sie zusammen mit den Anfragen deiner Programm-Instanzen dar. Beachte, dass sie aus technischen Gründen möglicherweise leicht asynchron zueinander angezeigt werden.