D. Sčítání lidu (census)

Časový limit: 1 sekund
Paměťový limit: 128 MiB

Málo známý fakt o Cesenaticu je, že je domovem tajné společnosti N informatiček. Tato společnost je vskutku velmi tajná, neboť žádná členka nezná žádnou jinou. Každá členka má unikátní ID, nezáporné celé číslo I.

Jediná komunikace mezi členkami probíhá nepřímo, pomocí čísel načmáraných křídou na různých místech po městě. Každých 100 let provádí společnost sčítání, aby spočítala své členky. Po dokončení sčítání by každá členka měla znát celkový počet členek ve společnosti.

Sčítání probíhá po několik dní. Každý den si každá členka, která se procesu stále účastní, vybere a provede právě jednu akci: **číst**, **psát**, nebo se **přestat** účastnit.

- Pokud se členka rozhodne **číst**, zvolí si místo P. Za bílého dne navštíví místo P a přečte si číslo, které je tam napsané.
- Pokud se členka rozhodne **psát**, zvolí si místo P a číslo V. Pozdě večer navštíví místo P a změní číslo, které tam bylo napsáno, na V. Protože už je tma, nemůže si staré číslo přečíst předtím, než napíše nové.
- Pokud se členka rozhodne **přestat**, v následujících dnech už nedělá žádné akce.

Pokud by jedna členka viděla druhou psát číslo, mohla by ji poznat. Proto je přísně zakázáno, aby se dvě nebo více členek rozhodlo psát na stejné místo ve stejný den. (Pro čtení takové omezení neplatí, protože to lze udělat nenápadně.)

Pokud jedna nebo více členek čte z místa, kde chce jiná členka ten samý den psát, všechna čtení proběhnou před psaním.

Jak by měla společnost naplánovat proces sčítání, aby minimalizovala počet dní, než se všechny dozví správný počet členek?

Implementace

⇨ Toto je interaktivní úloha, ve které bude současně spuštěn neznámý počet instancí ($1 \leq N \leq 100$) vašeho programu. Každá instance simuluje jednu členku společnosti.

Ve městě 10^{18} míst. Číslo místa P musí splňovat $0 \leq P < 10^{18}$. Na začátku je na všech místech napsaná hodnota $V = 0$.

Nová hodnota V napsaná na nějakém místě musí být vždy celé číslo takové, že $0 \leq V \leq 10^9$. Ve většině podúloh může být V pouze 0 nebo 1. Více podrobností naleznete v sekci Bodování.

Když instance vašeho programu začne běžet, měla by nejprve načíst řádek se dvěma celými čísly, I a M ($0 \leq I \leq M - 1$), která označují unikátní ID členky společnosti reprezentované touto instancí a celkový počet možných ID. V rámci každého vstupu dostanou všechny instance stejnou hodnotu M a různé hodnoty I. Upozorňujeme, že mohou existovat ID, která nejsou přiřazena žádné člence.

Následně by si váš program pro každý den procesu sčítání měl vybrat akci, kterou chce provést, a podle toho vypsat řádek:

Akce	Význam
`r` P	**Přečíst** místo P. Po vypsání tohoto řádku by měl váš program načíst řádek s aktuální hodnotou napsanou na P.
`w` P V	**Napsat** na místo P novou hodnotu V. Pokud více instancí píše na stejné P v ten samý den, získáte verdikt *Nesprávně* (*Not correct*). Až na příklady a podúlohu 3 musíte vypsat $0 \leq V \leq 1$, viz sekce Bodování.
`!` N	**Odpovědět a přestat**: nahlásit, že je celkem N členek a přestat se sčítání účastnit. Po odpovědi **by váš program měl normálně skončit**. (Upozorňujeme, že ostatní instance vašeho programu mohou dále běžet po více dní, než odpoví a skončí.)

Pokud jakákoliv instance vašeho programu odpoví špatnou hodnotou N, poruší protokol, použije více než 500 dní nebo překročí časový/paměťový limit (na proces), bude vaše řešení pro daný vstup ohodnoceno jako *Nesprávně* (*Not correct*).

V opačném případě bude váš program na vstupu ohodnocen jako *Správně* (*Correct*) nebo *Částečně správně* (*Partially Correct*) a obodován na základě hodnoty D označující maximální počet dní, který nějaká instance potřebovala k odpovědi. Pro plný počet bodů musíte vyřešit každý vstup s $D \leq 61$ a $V \leq 1$. Podrobnosti naleznete v sekci Bodování.

Vyprázdnění bufferu (flushing). Pokud nepoužíváte poskytnuté šablony, ujistěte se, že po vypsání každého řádku vyprázdníte standardní výstup (flush), jinak by váš program mohl být ohodnocen jako *Nesprávně* (*Not correct*). V Pythonu k tomu dochází automaticky, pokud k načítání řádků používáte `input()`. V C++ `cout << endl;` vyprázdní buffer navíc k vypsání nového řádku. Pokud používáte printf, použijte `fflush(stdout)`.

Omezení

- $1 \leq N \leq 100$.
- $1 \leq M \leq 100\,000$.
- Můžete využít nejvýše 500 dní.

Bodování

Váš program bude otestován na několika vstupech rozdělených do podúloh. Pro získání bodů za podúlohu musíte správně vyřešit všechny vstupy, které obsahuje.

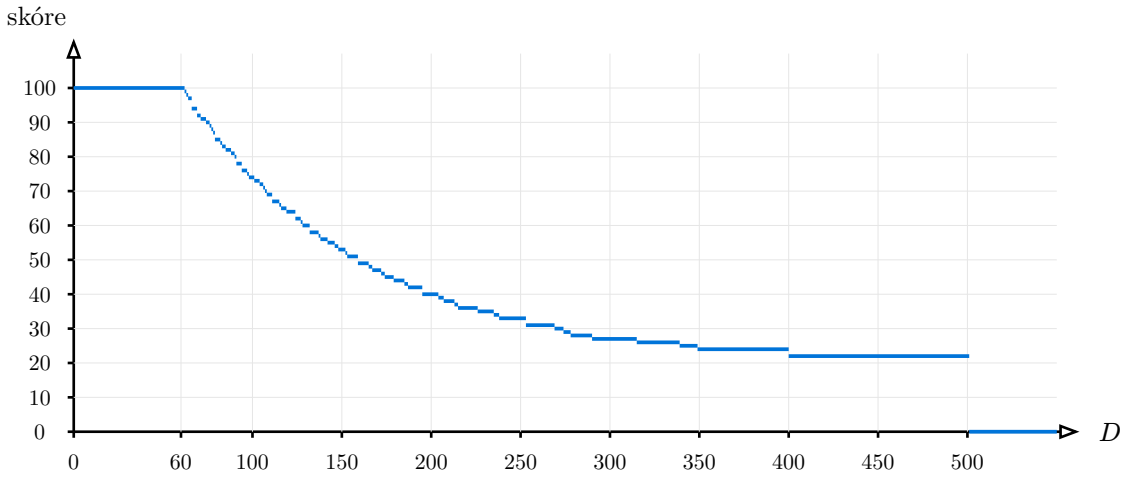
- **Podúloha 0 [0 bodů]**: Příklady (můžete psát libovolné celé číslo $0 \leq V \leq 1\,000\,000\,000$).
- **Podúloha 1 [11 bodů]**: $M \leq 100$ a N členek má ID $0, 1, ..., N-1$.
- **Podúloha 2 [12 bodů]**: $1 \leq N \leq 2$.
- **Podúloha 3 [22 bodů]**: $M \leq 8000$ a můžete psát libovolné celé číslo $0 \leq V \leq 1\,000\,000\,000$.
- **Podúloha 4 [55 bodů]**: Žádná další omezení.

V podúlohách 1, 2 a 4 můžete v každé akci Napsat psát pouze $V = 0$ nebo $V = 1$.

Nechť X_s jsou maximální body za podúlohu s (viz výše) a D_s je největší počet dní, který kterýkoli z vašich programů použije na vstup v podúloze s. Pak:

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

Hodnota skóre_s je zaokrouhlena na nejbližší celé číslo za každou podúlohu a vaše celkové skóre je jejich součtem. Abyste získali plný počet bodů za úlohu, musíte mít $D \leq 61$ a $V \leq 1$ pro každý vstup.



Obrázek 1: Celkové skóre za předpokladu, že každou podúlohu vyřešíte se stejným maximálním D.

Příklady

První příklad. Každá dvojice sloupců ukazuje komunikaci mezi graderem a jednou instancí.

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				

Druhý příklad.

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

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

Vysvětlení

První příklad. Máme $N = 5$ členek s po sobě jdoucími ID $0, 1, 2, 3, 4$ a $M = 100$ (možné v podúlohách 1, 3 a 4). Instance i odpovídá člence s ID i. Výše uvedená interakce je pouze jedna z možných platných posloupností operací a **není** zamýšlena jako efektivní nebo smysluplná strategie, je ukázána pouze pro ilustraci toho, jak funguje protokol.

Druhý příklad. Máme $N = 2$ členky s ID 0 a 3 a $M = 8000$ (možné v podúlohách 2, 3 a 4). První den členka s ID 0 napíše 0 na místo 0 (žádná změna) a členka s ID 3 napíše 1 na místo 2.

místo	0	1	2	3	4	...
číslo	0	0	1	0	0	...

Druhý den ID 0 napíše 1 na místo 1 a ID 3 čte na tom samém místě. Upozorňujeme, že čtení probíhá během dne, před večerním psaním. Proto ID 3 stále vidí 0.

místo	0	1	2	3	4	...
číslo	0	1	1	0	0	...

Třetí den obě čtou z místa 2, kde je napsána 1.

Čtvrtý den ID 0 odpoví, že jsou 2 členky (správně), zatímco ID 3 čte 1 na místě 1. ID 0 ihned poté skončí a neúčastní se v nadcházejících dnech.

Konečně, v den $D = 5$, zbývající členka také správně odpoví $N = 2$.

Testování

Pro usnadnění testování vašeho řešení poskytujeme jednoduchý nástroj, který si můžete stáhnout z CMS. Použití nástroje je volitelné. Vezměte na vědomí, že oficiální grader v CMS se od testovacího nástroje liší.

K použití nástroje potřebujete vstupní soubor. Můžete použít poskytnuté ukázkové vstupy `census.input0.txt` a `census.input1.txt`, nebo si vytvořit vlastní. Vstupní soubor by měl začínat počtem členek N a možných ID M, po kterých by měl následovat řádek s N čísly udávajícími ID členek společnosti.

Pro programy v Pythonu, řekněme `census.py` (normálně spouštěné jako `pypy3 census.py`) spusťte testovací nástroj takto:

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

Pro programy v C++ nejprve zkompilujte své řešení:

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

a poté spusťte testovací nástroj:

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

Vezměte na vědomí, že v této úloze se standardní výstup používá ke komunikaci s graderem, takže by neměl být používán pro ladění. Místo toho můžete použít standardní chybový výstup (stderr). V C++ můžete použít `cerr << msg << endl;`. V Pythonu můžete použít `print(msg, file=sys.stderr)`.

Testovací nástroj přečte a zobrazí tyto zprávy v stderr společně s dotazy provedenými všemi instancemi vašeho programu. Upozorňujeme, že z technických důvodů se mohou objevit mírně zpřeházené.