

D. Census (census)

Tidsgräns: 1 sekunder

Minnesgräns: 128 MiB

Ett mindre känt faktum om Cesenatico är att det är hem till ett hemligt sällskap av N kvinnliga informatiker. Det här sällskapet är verkligen väldigt hemligt; ingen medlem känner till identiteten på någon annan.

Den enda kommunikationen mellan medlemmarna sker indirekt, via siffror klottrade med krita på olika platser runt om i staden. Vart hundra år genomför sällskapet en folkräkning för att räkna sina medlemmar. När folkräkningen är klar ska varje medlem veta det totala antalet medlemmar i sällskapet.

Folkräkningen pågår över flera dagar. Varje dag kommer varje medlem som fortfarande deltar i processen att välja och utföra exakt en handling: att **läsa**, att **skriva** eller att **sluta** delta.

- Om en medlem väljer att **läsa**, väljer hon en plats P . Under dagen besöker hon platsen P och läser siffran som står skriven där.
- Om en medlem väljer att **skriva**, väljer hon en plats P och ett tal V . Sent på kvällen besöker hon platsen P och ändrar talet som stod där till V . Eftersom det redan blivit mörkt kan hon inte läsa det gamla talet innan hon skriver det nya.
- Om en medlem väljer att **sluta**, utför hon inga fler handlingar under de kommande dagarna.

Om en medlem ser en annan skriva ett tal skulle hon kunna lista ut hennes identitet. Därför är det strängt förbjudet för två eller fler medlemmar att välja att skriva på samma plats under en och samma dag. (Det finns ingen sådan begränsning för att läsa, eftersom det kan göras diskret.)

Om en eller flera medlemmar läser från en plats där en annan medlem vill skriva samma dag, sker alla läsningar innan skrivningen.

Hur ska sällskapet planera sin folkräkning för att minimera antalet dagar tills alla har listat ut det korrekta antalet medlemmar?

Implementering

⇒ Det här är ett interaktivt problem, där ett okänt antal instanser ($1 \leq N \leq 100$) av ditt program kommer att köras samtidigt. Varje instans simulerar en medlem i sällskapet.

Det finns 10^{18} platser. Numret P för en plats måste uppfylla $0 \leq P < 10^{18}$. Från början är värdet som står skrivet på alla platser $V = 0$.

Det nya värdet V som skrivs på en plats måste alltid vara ett heltal sådant att $0 \leq V \leq 10^9$. I de flesta testgrupper kan V bara vara 0 eller 1. Läs sektionen om Poängsättning för mer detaljer.

När en instans av ditt program startar ska det först läsa en rad med två heltal, I och M ($0 \leq I \leq M - 1$): det unika ID:t för den medlem av sällskapet som instansen representerar, och det totala antalet möjliga ID:n. I varje testfall får alla instanser samma värde på M och unika värden på I . Tänk på att det kan finnas ID:n som inte har delats ut till någon medlem.

Sen, för varje dag som folkräkningen pågår, ska ditt program välja vilken handling det vill utföra och printa en rad enligt följande:

Handling	Betydelse
$r \ P$	Läs från plats P . Efter att ha printat den här raden ska ditt program läsa in en rad med det nuvarande värdet som står på P .
$w \ P \ V$	Skriv det nya värdet V på plats P . Om flera instanser skriver på samma P under en och samma dag får du bedömningen <i>Not correct</i> . Förutom i exemplen och i testgrupp 3 måste du skriva $0 \leq V \leq 1$; se sektionen för Poängsättning.
$! \ N$	Svara och sluta: rapportera att det finns N medlemmar och sluta delta i folkräkningen. Efter att du svarat ska ditt program avslutas normalt . (Notera att andra instanser av ditt program kan fortsätta köra i ytterligare dagar innan de svarar och avslutas.)

Om någon instans av ditt program svarar med fel värde på N , bryter mot protokollet, använder mer än 500 dagar, eller överskrider tids- eller minnesgränsen (per process), kommer din inskickning att bedömas som *Not correct* för det testfallet.

Annars kommer ditt program att få (*Partially*) *Correct* på testfallet och poängsättas baserat på värdet D : det största antalet dagar det tog för någon instans att svara. För full poäng måste du lösa varje testfall med $D \leq 61$ och $V \leq 1$. Kolla in sektionen för Poängsättning för mer detaljer.

Flushning. Om du inte använder de medföljande mallarna måste du se till att flusha standard output efter att du printat varje rad, annars kan ditt program dömas som *Not correct*. I Python sker detta automatiskt om du använder `input()` för att läsa rader. I C++ flushar `cout << endl`; utöver att skriva en ny rad. Om du använder `printf`, använd `fflush(stdout)`.

Begränsningar

- $1 \leq N \leq 100$.
- $1 \leq M \leq 100\,000$.
- Du får använda som mest 500 dagar.

Poängsättning

Ditt program kommer att testas på flera testfall som är grupperade i testgrupper. För att få poängen för en testgrupp måste du lösa alla testfall som ingår i den.

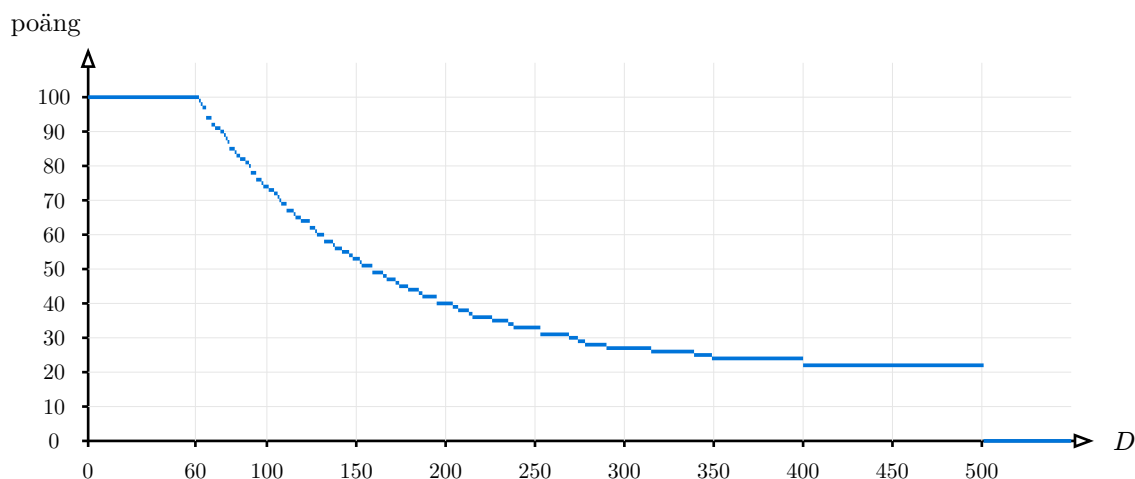
- **Testgrupp 0 [0 poäng]:** Exempel (du kan skriva valfritt heltal $0 \leq V \leq 1\,000\,000\,000$).
- **Testgrupp 1 [11 poäng]:** $M \leq 100$, och de N medlemmarna har ID:n $0, 1, \dots, N - 1$.
- **Testgrupp 2 [12 poäng]:** $1 \leq N \leq 2$.
- **Testgrupp 3 [22 poäng]:** $M \leq 8000$, och du kan skriva valfritt heltal $0 \leq V \leq 1\,000\,000\,000$.
- **Testgrupp 4 [55 poäng]:** Inga ytterligare begränsningar.

I testgrupp 1, 2 och 4 får du bara skriva $V = 0$ eller $V = 1$ i varje skriv-handling.

Låt X_s vara maxpoängen för testgrupp s (visas ovan), och D_s det största antalet dagar som något av dina program använder på ett test i testgrupp s . Då gäller:

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

Värdet för score_s avrundas till närmaste heltal per testgrupp, och din totala poäng är summan av dessa. För full poäng behöver du $D \leq 61$ och $V \leq 1$ på varje testfall.



Figur 1: Totalpoäng, förutsatt att varje testgrupp löses med samma maximala D .

Exempel

Första exemplet. Varje kolumnpar visar kommunikationen mellan domaren och en instans.

Dom.	Inst. 0	Dom.	Inst. 1	Dom.	Inst. 2	Dom.	Inst. 3	Dom.	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				

Andra exemplet.

Domare	Instans 0
0 8000	
	w 0 0
	w 1 1
	r 2
1	
	! 2

Domare	Instans 1
3 8000	
	w 2 1
	r 1
0	
	r 2
1	
	r 1
1	
	! 2

Förklaring

Första exemplet. Vi har $N = 5$ medlemmar med de på varandra följande ID:na 0, 1, 2, 3, 4 och $M = 100$ (giltigt för testgrupp 1, 3 och 4). Instans i motsvarar medlemmen med ID i . Interaktionen ovan är bara en möjlig giltig sekvens av operationer och det är **inte** tanken att vara en effektiv eller vettig strategi; den visas bara för att exemplifiera hur protokollet fungerar.

Andra exemplet. Vi har $N = 2$ medlemmar, med ID:n 0 och 3, samt $M = 8000$ (giltigt för testgrupp 2, 3 och 4). På den första dagen skriver medlemmen med ID 0 en nolla på plats 0 (ingen ändring), och medlemmen med ID 3 skriver en etta på plats 2.

plats	0	1	2	3	4	...
värde	0	0	1	0	0	...

På den andra dagen skriver ID 0 en etta på plats 1, och ID 3 läser från samma plats. Märk att läsningen sker på dagen, innan skrivningen på kvällen. Alltså ser ID 3 fortfarande en nolla.

plats	0	1	2	3	4	...
värde	0	1	1	0	0	...

På den tredje dagen läser båda två från plats 2, där det nu står en etta.

På den fjärde dagen svarar ID 0 att det finns 2 medlemmar (vilket stämmer), medan ID 3 läser ettan på plats 1. ID 0 avslutas direkt efter detta och deltar inte under kommande dagar.

Slutligen, på dag $D = 5$, svarar den kvarvarande medlemmen också korrekt att $N = 2$.

Testning

För att göra det lättare att testa din lösning tillhandahåller vi ett enkelt verktyg som du kan ladda ner från CMS. Verktöget är frivilligt att använda. Tänk på att den officiella domaren på CMS skiljer sig från testverktöget.

För att använda verktöget behöver du en indatafil. Du kan använda de medföljande exempelfilerna `census.input0.txt` och `census.input1.txt`, eller skapa en egen. Indatafilen ska börja med antalet medlemmar N och antalet möjliga ID:n M , följt av en rad med N tal som anger ID:na för sällskapets medlemmar.

För Python-program, låt säga `census.py` (körs normalt som `pypy3 census.py`), kör du testverktöget så här:

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

För C++-program, kompilera först din lösning:

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

och kör sedan testverktyget:

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

Notera att i det här problemet används standard output (stdout) för att kommunicera med domaren, så det ska inte användas för debuggning. Istället kan du använda standard error (stderr). I C++ kan du använda `cerr << msg << endl;`. I Python kan du använda `print(msg, file=sys.stderr)`.

Testverktyget kommer att läsa och visa dessa stderr-meddelanden tillsammans med de förfrågningar (queries) som görs av alla dina programinstanser. Tänk på att de av tekniska skäl kan dyka upp lite ur synk med varandra.