

D. Popis (census)

Časovna omejitev: 1 sekunde

Pomnilniška omejitev: 128 MiB

Manj znano dejstvo o mestu Cesenatico je, da je dom tajne družbe N informatičark. Ta družba je zares zelo skrivnostna; nobena članica ne pozna nobene druge. Vsaka članica ima edinstven ID: nenegativno celo število I .

Edina komunikacija med članicami poteka posredno, prek števil, napisanih s kredo na različnih lokacijah po mestu. Vsakih 100 let družba izvede popis, da prešteje svoje članice. Po končanem popisu mora vsaka članica vedeti skupno število članic v družbi.

Popis poteka več dni. Vsak dan vsaka članica, ki še vedno sodeluje v procesu, izbere in izvede natanko eno dejanje: **branje**, **pisanje** ali **prenehanje** sodelovanja.

- Če se članica odloči za **branje**, izbere lokacijo P . Čez dan obišče lokacijo P in prebere številko, ki je tam napisana.
- Če se članica odloči za **pisanje**, izbere lokacijo P in številko V . Pozno zvečer obišče lokacijo P in spremeni tam zapisano številko na V . Ker je že temno, ne more prebrati stare številke, preden zapiše novo.
- Če se članica odloči za **prenehanje**, v naslednjih dneh ne izvaja več nobenih dejanj.

Če ena članica vidi drugo, kako piše številko, jo lahko prepozna. Zato je strogo prepovedano, da bi dve ali več članic izbralo pisanje na isti lokaciji isti dan. (Za branje takšne omejitve ni, saj se to lahko stori diskretno.)

Če ena ali več članic bere z lokacije, kamor želi druga članica pisati isti dan, se vsa branja zgodijo pred pisanjem.

Kako naj družba načrtuje postopek popisa, da čim bolj zmanjša število dni, dokler vsi ne izvedo pravilnega števila članic?

Implementacija

⇒ To je interaktivni problem, pri katerem se bo hkrati izvajalo neznano število primerkov ($1 \leq N \leq 100$) tvojega programa. Vsak primerek simulira eno članico družbe.

Na voljo je 10^{18} lokacij. Številka lokacije P mora zadoščati $0 \leq P < 10^{18}$. Na začetku je vrednost, zapisana na vseh lokacijah, $V = 0$.

Nova vrednost V , zapisana na lokaciji, mora biti vedno celo število, tako da $0 \leq V \leq 10^9$. V večini podnalog je V lahko le 0 ali 1. Za več podrobnosti glej razdelek Točkovanje.

Ko se začne izvajati primerek tvojega programa, mora najprej prebrati vrstico z dvema celima številoma, I in M ($0 \leq I \leq M - 1$): edinstven ID članice družbe, ki jo predstavlja ta primerek, in skupno število možnih ID-jev. Znotraj vsakega testnega primera bodo vsi primerki prejeli enako vrednost M in različne vrednosti I . Upoštevaj, da morda obstajajo ID-ji, ki niso dodeljeni nobeni članici.

Nato mora tvoj program za vsak dan v postopku popisa izbrati dejanje, ki ga želi izvesti, in ustrezno izpisati vrstico:

Dejanje	Pomen
$r\ P$	Branje lokacije P . Po izpisu te vrstice mora tvoj program prebrati vrstico s trenutno vrednostjo, zapisano na P .
$w\ P\ V$	Pisanje na lokaciji P nove vrednosti V . Če več primerkov piše na isti P isti dan, boš prejela razsodbo <i>Ni pravilno</i> (Not correct). Razen pri primerih in podnalogi 3 moraš zapisati $0 \leq V \leq 1$; glej razdelek Točkovanje.
$!\ N$	Odgovori in končaj : poroča, da je N članic, in preneha s sodelovanjem v popisu. Po odgovoru mora tvoj program normalno končati izvajanje . (Upoštevaj, da lahko drugi primerki tvojega programa še naprej delujejo še nekaj dni, preden odgovorijo in končajo.)

Če kateri koli primerek tvojega programa odgovori z napačno vrednostjo N , krši protokol, porabi več kot 500 dni ali preseže časovno/pomnilniško omejitev (na proces), bo tvoja rešitev za dani testni primer ocenjena kot *Ni pravilno*.

V nasprotnem primeru bo tvoj program pri testnem primeru (*Delno*) *pravilen* in točkovan na podlagi vrednosti D : največje število dni, ki jih je kateri koli primerek potreboval za odgovor. Za polne točke moraš vsak testni primer rešiti z $D \leq 61$ in $V \leq 1$. Za podrobnosti glej razdelek Točkovanje.

Čiščenje izhoda (Flushing). Če ne uporabljaš priloženih predlog, poskrbi, da po izpisu vsake vrstice počistiš (flush) standardni izhod, sicer bo tvoj program morda ocenjen kot *Ni pravilno*. V Pythonu se to zgodi samodejno, če za branje vrstic uporabljaš `input()`. V C++ `cout << endl`; poleg izpisa nove vrstice počisti tudi medpomnilnik; če uporabljaš `printf`, uporabi `fflush(stdout)`.

Omejitve

- $1 \leq N \leq 100$.
- $1 \leq M \leq 100\,000$.
- Uporabiš lahko največ 500 dni.

Točkovanje

Tvoj program bo testiran na več testnih primerih, razdeljenih v podnaloge. Za pridobitev točk za podnalogo moraš pravilno rešiti vse teste, ki jih vsebuje.

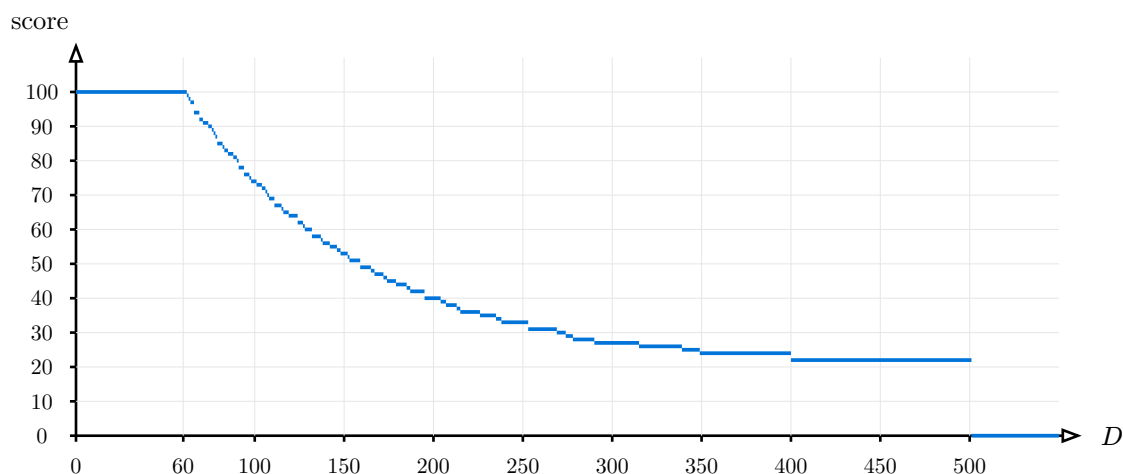
- **Podnaloga 0 [0 točk]**: Primeri (zapišeš lahko katero koli celo število $0 \leq V \leq 1\,000\,000\,000$).
- **Podnaloga 1 [11 točk]**: $M \leq 100$, in N članic ima ID-je $0, 1, \dots, N - 1$.
- **Podnaloga 2 [12 točk]**: $1 \leq N \leq 2$.
- **Podnaloga 3 [22 točk]**: $M \leq 8000$, in zapišeš lahko katero koli celo število $0 \leq V \leq 1\,000\,000\,000$.
- **Podnaloga 4 [55 točk]**: Brez dodatnih omejitev.

V podnalogah 1, 2 in 4 lahko pri vsakem dejanju pisanja zapišeš le $V = 0$ ali $V = 1$.

Naj bo X_s največje število točk za podnalogo s (prikazano zgoraj), in D_s največje število dni, ki jih kateri koli tvoj program porabi pri testu v podnalogi s . Potem:

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

Vrednost score_s se zaokroži na najbližje celo število za vsako podnalogo, tvoj skupni rezultat pa je vsota teh. Za polne točke pri nalogi moraš pri vsakem testnem primeru doseči $D \leq 61$ in $V \leq 1$.



Slika 1: Skupni rezultat, ob predpostavki, da je vsaka podnaloga rešena z istim največjim D .

Primeri vhoda/izhoda

Prvi primer. Vsak par stolpcev prikazuje komunikacijo med ocenjevalcem in enim primerkom.

Ocen.	Prim.	Ocen.	Prim.	Ocen.	Prim.	Ocen.	Prim.	Ocen.	Prim.
0	1	2	3	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							

Drugi primer.

Ocenjevalec	Primerek 0	Ocenjevalec	Primerek 1
0 8000	w 0 0	3 8000	w 2 1
	w 1 1		r 1
	r 2	0	r 2
1	! 2	1	r 1
		1	! 2

Razlaga

Prvi primer. Imamo $N = 5$ članic z zaporednimi ID-ji 0, 1, 2, 3, 4 in $M = 100$ (veljavno za podnaloge 1, 3 in 4). Primerek i ustreza članici z ID i . Zgornja interakcija je le eno od možnih zakonitih zaporedij operacij in **ni** mišljena kot učinkovita ali smiselna strategija; prikazana je le za ilustracijo delovanja protokola.

Drugi primer. Imamo $N = 2$ članici, z ID-jema 0 in 3, in $M = 8000$ (veljavno za podnaloge 2, 3 in 4). Prvi dan članica z ID 0 zapiše 0 na lokacijo 0 (brez spremembe), članica z ID 3 pa zapiše 1 na lokacijo 2.

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

Drugi dan ID 0 zapiše 1 na lokacijo 1, ID 3 pa bere z iste lokacije. Upoštevaj, da se branje zgodi čez dan, pred pisanjem zvečer. Zato ID 3 še vedno vidi 0.

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

Tretji dan obe bereta z lokacije 2, kjer je zapisana 1.

Četrty dan ID 0 odgovori, da je članic 2 (pravilno), medtem ko ID 3 bere 1 na lokaciji 1. ID 0 takoj po tem konča in v naslednjih dneh ne sodeluje več.

Končno, peti dan ($D = 5$), preostala članica prav tako pravilno odgovori, da je $N = 2$.

Testiranje

Za lažje testiranje svoje rešitve nudimo preprosto orodje, ki si ga lahko preneseš s sistema CMS. Uporaba orodja ni obvezna. Upoštevaj, da se uradni ocenjevalnik na CMS razlikuje od testnega orodja.

Za uporabo orodja potrebuješ vhodno datoteko. Uporabiš lahko priložene primere vhodnih datotek `census.input0.txt` in `census.input1.txt` ali pa ustvariš svojo. Vhodna datoteka se mora začeti s številom članic N in možnimi ID-ji M , čemur sledi vrstica z N števili, ki določajo ID-je članic družbe.

Za programe v Pythonu, recimo `census.py` (običajno se izvaja z `pypy3 census.py`), testno orodje zaženeš takole:

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

Za programe v C++, najprej prevedi svojo rešitev:

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

in nato zaženi testno orodje:

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

Upoštevaj, da se pri tem problemu standardni izhod uporablja za komunikacijo z ocenjevalnikom, zato ga ne uporabljaš za razhroščevanje (debugiranje). Namesto tega lahko uporabiš standardni izhod za napake (stderr). V C++ lahko uporabiš `cerr << msg << endl;`. V Pythonu lahko uporabiš `print(msg, file=sys.stderr)`.

Testno orodje bo prebralo in prikazalo ta sporočila iz stderr skupaj s poizvedbami, ki jih izvajajo vsi primerki tvojega programa. Upoštevaj, da so zaradi tehničnih razlogov lahko prikazana nekoliko časovno zamaknjena.