

## D. Recenseamento (census)

Limite de tempo: 1 segundos

Limite de memória: 128 MiB

Um facto pouco conhecido sobre Cesenatico é que alberga uma sociedade secreta de  $N$  informáticas. Esta sociedade é, de facto, muito secreta; nenhum membro conhece outro. Cada membro tem um único ID: um inteiro  $I$  tal que  $0 \leq I \leq M - 1$ .

A única comunicação entre os membros é indireta, através de números rabiscados com giz em diferentes locais da cidade. De 100 em 100 anos, a sociedade realiza um Recenseamento para contar os seus membros. Após o recenseamento estar concluído, cada membro deve saber o número total de membros na sociedade.

O recenseamento decorre ao longo de vários dias. Cada dia, cada membro que ainda esteja a participar no processo escolhe e realiza exatamente uma ação: **ler**, **escrever**, ou **parar** de participar.

- Se um membro escolhe **ler**, ela escolhe um local  $P$ . Durante o dia, ela visita o local  $P$  e lê o número lá escrito.
- Se um membro escolhe **escrever**, ela escolhe um local  $P$  e um número  $V$ . Ao final da tarde, ela visita o local  $P$  e altera o número lá escrito para  $V$ . Como já está escuro, ela não consegue ler o número antigo antes de escrever o novo.
- Se um membro escolhe **parar**, ela não realiza mais ações nos dias seguintes.

Se um membro vir outro a escrever um número, poderá descobrir a sua identidade. Portanto, é estritamente proibido que dois ou mais membros escolham escrever no mesmo local no mesmo dia. (Não existe tal restrição para a leitura, pois pode ser feita de forma discreta.)

Se um ou mais membros lerem de um local onde outro membro quer escrever no mesmo dia, todas as leituras ocorrem antes da escrita.

Como deve a sociedade planear o seu processo de recenseamento para minimizar o número de dias até que todos saibam a contagem correta de membros?

### Implementação

⇒ Este é um problema interativo, no qual um número desconhecido de instâncias ( $1 \leq N \leq 100$ ) do teu programa será executado simultaneamente. Cada instância simula um membro da sociedade.

Existem  $10^{18}$  locais. O número  $P$  de um local deve satisfazer  $0 \leq P < 10^{18}$ . Inicialmente, o valor escrito em todos os locais é  $V = 0$ .

O novo valor  $V$  escrito num local deve ser sempre um número inteiro tal que  $0 \leq V \leq 10^9$ . Na maioria das sub-tarefas,  $V$  só pode ser 0 ou 1. Vê a secção de Pontuação para mais detalhes.

Quando uma instância do teu programa arranca, deve primeiro ler uma linha com dois inteiros,  $I$  e  $M$  ( $0 \leq I \leq M - 1$ ): o ID único do membro da sociedade representado por esta instância e o número total de IDs possíveis. Dentro de cada caso de teste, todas as instâncias receberão o mesmo valor  $M$  e valores  $I$  distintos. Nota que pode haver IDs que não estão atribuídos a nenhum membro.

Depois, para cada dia no processo de recenseamento, o teu programa deve escolher a ação que pretende realizar e imprimir uma linha correspondente:

Ação	Significado
$r\ P$	<b>Ler</b> o local $P$ . Após imprimir esta linha, o teu programa deve ler uma linha com o valor atual escrito em $P$ .
$w\ P\ V$	<b>Escrever</b> no local $P$ o novo valor $V$ . Se múltiplas instâncias escreverem no mesmo $P$ no mesmo dia, receberás o veredito <i>Not correct</i> . Exceto nos exemplos e na sub-tarefa 3, tens de escrever $0 \leq V \leq 1$ ; vê a secção de Pontuação.
$! N$	<b>Responder e parar:</b> reporta que existem $N$ membros e pára de participar no recenseamento. Após responder, <b>o teu programa deve terminar normalmente</b> . (Nota que outras instâncias do teu programa podem continuar a correr durante dias adicionais antes de responderem e terminarem.)

Se alguma instância do teu programa responder com o valor errado de  $N$ , violar o protocolo, usar mais de 500 dias, ou exceder o limite de tempo/memória (por processo), a tua submissão será julgada como *Not correct* para o caso de teste dado.

Caso contrário, o teu programa será (*Partially*) *Correct* no caso de teste e pontuado com base no valor  $D$ : o número máximo de dias que qualquer instância demorou a responder. Para a pontuação máxima, precisas de resolver todos os casos de teste com  $D \leq 61$  e  $V \leq 1$ . Vê a secção de Pontuação para detalhes.

**Limpeza do buffer (Flushing).** Se não estiveres a usar os modelos fornecidos, certifica-te de limpar o buffer de standard output após imprimir cada linha, caso contrário o teu programa pode ser julgado como *Not correct*. Em Python, isto acontece automaticamente se usares `input()` para ler linhas. Em C++, `cout << endl;` limpa o buffer além de imprimir uma nova linha; se usares `printf`, usa `fflush(stdout)`.

## Restrições

- $1 \leq N \leq 100$ .
- $1 \leq M \leq 100\,000$ .
- Podes usar no máximo 500 dias.

## Pontuação

O teu programa será testado em vários casos de teste agrupados em sub-tarefas. Para obter a pontuação de uma sub-tarefa, tens de resolver corretamente todos os testes que ela contém.

- **Subtarefa 0 [ 0 pontos]:** Exemplos (podes escrever qualquer inteiro  $0 \leq V \leq 1\,000\,000\,000$ ).
- **Subtarefa 1 [11 pontos]:**  $M \leq 100$ , e os  $N$  membros têm IDs  $0, 1, \dots, N - 1$ .
- **Subtarefa 2 [12 pontos]:**  $1 \leq N \leq 2$ .
- **Subtarefa 3 [22 pontos]:**  $M \leq 8000$ , e podes escrever qualquer inteiro  $0 \leq V \leq 1\,000\,000\,000$ .
- **Subtarefa 4 [55 pontos]:** Sem restrições adicionais.

Nas sub-tarefas 1, 2 e 4, só podes escrever  $V = 0$  ou  $V = 1$  em cada ação de Escrita.

Seja  $X_s$  a pontuação máxima para a sub-tarefa  $s$  (mostrada acima), e  $D_s$  o maior número de dias que qualquer um dos teus programas usa num teste da sub-tarefa  $s$ . Então:

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

O valor de  $\text{score}_s$  é arredondado para o inteiro mais próximo por sub-tarefa, e a tua pontuação total é a soma destas. Para a pontuação máxima, precisas de  $D \leq 61$  e  $V \leq 1$  em todos os casos de teste.

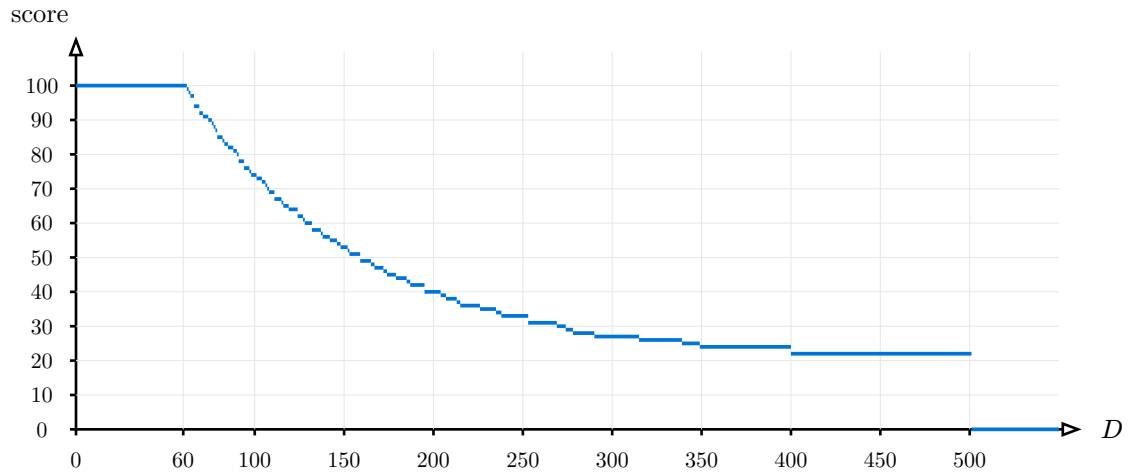


Figura 1: Pontuação total, assumindo que todas as sub-tarefas são resolvidas com o mesmo  $D$  máximo.

## Exemplos

Primeiro exemplo. Cada par de colunas mostra a comunicação entre o avaliador e uma instância.

Avali- ador	Ins- tância 0	Avali- ador	Ins- tância 1	Avali- ador	Ins- tância 2	Avali- ador	Ins- tância 3	Avali- ador	Ins- tância 4
0 100	w 12 1	1 100	w 50 1	2 100	w 99 0	3 100	w 7 1	4 100	r 5
	r 50		r 7		r 12		w 1 1	0	! 5
1	! 5	1	r 1	1	w 0 0		! 5		
		1	! 5		! 5				

Segundo exemplo.

Avaliador	Instância 0
0 8000	
	w 0 0
	w 1 1
	r 2
1	
	! 2

Avaliador	Instância 1
3 8000	
	w 2 1
	r 1
0	
	r 2
1	
	r 1
1	
	! 2

## Explicação

**Primeiro Exemplo.** Temos  $N = 5$  membros com IDs consecutivos 0, 1, 2, 3, 4 e  $M = 100$  (válido para as sub-tarefas 1, 3 e 4). A solução  $i$  corresponde ao membro com ID  $i$ . A interação acima é apenas uma possível sequência legal de operações e **não** pretende ser uma estratégia eficiente ou sensata; é mostrada apenas para ilustrar como funciona o protocolo.

**Segundo Exemplo.** Temos  $N = 2$  membros, com IDs 0 e 3, e  $M = 8000$  (válido para as sub-tarefas 2, 3 e 4). No primeiro dia, o membro com ID 0 escreve um 0 no local 0 (sem alteração), e o membro com ID 3 escreve um 1 no local 2.

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

No segundo dia, o ID 0 escreve um 1 no local 1, e o ID 3 lê esse mesmo local. Nota que a leitura ocorre durante o dia, antes da escrita ao final da tarde. Assim, o ID 3 ainda vê um 0.

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

No terceiro dia, ambos leem o local 2, onde um 1 está escrito.

No quarto dia, o ID 0 responde que existem 2 membros (correto), enquanto o ID 3 lê o 1 no local 1. O ID 0 termina imediatamente após isto e não participa nos dias seguintes.

Finalmente, no dia  $D = 5$ , o membro restante também responde corretamente  $N = 2$ .

## Testes

Para facilitar o teste da tua solução, fornecemos uma ferramenta simples que podes descarregar do CMS. A ferramenta é de uso opcional. Nota que o avaliador oficial no CMS é diferente da ferramenta de teste.

Para usares a ferramenta precisas de um ficheiro de entrada. Podes usar os exemplos fornecidos `census.input0.txt` e `census.input1.txt`, ou criar os teus próprios. O ficheiro de entrada deve começar com o número de membros  $N$  e IDs possíveis  $M$ , seguidos por uma linha com  $N$  números que especificam os IDs dos membros da sociedade.

Para programas em Python, digamos `census.py` (normalmente executado como `pypy3 census.py`) executa a ferramenta de teste da seguinte forma:

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

Para programas em C++, primeiro compila a tua solução:

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

e depois executa a ferramenta de teste:

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

Nota que neste problema a saída padrão é usada para comunicação com o avaliador, pelo que não deve ser usada para depuração. Em vez disso, podes usar a saída de erro padrão (stderr). Em C++ podes usar `cerr << msg << endl;`. Em Python podes usar `print(msg, file=sys.stderr)`.

A ferramenta de teste lerá e apresentará estas mensagens de stderr juntamente com as consultas realizadas por todas as instâncias do teu programa. Nota que, por razões técnicas, podem aparecer ligeiramente dessincronizadas entre si.