

Classroom Game

Uma importante escola secundária de Tashkent celebra o seu aniversário promovendo um jogo entre alunos e professores. Estão N alunos sentados na sala de aula, numerados de 0 a $N - 1$. Cada aluno segura uma folha de papel que contém um array de números inteiros. Inicialmente, cada folha está em branco, pelo que contém um array vazio.

Os alunos estão a jogar o jogo com M professores, numerados de 0 a $M - 1$, e o director. O jogo é jogado em M etapas, também numeradas de 0 a $M - 1$. Na etapa j , o professor j entra na sala de aula e ocorrem os seguintes acontecimentos:

- Alguns alunos (possivelmente nenhum) levantam a mão. É garantido que em cada etapa pelo menos um aluno **não** levanta a mão, sendo que cada aluno pode levantar a mão **no máximo uma vez** durante todo o jogo.
- O professor observa as folhas que os alunos seguram naquele momento e pode **modificar** as folhas dos alunos que **não** levantaram a mão nesta fase. Para cada aluno nesta situação, o professor pode substituir o conteúdo da folha por um novo array (eventualmente vazio) de números inteiros. Cada inteiro deve estar entre 0 e 63, inclusive, e o array pode ter no máximo 63 elementos.
- Após estas modificações, o professor j sai e os alunos trocam as suas folhas de acordo com uma permutação secreta P . Ou seja, cada aluno i ($0 \leq i < N$) entrega a sua folha ao aluno $P[i]$, em que $P[0], P[1], \dots, P[N - 1]$ são N inteiros **distintos** entre 0 e $N - 1$, inclusive. Os valores de P são **fixos** durante todas as etapas do jogo, mas os professores e o diretor não os conhecem.

Assim que todas as M etapas estiverem concluídos e a última troca de acordo com P for realizada, o diretor entra. Observando apenas as folhas que os alunos possuem, o director deve determinar, para cada aluno i ($0 \leq i < N$), o número da etapa j em que o aluno i levantou a mão, ou concluir que o aluno i nunca levantou a mão.

Os professores não podem comunicar com outros professores ou com o diretor, exceto através dos arrays de inteiros escritos nas folhas. Cada professor sabe a etapa em que entra na sala de aula.

A tua tarefa é elaborar e implementar uma estratégia para os professores e para o diretor que identifique corretamente se e quando cada aluno levantou a mão. A tua pontuação dependerá do tamanho máximo de qualquer sequência escrita numa folha: um tamanho máximo mais pequeno leva a uma pontuação maior ou igual.

Detalhes da Implementação

É preciso implementar duas funções, uma para os professores e outra para o diretor.

A função que deves implementar para os **professores** é:

```
std::vector<std::vector<int>> process_step(  
    int N, int M, int R,  
    std::vector<int> T,  
    std::vector<std::vector<int>> A)
```

- N : o número de alunos.
- M : o número de etapas e também o número de professores.
- R : o número da etapa actual (de 0 a $M - 1$).
- T : um array (possivelmente vazio) contendo os índices dos alunos que levantam a mão nesta etapa, ordenado de forma crescente.
- A : um array de tamanho N descrevendo as folhas, onde $A[i]$ ($0 \leq i < N$) é o array de inteiros na folha que o aluno i possui no início da etapa.
- Este procedimento é chamado exactamente M vezes por jogo, pela ordem $R = 0, 1, \dots, M - 1$.

A função deve devolver um array B de tamanho N , representando as folhas após as modificações do professor, no mesmo formato que A .

- Para cada aluno i que levantou a mão (i.e., i aparece em T), a folha não deve ser alterada, pelo que $B[i] = A[i]$.
- Para cada i tal que $0 \leq i < N$, o tamanho de $B[i]$ não deve exceder 63, e cada número inteiro em $B[i]$ deve estar compreendido entre 0 e 63, inclusive.

A função que deves implementar para o **diretor** é:

```
std::vector<int> determine_steps(  
    int N, int M, std::vector<std::vector<int>> A)
```

- N, M : o mesmo que acima.
- A : um array de tamanho N , onde $A[i]$ é o array de inteiros na folha segurado pelo aluno i após todos as M etapas.
- Este procedimento é chamado exactamente uma vez por jogo, após a chamada final a `process_step`.

O procedimento deve devolver um array D de tamanho N . Para cada i tal que $0 \leq i < N$, deve ser válido que

- $D[i] = j$ se o aluno i levantou a mão durante etapa j , ou

- $D[i] = -1$ se o aluno i nunca levantou a mão durante todo o jogo.

O teu programa não está autorizado a armazenar ou transmitir qualquer informação de uma chamada a `process_step` para outra, excepto através do valor de retorno de `process_step`.

Se o teu programa tentar fazer isto, a tua pontuação no CMS poderá estar incorrecta e poderá ser reduzida após o fim da competição. Nota que o avaliador pode executar várias instâncias do teu programa em simultâneo. Isto significa que as chamadas a `process_step` e `determine_steps` do mesmo jogo podem ser executadas em instâncias diferentes, e as chamadas a `process_step` e `determine_steps` de jogos diferentes podem ser executadas na mesma instância. Cada caso de teste inclui até 5 jogos.

Restrições

- $2 \leq N \leq 63$
- $1 \leq M \leq 63$
- Cada aluno levanta a mão **no máximo uma vez** durante todo o jogo. Por outras palavras, para cada aluno i , existe no máximo uma etapa j no qual i levanta a mão.
- Em cada etapa, pelo menos um aluno **não** levanta a mão.

Subtarefas e Pontuação

Subtarefa	Pontos	Restrições Adicionais
1	4	$M = 1$
2	6	$N = 2$
3	9	$P[i] = i$ para todo $0 \leq i \leq N - 1$.
4	25	No máximo um aluno levanta a mão em cada etapa.
5	56	Sem restrições adicionais.

Para cada caso de teste, a tua pontuação é 0 (reportada como `Output isn't correct` no CMS) se o valor de retorno de qualquer chamada ao procedimento `process_step` for inválido ou se o valor de retorno de qualquer chamada ao procedimento `determine_steps` estiver incorrecto.

Caso contrário, seja C o tamanho máximo de **qualquer** array em **qualquer** B devolvido por `process_step`. Então, a pontuação para um caso de teste numa subtarefa com pontuação S é calculada como $S \cdot X$, em que X depende de C de acordo com a seguinte tabela:

Condição	X
$C \leq 2$	1.00
$C = 3$	0.75
$C = 4$	0.55
$5 \leq C \leq 13$	$0.50 - 0.03 \cdot (C - 5)$
$14 \leq C \leq 63$	$0.19 \cdot \frac{64-C}{64-14} + 0.04$

Exemplo

Considera um cenário com $N = 4$ alunos, $M = 2$ professores e permutação $P = [0, 3, 1, 2]$. Inicialmente, todos as folhas estão em branco.

O avaliador chama primeiro:

```
process_step(4, 2, 0, [1], [[], [], [], []])
```

O aluno 1 levantou a mão, pelo que a sua folha não pode ser modificada. O professor 0 pode decidir escrever $[10]$ na folha do aluno 0, escrever $[1, 63, 4]$ na folha do aluno 3 e deixar a folha do aluno 2 em branco. Para tal, o procedimento deve devolver $B = [[10], [], [], [1, 63, 4]]$.

Após o professor 0 sair, os alunos trocam as suas folhas de acordo com P . Após a troca, as folhas são $A = [[10], [], [1, 63, 4], []]$.

O avaliador chama agora:

```
process_step(4, 2, 1, [0, 3], [[10], [], [1, 63, 4], []])
```

Os alunos 0 e 3 levantaram a mão, pelo que as suas folhas não podem ser modificadas. O professor 1 pode decidir escrever $[0, 40]$ na folha do aluno 1 e escrever $[1, 50]$ na folha do aluno 2. Para tal, a função deve devolver $B = [[10], [0, 40], [1, 50], []]$.

Após o professor 1 sair, as folhas são novamente trocadas de acordo com P . Após a troca, as folhas são $A = [[10], [1, 50], [], [0, 40]]$.

Finalmente, o avaliador chama:

```
determine_steps(4, 2, [[10], [1, 50], [], [0, 40]])
```

Este procedimento deve devolver o array $D = [1, 0, -1, 1]$ pois os alunos 0 e 3 levantaram a mão na etapa 1, o aluno 1 levantou a mão na etapa 0 e o aluno 2 nunca levantou a mão. Neste

exemplo, $C = 3$.

Avaliador de Exemplo

Formato de Input:

```
N M
Q[0] Q[1] ... Q[N-1]
P[0] P[1] ... P[N-1]
```

Aqui, Q é um array de tamanho N , em que $Q[i] = j$ se o aluno i levantar a mão durante a etapa j , ou $Q[i] = -1$ se o aluno i nunca levantar a mão durante todo o jogo.

Formato de Output:

Após cada chamada a `process_step`, o avaliador de exemplo apresenta o conteúdo das folhas. Seja K o tamanho de B e $L[i]$ o tamanho de $B[i]$ (para cada $0 \leq i < K$). O avaliador de exemplo imprime B no seguinte formato, **seguido de uma linha em branco**:

```
K
L[0] B[0][0] B[0][1] ... B[0][L[0]-1]
L[1] B[1][0] B[1][1] ... B[1][L[1]-1]
:
L[K-1] B[K-1][0] B[K-1][1] ... B[K-1][L[K-1]-1]
```

O avaliador de exemplo troca então as folhas de acordo com a permutação P . Se $K \neq N$, o avaliador imprime uma mensagem de erro e termina a execução.

Após a chamada `determine_steps`, o avaliador de exemplo produz o seguinte output:

```
C H
D[0] D[1] ... D[H-1]
```

Aqui, H é o tamanho do array D devolvido por `determine_steps`.