

Sfida in classe

Un'importante scuola superiore di Tashkent celebra il suo anniversario organizzando una sfida tra studenti e insegnanti. Ci sono N studenti seduti in classe, numerati da 0 a $N - 1$. Ognuno ha in mano un array di numeri interi, inizialmente vuoto.

Gli studenti sfidano il preside e M insegnanti (numerati da 0 a $M - 1$) nel corso di M turni (numerati da 0 a $M - 1$). Nel turno j , l'insegnante j entra in classe e in ordine:

- Alcuni (eventualmente zero) studenti alzano la mano. In ogni turno almeno uno studente **non** alza la mano, e ogni studente può alzare la mano **al massimo una volta** durante l'intera sfida.
- L'insegnante guarda gli array che gli studenti hanno ora in mano e può **modificare** quelli degli studenti che **non** hanno alzato la mano questo turno. Per ognuno di loro, può sostituire l'array con un qualunque nuovo array (eventualmente vuoto) di al massimo 63 numeri interi compresi tra 0 e 63 (inclusi).
- Dopo le modifiche, l'insegnante se ne va e gli studenti si scambiano gli array secondo una permutazione segreta P : ogni studente i ($0 \leq i < N$) dà il suo array allo studente $P[i]$, dove $P[0], P[1], \dots, P[N - 1]$ sono N numeri interi **distinti** compresi tra 0 e $N - 1$, inclusi. La permutazione è **fissa** durante tutti i turni, ma gli insegnanti e il preside non la conoscono.

Dopo che gli M turni sono finiti, e quindi dopo che l'ultimo scambio di array secondo P è stato effettuato, entra il preside. Osservando solo gli array che gli studenti hanno in mano, il preside deve determinare, per ogni studente i ($0 \leq i < N$), il numero del turno j in cui lo studente i ha alzato la mano, oppure concludere che lo studente i non ha mai alzato la mano.

Gli insegnanti non possono comunicare tra loro o con il preside, se non attraverso gli array. Ogni insegnante sa in quale turno entra in classe.

Implementa una strategia per gli insegnanti e il preside che identifichi correttamente se e quando ogni studente ha alzato la mano. Il tuo punteggio dipenderà dalla lunghezza massima degli array in mano agli studenti: una lunghezza massima inferiore produrrà un punteggio maggiore o uguale.

Implementazione

Devi implementare due funzioni, una per gli insegnanti e una per il preside. La funzione da implementare per gli **insegnanti** è la seguente:

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

- N : il numero di studenti.
- M : il numero di turni e di insegnanti.
- R : il numero del turno corrente (da 0 a $M - 1$).
- T : un array (eventualmente vuoto) con gli indici degli studenti che alzano la mano in questo turno, in ordine crescente.
- A : un array di lunghezza N , dove $A[i]$ ($0 \leq i < N$) è l'array di interi in mano allo studente i all'inizio del turno.
- La funzione viene chiamata esattamente M volte per sfida, nell'ordine $R = 0, 1, \dots, M - 1$.

La funzione deve restituire un array B di lunghezza N , che rappresenta gli array dopo le modifiche apportate dall'insegnante, nello stesso formato di A .

- Per ogni studente i che ha alzato la mano (per cui i compare in T), il foglio non deve essere cambiato, quindi $B[i] = A[i]$.
- Per ogni i tale che $0 \leq i < N$, la lunghezza di $B[i]$ non deve superare 63, e ogni intero in $B[i]$ deve essere compreso tra 0 e 63, inclusi.

La funzione da implementare per il **preside** è la seguente:

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

- N, M : come sopra.
- A : un array di lunghezza N , dove $A[i]$ è l'array dello studente i dopo tutti gli M turni.
- La funzione viene chiamata esattamente una volta per sfida, dopo la chiamata finale a `process_step`.

La funzione deve restituire un array D di lunghezza N , tale che per ogni i con $0 \leq i < N$:

- $D[i] = j$ se lo studente i ha alzato la mano durante il turno j , oppure
- $D[i] = -1$ se lo studente i non ha mai alzato la mano durante l'intera sfida.

Il tuo programma non è autorizzato a memorizzare o trasmettere alcuna informazione da una chiamata a `process_step` a un'altra, se non tramite il valore restituito da `process_step`. Se il tuo programma tenta di farlo ugualmente, il tuo punteggio in CMS potrebbe essere errato e potrebbe anche essere ridotto al termine della gara. Nota che il grader potrebbe eseguire più istanze del tuo programma contemporaneamente: quindi, le chiamate a `process_step` e `determine_steps` della stessa sfida potrebbero essere eseguite in istanze diverse e le chiamate a `process_step` e `determine_steps` di sfide diverse potrebbero essere eseguite nella stessa istanza. Ogni caso di test include fino a 5 sfide.

Assunzioni

- $2 \leq N \leq 63$.
- $1 \leq M \leq 63$.
- Ogni studente alza la mano **al massimo una volta** durante l'intera sfida: per ogni studente i , c'è al massimo un turno j in cui i alza la mano.
- Ad ogni turno, almeno uno studente **non** alza la mano.

Subtask e punteggio

Subtask	Punteggio	Assunzioni aggiuntive
1	4	$M = 1$.
2	6	$N = 2$.
3	9	$P[i] = i$ per ogni $0 \leq i \leq N - 1$.
4	25	Al massimo uno studente alza la mano in ogni turno.
5	56	Nessuna assunzione aggiuntiva.

Per ogni caso di test, il tuo punteggio è 0 (`Output isn't correct` in CMS) se il valore restituito da qualsiasi chiamata a `process_step` non è valido o se il valore restituito da qualsiasi chiamata a `determine_steps` non è corretto.

Altrimenti, sia C la lunghezza massima di **qualsiasi** array in **qualsiasi** B restituito da `process_step`. Il punteggio per un caso di test in un subtask con punteggio S viene calcolato come $S \cdot X$, dove X dipende da C secondo la seguente tabella:

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

Esempio

Consideriamo uno scenario con $N = 4$ studenti, $M = 2$ insegnanti e la permutazione $P = [0, 3, 1, 2]$. Inizialmente, tutti gli array sono vuoti e il grader chiama:

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

Lo studente 1 ha alzato la mano, quindi il suo array non può essere modificato. L'insegnante 0 può ad esempio decidere di impostare l'array dello studente 0 a $[10]$, quello dello studente 3 a $[1, 63, 4]$, lasciando vuoto quello dello studente 2. Per fare ciò, la funzione deve restituire $B = [[10], [], [], [1, 63, 4]]$.

Dopo che l'insegnante 0 se ne va, gli studenti si scambiano gli array secondo P , e dunque A diventa $[[10], [], [1, 63, 4], []]$. Il grader allora chiama:

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

Gli studenti 0 e 3 hanno alzato la mano, quindi i loro array non possono essere modificati. L'insegnante 1 può decidere di impostare l'array dello studente 1 a $[0, 40]$ e quello dello studente 2 a $[1, 50]$. Per fare ciò, la funzione deve restituire $B = [[10], [0, 40], [1, 50], []]$.

Dopo che l'insegnante 1 se ne va, gli array vengono nuovamente scambiati secondo P , e dunque A diventa $[[10], [1, 50], [], [0, 40]]$. A questo punto il grader chiama:

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

Questa funzione deve restituire l'array $D = [1, 0, -1, 1]$ poiché gli studenti 0 e 3 hanno alzato la mano al turno 1, lo studente 1 ha alzato la mano al turno 0 e lo studente 2 non ha mai alzato la mano. In questo esempio, $C = 3$.

Grader di esempio

Formato di input:

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

Dove Q è un array di lunghezza N e $Q[i] = j$ se lo studente i alza la mano durante il turno j , oppure $Q[i] = -1$ se lo studente i non alza mai la mano durante l'intera sfida.

Formato di output:

Dopo ogni chiamata a `process_step`, il grader di esempio stampa il contenuto degli array. Sia K la lunghezza di B e $L[i]$ la lunghezza di $B[i]$ (per ogni $0 \leq i < K$). Il grader stampa B nel seguente formato, **seguito da una riga vuota**:

```
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]
```

Se $K \neq N$, il grader di esempio stampa un messaggio di errore e termina, altrimenti scambia gli array secondo la permutazione P .

Dopo aver chiamato `determine_steps`, il grader stampa:

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

Dove H rappresenta la lunghezza dell'array D restituito da `determine_steps`.