

Klassenzimmerspiel

An einer angesehenen Schule in Wörgl feiert die Informatikolympiade ihr Jubiläum mit einem Spiel zwischen AOI-Teilnehmern und Trainern. Im Klassenzimmer sitzen N Teilnehmer, nummeriert von 0 bis $N - 1$. Jeder Teilnehmer hält einen Zettel, der ein Array von ganzen Zahlen enthält. Anfangs ist jeder Zettel leer und enthält somit ein leeres Array.

Die Teilnehmer spielen das Spiel mit M Trainern, nummeriert von 0 bis $M - 1$ und dem Bundeskoordinator Achi. Das Spiel wird in M Schritten gespielt, die ebenfalls von 0 bis $M - 1$ nummeriert sind. In Schritt j betritt Trainer j das Klassenzimmer und die folgenden Ereignisse finden statt:

- Einige Teilnehmer (möglicherweise keiner) heben die Hand. Es ist garantiert, dass in jedem Schritt mindestens ein Teilnehmer **nicht** die Hand hebt und jeder Teilnehmer während des gesamten Spiels **höchstens einmal** die Hand heben darf.
- Der Trainer sieht sich die Zettel der Teilnehmer an und kann die Zettel derjenigen Teilnehmer, die sich in diesem Schritt **nicht** gemeldet haben, **ändern**. Für jeden dieser Teilnehmer kann der Trainer den Inhalt des Zettels durch ein neues (möglicherweise leeres) Array von ganzen Zahlen ersetzen. Jede ganze Zahl muss zwischen einschließlich 0 und 63 liegen, und das Array darf maximal 63 Elemente enthalten.
- Nach diesen Änderungen verlässt Trainer j den Raum, und die Teilnehmer tauschen ihre Zettel gemäß einer geheimen Permutation P . Das heißt, jeder Teilnehmer i ($0 \leq i < N$) gibt seinen Zettel an Teilnehmer $P[i]$, wobei die N Elemente $P[0], P[1], \dots, P[N - 1]$ **verschiedene** ganze Zahlen zwischen 0 und $N - 1$ sind. Die Werte von P bleiben während des gesamten Spiels **unverändert**, sind aber weder den Trainern noch Achi bekannt.

Sobald alle M Schritte abgeschlossen und der letzte Austausch gemäß P durchgeführt ist, betritt Achi den Raum. Anhand der Zettel der Teilnehmer muss Achi für jeden Teilnehmer i ($0 \leq i < N$) die Schrittnummer j ermitteln, in der Teilnehmer i die Hand gehoben hat, oder feststellen, dass Teilnehmer i nie die Hand gehoben hat.

Die Trainer können nicht mit anderen Trainern oder Achi kommunizieren, außer über die auf die Zettel der Teilnehmer geschriebenen Arrays. Jeder Trainer weiß, in welchem Schritt er das Klassenzimmer betreten hat.

Deine Aufgabe ist es, eine Strategie für die Trainer und Achi zu entwickeln und umzusetzen, die korrekt erfasst, ob und wann jeder Teilnehmer die Hand gehoben hat. Deine Punktezahl hängt von der Länge des längsten auf einen Zettel geschriebenen Arrays ab – je kürzer dieses Arrays, desto höher die Punktezahl.

Implementierungsdetails

Du musst zwei Funktionen implementieren, eine für die Trainer und eine für Achi.

Für die **Trainer** sollst du folgende Funktion implementieren:

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

- N : die Anzahl der Teilnehmer.
- M : die Anzahl der Schritte und auch die Anzahl der Trainer.
- R : die aktuelle Schrittnummer (von 0 bis $M - 1$).
- T : ein (möglicherweise leeres) Array, das die Indizes der Teilnehmer in aufsteigender Reihenfolge enthält, die in diesem Schritt die Hand heben.
- A : ein Array der Länge N , das die Zettel beschreibt, wobei $A[i]$ ($0 \leq i < N$) das Array der ganzen Zahlen auf dem Zettel ist, den Teilnehmer i zu Beginn des Schritts hat.
- Diese Funktion wird genau M Mal pro Spiel aufgerufen, und zwar in der Reihenfolge $R = 0, 1, \dots, M - 1$.

Die Funktion sollte ein Array B der Länge N zurückgeben, das die Zettel nach den Änderungen des Trainers im gleichen Format wie A darstellt.

- Für jeden Teilnehmer i , der die Hand gehoben hat (d. h. i erscheint in T), darf der Zettel nicht gewechselt werden, also $B[i] = A[i]$.
- Für jedes i mit $0 \leq i < N$ darf die Länge von $B[i]$ nicht größer als 63 sein und jede ganze Zahl in $B[i]$ muss zwischen einschließlich 0 und 63 liegen.

Für **Achi** sollst du folgende Funktion implementieren:

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

- N, M : dasselbe wie oben.
- A : ein Array der Länge N , wobei $A[i]$ das Array der ganzen Zahlen auf dem Zettel ist, den Teilnehmer i nach allen M Schritten hat.
- Diese Funktion wird genau einmal pro Spiel aufgerufen, nach dem letzten Aufruf von `process_step`.

Die Funktion muss ein Array D der Länge N zurückgeben. Für jedes i mit $0 \leq i < N$ muss gelten:

- $D[i] = j$ falls Teilnehmer i während Schritt j die Hand gehoben hat, oder
- $D[i] = -1$ falls Teilnehmer i während des gesamten Spiels nie die Hand gehoben hat.

Dein Programm darf keine Informationen zwischen Aufrufen von `process_step` speichern oder übertragen, außer über den Rückgabewert von `process_step`. Versucht dein Programm dies, kann deine Punktezahl im CMS fehlerhaft sein und wird nach dem Ende des Contests von einem blauen Drachen (auralos) gefressen. Beachte, dass der Grader mehrere Instanzen deines Programms gleichzeitig ausführen kann. Das bedeutet, dass Aufrufe von `process_step` und `determine_steps` für dasselbe Spiel in verschiedenen Instanzen ausgeführt werden können, während Aufrufe für verschiedene Spiele in derselben Instanz ausgeführt werden können. Jeder Testfall umfasst bis zu 5 Spiele.

Beschränkungen

- $2 \leq N \leq 63$
- $1 \leq M \leq 63$
- Jeder Teilnehmer hebt während des gesamten Spiels **höchstens einmal** die Hand. Das heißt, für jeden Teilnehmer i gibt es höchstens einen Schritt j , in dem i die Hand hebt.
- Bei jedem Schritt hebt mindestens ein Teilnehmer **nicht** die Hand.

Teilaufgaben und Punktevergabe

Teilaufgabe	Punktezahl	Weitere Beschränkungen
1	4	$M = 1$
2	6	$N = 2$
3	9	$P[i] = i$ für alle $0 \leq i \leq N - 1$.
4	25	Bei jedem Schritt hebt höchstens ein Teilnehmer die Hand.
5	56	Keine weiteren Beschränkungen.

Für einen Testfall erhältst du 0 Punkte (wird im CMS als `Output isn't correct` angezeigt), wenn der Rückgabewert eines Aufrufs der Funktion `process_step` ungültig ist, oder wenn der Rückgabewert eines Aufrufs der Funktion `determine_steps` falsch ist.

Andernfalls sei C die Länge des **längsten** Arrays in einem B , das von `process_step` zurückgegeben wird. Dann wird die Punktezahl für einen Testfall in einer Teilaufgabe mit Punktezahl S als $S \cdot X$ berechnet, wobei X gemäß der folgenden Tabelle von C abhängt:

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

Beispiel

Betrachten wir ein Szenario mit $N = 4$ Teilnehmern, $M = 2$ Trainern und der Permutation $P = [0, 3, 1, 2]$. Anfangs sind alle Zettel leer.

Der Grader ruft zuerst auf:

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

Teilnehmer 1 hat die Hand gehoben, daher kann sein Zettel nicht mehr geändert werden. Trainer 0 kann entscheiden, $[10]$ auf den Zettel von Teilnehmer 0 zu schreiben, $[1, 63, 4]$ auf den Zettel von Teilnehmer 3 zu schreiben und den Zettel von Teilnehmer 2 leer zu lassen. Dazu muss die Funktion $B = [[10], [], [], [1, 63, 4]]$ zurückgeben.

Nachdem Trainer 0 den Raum verlassen hat, tauschen die Teilnehmer ihre Zettel gemäß P . Nach dem Tausch lauten die Zettel $A = [[10], [], [1, 63, 4], []]$.

Der Grader ruft nun auf:

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

Die Teilnehmer 0 und 3 haben die Hand gehoben, daher können ihre Zettel nicht mehr geändert werden. Trainer 1 kann beschließen, $[0, 40]$ auf den Zettel von Teilnehmer 1 und $[1, 50]$ auf den Zettel von Teilnehmer 2 zu schreiben. Dazu muss die Funktion $B = [[10], [0, 40], [1, 50], []]$ zurückgeben.

Nachdem Trainer 1 gegangen ist, werden die Zettel gemäß P erneut vertauscht. Nach dem Tausch lauten die Zettel $A = [[10], [1, 50], [], [0, 40]]$.

Schließlich ruft der Grader auf:

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

Diese Funktion muss das Array $D = [1, 0, -1, 1]$ zurückgeben, da die Teilnehmer 0 und 3 in Schritt 1 die Hand gehoben haben, Teilnehmer 1 in Schritt 0 die Hand gehoben hat und Teilnehmer 2 die Hand nicht gehoben hat. In diesem Beispiel ist $C = 3$.

Beispielgrader

Eingabeformat:

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

Hierbei ist Q ein Array der Länge N , wobei $Q[i] = j$ wenn Teilnehmer i während Schritt j die Hand hebt, oder $Q[i] = -1$ wenn Teilnehmer i während des gesamten Spiels nie die Hand hebt.

Ausgabeformat:

Nach jedem Aufruf von `process_step` gibt der Sample Grader den Inhalt der Zettel aus. Sei K die Länge von B und $L[i]$ die Länge von $B[i]$ (für jedes $0 \leq i < K$). Der Beispielgrader gibt B im folgenden Format aus, **gefolgt von einer Leerzeile**:

```
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]
```

Der Beispielgrader tauscht dann die Zettel gemäß der Permutation P aus. Falls $K \neq N$, gibt der Beispielgrader eine Fehlermeldung aus und beendet sich stattdessen.

Nach dem Aufruf von `determine_steps` gibt der Beispielgrader Folgendes aus:

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

Hierbei ist H die Länge des Arrays D das von `determine_steps` zurückgegeben wird.