

Klasseromslek

En videregående skole i Tasjkent markerer jubileum ved å arrangere en lek der elevene spiller mot lærerne. Det sitter N i et klasserom. De er nummerert fra 0 til $N - 1$. Hver elev holder et ark med en liste med heltall. I begynnelsen er arket blankt, altså inneholder det en tom liste.

Elevene spiller mot rektor og M lærere. Lærerne er nummerert fra 0 til $M - 1$. Leken lekes i M steg, disse er også nummerert fra 0 til $M - 1$. I steg j går lærer j inn i klasserommet og følgende hendelser skjer:

- Noen (potensielt null) elever rekker opp hånden. For hver runde er det garantert at det finnes minst én elev som **ikke** rekker opp hånda. Hver elev kan rekke opp hånda **maksimalt én gang** i løpet av hele leken.
- Læreren ser på arkene som elevene holder for øyeblikket, og kan **endre** arkene som holdes av elever som **ikke** rakk opp hånda. For hver av disse elevene kan læreren skifte ut lista på arket med en ny (potensielt tom) liste med heltall. Hvert heltall må være fra og med 0 til og med 63. Lista kan maksimalt inneholde 63 tall.
- Etter at tallene har blitt endret forlater lærer j rommet. Deretter utveksler elevene arkene sine i henhold til en hemmelig permutasjon P . Altså gir hver elev i ($0 \leq i < N$) arket sitt til elev $P[i]$, der $P[0], P[1], \dots, P[N - 1]$ er N **distinkte** heltall fra og med 0 til og med $N - 1$. Verdien til P er **faste** gjennom alle steg i leken, men hverken lærerne eller rektor kjenner til dem.

Etter at alle M stegene i leken har blitt gjennomført (altså har arkene blitt utvekslet en siste gang i henhold til P), kommer rektor inn i rommet. Ved å kun se på arkene som elevene holder må rektor finne ut hvilket steg-nummer (indeks) j hver elev i ($0 \leq i < N$) rakk opp hånda i, eller om eleven aldri rakk opp hånda.

Lærerne kan hverken kommunisere med hverandre eller med rektor utenom listene med heltall de skriver på elevenes ark. Lærerne kjenner det nåværende steg-nummeret når de går inn i klasserommet.

Din oppgave er å finne og implementere en strategi for lærerne og rektor som korrekt kan identifisere om og når en elev rakk opp hånda. Poengsummen din vil avhenge av den maksimale lengden av alle listene som blir skrevet på arkene – en lavere maksimal lengde vil gi en lik eller lavere poengsum.

Implementasjonsdetaljer

Du skal implementere to funksjoner, en for lærerne og en for rektor.

Funksjonen du skal implementere for **lærerne** er:

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

- N : antall elever.
- M : antall steg, som også er antall lærere.
- R : nåværende steg-nummer (fra 0 til $M - 1$).
- T : en (potensielt tom) liste over indeksene til elevene som rekker opp hånda, sortert i stigende rekkefølge.
- A : en liste av lengde N som beskriver arkene. $A[i]$ ($0 \leq i < N$) er lista med heltall som står på arket elev i holder når læreren kommer inn i rommet (i starten av steget).
- Denne funksjonen blir kalt eksakt M ganger per lek, i rekkefølge $R = 0, 1, \dots, M - 1$.

Funksjonen skal returnere en liste B av lengde N som representerer arkene etter lærerens endringer. Denne skal ha samme format som A .

- For hver elev i som rakk opp hånda (altså, i finnes i T), kan arket ikke modifiseres, altså må $B[i] = A[i]$.
- For hver i slik at $0 \leq i < N$, kan lengden av $B[i]$ ikke overstige 63. Alle tallene i $B[i]$ må være heltall fra og med 0 til og med 63.

Funksjonen du skal implementere for **rektor** er:

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

- N, M : samme som ovenfor
- A : en liste av lengde N , der $A[i]$ er lista av heltall som står på arket som elev i holder etter alle de M stegene.
- Denne funksjonen kalles én gang per lek, etter siste kall til `process_step`.

Funksjonen må returnere en liste D av lengde N . For hver i slik at $0 \leq i < N$, må følgende holde:

- $D[i] = j$ hvis i rakk opp hånda i steg j , eller
- $D[i] = -1$ hvis elev i ikke rakk opp hånda i løpet av hele leken.

Programmet ditt har ikke lov til å lagre eller sende informasjon mellom kallene til `process_step`, bortsett fra returverdien (som sendes videre i argumentet til neste kall). Hvis

programmet ditt prøver å gjøre dette, kan du få feil poengsum i CMS under konkurransen og den vil da bli redusert etter konkurransen er ferdig. Dette er fordi graderen kan kjøre flere instanser av programmet ditt samtidig. Dette betyr at kall til `process_step` og `determine_steps` fra samme lek kan kjøres i ulike instanser av programmet, og kall til `process_step` og `determine_steps` fra ulike leker kan kjøres i samme instans. Hvert testcase inneholder opp til 5 leker.

Begrensninger

- $2 \leq N \leq 63$
- $1 \leq M \leq 63$
- Hver elev rekker opp hånda **maksimalt én gang** i løpet av hele leken. Altså, for hver elev i , finnes det maksimalt ett steg j der i rekker opp hånda.
- I hvert steg finnes det minst én elev som **ikke** rekker opp hånda.

Deloppgaver og poeng

Deloppgaver	Poeng	Ytterligere Begrensninger
1	4	$M = 1$
2	6	$N = 2$
3	9	$P[i] = i$ for alle $0 \leq i \leq N - 1$.
4	25	Minst én elev rekker opp hånda i hvert steg.
5	56	Ingen ytterligere begrensninger.

For hvert testcase blir poengsummen din 0 (rapportert som `Output isn't correct` i CMS) hvis returverdien til noen av kallene til `process_step` er ugyldige eller om returverdien til `determine_steps` er feil.

Ellers, la C være den maksimale lengden av **noen** liste i **noen** B returnert av `process_step`. Da blir poengene for et testcase i en deloppgave med poengsum S beregnet som $S \cdot X$, der X avhenger av C i henhold til følgende tabell:

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

Eksempel

Gitt et scenario med $N = 4$ elever, $M = 2$ lærere, og permutasjon $P = [0, 3, 1, 2]$. Alle arkene er blanke i starten.

Graderen kaller først:

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

Elev 1 rekker opp hånda, så hennes ark kan ikke endres. Lærer 0 velger å skrive $[10]$ på arket til elev 0, skrive $[1, 63, 4]$ på arket til elev 3, og la elev 2 sitt ark være tomt. For å gjøre dette må funksjonen returnere $B = [[10], [], [], [1, 63, 4]]$.

Etter at lærer 0 har forlatt rommet, utveksler elevene arkene i henhold til P . Etter utvekslingen er arkene $A = [[10], [], [1, 63, 4], []]$.

Graderen kaller så:

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

Elevene 0 og 3 har rukket opp hånda, så deres ark kan ikke endres. Lærer 1 kan skrive $[0, 40]$ på arket til elev 1 og skrive $[1, 50]$ på arket til elev 2. For å gjøre dette må funksjonen returnere $B = [[10], [0, 40], [1, 50], []]$.

Etter at lærer 1 har forlatt rommet, blir arkene igjen utvekslet i henhold til P . Etter utvekslingen er arkene $A = [[10], [1, 50], [], [0, 40]]$.

Til slutt kaller graderen

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

Denne funksjonen må returnere lista $D = [1, 0, -1, 1]$ siden elevene 0 og 3 rakk opp hånda i steg 1, elev 1 rakk opp hånda i steg 0, og elev 2 aldri rakk opp hånda. I dette eksempelet er $C = 3$.

Sample Grader

Inputformat:

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

Her er Q en liste med lengde N , der $Q[i] = j$ hvis elev i rekker opp hånda i steg j , eller $Q[i] = -1$ hvis elev i aldri rekker opp hånda i løpet av leken.

Outputformat:

Etter hvert kall til `process_step` printer sample graderen innholdet på hvert av arkene. La K være lengden til B og $L[i]$ lengden til $B[i]$ (for alle $0 \leq i < K$). Sample graderen printer B på følgende format, **etterfulgt av en tom linje**:

```
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]
```

Sample graderen utveksler deretter arkene i henhold til permutasjonen P . Hvis $K \neq N$ printer sample graderen en feilmelding og terminerer i stedet.

Etter å ha kalt `determine_steps` printer sample graderen:

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

Her er H lengden av lista D som blir returnert av `determine_steps`.