بلیت‌های جشن (tickets)

رینگو در یک جشن در سنگاپور شرکت کرده است. او تعدادی بلیتِ جایزه در کیفش دارد که می‌خواهد از آن‌ها در غرفه‌ی بازی استفاده کند. بلیت‌ها n رنگ متفاوت دارند و روی هر بلیت یک عدد صحیح نامنفی نوشته شده است. اعداد بلیت‌های متفاوت ممکن است یکسان باشند. به خاطر قوانین جشن، تضمین می‌شود که n **زوج** است.

رینگو از هر رنگی m بلیت با خود دارد، که مجموعا $n \cdot m$ بلیت می‌شود. روی بلیت jام از رنگ iام عدد $x[i][j]$ نوشته شده است ($0 \leq i \leq n-1$ و $0 \leq j \leq m-1$).

بازی در k نوبت انجام می‌شود و نوبت‌ها از 0 تا $k-1$ شماره دارند. هر نوبت بازی این‌گونه است:

- رینگو یک **مجموعه** n عضوی از بلیت‌های داخل کیفش را انتخاب می‌کند، به نحوی که از هر رنگ یک بلیت انتخاب شود. سپس بلیت‌های انتخاب‌شده را به مسئول بازی تحویل می‌دهد.
- مسئول بازی شماره بلیت‌های انتخاب شده $a[0], a[1] \ \ldots \ a[n-1]$ را یادداشت می‌کند. ترتیب این n عدد اهمیتی ندارد.
- مسئول بازی از یک جعبه شانس، کارت مخصوصی را خارج می‌کند و عدد b که روی کارت نوشته شده است را یادداشت می‌کند.
- مسئول بازی به ازای هر i از 0 تا $n-1$، قدر مطلق تفاضل $a[i]$ و b را محاسبه می‌کند. فرض کنید، مجموع این قدرمطلق‌ها S باشد.
- مسئول بازی به ازای این نوبت بازی، جایزه‌ای به مقدار S به رینگو می‌دهد.
- بلیت‌های انتخاب شده، حذف می‌شوند و در نوبت‌های آینده قابل استفاده نیستند.

بلیت‌هایی که بعد از پایان k نوبت بازی در کیف رینگو اضافه می‌مانند، دور ریخته می‌شوند.

رینگو به‌دقت نگاه کرد و متوجه شد که بازی دست‌کاری می‌شود! در واقع داخل جعبه شانس، یک چاپگر وجود دارد. در هر نوبت، مسئول بازی عدد b را پیدا می‌کند که جایزه آن نوبت را کمینه می‌کند. عدد انتخاب شده توسط مسئول بازی، روی کارت مخصوص آن نوبت چاپ می‌شود.

رینگو با داشتن این اطلاعات، می‌خواهد بلیت‌ها را به نوبت‌های بازی تخصیص دهد. در واقع، او می‌خواهد برای هر نوبت بازی، مجموعه‌ای از بلیت‌ها را انتخاب کند که مجموع جایزه‌اش بیشینه شود.

جزئیات پیاده‌سازی

شما باید تابع زیر را پیاده‌سازی کنید:

```
int64 find_maximum(int k, int[][] x)
```

- k: تعداد نوبت‌های بازی
- x یک آرایه $n \times m$ که اعداد روی بلیت‌ها را نشان می‌دهد. بلیت‌های هر رنگ به ترتیب غیر نزولی اعدادشان

مرتب شده‌اند.
- این تابع دقیقا یک مرتبه فراخوانی می‌شود.
- این تابع باید دقیقا یک مرتبه تابع `allocate_tickets` را فراخوانی کند (ادامه توضیحات را ببینید)، و با این فراخوانی k مجموعه بلیت متعلق به هر نوبت بازی را مشخص کند. تخصیص بلیت‌ها باید مجموع مقدار جایزه‌ها را بیشینه کند.
- این تابع باید بیشینه مجموع مقدار جایزه‌ها را برگرداند.

تابع `allocate_tickets` به شکل زیر تعریف شده است:

```
void allocate_tickets(int[][] s)
```

- s: یک آرایه $n \times m$ است. مقدار $s[i][j]$ باید r باشد، اگر بلیت jام از رنگ iام در نوبت rام بازی استفاده می‌شود و در صورتی که این بلیت کلا مورد استفاده قرار نمی‌گیرد، این درایه مقدار -1 داشته باشد.
- به ازای هر $0 \leq i \leq n-1$، در بین $s[i][0], s[i][1], \ldots, s[i][m-1]$، هر یک از مقادیر $0, 1, 2, \ldots, k-1$ باید دقیقا یک مرتبه آمده باشد، و بقیه درایه‌ها باید -1 باشند.
- در صورتی که چند تخصیص متفاوت به مقدار بیشینه مجموع جایزه‌ها منجر شوند، گزارش هر کدام از آن‌ها قابل قبول است.

مثال‌ها

مثال ۱

فراخوانی زیر را در نظر بگیرید:

```
find_maximum(2, [[0, 2, 5],[1, 1, 3]])
```

این یعنی:

- بازی در دو نوبت انجام می‌شود؛
- اعداد بلیت‌های رنگ 0، به ترتیب 0، 2 و 5 هستند؛
- اعداد بلیت‌های رنگ 1، به ترتیب 1، 1 و 3 هستند.

یک تخصیص که به مجموع جایزه بیشینه منجر می‌شود این گونه است:

- در نوبت 0، رینگو بلیت 0 از رنگ 0 (که عدد 0 دارد) و بلیط 2 از رنگ 1 (که عدد 3 دارد) را انتخاب می‌کند. کم‌ترین جایزه ممکن در این نوبت بازی 3 است. برای مثال، مسئول بازی ممکن است $b = 1$ را انتخاب کند: $|1 - 0| + |1 - 3| = 1 + 2 = 3$.
- در نوبت 1، رینگو بلیت 2 از رنگ 0 (که عدد 5 دارد) و بلیت 1 از رنگ 1 (که عدد 1 دارد) را انتخاب می‌کند. کم‌ترین جایزه ممکن در این نوبت بازی 3 است. برای مثال، مسئول بازی ممکن است $b = 3$ را انتخاب کند: $|3 - 1| + |3 - 5| = 2 + 2 = 4$.
- به این ترتیب، مجموع جایزه‌ها $3 + 4 = 7$ خواهد بود.

برای گزارش این تخصیص، تابع `finx_maximum` باید تابع `allocate_tickets` را این گونه فراخوانی کند:

- `allocate_tickets([[0, -1, 1], [-1, 1, 0]])`

در نهایت، تابع `find_maximum` باید مقدار 7 را برگرداند.

مثال ۲

فراخوانی زیر را در نظر بگیرید:

```
find_maximum(1, [[5, 9], [1, 4], [3, 6], [2, 7]])
```

این یعنی:

- بازی تنها یک نوبت دارد،
- اعداد بلیت‌های رنگ 0، به ترتیب 5 و 9 هستند؛
- اعداد بلیت‌های رنگ 1، به ترتیب 1 و 4 هستند؛
- اعداد بلیت‌های رنگ 2، به ترتیب 3 و 6 هستند؛
- اعداد بلیت‌های رنگ 3، به ترتیب 2 و 7 هستند.

یک تخصیص که به مجموع جایزه بیشینه منجر می‌شود این گونه است:

- در نوبت 0، رینگو بلیت 1 از رنگ 0 (که عدد 9 دارد)، بلیط 0 از رنگ 1 (که عدد 1 دارد)، بلیت 0 از رنگ 2 (که عدد 3 دارد)، و بلیت 1 از رنگ 3 (که عدد 7 دارد) را انتخاب می‌کند. کم‌ترین جایزه ممکن در این نوبت بازی 12 است، در صورتی که مسئول بازی $b = 3$ را انتخاب کند: $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$.

برای گزارش این پاسخ، تابع `finx_maximum` باید تابع `allocate_tickets` را این گونه فراخوانی کند:

- `allocate_tickets([[-1, 0], [0, -1], [0, -1], [-1, 0]])`

در نهایت، تابع `find_maximum` باید مقدار 12 را برگرداند.

محدودیت‌ها

- $2 \le n \le 1500$ و n زوج است.
- $1 \le k \le m \le 1500$
- $0 \le x[i][j] \le 10^9$ (به ازای هر $0 \le i \le n - 1$ و $0 \le j \le m - 1$)
- $x[i][j - 1] \le x[i][j]$ (به ازای هر $0 \le i \le n - 1$ و $1 \le j \le m - 1$)

زیرمسئله‌ها

1. (۱۱ امتیاز) $m = 1$
2. (۱۶ امتیاز) $k = 1$
3. (۱۴ امتیاز) $0 \le x[i][j] \le 1$ (به ازای هر $0 \le i \le n - 1$ و $0 \le j \le m - 1$)
4. (۱۴ امتیاز) $k = m$

5. (۱۲ امتیاز) $n, m \leq 80$
6. (۲۳ امتیاز) $n, m \leq 300$
7. (۱۰ امتیاز) بدون محدودیت اضافه.

ارزیاب نمونه

ارزیاب نمونه، ورودی را در قالب زیر می‌خواند.

- خط 1: $n \;\; m \;\; k$
- خط $2 + i$ $(0 \leq i \leq n - 1)$: $x[i][0] \;\; x[i][1] \;\; \ldots \;\; x[i][m - 1]$

ارزیاب نمونه، پاسخ شما را در قالب زیر چاپ می‌کند:

- خط 1: مقدار بازگردانده‌شده‌ی تابع `find-maximum`
- خط $2 + i$ $(0 \leq i \leq n - 1)$: $s[i][0] \;\; s[i][1] \;\; \ldots \;\; s[i][m - 1]$