Квитки на карнавал (tickets)

Рінго потрапив на карнавал у Сінгапурі. У нього в сумці є призові квитки які він хоче використати у ігровому кіоску. Кожен квиток може бути одного з n кольорів, та на ньому надруковано невід'ємне число. Числа надруковані на різних квитках можуть співпадати. Зважаючи на особливість правил карнавалу, гарантується що n є **парним**.

Рінго має у сумці m квитків кожного кольору, всього $n \cdot m$ квитків. На квитку j кольору i надруковано ціле число $x[i][j]$ ($0 \leq i \leq n-1$ та $0 \leq j \leq m-1$).

Гра складається з k раундів, пронумерованих від 0 до $k-1$. Кожен раунд грається у наступному порядку:

- Рінго обирає зі своєї сумки **набір** з n квитків, по одному квитку кожного кольору. Після цього він передає квитки ведучому гри.
- Ведучий записує числа $a[0], a[1] \ \ldots \ a[n-1]$ що надруковані на квитках з набору. Порядок цих n чисел значення не має.
- Ведучий дістає спеціальну картку з лотерейної коробки та записує ціле число b надруковане на цій картці.
- Ведучий обчислює модулі різниць між $a[i]$ та b для кожного i від 0 до $n-1$. Нехай S буде сумою цих модулів різниць.
- У цьому раунді ведучий дає Рінго приз що має вартість S.
- Квитки з цього раунда викидаються і не можуть бути використані у наступних раундах.

Решта квитків з сумки Рінго викидаються після k раундів гри.

Придивившись, Рінго помітив що гра є сфальсифікованою! Насправді, у лотерейній коробці стоїть принтер. У кожному раунді ведучий знаходить ціле число b що мінімізує вартість приза у цьому раунді. Значення, що знайшов ведучий, друкується на картці для цього раунда.

Маючи всю цю інформацію, Рінго хоче розподілити квитки між раундами гри. Він хоче вибрати набір квитків для використання у кожному раунді так, щоб максимізувати загальну вартість призів.

Деталі реалізації

Ви маєте реалізувати наступну процедуру:

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

- k: кількість раундів.

- x: масив $n \times m$ що задає цілі числа для кожного з квитків. Квитки кожного кольору відсортовані за неспаданням чисел на них.
- Ця процедура викликається рівно один раз.
- Ця процедура має рівно один раз викликати `allocate_tickets` (див. нижче), описуючи k наборів квитків, по одному набору на кожен раунд. Розподіл квитків має максимізувати сумарну вартість призів.
- Ця процедура має повернути максимальну загальну вартість призів.

Процедуру `allocate_tickets` визначено так:

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

- s: масив $n \times m$. Значення $s[i][j]$ має бути r якщо квиток j кольору i використовується у наборі для раунда r гри, або -1 якщо не використовується взагалі.
- Для кожного $0 \leq i \leq n-1$, серед $s[i][0], s[i][1], \ldots, s[i][m-1]$ кожне значення $0, 1, 2, \ldots, k-1$ має зустрічатись рівно один раз, а решта значень мають бути -1.
- Якщо існує декілька розподілів, що призводять до максимальної загальної вартості призів, можна повернути довільний з них.

Приклади

Приклад 1

Розглянемо наступний виклик:

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

Це означає що:

- є $k = 2$ раундів;
- числа, надруковані на квитках кольору 0, є 0, 2 та 5, відповідно;
- числа, надруковані на квитках кольору 1, є 1, 1 та 3, відповідно.

Можливий розподіл що дає максимальну загальну вартість призів є:

- У раунді 0 Рінго обирає квиток 0 кольору 0 (з числом 0) та квиток 2 кольору 1 (з числом 3). Найменша можлива вартість приза у цьому раунді є 3. Наприклад, ведучий може обрати $b = 1$: $|1 - 0| + |1 - 3| = 1 + 2 = 3$.
- У раунді 1 Рінго обирає квиток 2 кольору 0 (з числом 5) та квиток 1 кольору1 (з числом 1). Найменша можлива вартість приза у цьому раунді є 4. Наприклад, ведучий може обрати $b = 3$: $|3 - 1| + |3 - 5| = 2 + 2 = 4$.
- Отже, загальна вартість призів буде $3 + 4 = 7$.

Щоб повідомити цей розподіл, процедура `find_maximum` має зробити наступний виклик `allocate_tickets`:

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

Зрештою, процедура `find_maximum` має повернути 7.

Приклад 2

Розглянемо наступний виклик:

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

Це означає що:

- є тільки один раунд,
- числа, надруковані на квитках кольору 0, є 5 та 9, відповідно;
- числа, надруковані на квитках кольору 1, є 1 та 4, відповідно;
- числа, надруковані на квитках кольору 2, є 3 та 6, відповідно;
- числа, надруковані на квитках кольору 3, є 2 та 7, відповідно.

Можливий розподіл що дає максимальну загальну вартість призів є:

- У раунді 0 Рінго обирає квиток 1 кольору 0 (з числом 9), квиток 0 кольору 1 (з числом 1), квиток 0 кольору 2 (з числом 3), та квиток 1 кольору 3 (з числом 7). Найменша вартість приза у цьому раунді є 12, коли ведучий обирає $b = 3$: $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$.

Щоб повідомити цей розподіл, процедура `find_maximum` має зробити наступний виклик `allocate_tickets`:

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

Зрештою, процедура `find_maximum` має повернути 12.

Обмеження

- $2 \leq n \leq 1500$ та n є парним.
- $1 \leq k \leq m \leq 1500$
- $0 \leq x[i][j] \leq 10^9$ (для всіх $0 \leq i \leq n - 1$ та $0 \leq j \leq m - 1$)
- $x[i][j - 1] \leq x[i][j]$ (для всіх $0 \leq i \leq n - 1$ та $1 \leq j \leq m - 1$)

Підзадачі

1. (11 балів) $m = 1$
2. (16 балів) $k = 1$
3. (14 балів) $0 \leq x[i][j] \leq 1$ (для всіх $0 \leq i \leq n - 1$ та $0 \leq j \leq m - 1$)
4. (14 балів) $k = m$
5. (12 балів) $n, m \leq 80$

6. (23 бали) $n, m \leq 300$
7. (10 балів) Без додаткових обмежень.

Приклад модуля перевірки

Приклад модуля перевірки читає вхідні дані у наступному форматі:

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

Приклад модуля перевірки друкує ваші відповіді у наступному форматі:

- рядок 1: значення, що повернула `find_maximum`
- рядок $2 + i$ ($0 \leq i \leq n - 1$): $s[i][0] \;\; s[i][1] \;\; \ldots \;\; s[i][m - 1]$