Карнавал Билеттери (tickets)

Ринго Сингапурдагы карнавалда. Анын сумкасында байге билеттери бар, аларды байге ойноочу жайда колдонгусу келет. Ар бир билет n түстөрдүн биринде болот, анда терс эмес бүтүн сан басылды. Ар кандай билеттерде басылган сандар барабар болушу мүмкүн. Карнавал эрежелери боюнча, n **жуп** сан.

Рингонун сумкасында ар бир түстөгү m билет бар, ошондуктан бардык $n * m$ билет бар. i өңүндөгү j-билетте $x[i][j]$ бүтүн сан басылган ($0 <= i <= n-1$ жана $0 <= j <= m-1$).

Оюн-байгеси 0 дөн $(k-1)$ге чейин k раундда ойнолот. Ар бир раунд төмөнкүдөй тартипте ойнолот:

- Ринго сумкасынан n билеттин **тобун** , ар бир түстөн бирден билет тандайт. Андан кийин топ оюн-мастерине берилет.
- Оюн-мастери топтун $a[0], a[1], ..., a[n-1]$ билеттерине басылган сандарды жазып алат. Ал n сандын тартиби маанилүү эмес.
- Оюн-мастери атайын кутусунан b бүтүн санын алат.
- Оюн-мастери ар бир i үчүн 0 дан $(n-1)$ га чейинки $a[i]$ жана b абсолюттук айырмачылыктарды эсептейт. S ушул абсолюттук айырмачылыктардын суммасы болсун.
- Бул раунд үчүн оюн-мастери Рингого Sке барабар байгени берет.
- Топтогу билеттер жокко чыгарылып, кийинки турларда колдонууга болбойт.

Оюндун k рунддан кийин Рингонун сумкасында калган билеттер жокко чыгарылат.

Жакшылап байкоо жүргүзүү менен, Ринго оюн байгеси бурмаланганын түшүндү! Атайын кутунун ичинде акылдуу принтер бар. Ар бир раундда оюн-мастери ошол рунддун байгесинин маанисин минималдаштырган b бүтүн санын табат.

Ушул маалыматтардын бардыгына ээ болгон Ринго оюндун рунддарына билеттеринин топторун оптималдуу тандоону каалайт. Башкача айтканда, ал байгелердин жалпы наркын жогорулатуу үчүн ар бир раундда колдонула турган билеттерди тандап алгысы келет.

Implementation details

You should implement the following procedure:

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

- k: the number of rounds.

- x: an $n \times m$ array describing the integers on each ticket. Tickets of each color are sorted in non-decreasing order of their integers.
- This procedure is called exactly once.
- This procedure should make exactly one call to `allocate_tickets` (see below), describing k ticket sets, one for each round. The allocation should maximize the total value of the prizes.
- This procedure should return the maximum total value of the prizes.

The procedure `allocate_tickets` is defined as follows:

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

- s: an $n \times m$ array. The value of $s[i][j]$ should be r if the ticket j of the colour i is used in the set of round r of the game, or -1 if it is not used at all.
- For each $0 \leq i \leq n-1$, among $s[i][0], s[i][1], \ldots, s[i][m-1]$ each value $0, 1, 2, \ldots, k-1$ must occur exactly once, and all other entries must be -1.
- If there are multiple allocations resulting in the maximum total prize value, it is allowed to report any of them.

Examples

Example 1

Consider the following call:

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

This means that:

- there are $k = 2$ rounds;
- the integers printed on the tickets of colour 0 are 0, 2 and 5, respectively;
- the integers printed on the tickets of colour 1 are 1, 1 and 3, respectively.

A possible allocation that gives the maximum total prize value is:

- In round 0, Ringo picks ticket 0 of colour 0 (with the integer 0) and ticket 2 of colour 1 (with the integer 3). The lowest possible value of the prize in this round is 3. E.g., the game master may choose $b = 1$: $|1 - 0| + |1 - 3| = 1 + 2 = 3$.
- In round 1, Ringo picks ticket 2 of colour 0 (with the integer 5) and ticket 1 of colour 1 (with the integer 1). The lowest possible value of the prize in this round is 4. E.g., the game master may choose $b = 3$: $|3 - 1| + |3 - 5| = 2 + 2 = 4$.
- Therefore, the total value of the prizes would be $3 + 4 = 7$.

To report this allocation, the procedure `find_maximum` should make the following call to `allocate_tickets`:

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

Finally, the procedure `find_maximum` should return 7.

Example 2

Consider the following call:

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

This means that:

- there is only one round,
- the integers printed on the tickets of colour 0 are 5 and 9, respectively;
- the integers printed on the tickets of colour 1 are 1 and 4, respectively;
- the integers printed on the tickets of colour 2 are 3 and 6, respectively;
- the integers printed on the tickets of colour 3 are 2 and 7, respectively.

A possible allocation that gives the maximum total prize value is:

- In round 0, Ringo picks ticket 1 of colour 0 (with the integer 9), ticket 0 of colour 1 (with the integer 1), ticket 0 of colour 2 (with the integer 3), and ticket 1 of colour 3 (with the integer 7). The lowest possible value of the prize in this round is 12, when the game master chooses $b = 3$: $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$.

To report this solution, the procedure `find_maximum` should make the following call to `allocate_tickets`:

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

Finally, the procedure `find_maximum` should return 12.

Constraints

- $2 \leq n \leq 1500$ and n is even.
- $1 \leq k \leq m \leq 1500$
- $0 \leq x[i][j] \leq 10^9$ (for all $0 \leq i \leq n - 1$ and $0 \leq j \leq m - 1$)
- $x[i][j - 1] \leq x[i][j]$ (for all $0 \leq i \leq n - 1$ and $1 \leq j \leq m - 1$)

Subtasks

1. (11 points) $m = 1$
2. (16 points) $k = 1$
3. (14 points) $0 \leq x[i][j] \leq 1$ (for all $0 \leq i \leq n - 1$ and $0 \leq j \leq m - 1$)
4. (14 points) $k = m$
5. (12 points) $n, m \leq 80$

6. (23 points) $n, m \leq 300$
7. (10 points) No additional constraints.

Sample grader

The sample grader reads the input in the following format:

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

The sample grader prints your answer in the following format:

- line 1: the return value of `find_maximum`
- line $2 + i$ ($0 \leq i \leq n - 1$): $s[i][0]$ $s[i][1]$ $\ldots$ $s[i][m - 1]$