Karnawałowe kupony (tickets)

Ringo jest na karnawale w Singapurze. W plecaku ma pewną liczbę kuponów, które może wymienić na nagrody na specjalnym stoisku. Każdy kupon jest w jednym z n kolorów i ma pewną nieujemną liczbę całkowitą wydrukowaną na nim. Liczby wydrukowane na różnych biletach mogą być takie same. Ze względu na pewien kaprys organizatorów karnawału, n jest zawsze liczbą **parzystą**.

Ringo ma po m kuponów każdego koloru w plecaku, zatem w sumie ma ich $n \cdot m$. Kupon j w kolorze i ma wydrukowaną na sobie liczbę $x[i][j]$ ($0 \leq i \leq n-1$ oraz $0 \leq j \leq m-1$).

Aby zdobyć nagrody, należy zagrać w grę złożoną z k rund ponumerowanych od 0 do $k-1$. Każda runda składa się z następujących czynności:

- Ze swojego plecaka Ringo wybiera **zbiór** n kuponów, po jednym kuponie w każdym z kolorów. Następnie przekazuje te kupony mistrzowi gry.
- Mistrz gry zapisuje wszystkie liczby $a[0], a[1] \ \ldots \ a[n-1]$ wydrukowane na kuponach ze zbioru. Kolejność w jakiej zapisuje te liczby, nie ma znaczenia.
- Mistrz gry wyciąga specjalną kartę ze swojego pudełka do losowania i zapisuje liczbę calkowitą b z tej karty.
- Mistrz gry oblicza wartość bezwzględną różnicy pomiędzy $a[i]$ oraz b dla każdego i od 0 do $n-1$. Przez S oznaczmy sumę tych bezwzględnych różnic.
- Za tę rundę, mistrz gry przekazuje Ringo nagrodę o wartości równej S.
- Kupony ze zbioru użytego do tej rundy są odrzucane i nie mogą już być używane w kolejnych rundach.

Wszystkie bilety, które pozostaną w plecaku Ringo po k rundach są odrzucane.

Po uważnej obserwacji, Ringo zauważył, że gra z nagrodami jest ustawiona! W środku pudełka do losowania mistrza gry znajduje się drukarka. W każdej rundzie, mistrz gry znajduje liczbę całkowitą b, która minimalizuje wartość nagrody w tej rundzie. Wartość ta jest drukowana przez mistrza gry na specjalnej karcie podczas tej rundy.

Wiedząc te wszystkie rzeczy, Ringo chciałby teraz przydzielić kupony do rund gry z nagrodami. To jest, chce on wybrać kupony do zbiorów, które użyje podczas każdej rundy, aby zmaksymalizować sumaryczną wartość nagród.

Szczegóły implementacji

Twoim zadaniem jest zaimplementować następującą funkcję:

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

- k: liczba rund.
- x: tablica o rozmiarze $n \times m$, która opisuje liczby całkowite na każdym kuponie. Kupony każdego koloru są posortowane niemalejąco względem liczb na tych kuponach.
- Funkcja ta jest wywoływana dokładnie raz.
- Funkcja ta powinna wykonać dokładnie jedno wywołanie procedury `allocate_tickets` (zobacz poniżej), opisując k zbiorów kuponów, po jednym na rundę. Przydział ten powininen maksymalizować sumaryczną wartość nagród.
- Funkcja ta powinna zwrócić maksymalną sumaryczną wartość nagród.

Procedura `allocate_tickets` jest zdefiniowana następująco:

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

- s: tablica o rozmiarze $n \times m$. Wartość $s[i][j]$ powinna być równa r, jeżeli bilet j w kolorze i jest wzięty do zbioru podczas rundy r, lub -1, jeżeli nie będzie w ogóle użyty.
- Dla każdego $0 \leq i \leq n-1$, spośród $s[i][0], s[i][1], \ldots, s[i][m-1]$ każda wartość $0, 1, 2, \ldots, k-1$ musi wystąpić dokładnie raz, a wszystkie inne wartości powinny być równe -1.
- Jeżeli istnieje wiele możliwych przydziałów kuponów, które maksymalizują sumaryczną wartość nagród, możesz wyznaczyć dowolny z nich.

Przykłady

Przykład 1

Rozważmy następujące wywołanie funkcji:

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

Oznacza to, że:

- mamy $k = 2$ rundy;
- liczby wydrukowane na kuponach w kolorze 0 to kolejno 0, 2, 5;
- liczby wydrukowane na kuponach w kolorze 1 to kolejno 1, 1, 3.

Możliwy przydział kuponów, który maksymalizuje sumaryczną wartość nagród, jest następujący:

- W rundzie 0, Ringo wybiera kupon 0 w kolorze 0 (z liczbą 0) oraz kupon 2 w kolorze 1 (z liczbą 3). Najmniejsza możliwa wartość nagrody w tej rundzie to 3. Na przykład, mistrz gry może wybrać wartość $b = 1$: $|1 - 0| + |1 - 3| = 1 + 2 = 3$.
- W rundzie 1, Ringo wybiera kupon 2 w kolorze 0 (z liczbą 5) oraz kupon 1 w kolorze 1 (z liczbą 1). Najmniejsza możliwa wartość nagrody w tej rundzie to 4. Na przykład, mistrz gry może

wybrać wartość $b = 3$: $|3 - 1| + |3 - 5| = 2 + 2 = 4$.
- Sumaryczna wartość nagród wyniesie zatem $3 + 4 = 7$.

Aby zgłosić taki przydział, funkcja `find_maximum` powinna wywołać procedurę `allocate_tickets` w poniższy sposób:

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

Na końcu, funkcja `find_maximum` powinna zwrócić 7.

Przykład 2

Rozważmy następujące wywołanie funkcji:

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

Oznacza to, że:

- mamy tylko jedną rundę;
- liczby wydrukowane na kuponach w kolorze 0 to kolejno 5 i 9;
- liczby wydrukowane na kuponach w kolorze 1 to kolejno 1 i 4;
- liczby wydrukowane na kuponach w kolorze 2 to kolejno 3 i 6;
- liczby wydrukowane na kuponach w kolorze 3 to kolejno 2 i 7.

Możliwy przydział kuponów, który maksymalizuje sumaryczną wartość nagród, jest następujący:

- W rundzie 0, Ringo wybiera kupon 1 w kolorze 0 (z liczbą 9), kupon 0 w kolorze 1 (z liczbą 1), kupon 0 w kolorze 2 (z liczbą 3), oraz kupon 1 w kolorze 3 (z liczbą 7). Najmniejsza możliwa wartość nagrody w tej rundzie to 12. Na przykład, mistrz gry może wybrać wartość $b = 3$: $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$.

Aby zgłosić taki przydział, funkcja `find_maximum` powinna wywołać procedurę `allocate_tickets` w poniższy sposób:

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

Na końcu, funkcja `find_maximum` powinna zwrócić 12.

Ograniczenia

- $2 \leq n \leq 1500$ oraz n jest parzyste.
- $1 \leq k \leq m \leq 1500$
- $0 \leq x[i][j] \leq 10^9$ (dla $0 \leq i \leq n - 1$ oraz $0 \leq j \leq m - 1$)
- $x[i][j - 1] \leq x[i][j]$ (dla $0 \leq i \leq n - 1$ oraz $1 \leq j \leq m - 1$)

Podzadania

1. (11 punktów) $m = 1$
2. (16 punktów) $k = 1$
3. (14 punktów) $0 \leq x[i][j] \leq 1$ (dla $0 \leq i \leq n-1$ oraz $0 \leq j \leq m-1$)
4. (14 punktów) $k = m$
5. (12 punktów) $n, m \leq 80$
6. (23 punkty) $n, m \leq 300$
7. (10 punktów) Brak dodatkowych ograniczeń.

Przykładowy program oceniający

Przykładowy progam oceniający wczytuje wejście w następującym formacie:

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

Przykładowy program oceniający wypisuje wyjście w następującym formacie:

- wiersz 1: wynik wywołania funkcji `find_maximum`
- wiersze $2 + i$ ($0 \leq i \leq n-1$): $s[i][0]$ $s[i][1]$ $\ldots$ $s[i][m-1]$