Constructor

СЕЗОН 10 – ВТОРИ РУНД

Ани и Боби играят с новия си конструктор, състоящ се от разноцветни кубчета. Цветовете на кубчетата са общо $\boldsymbol{M}$ и са означени с малки латински букви. Известно е, че от цвета L_i има C_i на брой кубчета. Децата се редуват да добавят кубчетата в редичка като в началото си избират един цвят и поставят всички кубчета от този цвят. След това си избират друг цвят кубчета и поставят една част от тях в началото на редичката, а останалите – в края. Те правят това, докато наредят всички кубчета. Възможно е всички кубчета от даден цвят да бъдат поставени само в началото или само в края.

Напишете програма, която определя дали по този начин Ани и Боби могат да конструират $\boldsymbol{N}$ зададени редици от кубчета, представени като символни низове.

Вход

От първия ред на входния файл `constructor.in` се въвеждат числата N и M. Следващите M реда съдържат по един символ L_i и едно число C_i, разделени с интервал. От всеки от последните N реда се въвежда по един символен низ, описващ поредната редица, за която трябва да проверите дали е възможно да бъде получена от наличните кубчета.

Изход

На N реда от изходния файл `constructor.out` изведете "Yes" или "No", в зависимост от това дали поредната редица от кубчета може да бъде получена.

Ограничения

$1 \leq N \leq 100$
$1 \leq M \leq 26$
Дължината на всеки един от въведените низове е не по-голяма от 100.
Сумата от всички C_i е не по-голяма от 100.

Пример

Вход	Изход
5 5 a 2 b 3 c 1 x 2 y 4 aabbbcxxyyyy ycbaabbxxyyy ababbxxyyycy xxbacabbyyy yaabbbcxyyyx	Yes Yes No No No

Partsort

СЕЗОН 10 – ВТОРИ РУНД

Дадена е редица, която е пермутация на числата от 1 до $\boldsymbol{N}$. Казваме, че редицата е сортирана, ако за всяко $i = 1, 2, \ldots, N - 1$ е изпълнено неравенството $A_i < A_{i+1}$, където с A_i означаваме i-тото число в редицата. В недалечното минало компютрите са разполагали с доста по-малко оперативна памет и именно заради това сортирането на редици не е било толкова тривиална задача.

Нека знаем, че в паметта може да поддържаме едновременно $\boldsymbol{K}$ от елементите на редицата. Ето защо сортираме числата с индекси между 1 и K, след това между 2 и $K + 1$ и т.н. до тези между $N - K + 1$ и N. За съжаление, това невинаги е достатъчно, за да сортираме редицата. Например, ако $N = 5$, $K = 3$ и $A = \{4, 5, 3, 1, 2\}$, последователно получаваме следните промени: $\{\mathbf{3}, \mathbf{4}, \mathbf{5}, 1, 2\} \rightarrow \{3, \mathbf{1}, \mathbf{4}, \mathbf{5}, 2\} \rightarrow \{3, 1, \mathbf{2}, \mathbf{4}, \mathbf{5}\}$. Така за една стъпка от началната редица получаваме $\{3, 1, 2, 4, 5\}$ и се налага да направим втора стъпка, за да сортираме редицата.

Напишете програма, която намира броя на стъпките, които ще извърши алгоритъмът, за да сортира редицата. Програмата трябва да обработва $\boldsymbol{T}$ тестови случая.

Вход

От първия ред на входния файл `partsort.in` се въвежда числото T. Следват T реда, всеки от които описва по един тестов случай във формат – $N, K, A_1, A_2, \ldots, A_N$.

Изход

На N реда от изходния файл `partsort.out` изведете по едно число, равно на търсения брой повторения на описания алгоритъм, за да се сортира редицата от съответния тестов случай.

Ограничения

$2 \leq K \leq N \leq 10\,000$

Сумата от $N \div K$ за всички тестови случаи е не по-голяма от 100.

Пример

Вход	Изход
3 5 3 4 5 3 1 2 3 2 1 2 3 7 7 7 6 5 4 3 2 1	2 0 1

Bit inversion

СЕЗОН 10 – ВТОРИ РУНД

Дадени са ви **N** двоични числа, всяко от които с по **M** бита (могат да имат водещи нули). Разполагате със специално устройство, което може да извършва следната операция по зададени от вас номер на число и позиция на бит: обръща всички битове на зададеното число (всяка нула става единица и всяка единица става нула) и освен това обръща бита на зададената позиция във всички останали числа. Целта ви е да използвате това устройство така, че всичките ви числа да станат равни на нула.

Вход (bitinversion.in)

На първия ред на входа се въвеждат числата **N** и **M**. На всеки от следващите N реда се въвежда по едно двоично число с M бита.

Изход (bitinversion.out)

Ако е невъзможно всичките числа да станат равни на нула чрез прилагане на гореописаната операция, изведете на единствен ред числото **-1**. В противен случай, на първия ред от изхода изведете числото **K** ($0 \le K \le 1\,000\,000$), равно на броя операции, които ще използвате. На всеки от следващите K реда изведете две числа $\mathbf{x_i}$ и $\mathbf{y_i}$ ($0 \le x_i < N$, $0 \le y_i < M$, най-младшият бит на всяко число има индекс 0, а най-старшият има индекс M-1) - номера на число и позицията на бит, които задавате на устройството за i-тата операция. Може да се докаже, че ако е възможно да направите всички числа равни на нула, то можете да го направите с до 1 000 000 операции. Ако има повече от един начина да направите всички числа равни на нула с не повече от 1 000 000 операции, изведете който и да е от тях.

Ограничения

$1 \le N*M \le 1\,000\,000$

Пример

Вход	Изход
2 3 010 100	3 0 1 0 0 1 1

Обяснение

Числата се изменят по следния начин:
(010, 100) → (101, 110) → (010, 111) → (000, 000)