Задача A. Автоматизация делимости

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

Даны два числа N и d. От вас требуется построить конечный детерминированный автомат над языком $\{0, 1...d-1\}$ такой, что распознаваемый им язык это все числа, записанные в d-ичной системе счисления, кратные N.

Под числом здесь понимается последовательность цифр, начинающаяся с ненулевой цифры.

Формат входных данных

В единственной строке входного файла даны два целых числа N и d ($1 \leqslant N \leqslant 10^5$, $2 \leqslant d \leqslant 10$).

Формат выходных данных

Выведите автомат в описанном ниже формате. У него должно быть не более чем $2 \cdot 10^5$ состояний. Для каждого теста гарантируется, что найдется автомат с не более чем $2 \cdot 10^5$ состояниями, распознающий искомый язык.

В первой строке выведите числа n, s, k : число состояний автомата ($1 \leqslant n \leqslant 2 \cdot 10^5$), начальное состояние ($0 \leqslant s < n$) и число терминальных состояний ($0 \leqslant k \leqslant n$), соответственно. В следующей строке выведите через пробел k чисел $t_1 \dots t_k$ — номера терминальных состояний ($0 \leqslant t_i < n$). В i-й из следующих n строк через пробел выведите по d чисел: $u_{i,0} \dots u_{i,d-1}$. В автомате переход из состояния i по цифре j ведет в состояние с номером $u_{i,j}$.

Примеры

стандартный ввод	стандартный вывод
1 3	3 1 1 0 0 0 0 2 0 0 2 2 2
2 2	4 2 1 0 0 1 0 1 3 1 3 3

Задача B. Регуляризация делимости

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Напомним определение *регулярного выражения*. В этом случае, над алфавитом из маленьких латинских букв (обозначим его Σ). Для регулярного выражения r обозначим язык, им задаваемый, за $L(r)$. Пустое слово обозначается ε. Для двух языков L_1, L_2 их конкатенация L_1L_2 это по определению $\{w_1w_2|w_1 \in L_1, w_2 \in L_2\}$. Под степенью языка L^k понимается конкатенация k копий языка L, $L^0 = \{\varepsilon\}$.

- 0 — регулярное выражение, $L(0) = \emptyset$.
- 1 — регулярное выражение, $L(1) = \{\varepsilon\}$.
- Если $\sigma \in \Sigma$, то σ — регулярное выражение, $L(\sigma) = \{\sigma\}$.
- Если r_1, r_2 — регулярные выражения, то $(r_1 + r_2)$ — регулярное выражение, $L((r_1 + r_2)) = L(r_1) \cup L(r_2)$.
- Если r_1, r_2 — регулярные выражения, то (r_1r_2) — регулярное выражение, $L((r_1r_2)) = \{w_1w_2|w_1 \in L(r_1), w_2 \in L(r_2)\}$.
- Если r — регулярное выражение, то r^* — регулярное выражение, $L(r^*) = \bigcup_{i=0}^{\infty} L(r)^i$.

Например, регулярное выражение $(a + b)^*$ задает язык из всех слов из букв a и b. А регулярное выражение $((((1 + a) + b)((1 + a) + b))((1 + a) + b))$ задает язык из всех слов из букв a и b длины не более 3. Смотрите примеры для лучшего понимания формата записи.

Даны два числа N и d. Обозначим за $\text{number}(a) = 0$, $\text{number}(b) = 1$ и так далее (то есть $\text{number}(\sigma)$, где σ это буква латинского алфавита, это порядковый номер σ в 0-индексации). Рассмотрим слово $w = \sigma_{k-1}\sigma_{k-2}...\sigma_0$. Положим $\text{number}(w) = \sum_{i=0}^{k-1} \text{number}(\sigma_i)d^i$. То есть, $\text{number}(w)$ это значение, полученное, если прочитать w как число в d-ичной системе счисления с цифрами от a до $(d-1)$-й буквы латинского алфавита. От вас требуется построить регулярное выражение, задающее язык $L = \{w|w \text{ не начинается с } a \text{ и } \text{number}(w) \vdots N\}$. То есть, задающее язык чисел, записанных в d-ичной системе счисления, кратных N.

Формат входных данных

В единственной строке входных данных вводятся два натуральных числа N и d ($1 \leqslant N \leqslant 10$, $2 \leqslant d \leqslant 20$). Гарантируется, что $N \cdot d \leqslant 20$.

Формат выходных данных

В единственной строке выведите регулярное выражение, определенное по правилам выше. Его длина должна не превосходить 20 000. Гарантируется, что такое регулярное выражение существует.

Примеры

стандартный ввод	стандартный вывод
1 3	((b+c)((a+b)+c)*)
2 2	(b((b+(a(a*b)))*(aa*)))

Задача C. Автоматизация требований

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Вам даны N строк $s_1 \dots s_n$, состоящих из первых k букв латинского алфавита. Строка называется *хорошей*, если у неё есть подстрока, равная какому-то s_i. От вас требуется построить полный детерминированный конечный автомат, распознающий язык *хороших* строк (над алфавитом из первых k букв латинского алфавита).

Формат входных данных

В первой строке входных данных вводится число k — размер алфавита ($1 \leqslant k \leqslant 5$). В следующей строке вводится число n — количество строк ($1 \leqslant n \leqslant 5 \cdot 10^5$). В каждой из следующих n строк вводится s_i. Гарантируется, что $\sum_{i=1}^{n} |s_i| \leqslant 5 \cdot 10^5$.

Формат выходных данных

Выведите автомат в описанном ниже формате. У него должно быть не более чем $5 \cdot 10^5 + 1$ состояний. Для каждого теста гарантируется, что найдется автомат с не более чем $5 \cdot 10^5 + 1$ состояниями, распознающий искомый язык.

В первой строке выведите числа n, s, f : число состояний автомата ($1 \leqslant n \leqslant 5 \cdot 10^5 + 1$), начальное состояние ($0 \leqslant s < n$) и число терминальных состояний ($0 \leqslant f \leqslant n$), соответственно. В следующей строке выведите через пробел $t_1 ... t_f$ — номера терминальных состояний ($0 \leqslant t_i < n$). В i-й из следующих n строк через пробел выведите по k чисел: $u_{i,1} \dots u_{i,k}$. В автомате переход из состояния i по j-й букве латинского алфавита ведет в состояние с номером $u_{i,j}$.

Примеры

стандартный ввод	стандартный вывод
1 1 a	2 0 1 1 1 1
2 2 ab ba	4 0 1 3 1 2 1 3 3 2 3 3

Задача D. Автоматизация эквивалентности

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

Даны два полных детерминированных конечных автомата над алфавитом из первых k латинских букв. Проверьте, эквивалентны ли они (то есть совпадают ли распознаваемые ими языки).

Формат входных данных

В первой строке входных данных вводится число k ($1 \leqslant k \leqslant 26$) — размер алфавита. Далее вводятся два автомата в формате, описанном ниже.

В первой строке описания автомата вводятся числа n, s, f : число состояний автомата ($1 \leqslant n \leqslant 10^5$), начальное состояние ($0 \leqslant s < n$) и число терминальных состояний ($0 \leqslant f \leqslant n$), соответственно. В следующей строке вводятся через пробел f чисел $t_1 ... t_f$ — номера терминальных состояний ($0 \leqslant t_i < n$). В i-й из следующих n строк через пробел вводятся по k чисел: $u_{i,1}$ ... $u_{i,k}$. Переход из состояния i по j-й букве латинского алфавита ведет в состояние с номером $u_{i,j}$.

Формат выходных данных

В единственной строке вывода выведите «Yes», если автоматы эквивалентны, и «No», иначе (без кавычек).

Примеры

стандартный ввод	стандартный вывод
3 3 0 2 0 1 1 0 0 2 0 2 2 2 2 5 4 4 4 0 3 1 2 4 2 0 1 4 2 2 2 2 1 2 3 1 4	Yes
2 1 0 1 0 0 0 3 0 2 0 1 1 2 2 0 0 1	No

Задача E. Минимизация автомата

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	10 секунд
Ограничение по памяти:	512 мегабайт

Вам дан полный детерминированный конечный автомат над алфавитом, состоящим из первых k маленьких латинских букв. От вас требуется построить минимальный полный детерминированный конечный автомат, эквивалентный данному.

Формат входных данных

В первой строке входных данных вводится одно число k — размер алфавита ($1 \leqslant k \leqslant 26$).

В следующей строке вводятся числа n, s, f : число состояний автомата ($1 \leqslant n \leqslant 10^5$), начальное состояние ($0 \leqslant s < n$) и число терминальных состояний ($0 \leqslant f \leqslant n$), соответственно. В следующей строке вводятся через пробел f чисел $t_1 ... t_f$ — номера терминальных состояний ($0 \leqslant t_i < n$). В i-й из следующих n строк через пробел вводятся по k чисел: $u_{i,1}$ ... $u_{i,k}$. Переход из состояния i по j-й букве латинского алфавита ведет в состояние с номером $u_{i,j}$.

Формат выходных данных

Выведите искомый минимальный автомат в том же формате, что и во входных данных (но без строки с числом k в начале).

Примеры

стандартный ввод	стандартный вывод
2 5 0 2 1 2 1 2 0 3 3 0 2 1 4 4	2 0 1 1 1 1 0 0
1 1 0 0 0	1 0 0 0

Задача F. Регуляризация автоматизации

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Напомним определение *регулярного выражения*. В этом случае, над алфавитом из маленьких латинских букв (обозначим его Σ). Для регулярного выражения r обозначим язык, им задаваемый, за $L(r)$. Пустое слово обозначается ε. Для двух языков L_1, L_2 их конкатенация L_1L_2 это по определению $\{w_1w_2|w_1 \in L_1, w_2 \in L_2\}$. Под степенью языка L^k понимается конкатенация k копий языка L, $L^0 = \{\varepsilon\}$.

- 0 — регулярное выражение, $L(0) = \emptyset$.
- 1 — регулярное выражение, $L(1) = \{\varepsilon\}$.
- Если $\sigma \in \Sigma$, то σ — регулярное выражение, $L(\sigma) = \{\sigma\}$.
- Если r_1, r_2 — регулярные выражения, то $(r_1 + r_2)$ — регулярное выражение, $L((r_1 + r_2)) = L(r_1) \cup L(r_2)$.
- Если r_1, r_2 — регулярные выражения, то (r_1r_2) — регулярное выражение, $L((r_1r_2)) = \{w_1w_2|w_1 \in L(r_1), w_2 \in L(r_2)\}$.
- Если r — регулярное выражение, то r^* — регулярное выражение, $L(r^*) = \bigcup_{i=0}^{\infty} L(r)^i$.

Например, регулярное выражение $(a + b)^*$ задает язык из всех слов из букв a и b. А регулярное выражение $((((1 + a) + b)((1 + a) + b))((1 + a) + b))$ задает язык из всех слов из букв a и b длины не более 3. Смотрите примеры для лучшего понимания формата записи.

Дано число k. Далее языки будут рассматриваться над алфавитом, состоящим из первых k латинских букв. Дано также регулярное выражение R. От вас требуется построить полный детерминированный конечный автомат такой, что язык, который им распознается, совпадает с языком, задаваемым R.

Формат входных данных

В первой строке входных данных находится одно число k ($1 \leqslant k \leqslant 26$) — размер алфавита. В следующей строке находится регулярное выражение R ($1 \leqslant |R| \leqslant 500$).

Формат выходных данных

Выведите автомат в описанном ниже формате. У него должно быть не более чем $2 \cdot 10^5$ состояний. Для каждого теста гарантируется, что найдется автомат с не более чем $2 \cdot 10^5$ состояниями, распознающий искомый язык.

В первой строке выведите числа n, s, f : число состояний автомата ($1 \leqslant n \leqslant 2 \cdot 10^5$), начальное состояние ($0 \leqslant s < n$) и число терминальных состояний ($0 \leqslant f \leqslant n$), соответственно. В следующей строке выведите через пробел числа $t_1...t_f$ — номера терминальных состояний ($0 \leqslant t_i < n$). В i-й из следующих n строк через пробел выведите по k чисел: $u_{i,1}$ ... $u_{i,k}$. В автомате переход из состояния i по j-й букве латинского алфавита ведет в состояние с номером $u_{i,j}$.

Примеры

стандартный ввод	стандартный вывод
1 a*	2 0 2 0 1 1 1
2 (a+b)	3 0 1 1 1 1 2 2 2 2
3 (a*c)*	4 0 2 0 3 1 2 3 1 2 3 2 2 2 1 2 3

Задача G. Палиндром?

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В данной задаче вам требуется построить детерминированную машину Тьюринга, которая будет проверять палиндромность бинарной строки.

Машина Тьюринга работает с бесконечной (в обе стороны) лентой над конечным алфавитом Σ ={0-1, a-z, #}. Символ # обозначает пустую ячейку ленты, символы a-z зарезервированы под специальные символы, которые может использовать ваша машина.

Машина имеет Q состояний, пронумеруем их целыми числами от 0 до $Q-1$. Среди них выделены три состояния $q_s, q_a, q_r (0 \leqslant q_s, q_a, q_r < Q; q_a \neq q_r)$, q_s — стартовое состояние, q_a — принимающее терминальное состояние, q_r — отвергающее терминальное состояние.

Работа машины Тьюринга проходит в соответствии с R заданными правилами $(q, c, nq, nc, act)(0 \leqslant q, nq < Q)(c, nc \in \Sigma)(act \in \{'L', 'N', 'R'\})$, которое означает, что если машина находится в состоянии q и в ячейке, на которую смотрит указатель, написан символ c, то машина записывает в эту ячейку символ nc, переходит в состояние nq и сдвигает указатель (на одну ячейку влево, если $act =' L'$, вправо, если $act =' R'$, и не сдвигает вовсе, если $act =' N'$). Машина должна быть **детерминированной**, то есть не может быть двух правил с одинаковой парой (q, c).

Работа машины останавливается, как только она попадает в одно из терминальных состояний. Работа также прекращается с ошибкой, если нет подходящего для применения правила. Также заметим, что если машина Тьюринга делает 10^5 шагов, после которых не оказывается в терминальном состоянии, мы считаем, что она **зациклилась** и ваше решение считается неверным.

Машина Тьюринга получает на вход строку s, это значит, что в некоторых |s| подряд идущих ячейках записаны символы $s_1, \dots s_n$, во всех остальных ячейках записан символ #, машина находится в состоянии q_s и указатель смотрит на ячейку, в которой записан s_1.

Определим вывод машины Тьюринга, если машина завершилась в состоянии q_a. Пусть указатель смотрит на какую-то ячейку, будем строить строку из символов, записанных в подряд идущих ячейках вправо, пока не встретим символ #. В том числе, если указатель смотрит на ячейку, где написан #, выводом машины считается пустое слово. Заметим, что символы, записанные левее ячейки, на которую смотрит указатель, и в ячейках после ближайшего # **не считаются** частью вывода машины.

Формат выходных данных

Ваша программа должна вывести описание машины Тьюринга. Обратите внимание на **ограничения**.

В первой строке выводится $Q (2 \leqslant Q \leqslant 16)$ — кол-во состояний машины Тьюринга.

Во второй строке, через пробел, выводятся три числа $q_s, q_a, q_r (0 \leqslant q_s, q_a, q_r < Q; q_a \neq q_r)$ — выделенные состояния машины.

В третьей строке выводится одно число R $(0 \leqslant R \leqslant 10^3)$ — кол-во правил машины.

В следующих R строчках заданы правила $q, c, nq, nc, act (0 \leqslant q, nq < Q)(c, nc \in \Sigma)(act \in \{'L', 'N', 'R'\})$ — правила должны задавать детерминированную машину Тьюринга.

После вывода описания машины **очистите буфер** вывода, как это делается в интерактивных задачах.

Протокол взаимодействия

На вход машине Тьюринга будет подаваться бинарная строка s длины не более 100 символов.

Ваша машина должна корректно отрабатывать на любой входной строке и завершаться в состоянии q_a, если введённая строка является палиндромом, и в q_r — иначе.

Пример

стандартный ввод	стандартный вывод
10101	3 0 1 2 3 0 0 0 # R 0 1 1 1 N 0 # 2 # L

Замечание

Машина Тьюринга из примера корректно завершает работу на любой бинарной строке и принимает (завершается в q_a) строки, которые содержат в себе хотя бы одну единицу.

Задача H. A+B

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В данной задаче вам требуется построить детерминированную машину Тьюринга, которая будет складывать два двоичных числа.

Машина Тьюринга работает с бесконечной (в обе стороны) лентой над конечным алфавитом $\Sigma =$\{0-1, +, a-z, #\}. Символ # обозначает пустую ячейку ленты, символы a-z зарезервированы под специальные символы, которые может использовать ваша машина.

Машина имеет Q состояний, пронумеруем их целыми числами от 0 до $Q-1$. Среди них выделены три состояния $q_s, q_a, q_r (0 \leqslant q_s, q_a, q_r < Q; q_a \neq q_r)$, q_s — стартовое состояние, q_a — принимающее терминальное состояние, q_r — отвергающее терминальное состояние.

Работа машины Тьюринга проходит в соответствии с R заданными правилами $(q, c, nq, nc, act)(0 \leqslant q, nq < Q)(c, nc \in \Sigma)(act \in \{'L', 'N', 'R'\})$, которое означает, что если машина находится в состоянии q и в ячейке, на которую смотрит указатель, написан символ c, то машина записывает в эту ячейку символ nc, переходит в состояние nq и сдвигает указатель (на одну ячейку влево, если $act =' L'$, вправо, если $act =' R'$ и не сдвигает вовсе, если $act =' N'$). Машина должна быть **детерминированной**, то есть не может быть двух правил с одинаковой парой (q, c).

Работа машины останавливается, как только она попадает в одно из терминальных состояний. Работа также прекращается с ошибкой, если нет подходящего для применения правила. Также заметим, что если машина Тьюринга делает 10^5 шагов, после которых не оказывается в терминальном состоянии, мы считаем, что она **зациклилась** и ваше решение считается неверным.

Машина Тьюринга получает на вход строку s, это значит, что в некоторых |s| подряд идущих ячейках записаны символы $s_1, \dots s_n$, во всех остальных ячейках записан символ #, машина находится в состоянии q_s и указатель смотрит на ячейку, в которой записан s_1.

Определим вывод машины Тьюринга, если машина завершилась в состоянии q_a. Пусть указатель смотрит на какую-то ячейку, будем строить строку из символов, записанных в подряд идущих ячейках вправо, пока не встретим символ #. В том числе, если указатель смотрит на ячейку, где написан #, выводом машины считается пустое слово. Заметим, что символы, записанные левее ячейки, на которую смотрит указатель, и в ячейках после ближайшего # **не считаются** частью вывода машины.

Формат выходных данных

Ваша программа должна вывести описание машины Тьюринга. Обратите внимание на **ограничения**.

В первой строке выводится $Q (2 \leqslant Q \leqslant 30)$ — кол-во состояний машины Тьюринга.

Во второй строке, через пробел, выводятся три числа $q_s, q_a, q_r (0 \leqslant q_s, q_a, q_r < Q; q_a \neq q_r)$ — выделенные состояния машины.

В третьей строке выводится одно число R $(0 \leqslant R \leqslant 10^3)$ — кол-во правил машины.

В следующих R строчках заданы правила $q, c, nq, nc, act (0 \leqslant q, nq < Q)(c, nc \in \Sigma)(act \in \{'L', 'N', 'R'\})$ — правила должны задавать детерминированную машину Тьюринга.

После вывода описания машины **очистите буфер** вывода, как это делается в интерактивных задачах.

Протокол взаимодействия

На вход машине Тьюринга будет подаваться строка A+B, где A, B $(0 \leqslant 0 < 2^{100})$ это двоичные числа, записанные без ведущих нулей.

Ваша машина должна корректно отрабатывать на любой входной строке, завершаться в состоянии q_a и выводить сумму чисел.

Пример

стандартный ввод	стандартный вывод
0+100	3 0 1 2 3 0 0 0 # R 0 1 0 # R 0 + 1 # R

Замечание

Машина Тьюринга из примера корректно отрабатывает на любом входе, завершается в состоянии q_a и выводит B в качестве суммы.