

## Задача А. 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  $\{0, 1 \dots d-1\}$  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  $d$  ( $1 \leq N \leq 10^5$ ,  $2 \leq d \leq 10$ ).

### 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 ( $1 \leq n \leq 2 \cdot 10^5$ ), начальное состояние ( $0 \leq s < n$ ) и число терминальных состояний ( $0 \leq k \leq n$ ), соответственно. В следующей строке выведите через пробел  $k$  чисел  $t_1 \dots t_k$  — номера терминальных состояний ( $0 \leq t_i < n$ ). 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 В. 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  $\Sigma$ ). Для регулярного выражения  $r$  обозначим язык, им задаваемый, за  $L(r)$ . Пустое слово обозначается  $\varepsilon$ . 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  $\sigma$  — 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  $\sigma$  это буква латинского алфавита, это порядковый номер  $\sigma$  в 0-индексации). Рассмотрим слово  $w = \sigma_{k-1}\sigma_{k-2}\dots\sigma_0$ . 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  $L = \{w | w \text{ не начинается с } a \text{ и } \text{number}(w) : N\}$ . 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  $d$  ( $1 \leq N \leq 10$ ,  $2 \leq d \leq 20$ ). Гарантируется, что  $N \cdot d \leq 20$ .

### 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 С. 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 ( $1 \leq k \leq 5$ ). В следующей строке вводится число  $n$  — количество строк ( $1 \leq n \leq 5 \cdot 10^5$ ). В каждой из следующих  $n$  строк вводится  $s_i$ . Гарантируется, что  $\sum_{i=1}^n |s_i| \leq 5 \cdot 10^5$ .

### 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 ( $1 \leq n \leq 5 \cdot 10^5 + 1$ ), начальное состояние ( $0 \leq s < n$ ) и число терминальных состояний ( $0 \leq f \leq n$ ), соответственно. В следующей строке выведите через пробел  $t_1 \dots t_f$  — номера терминальных состояний ( $0 \leq t_i < n$ ). 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	2 0 1
1	1
a	1
	1
2	4 0 1
2	3
ab	1 2
ba	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  $k$  ( $1 \leq k \leq 26$ ) — размер алфавита. Далее вводятся два автомата в формате, описанном ниже.

В первой строке описания автомата вводятся числа  $n, s, f$  : число состояний автомата ( $1 \leq n \leq 10^5$ ), начальное состояние ( $0 \leq s < n$ ) и число терминальных состояний ( $0 \leq f \leq n$ ), соответственно. В следующей строке вводятся через пробел  $f$  чисел  $t_1 \dots t_f$  — номера терминальных состояний ( $0 \leq t_i < n$ ). В  $i$ -й из следующих  $n$  строк через пробел вводятся по  $k$  чисел:  $u_{i,1} \dots u_{i,k}$ . 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 ( $1 \leq k \leq 26$ ).

В следующей строке вводятся числа  $n, s, f$ : число состояний автомата ( $1 \leq n \leq 10^5$ ), начальное состояние ( $0 \leq s < n$ ) и число терминальных состояний ( $0 \leq f \leq n$ ), соответственно. В следующей строке вводятся через пробел  $f$  чисел  $t_1 \dots t_f$  — номера терминальных состояний ( $0 \leq t_i < n$ ). В  $i$ -й из следующих  $n$  строк через пробел вводятся по  $k$  чисел:  $u_{i,1} \dots u_{i,k}$ . 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	2 0 1
5 0 2	1
1 2	1 1
1 2	0 0
0 3	
3 0	
2 1	
4 4	
1	1 0 0
1 0 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  $\Sigma$ ). Для регулярного выражения  $r$  обозначим язык, им задаваемый, за  $L(r)$ . Пустое слово обозначается  $\varepsilon$ . 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  $\sigma$  — 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  $k$  ( $1 \leq k \leq 26$ ) — размер алфавита. В следующей строке находится регулярное выражение  $R$  ( $1 \leq |R| \leq 500$ ).

### 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 ( $1 \leq n \leq 2 \cdot 10^5$ ), начальное состояние ( $0 \leq s < n$ ) и число терминальных состояний ( $0 \leq f \leq n$ ), соответственно. В следующей строке выведите через пробел числа  $t_1 \dots t_f$  — номера терминальных состояний ( $0 \leq t_i < n$ ). 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 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