Задача A. Расстояние Хэмминга

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Поздравляем! После многих лет обучения и подготовки вы были выбраны для работы в исследовательских лабораториях Зоны 51. Ваша первая задача - проанализировать ДНК пришельцев. В отличие от нашей, инопланетная ДНК построена из 26 различных типов нуклеотидов, которые мы обозначаем строчными латинскими буквами от '`a`' до '`z`'.

Одной из мер различия между строками a, b одинаковой длины l, представляющими нити ДНК, является количество индексов, в которых a, b различаются. Точнее, пусть $d(a, b)$ - это расстояние Хэмминга между a и b, которое представляет собой число индексов i таких, что $a[i] \neq b[i]$. Например, $d($"`gatyd`","`taxyd`"$) = 2$, так как первый и третий символы отличаются.

Ваша первая задача - проанализировать набор строк одинаковой длины, представляющих собой нити ДНК. Вы подозреваете, что все они являются небольшими мутациями какой-то общей нити ДНК.

То есть, вам дана коллекция строк S одинаковой длины l, представляющих ДНК пришельца, а также дано значение d_{lim}. Ваша задача - определить, существует ли строка s длины l такая, что для каждой строки $a \in S$ у вас есть $d(a, s) \leqslant d_{lim}$.

Формат входных данных

В первой строке записаны три целых числа n ($1 \leqslant n \leqslant 50$), l ($1 \leqslant l \leqslant 10^6$), and d_{lim} ($0 \leqslant d_{lim} \leqslant 8$). Далее следуют еще n строк, каждая из которых содержит по l символов. Каждая из строк содержит только латинские строчные буквы.

Также, гарантируется, что $n \cdot l \leqslant 10^6$.

Формат выходных данных

Если существует строка s, удовлетворяющая $d(s, a) \leqslant d_{lim}$ для каждой из n данных строк, выведите ее. Иначе, выведите единственную цифру '`0`'.

Примеры

стандартный ввод	стандартный вывод
2 4 1 abba bbca	abca
3 4 3 abcd efgh ijkl	abgl
3 4 2 abcd efgh ijkl	0

Задача B. Простота - высшая изысканность

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	8 секунд
Ограничение по памяти:	256 мегабайт

В мире существует n видов конфет, от 1 до n, и вы любите их все! Однако некоторые из них вы любите больше, чем другие. Точнее, для каждых двух разных видов конфет $i \neq j$ вы либо предпочитаете i, чем j, либо предпочитаете j, чем i.

Ваши предпочтения иногда доставляют вам головную боль. Когда вам дают какой-то набор конфет на выбор, вы хотите выбрать из них ту, которая вам больше всего нравится. Но ваши предпочтения иногда могут образовывать цикл, в результате чего не существует ни одного вида конфет, который вы предпочитаете всем остальным. Например, предположим, вы предпочитаете тип 1 типу 3, тип 3 типу 7, а тип 7 типу 1. Теперь, если вам дадут на выбор три конфеты 1, 3 и 7 типа, хорошего выбора не будет! Если вы выберете конфету 1-го типа, вы будете жалеть, что не выбрали конфету 7-го типа; но если вы действительно выберете конфету 7-го типа, вы начнете тосковать по конфете 3-го типа, и так далее.

Мы будем называть набор предпочтений логичным, если вышеописанная ситуация не может произойти— другими словами, если набор предпочтений не содержит циклов.

Вы обнаружили, что если запретить некоторое небольшое множество x (не более 15 элементов) типов конфет, то ваши предпочтения среди оставшихся типов конфет будут логичными. Однако вы не хотите запрещать слишком много типов конфет, поэтому вы хотите найти (возможно, другое) множество y типов конфет, которое содержит минимальное количество типов конфет и такое, что запрет всех типов из y приводит к тому, что оставшиеся предпочтения становятся логичными.

Формат входных данных

Первая строка входных данных содержит одно целое число n ($1 \leqslant n \leqslant 500$)— количество типов конфет.

Каждая из последующих n строк содержит по n символов без пробелов: j-й символ i-й строки равен '`1`' если Вы предпочитаете тип i типу j, или '`0`' если Вы предпочитаете тип j типу i или если $i = j$. Символ с координатами i, j никогда не равен символу с координатами j, i, если $i \neq j$.

Следующая строка содержит одно целое число k ($1 \leqslant k \leqslant 15$) — количество элементов во множестве x. Следующая строка содержит k различных чисел, разделенных пробелом, каждое между 1 и n— типы конфет в x. Гарантируется, что предпочтения конфет вне x логичны.

Формат выходных данных

В первой строки выведите m— число элементов во множестве y. На второй строке выведите m различных целых чисел от 1 до n— типы конфет в y. Предпочтения между конфетами вне y должны быть логичны, а m должно быть минимально возможным.

Примеры

стандартный ввод	стандартный вывод
4 0110 0011 0001 1000 2 2 3	1 4
3 011 001 000 3 1 2 3	0

Задача C. Количество решений

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дана логическая формула F от n переменных $x_1, x_2, \ldots, x_n$, имеющая вид $\bigwedge_{i=1}^{m} a_i \vee b_i$, где каждое из a_i и b_i равно либо одной из переменных, либо её отрицанию. Каждая переменная x_j может принимать два значения: истина и ложь.

Необходимо посчитать количество выполняющих наборов для этой формулы, то есть таких способов выбрать значения для всех x_j, что для всех i из a_i и b_i истинно хотя бы одно.

Посмотрите на пояснение к примеру для лучшего понимания.

Формат входных данных

В первой строке даны два целых числа n и m ($1 \leqslant n \leqslant 50$). В следующих m строках даны пары целых чисел a_i и b_i ($1 \leqslant |a_i|, |b_i| \leqslant n$). Положительным числом k от 1 до n обозначена переменная x_k, отрицательным числом k от -1 до $-n$ — отрицание $\neg x_{|k|}$.

Гарантируется, что для всех i выполнено $a_i < b_i$, а также что все пары (a_i, b_i) различны.

Формат выходных данных

Выведите одно целое число — количество выполняющих наборов формулы.

Пример

стандартный ввод	стандартный вывод
3 2 -2 1 2 3	4

Замечание

Пример задаёт формулу $F(x_1, x_2, x_3) = (\neg x_2 \vee x_1) \wedge (x_2 \vee x_3)$. В таблице ниже представлены значения этой формулы во всех возможных случаях. Видно, что есть четыре выполняющих набора.

x_1	x_2	x_3	$\neg x_2 \vee x_1$	$x_2 \vee x_3$	F
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

Задача D. Построитель графов

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Вася очень любит решать задачи на графах. Для этого он изобрёл специальный построитель графов, который строит только такие графы, которые ему нравятся.

У построителя есть память, в которой он может хранить не более k вершин одновременно. Он умеет делать следующие операции:

- Создать новую вершину и соединить ее с частью вершин в памяти. Для того, чтобы выполнить операцию, необходимо, чтобы в памяти было строго менее k вершин. Новая вершина помещается в память.
- Удалить вершину из памяти. При этом вернуть вершину в память позднее будет невозможно.
- Разветвиться. При этой операции построитель создаёт копию текущего состояния, выполняет первую ветку, после чего возвращается к текущему состоянию и выполняет вторую ветку операций. Обратите внимание, что при этом рёбер между вершинами, добавленными в разных ветках, появиться не может.

Вам дана программа для построителя графов. Необходимо построить тот же граф, а после этого найти в нем множесто вершин A такое, что у всех вершин, не лежащих в A, есть сосед из A. Среди всех таких множеств нужно выбрать наименьшее по размеру.

Формат входных данных

В первой строке задано число k ($1 \leqslant k \leqslant 7$), а также число n ($1 \leqslant n \leqslant 10\,000$) — длина программы для построителя. Программа задаётся в следующем формате:

- `New` t v_1 v_2 ... v_t для создания новой вершины. Вершине присваивается наименьший положительный номер, который ещё не был использован. Вершина соеденяется с перечисленными t вершинами. Гарантируется, что все эти вершины в этот момент находятся в памяти построителя.
- `Remove` v для удаления вершины v. Гарантируется, что в этот момент вершина v находится в памяти построителя.
- `Fork` для команды разветвления. После этой команды следуют команды для первой ветки, а за ними — команда `Return`, после которой идут команды для второй ветки. Команду `Return` следует считать относящейся к наиболее вложенной команде `Fork`, для которой ещё не было команды `Return`.

Формат выходных данных

В первой строке выведите одно число — Размер множества A. Во второй строке выведите номера вершин, содержащихся в множестве A, в любом порядке.

Пример

стандартный ввод	стандартный вывод
3 8 New 0 Fork New 1 1 Remove 1 New 1 2 Return New 1 1 New 2 1 4	2 2 5

Замечание

Память до	Действие	Память после
пусто	Добавлена вершина 1	1
1	Разделение в две ветки	1
1	Добавлены вершина 2 и ребро (1, 2)	1, 2
1, 2	Из памяти удалена вершина 1	2
2	Добавлены вершина 3 и ребро (2, 3)	2, 3
2, 3	Возврат к команде Fork	1
1	Добавлены вершина 4 и ребро (1, 4)	1, 4
1,4	Добавлены вершина 5 и ребро (1, 5), (4, 5)	1, 4, 5

Таким образом в конечном графе 5 вершин и 5 ребер: (1, 2), (2, 3), (1, 4), (1, 5), (4, 5);