# Задача A. Максимальное остовное дерево

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

Рассмотрим неориентированный граф. *Остовным деревом* называется подграф, который является деревом и покрывает все вершины графа. Если каждому ребру приписан вес, то можно говорить о *максимальном остовном дереве* графа.

Вам необходимо найти максимальное остовное дерево в неявно заданном полном графе. А именно, в вершинах графа записаны строки. Вес ребра, соединяющего две вершины — длина наибольшей общей подстроки двух соответствующих строк.

## Формат входных данных

В первой строке находится натуральное число $n$ — количество строк. Далее заданы сами строки. Каждая из них непуста и состоит из строчных букв латинского алфавита.

Суммарная длина всех строк не превосходит 500 000 символов.

## Формат выходных данных

В первую строку выведите одно число — вес максимального остовного дерева. Далее выведите $n - 1$ строку — описание рёбер искомого остовного дерева. Описание каждого ребра должно состоять из пяти чисел: номеров концов ребра и трёх чисел, которые описывают соответствующую наибольшую общую подстроку. Подстрока задается начальными позициями в первой и второй строке и своей длиной.

Вершины графа и позиции в строке нумеруются с единицы.

## Пример

| стандартный ввод | стандартный вывод |
|---|---|
| 2<br>xaby<br>abuv | 2<br>2 1 1 2 2 |

# Задача B. Парковка

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

Иван работает менеджером автомобильной парковки. Из-за МФК[1] его работа стала заметно скучнее: люди реже используют машины, в результате те всё больше времени стоят без дела на парковке. В результате Иван решил поразвлекаться. Парковка довольно новая, так что он может перемещать машины с места на место с помощью автоматики.

Однажды ему стало скучно (снова!) просто перемещать автомобили, и он придумал правило: теперь он никогда не ставит машину на то место, где раньше стояла машина того же производителя.

Все парковочные места пронумерованы целыми числами, начиная с 1. Оказалось, что автомобили произведены $n$ компаниями, при этом всего $a_i$ автомобилей $i$-й компании.

Иван уже поперемещал все машины, которые хотел, в результате чего все места с 1 по $a_1$ заняты продукцией 1-го производителя, с $a_1 + 1$ по $a_1 + a_2$ — 2-го, и так далее.

Через пять часов — открытие парковки, поэтому ему интересно начальное расположение автомобилей (владельцы не обрадуются, не найдя свою движимую собственность на месте...).

Ваша задача, во-первых, проверить, была ли совершена ошибка (то есть нарушил ли Иван правило перемещения). Если же этого не произошло, Иван планирует понадеяться на удачу и выбрать лексикографически минимальное расположение, которое могло быть изначально.

Одно расположение называется лексикографически меньше другого, если существует позиция $i$ такая, что во всех позициях $j < i$ они совпадают, а номер производителя автомобиля в $i$-й позиции в первом расположении меньше, чем во втором.

Иван решил, что заботиться о том, чтобы сами конкретные автомобили оказывались на местах, не обязательно. Ведь бензин, скорее всего, подорожает, и большинство владельцев автомобилей придёт, взглянет издали на своё парковочное место, убедится в том, что значок любимой марки на месте, вздохнет и двинется в сторону метро.

[1] МФК — мировой финансовый кризис

После того, как расположение выбрано, Иван решил посмотреть на некоторые части последовательности, которые он помнит лучше других. От вас требуется выяснить, как они выглядят.

Рассмотрим пример: на парковке 3 автомобиля первого производителя, 2 — второго и 2 — третьего. Тогда на первых трёх местах должны стоять автомобили производителей 2, 2, 3, соответственно (так как сейчас там стоят автомобили первого производителя). Следующее место будет занято автомобилем первого производителя. На пятое место следует поставить автомобиль третьего производителя, так как иначе на одной из двух последних позиций произойдёт нарушение правил. Наконец, на две последние позиции можно поставить два автомобиля первого производителя.

### Формат входных данных

В первой строке входных данных записано число $n$ — количество известных на парковке производителей автомобилей ($1 \leqslant n \leqslant 10\,000$). Затем следует $n$ чисел $a_i$ — количество автомобилей $i$-го производителя ($0 < a_i \leqslant 10^9$).

Затем следует число $k$ ($1 \leqslant k \leqslant 10\,000$) и $k$ пар чисел $l_i$, $r_i$ — отрезки, информацию о которых нужно вывести ($1 \leqslant l_i \leqslant r_i \leqslant \sum_{i=1}^{n} a_i$). Всего потребуется вывести не более 10 000 чисел.

### Формат выходных данных

Если Иван не мог не допустить ошибку, выведите единственную строку “`Bad luck`”. Иначе выведите $k + 1$ строку. На первой из них выведите сообщение “`Ok`”. В $i$-й из оставшихся выведите числа, соответствующие номерам производителей автомобилей на позициях с $l_i$ по $r_i$ в рассматриваемом Иваном расположении.

### Примеры

| стандартный ввод | стандартный вывод |
|---|---|
| 3<br>3 2 2<br>1<br>1 7 | Ok<br>2 2 3 1 3 1 1 |
| 2<br>4 1<br>1<br>2 3 | Bad luck |

## Задача C. Pattern Matching

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

Регулярные выражения — распространённый способ задания множества строк (возможно, бесконечного) в компактном виде. Регулярное выражение — это строка, состоящая из строчных латинских букв, знаков вопроса и звёздочек. Знак вопроса соответствует любой букве, а звёздочка — любой строке (возможно, пустой).

Вам задано регулярное выражение длиной от 1 до 5 символов и длинная строка, с которой проделываются следующие операции:

- вставить символ в строку,
- удалить символ из строки.

После каждой операции вы должны сообщать количество непустых подстрок текущей строки, которые удовлетворяют регулярному выражению.

Отметим, что, пока вы не выведете очередной ответ, информация о следующей операции доступна не будет.

### Формат входных данных

В первой строке находится регулярное выражение. Во второй строке находится непустая строка, состоящая из строчных латинских букв. Её длина не превосходит 100 000. В следующей строке находится натуральное число $k$ ($1 \leqslant k \leqslant 10\,000$) — количество операций. После считывания этих данных вы должны организовать интерактивное общение с тестирующей системой. Вы должны $k$ раз проделать следующую последовательность действий:

- считать описание очередной операции,
- выполнить эту операцию со строкой,
- вывести очередной ответ на отдельной строке,
- выполнить команду `flush(output)` в Паскале и `fflush(stdout)` в C и C++.

Операции задаются в формате ‘`+ c k`’ (добавление символа $c$ после позиции $k$) и ‘`- k`’ (удаление символа, который находится на позиции $k$). Гарантируется, что все требуемые операции корректны (то есть в операции добавления $0 \leqslant k \leqslant l$, где $l$ — текущая длина строки, а в операции удаления $1 \leqslant k \leqslant l$).

Не рекомендуется использовать потоки ввода/вывода C++ (`fstream`), так как они работают существенно медленней `FILE *` и могут стать причиной превышения ограничения по времени.

### Формат выходных данных

Смотрите описание входного файла.

### Пример

| стандартный ввод | стандартный вывод |
|---|---|
| a*b<br>aaab<br>2<br>+ b 1<br>- 2 | 4<br>3 |

## Задача D. Электрички

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

Между двумя секретными железнодорожными станциями (условно назовем их «Ма» и «Мо») установлено стабильное сообщение — каждый день от станции «Ма» в сторону станции «Мо» отправляется $k$ электричек. Начальник станции «Ма» решил оптимизировать схему движения электричек таким образом, чтобы каждый день перевозить как можно больше пассажиров.

Исследовав поток пассажиров, начальник станции выяснил, что пассажир номер $i$ приходит на станцию «Ма» в момент времени $a_i$, а уходит в момент времени $b_i$. При этом, если электричка приходит в любой момент времени между $a_i$ и $b_i$, включительно, то пассажир может в неё сесть. В каждую электричку может сесть не более $m$ пассажиров, причём начальник станции имеет право указывать пассажирам, садиться в электричку или нет. Также начальник станции может выбирать моменты отправления каждой электрички. Ему требуется составить такое расписание движения электричек и пассажиров, чтобы как можно больше пассажиров успешно отправилось в сторону станции «Мо». Будем считать, что временем посадки любого числа пассажиров на электричку можно пренебречь.

### Формат входных данных

В первой строке записано три числа $n$, $m$ и $k$ — число пассажиров, вместимость каждой электрички и количество электричек, которые можно отправить в течение дня ($1 \leqslant n \leqslant 100\,000$, $1 \leqslant m \leqslant n$, $1 \leqslant k \leqslant \min(n, 1000)$). Далее следует $n$ строк по два числа $a_i$, $b_i$ в каждой — моменты, в которые приходит и уходит $i$-й пассажир, соответственно ($-1\,000\,000\,000 \leqslant a_i \leqslant b_i \leqslant 1\,000\,000\,000$). Электрички и пассажиры нумеруются, начиная с единицы.

### Формат выходных данных

В первой строке выведите $q$ — максимальное количество пассажиров, которое можно отправить в сторону станции «Мо». Во второй строке выведите $k$ чисел $t_i$ — моменты времени отправления электричек ($-1\,000\,000\,000 \leqslant t_i \leqslant 1\,000\,000\,000$). В третьей строке выведите $n$ чисел $z_i$ — для каждого пассажира $i$ номер электрички, на которой он поедет ($0 \leqslant z_i \leqslant n$). Если пассажир не отправляется в сторону станции «Мо», выведите в качестве номера электрички для него ноль. Моменты времени отправления электричек выводите в произвольном порядке. Если решений несколько, выведите любое.

### Пример

| стандартный ввод | стандартный вывод |
|---|---|
| 5 2 2<br>1 3<br>3 5<br>2 4<br>6 8<br>6 7 | 4<br>3 6<br>1 1 0 2 2 |