

Задача А. Распределение призов

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

В спортивном соревновании приняло участие N команд. Результат каждого участника каждой команды оценивается в некоторую сумму призовых денег. Однако по правилам соревнования команде выдаётся только один приз. В качестве суммы приза команды организаторы хотели выбрать медиану призовых денег её участников, но призовой бюджет уже зафиксирован. Жюри хочет избежать обвинений в предвзятости, поэтому было решено распределить призы так, чтобы сумма абсолютных разностей по всем участникам между заработанными и полученными деньгами была минимальной.

Распределите призы между командами.

Формат входных данных

В первой строке задано $1 \leq N \leq 1000$ — число команд. В следующих N строках записаны команды. Первое число в строке $1 \leq M_i \leq 100$ — размер команды. Оставшиеся M_i целых чисел $0 \leq D_{ij} \leq 10^6$ — призовые деньги заработанные участниками команды. В последней строке записано единственное целое число $1 \leq T \leq 10^9$ — размер призового фонда.

Формат выходных данных

Выведите в одной строке N целых чисел $0 \leq T_i \leq 10^9$ — призы полученные командами. Весь призовой бюджет должен быть полностью израсходован, то есть $\sum_{i=1}^N T_i = T$. Также должна быть минимизирована величина $\sum_{i=1}^N \sum_{j=1}^{M_i} |D_{ij} - T_i|$. Если есть несколько оптимальных распределений призов, выведите любое из них.

Примеры

стандартный ввод	стандартный вывод
2 3 5 4 1 3 1 2 3 6	4 2
2 2 1 1 2 1 1 4	3 1
2 1 0 2 0 1 3	2 1
4 1 1 1 1 1 1 1 1 1	0 0 0 1

Замечание

В первом тесте организаторы могут выдать каждой команде её медиану. В остальных тестах им приходится изменять выдаваемые призы.

Задача В. Таргетинг на погоду

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

Лето уже в самом разгаре — а это значит, что нас всех ждут и дождливые, и жаркие дни. В дождь людям нужны зонты и дождевики, и производителям этих товаров крайне важно отслеживать рост интереса к ним. С другой стороны, производителям прохладительных напитков важны жаркие дни, когда на продвижение можно потратить большие бюджеты.

Итак, владельцам рекламных объявлений определённых тематик требуется возможность управлять настройками показов в зависимости от погоды. В этой задаче вам потребуется реализовать простейшую версию этой функциональности — таргетинг на температуру в определённые диапазоны дней.

В вашем распоряжении есть исторические данные о температуре, а также прогноз на ближайшее будущее. А именно:

- температура сегодня (этот день имеет номер 0);
- температура за прошедшие N дней (вчерашний имеет номер -1 , позавчерашний — -2 и так далее до дня $-N$, самого раннего из известных);
- прогноз температуры на ближайшие N дней (завтрашний имеет номер 1, послезавтрашний — 2 и так далее до дня N).

Кроме того, есть данные о настройках показа B рекламных объявлений. Каждому из них соответствуют четыре числа t_{min} , t_{max} , d_{min} , d_{max} , означающие, что объявление можно показывать только в том случае, если за период от дня d_{min} до дня d_{max} включительно температура принимала или примет значение от t_{min} до t_{max} включительно. При этом гарантируется, что $d_{min} \leq 0 \leq d_{max}$, то есть диапазон дней должен включать сегодняшний.

Обратите внимание, что **температура не дискретна!** Например, если вчера значение температуры было -5 , а сегодня -2 , то необходимо считать, что в период, включающий сегодняшний и вчерашний дни, температура принимала не только значения -5 и -2 , но и -4 и -3 (и даже $-3,14$, но в этой задаче все входные данные целочисленны).

Для каждого рекламного объявления определите, может ли оно быть показано в текущий день с учётом настроек и известных погодных данных.

Формат входных данных

В первой строке входного файла вводится единственное целое неотрицательное число N ($N \leq 100\,000$).

В следующей строке вводится $2N + 1$ целых чисел — значения температуры в дни $-N$, $-N + 1$, ..., -1 , 0 , 1 , ..., N . Значения температуры по модулю не превосходят 10^9 .

В третьей строке содержится единственное целое положительное число B ($B \leq 100\,000$) — количество рекламных объявлений.

Наконец, в каждой из следующих B строк содержатся 4 целых числа t_{min} , t_{max} , d_{min} , d_{max} , описывающие настройки соответствующего объявления. Гарантируется, что $-10^9 \leq t_{min} \leq t_{max} \leq 10^9$ и $-N \leq d_{min} \leq 0 \leq d_{max} \leq N$.

Формат выходных данных

Для каждого из объявлений выведите в отдельной строке «yes», если оно может быть показано согласно настройкам, и «no» в противном случае.

Пример

стандартный ввод	стандартный вывод
2	yes
2 4 5 -2 2	yes
4	no
-1 -1 -1 1	no
0 10 -1 0	
6 8 -2 2	
3 3 -1 0	

Задача С. Разбиение на рекламные блоки

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

Уже не первый год в Байтландии идёт тяжёлая борьба за быструю отрисовку страниц в интернете...

Одним из шагов к победе стало нововведение Баннерной крутилки: отдавать баннеры для нескольких блоков в одном ответе, тем самым сохранив дополнительные вызовы, а с ними и время. Однако оказалось, что совсем не просто разумно разбить полученные баннеры по блокам.

Имеется N баннеров, которые мы хотим разбить на K блоков. В ответе есть массив P , где P_i — прогнозируемая польза от показа i -го баннера, а также массив L , где L_j — возможная потеря внимания пользователя на j -й по счёту блок.

Аналитик Виталик тестирует гипотезы, которые должны в первую очередь повысить счастье пользователя. Сейчас он прорабатывает следующую метрику: для блока вводим функцию $f(Block_s) = P_{min} \cdot len(Block_s) - L_s P_{max}$, где P_{min} — минимальное значение P среди баннеров в блоке, P_{max} — максимальное, $len(Block_s)$ — количество баннеров в блоке, s — номер блока в разбиении. Значением метрики для конкретного разбиения является сумма значений f для всех блоков в этом разбиении. На каждом запросе необходимо находить разбиение, которое максимизирует эту метрику.

Виталик умеет решать эту задачу, но его алгоритм оказался не очень быстрым. Вас, как эксперта, попросили помочь ему и ускорить отбор!

Формат входных данных

Первая строка входных данных содержит 2 целых числа N и K ($1 \leq N \leq 5 \cdot 10^4$, $1 \leq K \leq \min(100, N)$).

Вторая строка содержит N целых чисел $1 \leq P_i \leq 10^6$; гарантируется, что $P_i > P_{i+1}$.

Третья строка содержит K целых чисел $0 \leq L_i \leq 10^6$.

Формат выходных данных

Выведите одно целое число — значение метрики для наилучшего разбиения.

Примеры

стандартный ввод	стандартный вывод
4 2 6 4 3 1 0 3	7
10 3 10 9 8 7 6 5 4 3 2 1 0 4 2	19

Замечание

В первом примере возможны три разбиения:

- $(6)|(4\ 3\ 1)$, со значением метрики -3 ;
- $(6\ 4)|(3\ 1)$, со значением метрики 1 ;
- $(6\ 4\ 3)|(1)$, со значением метрики 7 .

Задача D. Задача про потерянные данные

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

В распределённых системах одна из наиболее важных задач — это умение переживать сбои оборудования. Классическим подходом к решению проблемы сбоев в случае хранения данных является тройная репликация, то есть хранение каждого блока данных на трёх разных хостах в кластере. В таком случае данные не потеряются при одновременном выходе из строя двух дисков. Одновременная поломка трёх дисков считается крайне маловероятным событием и на практике практически никогда не случается.

Однако, тройная репликация приводит к тому, что для хранения X данных на кластере надо иметь $3X$ свободного места, что достаточно расточительно. Для экономии места и сохранения гарантий надёжности используются коды, исправляющие ошибки (а точнее, в нашем случае, коды, исправляющие потерю части данных).

Рассмотрим один из таких кодов. Представим, что наши данные состоят из числа битов, кратного 20. Для простоты будем считать, что ровно из 20 бит и научимся кодировать эти 20 бит (для большего числа бит задача решается простым повторением данного алгоритма).

Итак, представим наши 20 бит в виде матрицы $\{a_{i,j}\}$ размером 4 на 5, то есть $i = 1 \dots 4, j = 1 \dots 5$. Мы дополним матрицу с данными двумя контрольными столбцами $\{a_{i,6}\}$ и $\{a_{i,7}\}$.

Шестой столбец будет просто представлять собой хог-сумму столбцов, то есть:

$$a_{i,6} = \bigoplus_{j=1}^5 a_{i,j}, \quad i = 1 \dots 4$$

Седьмой столбец будет устроен более хитро. Он будет представлять собой определённые хог-суммы диагоналей. А именно, положим $S = \bigoplus_{k=1..4} a_{k,6-k}$, тогда:

$$a_{i,7} = S \oplus \bigoplus_{k=1..4} a_{k, \text{shift}(6+i-k)}, \quad i = 1 \dots 4$$

Здесь *shift* обозначает функцию, которая шагом величины 5 сдвигает число в диапазон $1 \dots 5$, (то есть $\text{shift}(x) \in \{1 \dots 5\}$ и $x - \text{shift}(x)$ кратно 5).

Изобразим вклад элементов в контрольные суммы 7-го столбца в виде матрицы:

$$\begin{pmatrix} a_{7,1} & a_{7,2} & a_{7,3} & a_{7,4} & S \\ a_{7,2} & a_{7,3} & a_{7,4} & S & a_{7,1} \\ a_{7,3} & a_{7,4} & S & a_{7,1} & a_{7,2} \\ a_{7,4} & S & a_{7,1} & a_{7,2} & a_{7,3} \end{pmatrix}$$

После этого мы будем писать каждый столбец матрицы на отдельный хост. Утверждается, что при потере любых двух столбцов их можно восстановить из имеющихся, то есть полученный код в определённом смысле не менее надёжен, чем тройная репликация. При этом видно, что накладные расходы указанного кода составляют лишь 40%, в отличие от 200%, которые были у тройной репликации.

Заметим, что при подсчёте контрольных сумм используется только операция хог, поэтому можно в качестве $a_{i,j}$ использовать числа фиксированной битовой ширины, а не просто биты. В нашем случае мы будем кодировать 32-битные беззнаковые числа.

Опишем немного точнее, как мы будем кодировать данные. Для кодирования данные разбиваются на пять непрерывных частей одинакового размера, причём размер части должен быть кратен четырём. Каждая из частей соответствует фиксированному столбцу, при этом первые четыре числа в части относятся к первой матрице, которую мы кодируем, вторые четыре числа ко второй матрице и так далее.

Ваша задача состоит в том, чтобы написать алгоритм, который восстанавливает потерянные данные при использовании указанной схемы кодирования.

Формат входных данных

В первой строке входных данных указано число n — количество кодируемых чисел в одной части и числа e_1, e_2 , обозначающие номера потерянных частей ($1 \leq e_1 < e_2 \leq 7, i = 1, 2$). Число n кратно четырём и не превосходит $2 \cdot 10^5$.

Далее в 5 строках заданы данные, отвечающие частям, которые остались в сохранности. В каждой из 5 строк записано n чисел, разделенных пробелами, относящихся к очередному сохранившемуся столбцу в порядке возрастания индексов столбцов.

Например, если были потеряны 1-й и 4-й столбцы, то в пяти строках будут указаны данные, относящиеся к 2-му, 3-му, 5-му, 6-му и 7-му столбцам соответственно.

Гарантируется, что входные данные были получены алгоритмом, описанным в условии.

Формат выходных данных

Выведите две строки по n чисел в каждой, соответствующие потерянным данным.

Примеры

стандартный ввод	стандартный вывод
4 6 7 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1 0 0 1 0 1 1 1
8 1 2 0 0 0 0 2 2 2 2 0 0 0 0 2 2 2 2 0 0 0 0 2 2 2 2 1 0 0 1 2 2 2 2 0 1 1 1 0 0 0 0	1 0 0 0 2 2 2 2 0 0 0 1 2 2 2 2

Замечание

Обратите внимание, что у задачи большой вход, поэтому в случае использования языка C++ рекомендуется выключить синхронизацию C/C++ способов чтения, то есть вызвать метод `std::ios::sync_with_stdio(false)`.

Задача Е. Критическая уязвимость

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

В n кластерах расположены сервера, на каждом из которых установлена одна и та же операционная система. В её ядре была найдена уязвимость, поэтому сервера нужно обновить на исправленную версию ядра. В i -м кластере расположено x_i серверов. Обновление одного сервера занимает одну единицу времени. Сервера обновляются строго последовательно, поэтому их суммарное время обновления занимает x_i единиц времени. Начатый процесс обновления в одном кластере нельзя прерывать до того, как все сервера в нём будут обновлены. Также нельзя выполнять обновление в двух кластерах одновременно. В каждом кластере под обновление было выделено окно времени $[a_i, a_i + x_i]$, эти окна могут пересекаться. Необходимо выбрать кластера, в которых провести обновление, чтобы на максимальном количестве серверов была установлена новая версия ядра.

Формат входных данных

В первой строке дано число кластеров n ($1 \leq n \leq 10^5$). В каждой из n следующих строк дана пара чисел a_i и x_i , разделённая пробелом ($1 \leq a_i, x_i \leq 10^9$).

Формат выходных данных

В первой строке выходного файла нужно вывести максимальное суммарное количество серверов, на которых возможно обновить ядро. Во второй строке выходного файла нужно вывести разделённые пробелом номера кластеров, в которых следует обновить ядро. Кластера нумеруются с нуля. Их номера можно вывести в произвольном порядке.

Примеры

стандартный ввод	стандартный вывод
4 1 4 4 11 8 5 12 5	11 1
4 1 4 4 11 8 3 12 5	12 3 2 0
2 1 1 2 1	2 1 0

Задача F. Трансформация директории

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

У разработчика была директория A , в которой размещались директории и файлы. Для каждого файла посчитана его хэш сумма. В процессе работы известно, что разработчик создавал хардлинки, удалял хардлинки, создавал и удалял директории. Более формально, изменения были одного из четырех типов:

1. `mkdir /some/dir/` — создать директорию с именем `/some/dir/` в директории A .
2. `rmdir /some/dir/` — удалить директорию с именем `/some/dir/` в директории A .
3. `unlink /some/file` — удалить хардлинку с именем `/some/file` в директории A .
4. `link /from/file /to/file` — создать хардлинку с именем `/to/file` на `/from/file` в директории A . Тогда хэш сумма файла `/to/file` будет совпадать с хэш суммой файла `/from/file`.

При каждом изменении разработчик соблюдал правила:

1. Нельзя создавать новые файлы, только хардлинки.
2. Нельзя изменять файлы и директории вне директории A .
3. Новые хардлинки можно создавать только на существующие в A файлы.
4. Можно удалять только пустые директории.
5. Нельзя создавать хардлинку в директории, которой не существует.
6. Нельзя создавать поддиректорию в директории, которой не существует.

В результате своей работы разработчик пришел к выводу, что это уже не директория A , а директория B . Тогда он решил рассмотреть все свои изменения от изначального состояния директории A (далее просто "от A ") до нынешнего состояния (далее "до B "). Но, к сожалению, он не смог найти историю изменений. Зато он смог найти список файлов с их хэш суммами и поддиректорий директории A и список файлов с их хэш суммами и поддиректорий директории B .

Разработчик заметил, что пространства имен директорий в A и файлов в B не пересекаются, то есть нет файла в B с именем, совпадающим с именем какой-либо директории в A . Аналогично, пространства имен файлов в A и директорий в B тоже не пересекаются, то есть нет файла в A с именем, совпадающим с именем какой-либо директории в B . К тому же, если в A и в B есть файлы с одинаковым именем, то их хэш суммы тоже одинаковые.

Разработчик хотел бы восстановить последовательность своих изменений, соответствующих всем описанным правилам, но это невозможно, поэтому он просит Вас помочь найти произвольную минимальную последовательность.

Найдите минимальную последовательность изменений, соответствующих описанным правилам, которая преобразует директорию A в директорию B .

Формат входных данных

В первой строке через пробел заданы 2 целых числа n и m ($0 \leq n, m \leq 10^4$) — количество файлов и директорий в A и количество файлов и директорий в B соответственно.

Каждая из следующих n строк содержит имя директории (оканчивается на `'/'`) или имя файла (не заканчивается на `'/'`) в A . Имя состоит из букв английского алфавита, цифр, знаков `'.'` и `'/'` и не содержит двух подряд идущих знаков `'/'`. Если это имя файла, то следом за ним через пробел указана хэш сумма — строка, состоящая из строчных букв английского алфавита и цифр.

Каждая из следующих m строк содержит имя директории (оканчивается на '/') или имя файла (не заканчивается на '/') в B . Имя состоит из букв английского алфавита, цифр, знаков '.' и '/' и не содержит двух подряд идущих знаков '/'. Если это имя файла, то следом за ним через пробел указана хэш сумма — строка, состоящая из строчных букв английского алфавита и цифр.

Длины имен от 2 до 256, длины хэш сумм от 1 до 256.

Формат выходных данных

В первой строке выведите целое число k — минимальное количество операций, за которое можно получить директорию B из директории A . В следующих k строках выведите сами операции в формате, описанном в условии.

Пример

стандартный ввод	стандартный вывод
4 3 /a/ /a/b.txt hash1 /a/d.txt hash2 /f/ /a/ /a/e/ /a/e/c.txt hash1	5 mkdir /a/e/ rmdir /f/ unlink /a/d.txt link /a/b.txt /a/e/c.txt unlink /a/b.txt

Задача G. Скачивание ресурсов в дата-центре

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1.5 секунд
Ограничение по памяти:	64 мегабайта

Сеть компании состоит из серверов, каждый из которых имеет уникальный целочисленный идентификатор. N пар серверов соединены друг с другом. Соединённые серверы образуют кластеры: два сервера относятся к одному и тому же кластеру, если от одного из них можно добраться до другого, перемещаясь по связям.

Периодически возникает необходимость скачать определённый файл на некоторый сервер. В сети работает сервис, аналогичный торрент-трекеру, который может сообщить, на каких серверах уже имеется необходимый файл.

Проблема, однако, заключается в том, что любой сервер может скачивать файлы только с серверов в его кластере.

Составьте программу, которая будет анализировать конфигурацию сети и сообщать, из каких источников определённый сервер может скачать необходимый файл.

Формат входных данных

Первая строка содержит целое число N ($0 \leq N \leq 10^6$) — количество связей между серверами.

Следующие N строк описывают связи между серверами. Каждая из них содержит целые числа A_i и B_i ($1 \leq A_i, B_i \leq 10^9$) — идентификаторы соединённых серверов.

Следующая строка содержит целое число Q ($1 \leq Q \leq 10^3$) — количество запросов на скачивание файлов.

Следующие $(2 \cdot Q)$ строк описывают запросы на скачивание файлов. Первая строка каждой пары содержит целые числа X_i и K_i ($1 \leq X_i \leq 10^9$, $1 \leq K_i \leq 100$) — соответственно идентификатор сервера, на который нужно скачать файл, и количество серверов, содержащих файл. Вторая строка каждой пары содержит K_i целых чисел Y_{ij} ($1 \leq Y_{ij} \leq 10^9$) — идентификаторы серверов, содержащих файл.

Формат выходных данных

Для каждого запроса в отдельной строке выведите сначала целое число R_j — количество серверов, с которых можно скачать файл. Затем выведите R_j целых чисел — идентификаторы серверов, с которых можно скачать файл. Серверы следует выводить в том порядке, в котором они перечислены в описании соответствующего запроса во входных данных.

Примеры

стандартный ввод	стандартный вывод
8 54578972 99368556 79699669 54578972 64508306 99368556 56554555 64508306 27101564 81480071 89445516 27101564 93762227 81480071 89808815 93762227 4 56554555 2 93762227 79699669 99368556 2 64508306 56554555 27101564 2 99368556 79699669 93762227 2 56554555 54578972	1 79699669 2 64508306 56554555 0 0
10 85619126 64616465 98159933 85619126 73978229 85619126 29978081 64616465 72404745 29978081 97698445 75243921 36815728 97698445 18360411 97698445 66445821 75243921 92142380 66445821 4 97698445 4 75243921 92142380 98159933 73978229 66445821 4 29978081 92142380 73978229 97698445 18360411 4 29978081 92142380 98159933 97698445 36815728 4 64616465 92142380 97698445 29978081	2 75243921 92142380 2 92142380 97698445 2 92142380 97698445 2 92142380 97698445

Задача Н. Рандомный Шард

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

Метаданные пользователей Яндекс.Почты лежат в БД PostgreSQL и разбиты на N шардов (теоретически N может быть очень большим). У каждого шарда есть вес, который постоянно пересчитывается пропорционально загрузке процессора на мастере шарда. Новому пользователю при регистрации автоматически присваивается шард случайно с учётом веса этого шарда. Нужно реализовать класс который хранит веса шардов и отвечает на следующие типы запросов:

1. Обновить вес какого-то шарда.
2. Получить по заданному $weight$ номер шарда i такого, что сумма весов шардов с номерами, находящимися на отрезке $[0, i]$ меньше или равна заданному весу, а сумма весов шардов с номерами, находящимися на отрезке $[0, i + 1]$ строго больше. Если сумма весов всех шардов не превосходит заданный вес, нужно вывести $n - 1$. Если вес нулевого шарда больше, чем $weight$, то надо вывести -1 .

Формат входных данных

Первая строка содержит два целых числа n ($1 \leq n \leq 10^6$) — количество шардов (шарды нумеруются с 0); q ($1 \leq q \leq 10^5$) — количество запросов.

Каждая из следующих q строк содержит запрос. Запрос описан в одном из форматов:

- $+ shard\ weight$ ($0 \leq shard < n, 1 \leq weight \leq 10^7$) — означает, что надо увеличить вес шарда с номером $shard$ на $weight$ единиц.
- $? weight$ ($0 \leq weight \leq 10^{15}$) — означает, что надо отдать номер шарда i который удовлетворяет следующему условию: сумма весов шардов с номерами, находящимися на отрезке $[0, i]$, не превосходит $weight$, а сумма весов шардов с номерами, находящимися на отрезке $[0, i + 1]$, превосходит $weight$. При этом если сумма весов всех шардов не превосходит $weight$, то надо вывести $n - 1$. Если вес нулевого шарда больше, чем $weight$, то надо вывести -1 .

Формат выходных данных

Для каждого запроса формата 2 выведите номер шарда, который удовлетворяет условию, описанному выше.

Пример

стандартный ввод	стандартный вывод
1 5	0
+ 0 2	-1
+ 0 1	
+ 0 3	
? 6	
? 5	

Задача I. Михаил и билборды

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

Индустриализация добралась до самых отдалённых уголков страны, и наружная реклама уже повсюду показывается в Байдландии.

Реклама на билбордах показывается по определённым правилам:

- каждый рекламодатель назначает ставку, которую он готов заплатить за своё объявление;
- в момент предполагаемого показа определяется список рекламных объявлений, готовых показаться на билборде, который находится в определённой точке;
- RTB-система (Real-time bidding) проводит аукцион второй цены между всеми объявлениями.

Аукцион второй цены проходит по следующей схеме:

- отсеиваются объявления, которые не готовы заплатить не менее минимальной ставки $MinCost$;
- рекламные объявления сортируются по убыванию ставок;
- первое объявление назначается победителем;
- со счёта рекламодателя списывается ставка второго объявления или $MinCost$, если объявление в аукционе одно.
- если есть несколько объявлений с максимальной ставкой $Cost$, то победитель из них выбирается случайно; в таком случае с рекламодателя списывается $Cost$.

Одной из новинок в индустрии рекламы является геотаргетинг — возможность задать окружность на карте, ограничивающую множество щитов, на которых необходимо производить рекламные показы конкретного объявления. Волевым решением Байдландия была выбрана в качестве испытательного полигона для новой функциональности. Перед аналитиком Михаилом стоит задача: определить места в городе, в которых рекламодатели готовы заплатить наибольшее количество денег. Для этого Михаил прикрепил огромный билборд на свой автомобиль и решил проехать по прямой улице из точки (X_{start}, Y_{start}) . Автомобиль Михаила может ехать только с постоянной скоростью. В некоторые моменты времени аналитик останавливается и проверяет, какая реклама показывается на его билборде.

Помогите аналитику Михаилу определить лучшие места для расположения билбордов.

Формат входных данных

Первая строка входных данных содержит 4 целых числа $-10^9 \leq X_{start}, Y_{start}, dx, dy \leq 10^9$ — стартовую точку и вектор движения, который автомобиль проезжает за единицу времени. Гарантируется, что либо dx , либо dy равен нулю, но при этом они не равны нулю одновременно.

Вторая строка содержит количество остановок $1 \leq N \leq 3 \cdot 10^5$. Третья строка содержит N возрастающих целых чисел $0 \leq t_i \leq 10^9$. Четвёртая строка содержит 2 числа: количество объявлений $1 \leq M \leq 10^5$ и минимальную ставку $1 \leq MinCost \leq 10^9$.

Далее следует M окружностей, описывающих настройки соответствующих объявлений. Каждая окружность описывается 4 целыми числами: $-10^9 \leq X, Y \leq 10^9$ — центр окружности, радиус $1 \leq R \leq 10^9$ и ставка $1 \leq Cost \leq 10^9$.

Формат выходных данных

Выведите N чисел — стоимость победившего объявления в моменты времени $t_1, \dots, t_N$.

Примеры

стандартный ввод	стандартный вывод
0 0 0 1 10 0 1 2 3 4 5 6 7 8 9 4 50 0 2 1 100 0 4 1 200 0 6 1 300 0 7 1 250	50 50 50 100 50 200 250 250 50 50
-3 -1 3 0 3 1 3 20 3 100 6 -1 1 400 8 -1 9 300 0 -3 3 200	200 300 100

Задача J. Спасти JSON

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

У вас есть записанный в строку невалидный json. Заранее известно что в нём был потерян один символ. Надо восстановить json до валидного добавив один символ. Об исходном JSON сообщении нам известно:

1. json или строка или словарь.
2. Словарь состоит из пар ключ значение разделенных двоеточием, следующая пара отделяется запятой.
3. Ключ словаря всегда строка. Уникальность ключей не требуется. Значение может быть строкой или словарем.
4. Строки выделяются только двойными кавычками " .
5. Строки не бывают пустыми и состоят только из символов [a-zA-Z0-9].
6. Словарь может быть пустым.
7. Пробелы отсутствуют.

Формат входных данных

Строка с не валидным json длиной от 1 до 10^5 .

Формат выходных данных

Строка с валидным json отличающаяся от входной одним добавленным символом.

Примеры

стандартный ввод	стандартный вывод
{"key": "value"}	{"key": "value"}
{"key" "value"}	{"key": "value"}

Задача К. Список писем

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

Электронное письмо можно упрощённо представить в виде структуры

```
struct Message {  
    int userId; // id получателя  
    string from; // email-адрес отправителя  
    unsigned receiveTimestamp; // время получения  
    int folderId; // id папки, в которую доставлено письмо  
    int threadId; // id цепочки (треда), в которую группируются письма при обсуждении  
};
```

Нужно реализовать хранилище писем, в которое можно складывать письма и у которого можно запрашивать список писем по параметрам. Параметрами для запроса являются:

1. id юзера *userId*
2. id папки *folderId*
3. отрезок времени получения письма [*since*, *till*], включительно
4. максимальное количество элементов в выдаче *count*

Письма в выдаче должны быть сгруппированы в треда (сначала идут письма первого треда, затем второго и так далее). Таймстемпом треда считается таймстемп самого свежего письма в этом треде.

Должны выбираться только треда, принадлежащие папке *folderId*. Считается, что тред принадлежит папке *folderId*, если хотя бы одно письмо из этого треда лежит в папке *folderId*. Таким образом в выдаче писем могут присутствовать письма с *folderId*, отличающимся от переданного.

В выдачу должны попасть только треда, таймстемпы которых попадают в отрезок [*since*, *till*], включительно.

Треда в выдаче должны быть отсортированы по убыванию своих таймстемпов. Для тредов с одинаковым таймстемпом сортировка идет по увеличению *threadId*. Письма внутри треда должны быть отсортированы по убыванию таймстемпа. Для писем с одинаковым таймстемпом сортировка идет по порядку сохранения в хранилище.

После всех группировок применяется правило для ограничения количества элементов в выдаче: надо выбрать *count* первых.

Формат входных данных

На вход программе подается последовательность строк вида

```
n q  
+ <userId> <from> <timestamp> <folderId> <threadId>  
...  
? <userId> <folderId> <since> <till> <count>  
...
```

1. Первая строка содержит два целых числа n (количество запросов на добавление письма, $0 \leq n \leq 10^5$) и q (кол-во запросов на выборку писем, $0 \leq q \leq 10^4$)
2. после первой строки идет $n + q$ запросов: $+$ (действие "сложить письмо") или $?$ (действие "получить выдачу"), действия могут чередоваться
3. *userId*, *folderId*, *threadId* - произвольные целые числа, $0 \leq userId \leq 10^5$, $0 \leq folderId \leq 10^2$, $0 \leq threadId \leq 10^5$

4. *timestamp*, *since*, *till* - произвольные целые числа в отрезке $[0, 10^9]$, причем $since \leq till$
5. $0 \leq count \leq 10^2$
6. *from* - строка из символов $[a - z0 - 9@.]$, длиной M , где $1 \leq M \leq 256$

Формат выходных данных

Программа должна для каждой операции ? выдать следующую секцию:

M
<userId> <from> <timestamp> <folderId> <threadId>
...
<userId> <from> <timestamp> <folderId> <threadId>

где M - количество элементов в выдаче на запрос ?. После строки с числом M следует столько же строк, каждое обозначает письмо. Количество таких секций равняется количеству действий ? во входном файле.

Примеры

стандартный ввод	стандартный вывод
8 1 + 0 hello0.0.0@yandex.ru 39 0 0 + 0 hello0.0.1@yandex.ru 83 0 0 + 0 hello0.1.0@yandex.ru 91 1 0 + 0 hello0.1.1@yandex.ru 0 1 0 + 1 hello1.0.0@yandex.ru 61 0 0 + 1 hello1.0.1@yandex.ru 64 0 0 + 1 hello1.1.0@yandex.ru 89 1 0 + 1 hello1.1.1@yandex.ru 45 1 0 ? 1 0 3 90 2	2 1 hello1.1.0@yandex.ru 89 1 0 1 hello1.0.1@yandex.ru 64 0 0
2 1 + 1 hello1@yandex.ru 1 1 1 + 1 hello2@yandex.ru 2 1 2 ? 1 1 0 10 1	1 1 hello2@yandex.ru 2 1 2

Задача L. Набор классификаторов

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

При решении различных задач используются классификационные алгоритмы (или *классификаторы*), которые относят наблюдаемый объект к определённому классу. Например, рассмотрим *запросные* классификаторы, которые относят запрос пользователя в поисковую систему к той или иной группе запросов: спортивные события, новости, рецепты и пр. Такие классификаторы помогают улучшить взаимодействие пользователя с сервисом.

При внедрении классификаторов смотрят на метрики и на время работы. Метрики отвечают за пользу, которую классификаторы приносят сервису, а время работы должно быть небольшим — для удобства пользователя. Опытным путём было выявлено, что для быстрого и комфортного взаимодействия с поисковой системой, одновременно может работать не более M классификаторов.

Всего есть K метрик, которые определены для каждого классификатора. Для набора классификаторов значение метрики определяется как **максимум** по ней среди классификаторов, входящих в набор. **Пользой** набора классификаторов считается сумма его метрик.

Вам дано N классификаторов. Необходимо найти набор из M классификаторов с максимальной пользой.

Формат входных данных

Первая строка содержит три целых числа N , M и K ($1 \leq N \leq 2000$, $1 \leq M \leq N$, $1 \leq K \leq 15$) — количество классификаторов, размер искомого набора и количество метрик.

Следующие N строк описывают классификаторы. В i -й строке содержатся K значений метрик a_{ij} ($1 \leq a_{ij} \leq 10^8$) i -го классификатора.

Формат выходных данных

В первой строке выведите одно число — максимальную пользу, которую можно получить для набора из M классификаторов. Во второй строке выведите через пробел M чисел — номера классификаторов из набора с максимальной пользой.

Если таких наборов несколько, можно вывести любой.

Примеры

стандартный ввод	стандартный вывод
6 2 3 4 1 1 1 4 1 1 1 4 1 3 3 3 1 3 3 3 1	10 1 4
3 3 2 10 10 5 5 1 1	20 1 2 3