A. Сумматор (5 баллов)

3 секунды, 256 мегабайт

Эта задача познакомит вас с тестирующей системой Codeforces. Правильные решения задач должны проходить все заранее заготовленные тесты жюри и укладываться в ограничения по времени/памяти на каждом тесте. Ниже перечислены технические требования к решениям:

- решение располагается в одном файле исходного кода;
- решение читает входные данные со стандартного ввода (экрана);
- решение пишет выходные данные на стандартный вывод (экран);
- решение не взаимодействует как-либо с другими ресурсами компьютера (сеть, жесткий диск, процессы и прочее);
- решение использует только стандартную библиотеку языка;
- решение располагается в пакете по-умолчанию (или его аналоге для вашего языка), имеют стандартную точку входа для консольных программ;
- гарантируется, что во всех тестах выполняются все ограничения, что содержатся в условии задачи — как-либо проверять входные данные на корректность не надо, все тесты строго соответствуют описанному в задаче формату;
- выводи ответ в точности в том формате, как написано в условии задачи (не надо выводить «поясняющих» комментариев типа *введите число* или *ответ равен*);
- решения можно отправлять сколько угодно раз (пожалуйста, только без абьюза системы).

Для вашего удобства тесты, на которых будут тестироваться ваши решения, являются открытыми. В каждой задаче можно скачать архив тестов (смотрите сайдбар справа, раздел «Материалы соревнования»).

Перейдём к задаче.

Напишите программу, которая выводит сумму двух целых чисел.

Так как это ознакомительная задача, то вы можете посмотреть и отправить авторские примеры решений (смотрите ссылку в сайдбаре в разделе «Материалы соревнования»). Конечно, в других задачах примеры решений не предоставляются.

Входные данные

В первой строке входных данных содержится целое число t ($1 \le t \le 10^4$) — количество наборов входных данных в тесте.

Далее следуют описания t наборов входных данных, один набор в строке.

В первой (и единственной) строке набора записаны два целых числа a и b ($-1000 \le a, b \le 1000$).

Выходные данные

Для каждого набора входных данных выведите сумму двух заданных чисел, то есть $a + b$.

входные данные
5 256 42 1000 1000 -1000 1000 -1000 1000 20 22

выходные данные
298 2000 0 0 42

B. Сумма к оплате (10 баллов)

1 секунда, 256 мегабайт

В магазине акция: «купи три одинаковых товара и заплати только за два». Конечно, каждый купленный товар может участвовать лишь в одной акции. Акцию можно использовать многократно.

Например, если будут куплены 7 товаров одного вида по цене 2 за штуку и 5 товаров другого вида по цене 3 за штуку, то вместо $7 \cdot 2 + 5 \cdot 3$ надо будет оплатить $5 \cdot 2 + 4 \cdot 3 = 22$.

Считая, что одинаковые цены имеют только одинаковые товары, найдите сумму к оплате.

Неполные решения этой задачи (например, недостаточно эффективные) могут быть оценены частичным баллом.

Входные данные

В первой строке записано целое число t ($1 \le t \le 10^4$) — количество наборов входных данных.

Далее записаны наборы входных данных. Каждый начинается строкой, которая содержит n ($1 \le n \le 2 \cdot 10^5$) — количество купленных товаров. Следующая строка содержит их цены $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le 10^4$). Если цены двух товаров одинаковые, то надо считать, что это один и тот товар.

Гарантируется, что сумма значений n по всем тестам не превосходит $2 \cdot 10^5$.

Выходные данные

Выведите t целых чисел — суммы к оплате для каждого из наборов входных данных.

входные данные
6 12 2 2 2 2 2 2 3 3 3 3 3 3 12 2 3 2 3 2 2 3 2 3 2 2 3 1 10000 9 1 2 3 1 2 3 1 2 3 6 10000 10000 10000 10000 10000 10000 6 300 100 200 300 200 300

выходные данные
22 22 10000 12 40000 1100

C. Парное программирование (10 баллов)

1 секунда, 512 мегабайт

В компании работает n разработчиков, где n — **четное** число. Сумасшедший менеджер решил разбить всех разработчиков на команды по два человека.

Для этого он составил список всех разработчиков и назначил каждому из них номер по списку (от 1 до n) и значение a_i — уровень мастерства i-го в списке разработчика.

Очередную команду он составляет следующим образом:

1. первый разработчик в команде тот, кто идет первым в списке;
2. ему в пару подбирается такой, что разница их уровней минимальна (то есть минимально значение $|a_i - a_j|$, где $|x|$ — это модуль числа x); если таких кандидатов несколько, то выбирается из них тот, кто находится раньше в списке;
3. эти два разработчика образуют команду и удаляются из списка.

Например, если массив a равен $[2, 1, 3, 1, 1, 4]$, то формирование команд будет происходить следующим образом:

1. назначим разработчикам номера $[1, 2, 3, 4, 5, 6]$ в соответствии с их положением в списке, первый среди них имеет номер 1, его уровень мастерства $a_1 = 2$, подходящими (с минимальной абсолютной разностью) являются разработчики с номерами $2, 3, 4, 5$, первый среди них 2, таким образом первая команда — это разработчики с номерами 1 и 2;
2. оставшиеся разработчики теперь имеют номера $[3, 4, 5, 6]$, первый среди них 3, его уровень $a_3 = 3$, разработчик с минимальной абсолютной разностью только один (номер 6), таким образом команда — разработчики с номерами 3 и 6;
3. оставшиеся разработчики имеют номера $[4, 5]$, первый среди них 4, его уровень $a_4 = 1$, остался только разработчик с номером 5, таким образом третья команда — разработчики с номерами 4 и 5.

Ваша задача — помочь сумасшедшему менеджеру промоделировать процесс разбиения на команды. Обратите внимание, что команды должны быть выведены в порядке, описанном выше в условии.

Входные данные

Первая строка содержит одно целое число t ($1 \le t \le 50$) — количество наборов входных данных.

Первая строка каждого набора содержит одно целое число n ($2 \le n \le 50$; n четное) — количество разработчиков.

Вторая строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100$), где a_i — уровень мастерства i-го разработчика.

Выходные данные

Для каждого набора входных данных выведите $\frac{n}{2}$ строк, i-я строка должна содержать пару чисел — номер первого и второго разработчика в i-й команде в порядке, описанном в условии.

Выводите пустую строку между выводами для наборов входных данных.

входные данные
3 6 2 1 3 1 1 4 2 5 5 8 1 4 2 5 4 2 6 3

выходные данные
1 2 3 6 4 5 1 2 1 3 2 5 4 7 6 8

Первый набор входных данных из примера разобран в условии задачи.

D. Электронная таблица (10 баллов)

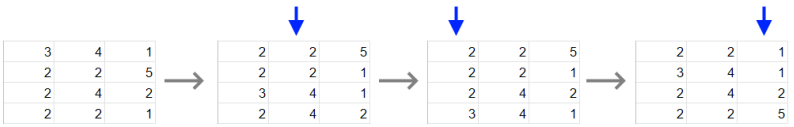
1 секунда, 256 мегабайт

Вам необходимо написать часть функциональности обработки сортировок в электронных таблицах.

Задана прямоугольная таблица $n \times m$ (n строк по m столбцов) из целых чисел.

Если кликнуть по заголовку i-го столбца, то строки таблицы пересортируются таким образом, что в этом столбце значения будут идти по неубыванию (то есть возрастанию или равенству). При этом, если у двух строк одинаковое значение в этом столбце, то относительный порядок строк не изменится.

Рассмотрим пример.



В этом примере сначала клик был совершен по второму столбцу, затем по первому и, наконец, по третьему.

Заметим, что если кликнуть подряд два раза в один столбец, то после второго клика таблица не изменится (в момент второго клика она уже отсортирована по этому столбцу).

Обработайте последовательность кликов и выведите состояние таблицы после всех кликов.

Неполные решения этой задачи (например, недостаточно эффективные) могут быть оценены частичным баллом.

Входные данные

В первой строке записано целое число t ($1 \le t \le 100$) — количество наборов входных данных в файле. Далее следуют описания наборов, перед каждым из них записана пустая строка.

В первой строке набора записаны два целых числа n и m ($1 \le n, m \le 30$) — количество строк и столбцов в таблице.

Далее следуют n строк по m целых чисел в каждой — начальное состояние таблицы. Все элементы таблицы от 1 до 100.

Затем входные данные содержат строку с один целым числом k ($1 \le k \le 30$) — количество кликов.

Следующая строка содержит k целых чисел $c_1, c_2, \ldots, c_k$ ($1 \le c_i \le m$) — номера столбцов, по которым были осуществлены клики. Клики даны в порядке их совершения.

Выходные данные

Для каждого набора входных данных выведите n строк по m чисел в каждой — итоговое состояние таблицы. После каждого набора выходных данных выводите дополнительный перевод строки.

входные данные
3 4 3 3 4 1 2 2 5 2 4 2 2 2 1 3 2 1 3 3 1 100 9 10 2 1 1 3 3 2 11 72 99 11 13 2 8 13 5 2 3 2 1 2

выходные данные
2 2 1 3 4 1 2 4 2 2 2 5 9 10 100 2 8 13 2 11 72 99 11 13

E. Отчет (15 баллов)

2 секунды, 512 мегабайт

Директор IT-корпорации оценивает эффективность работы сотрудников по различным показателям и критериям. Один из этих критериев сформулирован следующим образом: приступив к некоторому заданию, сотрудник должен завершить его, не переключаясь на другие задания.

Чтобы проверить сотрудников на соответствие этому критерию, директор потребовал от каждого сотрудника отчет о том, какие задания он выполнял в последние n дней. Отчет — это последовательность из n целых чисел $a_1, a_2, \ldots, a_n$, где a_i — идентификатор задания, которое сотрудник выполнял в i-й день.

Вам необходимо написать программу, проверяющую, соответствует ли сотрудник критерию по его отчету. Сотрудник соответствует этому критерию, если не существует такого задания x, которое выполнялось с перерывом (т. е. в некоторый день i сотрудник выполнял задание x, в дни с $i+1$ по $j-1$ он занимался другими заданиями, а в день j сотрудник продолжил выполнение задания x, при этом $j > i+1$). Иными словами, каждое задание, которое выполнял сотрудник, должно занимать один непрерывный отрезок дней.

Неполные решения этой задачи (например, недостаточно эффективные) могут быть оценены частичным баллом.

Входные данные

В первой строке задано одно целое число t ($1 \le t \le 10$) — количество наборов входных данных.

Каждый набор входных данных состоит из двух строк. В первой строке задано одно целое число n ($3 \le n \le 50000$). Во второй строке заданы n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — отчет сотрудника.

Выходные данные

Для каждого набора входных данных выведите ответ на отдельной строке. Если отчет соответствует критерию, выведите `YES`, иначе выведите `NO`.

входные данные
5 5 1 2 3 4 5 4 1 2 3 1 8 2 3 4 8 5 5 5 5 5 1 1 3 2 2 5 1 1 2 3 2

выходные данные
YES NO YES YES NO

F. Отрезки времени (20 баллов)

2 секунды, 512 мегабайт

Вам задан набор отрезков времени. Каждый отрезок задан в формате `HH:MM:SS-HH:MM:SS`, то есть сначала заданы часы, минуты и секунды левой границы отрезка, а затем часы, минуты и секунды правой границы.

Вам необходимо выполнить валидацию заданного набора отрезков времени. Иными словами, вам нужно проверить следующие условия:

- часы, минуты и секунды заданы корректно (то есть часы находятся в промежутке от 0 до 23, а минуты и секунды — в промежутке от 0 до 59);
- левая граница отрезка находится не позже его правой границы (но границы могут быть равны);
- никакая пара отрезков не пересекается (даже в граничных моментах времени).

Вам необходимо вывести `YES`, если заданный набор отрезков времени проходит валидацию, и `NO` в противном случае.

Вам необходимо ответить на t независимых наборов тестовых данных.

Неполные решения этой задачи (например, недостаточно эффективные) могут быть оценены частичным баллом.

Входные данные
Первая строка входных данных содержит одно целое число t ($1 \le t \le 10$) — количество наборов тестовых данных. Затем следуют t наборов.

Первая строка набора содержит одно целое число n ($1 \le n \le 2 \cdot 10^4$) — количество отрезков времени. В следующих n строках следуют описания отрезков.

Описание отрезка времени задано в формате `HH:MM:SS-HH:MM:SS`, где `HH`, `MM` и `SS` — последовательности из двух цифр. Заметьте, что никаких пробелов в описании формата нет. Также ни в одном описании нет пробелов в начале и конце строки.

Выходные данные
Для каждого набора тестовых данных выведите ответ — `YES`, если заданный набор отрезков времени проходит валидацию, и `NO` в противном случае. Ответы выводите в порядке следования наборов во входных данных.

входные данные
6 1 02:46:00-03:14:59 2 23:59:59-23:59:59 00:00:00-23:59:58 2 23:59:58-23:59:59 00:00:00-23:59:58 2 23:59:59-23:59:58 00:00:00-23:59:57 6 17:53:39-20:20:02 10:39:17-11:00:52 08:42:47-09:02:14 09:44:26-10:21:41 00:46:17-02:07:19 22:42:50-23:17:46 1 24:00:00-23:59:59

выходные данные
YES YES NO NO YES NO

G. Возможные друзья (20 баллов)

3 секунды, 512 мегабайт

Во многих социальных сетях у пользователей есть возможность указать других пользователей как своих друзей. Помимо этого, часто существует система рекомендации друзей, которая показывает пользователям людей, с которыми они знакомы косвенно (через кого-то из своих друзей), и предлагает добавить этих людей в список друзей. Вам предстоит разработать систему рекомендации друзей.

В интересующей нас социальной сети n пользователей, каждому из которых присвоен уникальный id от 1 до n. У каждого пользователя этой сети не более 5 друзей. Очевидно, ни один пользователь не является другом самому себе, и если пользователь x в списке друзей у пользователя y, то и пользователь y входит в список друзей пользователя x.

Опишем, как должен формироваться список возможных друзей для каждого пользователя. Для пользователя x в список должны входить такие пользователи y, что:

- y не является другом x и не совпадает с x;
- у пользователя y и у пользователя x есть хотя бы один общий друг;
- не существует такого пользователя y', который удовлетворяет первым двум ограничениям, и у которого **строго больше** общих друзей с x, чем у y с x.

Иными словами, в список возможных друзей пользователя x входят все такие пользователи, не являющиеся его друзьями, для которых количество общих друзей с x максимально. Обратите внимание, что список возможных друзей может быть пустым.

Вы должны написать программу, которая по заданной структуре социальной сети формирует списки возможных друзей для всех пользователей сети.

Неполные решения этой задачи (например, недостаточно эффективные) могут быть оценены частичным баллом.

Входные данные
В первой строке заданы два целых числа n и m ($2 \le n \le 50000$; $0 \le m \le \min(\frac{n(n-1)}{2}, \frac{5n}{2})$) — количество пользователей и количество пар друзей, соответственно.

Далее следуют m строк, в каждой из которых заданы два целых числа x_i и y_i ($1 \le x_i, y_i \le n; x_i \ne y_i$) — очередная пара друзей в социальной сети. Каждая пара друзей задается не более одного раза; у каждого пользователя не более 5 друзей.

Выходные данные
Для каждого пользователя от 1 до n выведите **в отдельной строке** список его возможных друзей в следующем формате:

- если список возможных друзей пуст, выведите одно целое число 0;
- иначе выведите id возможных друзей пользователя **в возрастающем порядке**.

входные данные
8 6 4 3 3 1 1 2 2 4 2 5 6 8

выходные данные
4 3 2 1 1 4 0 0 0

входные данные
8 10 1 2 1 3 1 4 4 3 3 2 2 4 1 8 5 6 7 6 5 7

выходные данные
0 8 8 8 0 0 0 2 3 4

Рассмотрим первый пример из условия.

Для начала сформируем списки друзей всех пользователей:

- друзья пользователя 1: $[2, 3]$.
- друзья пользователя 2: $[1, 4, 5]$.
- друзья пользователя 3: $[1, 4]$.
- друзья пользователя 4: $[2, 3]$.
- друзья пользователя 5: $[2]$.
- друзья пользователя 6: $[8]$.
- друзья пользователя 7: $[]$ (список друзей пуст).
- друзья пользователя 8: $[6]$.

Рассмотрим, как формируются списки возможных друзей для некоторых пользователей.

У пользователя 1 есть два пользователя, которые не являются его друзьями и с которыми у него есть хотя бы один общий друг: это пользователь 4 (общие друзья 2 и 3) и пользователь 5 (общий друг 2). С пользователем 4 общих друзей больше, поэтому в список возможных друзей попадает только он.

У пользователя 5 есть два пользователя, которые не являются его друзьями и с которыми у него есть хотя бы один общий друг: это пользователь 1 (общий друг 2) и пользователь 4 (общий друг 2). Количество общих друзей одинаковое, поэтому оба этих пользователя попадают в список возможных друзей.

У пользователя 7 вообще нет друзей, поэтому ни один пользователь не удовлетворяет требованиям списка возможных друзей.

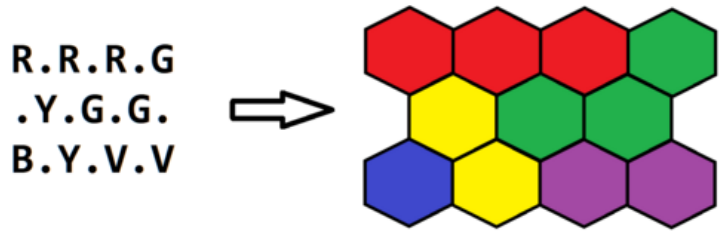
H. Валидация карты (25 баллов)

1 секунда, 512 мегабайт

В этой задаче вам необходимо реализовать валидацию корректности карты для стратегической компьютерной игры.

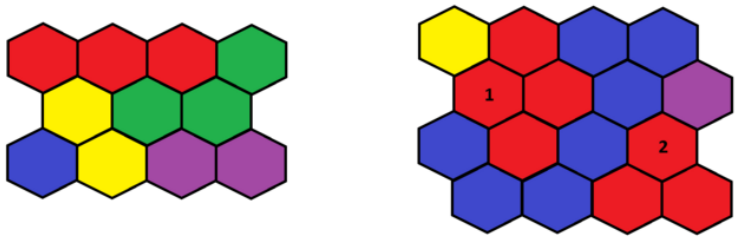
Карта состоит из гексагонов (шестиугольников), каждый из которых принадлежит какому-то региону карты. В файлах игры карта представлена как n строк по m символов в каждой (строки и символы в них нумеруются с единицы). Каждый нечетный символ каждой четной строки и каждый четный символ каждой нечетной строки — точка (символ `.` с ASCII кодом 46); все остальные символы соответствуют гексагонам и являются заглавными буквами латинского алфавита. Буква указывает на то, какому региону принадлежит гексагон.

Посмотрите на картинку ниже, чтобы понять, как описание карты в файлах игры соответствует карте из шестиугольников.



Соответствие описания карты в файле (слева) и самой карты (справа). Регионы `R`, `G`, `V`, `Y` и `B` окрашены в красный, зеленый, фиолетовый, желтый и синий цвет, соответственно.

Вы должны проверить, что каждый регион карты является одной связной областью. Иными словами, не должно быть двух гексагонов, принадлежащих одному и тому же региону, которые не соединены другими гексагонами этого же региона.



Карта слева является корректной. Карта справа не является корректной, так как гексагоны, обозначенные цифрами 1 и 2, принадлежат одному и тому же региону (обозначенному красным цветом), но не соединены другими гексагонами этого региона.

Неполные решения этой задачи (например, недостаточно эффективные) могут быть оценены частичным баллом.

Входные данные

В первой строке задано одно целое число t ($1 \le t \le 100$) — количество наборов входных данных.

Первая строка набора входных данных содержит два целых числа n и m ($2 \le n, m \le 20$) — количество строк и количество символов в каждой строке в описании карты.

Далее следуют n строк по m символов в каждой — описание карты. Каждый нечетный символ каждой четной строки и каждый четный символ каждой нечетной строки — точка (символ `.` с ASCII кодом 46); все остальные символы соответствуют гексагонам и являются заглавными буквами латинского алфавита.

Выходные данные

На каждый набор входных данных выведите ответ в отдельной строке — `YES`, если каждый регион карты представляет связную область, или `NO`, если это не так.

входные данные
3 3 7 R.R.R.G .Y.G.G. B.Y.V.V 4 8 Y.R.B.B. .R.R.B.V B.R.B.R. .B.B.R.R 2 7 G.B.R.G .G.G.G.
выходные данные
YES NO YES

Первые два набора входных данных из примера показаны на второй картинке в условии.

I. Планировщик задач (30 баллов)

3 секунды, 256 мегабайт

Представьте, вы собрали собственный сервер из n разнородных процессоров и теперь решили создать для него простейший планировщик задач.

Ваш сервер состоит из n процессоров. Но так как процессоры разные, то и достигают они одинаковой скорости работы при разном энергопотреблении. А именно, i-й процессор в нагрузке тратит a_i энергии за одну секунду.

Вашему серверу в качестве тестовой нагрузки придет m задач. Про каждую задачу вам известны два значения: t_j и l_j — момент времени, когда задача j придет и время выполнения задачи в секундах.

Для начала вы решили реализовать простейший планировщик, ведущий себя следующим образом: в момент t_j прихода задачи, вы выбираете свободный процессор с минимальным энергопотреблением и выполняете данную задачу на выбранном процессоре все заданное время. Если к моменту прихода задачи свободных процессоров нет, то вы просто отбрасываете задачу.

Процессор, на котором запущена задача j будет занят ровно l_j секунд, то есть освободится ровно в момент $t_j + l_j$ и в этот же момент уже может быть назначен для выполнения какой-то другой задачи.

Определите суммарное энергопотребление вашего сервера при обработке m заданных задач (будем считать, что процессоры в простое не потребляют энергию).

Неполные решения этой задачи (например, недостаточно эффективные) могут быть оценены частичным баллом.

Входные данные

В первой строке заданы два целых числа n и m ($1 \le n, m \le 3 \cdot 10^5$) — количество процессоров и задач соответственно.

Во второй строке заданы n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$) — энергопотребление соответствующих процессоров под нагрузкой **в секунду**. *Все энергопотребления различны.*

В следующих m строках заданы описания задач: по одному в строке. В j-й строке заданы два целых числа t_j и l_j ($1 \le t_j \le 10^9$; $1 \le l_j \le 10^6$) — момент прихода j-й задачи и время ее выполнения.

Все времена прихода t_j различны, и задачи заданы в порядке времени прихода.

Выходные данные

Выведите единственное число — суммарное энергопотребление сервера, если потреблением энергии в простое можно пренебречь.

входные данные
4 7 3 2 6 4 1 3 2 5 3 7 4 10 5 5 6 100 9 2
выходные данные
105

Рассмотрим работу планировщика посекундно:

- $t = 1$: приходит первая задача, все процессоры свободны. Задача занимает второй процессор на 3 секунды.
- $t = 2$: приходит вторая задача. Второй процессор занят, а потому задача занимает первый процессор на 5 секунд.
- $t = 3$: приходит третья задача и занимает четвертый процессор на 7 секунд.
- $t = 4$: приходит четвертая задача. Второй процессор освободился в данный момент, а потому его и занимает задача на 10 секунд.
- $t = 5$: приходит пятая задача и занимает последний свободный на данный момент процессор (третий) на 5 секунд.
- $t = 6$: приходит шестая задача. Все процессоры еще заняты, а потому задача отбрасывается.
- $t = 7$: освобождается первый процессор.
- $t = 9$: приходит седьмая задача и занимает первый процессор на 2 секунды.
- $t = 10$: освобождаются третий и четвертый процессоры.
- $t = 11$: освобождается первый процессор.
- $t = 14$: освобождается второй процессор.

Общее энергопотребление равно $3 \cdot 2 + 5 \cdot 3 + 7 \cdot 4 + 10 \cdot 2 + 5 \cdot 6 + 2 \cdot 3 = 6 + 15 + 28 + 20 + 30 + 6 = 105$.

J. Рифмы (30 баллов)

2 секунды, 512 мегабайт

Вы разрабатываете программу автоматической генерации стихотворений. Один из модулей этой программы должен подбирать рифмы к словам из некоторого словаря.

Словарь содержит n различных слов. Словами будем называть последовательности из 1—10 строчных букв латинского алфавита.

Зарифмованность двух слов — это длина их наибольшего общего суффикса (суффиксом будем называть какое-то количество букв в конце слова). Например:

- `task` и `flask` имеют зарифмованность 3 (наибольший общий суффикс — `ask`);
- `decide` и `code` имеют зарифмованность 2 (наибольший общий суффикс — `de`);
- `id` и `void` имеют зарифмованность 2 (наибольший общий суффикс — `id`);
- `code` и `forces` имеют зарифмованность 0.

Ваша программа должна обработать q запросов следующего вида: дано слово t_i (возможно, принадлежащее словарю), необходимо найти слово из словаря, которое **не совпадает** с t_i и имеет максимальную зарифмованность с t_i среди всех слов словаря, не совпадающих с t_i. Если подходящих слов несколько — выведите любое из них.

Неполные решения этой задачи (например, недостаточно эффективные) могут быть оценены частичным баллом.

Входные данные
Первая строка содержит одно целое число n ($2 \le n \le 50000$) — размер словаря.

Далее следуют n строк, i-я строка содержит одну строку s_i ($1 \le |s_i| \le 10$) — i-е слово из словаря. В словаре все слова различны.

Следующая строка содержит одно целое число q ($1 \le q \le 50000$) — количество запросов.

Далее следуют q строк, i-я строка содержит одну строку t_i ($1 \le |t_i| \le 10$) — i-й запрос.

Каждая строка s_i и каждая строка t_i состоит только из строчных букв латинского алфавита.

Выходные данные
Для каждого запроса выведите одну строку — слово из словаря, которое не совпадает с заданным в запросе и имеет с ним максимальную зарифмованность (если таких несколько — выведите любое).

входные данные
3 task decide id 6 flask code void forces id ask
выходные данные
task decide id task decide task

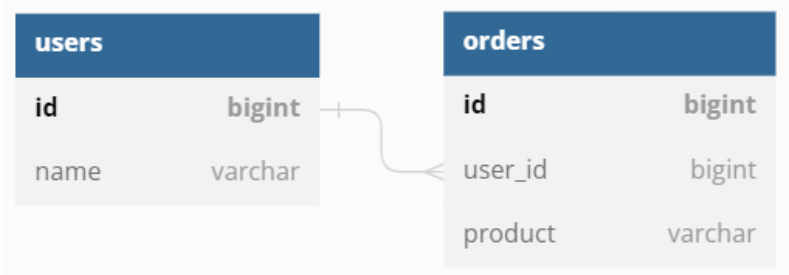
K. Пользователи и заказы (SQL, 10 баллов)

15 секунд, 1024 мегабайта

Это необычная задача — вам надо написать SQL-запрос. В качестве решения вы должны отослать один запрос к базе данных, который возвращает требуемые данные. При проверке вашего решения используется PostgreSQL 15.1. В качестве входных данных вам предоставляется дамп состояния базы данных. Обратите внимание, что время работы вашего решения на тесте включает восстановление состояния базы данных из дампа, но это время значительно меньше ограничения по времени. Вы можете использовать сервис http://sqlfiddle.com/ как инструмент для запуска запросов.

Напишите запрос к базе данных, который возвращает всех пользователей, сделавших хотя бы один заказ. Выведите всех таких пользователей, отсортировав их по имени (при равенстве по `id`).

Схема базы данных содержит две таблицы: `users` и `orders`, которые связаны отношением «один ко многим». Изучите входные данные примера, чтобы подробно ознакомиться со схемой базы данных. Диаграмма ниже иллюстрирует схему базы данных.



Входные данные
Входными данными в этой задаче является дамп базы данных. Вам он может быть полезен для ознакомления с состоянием базы данных для конкретного теста. В качестве решения вы должны отправить один SQL-запрос.

Выходные данные
Ваш SQL-запрос должен вывести всех пользователей в порядке неубывания их имён (по возрастанию id при равенстве имён). Используйте collation по умолчанию.

входные данные

```
create table users
(
    id   bigint primary key,
    name varchar not null
);

insert into users
values (1, 'john'),
       (2, 'Liza'),
       (7, 'Odin'),
       (11, 'donatello'),
       (17, 'spider-man'),
       (19, 'Elen'),
       (20, 'Liza');

create table orders
(
    id         bigint primary key,
    user_id    bigint  not null,
    product    varchar not null,
    constraint fk_orders_user_id foreign key (user_id) references
users (id)
);

insert into orders
values (101, 17, 'pizza'),
       (107, 2, 'toothpaste'),
       (108, 19, 'candies'),
       (109, 20, 'pizza'),
       (200, 17, 'shampoo'),
       (205, 2, 'pizza'),
       (210, 19, 'toothpaste'),
       (220, 19, 'pizza'),
       (221, 11, 'shampoo'),
       (222, 19, 'pizza');
```

выходные данные

```
 id |    name
----+------------
 11 | donatello
 19 | Elen
  2 | Liza
 20 | Liza
 17 | spider-man
(5 rows)
```