Задача A. Условие Фано

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Как известно из теории кодирования, для так называемых префиксных кодов должно выполняться условие Фано: никакое кодовое слово не должно быть началом другого кодового слова.

Вам будут даны различные наборы кодовых слов. Каждое кодовое слово представляет собой последовательность цифр. Длина последовательности не превышает 10 цифр.

Для каждого набора кодовых слов определите, удовлетворяет ли этот набор условию Фано.

Формат входных данных

Первая строка входных данных содержит целое число T ($1 \leqslant T \leqslant 40$) - количество наборов кодовых слов.

Каждый набор начинается записанным в отдельной строке числом N ($1 \leqslant N \leqslant 10000$) - количеством кодовых слов в наборе. Каждая из последующих N строк содержит кодовое слово, представляющее собой последовательность цифр длиной не более 10.

Формат выходных данных

Для каждого набора кодовых слов из входных данных выведите YES, если набор удовлетворяет условию Фано. Иначе выведите NO.

Примеры

стандартный ввод	стандартный вывод
2 3 512 123456789 51225426 5 112 122220 113440 54321 98346	NO YES

Задача B. Хэширование

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Рассмотрим следующую задачу. Вам дано некоторое множество слов D, образующее словарь, и вы хотите быстро отвечать на запрос, есть ли этом словаре заданное слово q. Одним из наиболее распространенных подходов к решению этой задачи является хэширование. Идея хэширования заключается в том, чтобы придумать функцию $h : \Sigma^* \to [0..n-1]$, которая ставит в соответствие каждой строке, состоящей из символов алфавита Σ, некоторое число из диапазона $0, 1, 2, ..., n-1$.

Другими словами, нужно придумать быстрый алгоритм, который получает в качестве входных данных строку, а возвращает в качестве результата своей работы целое число из диапазона от 0 до $n-1$ включительно. Если такой алгоритм придуман, то тогда можно завести массив T, называемый хэш-таблицей, и каждое слово w из словаря D записать в ячейку с индексом $h(w)$, то есть выполнить присваивание $T[h(w)] = w$. Теперь на запрос ответить очень просто. Для заданного слова q вычисляем значение хэш-функции $h(q)$ и если $T[h(q)]$ равно q, то слово q есть в словаре.

Но есть одна проблема - что если два разных слова из словаря D дают одно и то же значение хэш-функции, то есть должны быть размещены в одной и той же ячейке массива T? Такая ситуация называется коллизией. Одним из способов борьбы с коллизиями в хэш-таблицах является так называемое "кукушкино хэширование". Идея такого хэширования заключается в использовании двух хэш-функций h_1 и h_2. В этом случае каждому слову w из словаря будут соответствовать две ячейки в таблице - ячейка с индексом $h_1(w)$ и ячейка с индексом $h_2(w)$.

Будем строить таблицу T следующим образом. Будем перебирать слова w из словаря D и добавлять их в таблицу последовательно, одно за другим. Если ячейка $T[h_1(w)]$ свободна, записываем туда слово w, выполняя присваивание $T[h_1(w)] = w$. Если ячейка $T[h_1(w)]$ занята, а $T[h_2(w)]$ свободна, то записываем слово w в ячейку $T[h_2(w)]$. Если же обе ячейки заняты, то забираем слово r из ячейки $h_1(w)$, помещаем в нее слово w, а слово r пытаемся разместить в его вторую позицию $h_2(r)$. Если вдруг эта ячека окажется занятой, то извлекаем из нее слово, помещаем туда слово r, а извлеченное слово опять пытаемся поместить в таблицу в альтернативную позицию. И так далее. Конечно, может так получиться, что мы войдем в вечный цикл, но вероятность этого очень мала. Если такое случается, то это означает, что нужно перестроить всю таблицу с новыми функциями h_1 и h_2.

Формат входных данных

В первой строке входных данных содержится единственное целое положительное число T ($1 \leqslant T \leqslant 50$) - количество тестовых случаев.

Каждый тестовый случай начинается с двух целых положительных чисел $1 \leqslant m \leqslant n \leqslant 10000$, записаных в одной строке. m - это количество слов в словаре, а n - размер хэш-таблицы для этого тестового случая.

Далее следуют m строк, каждая из которых описывает i-ое слово в словаре с помощью двух неотрицательных чисел - соответствующих этому слову значений функции h_1 и h_2. Оба значения меньше n и могут совпадать.

Формат выходных данных

Для каждого тестового случая выведите единственную строку, содержащую слово *success*, если все слова можно записать в хэш-таблицу, или слово *rehash*, если нельзя.

Примеры

стандартный ввод	стандартный вывод
2 3 3 0 1 1 2 2 0 5 6 2 3 3 1 1 2 5 1 2 5	success rehash

Задача C. Магазины

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Представьте себе длинную улицу, вдоль которой расположены магазины. Будем считать, что улица - это координатная ось OX, а все магазины расположены в точках с целыми координатами. Вам необходимо обойти все магазины и вернуться в исходную точку, при этом начать обход вы можете из любой точки оси OX. Какой минимальный путь вам придется пройти, чтобы обойти все магазины и вернуться в начальную точку при оптимальном выборе точки, из который вы начнете обход?

Формат входных данных

В первой строке входных данных содержится единственное целое число T — количество тестовых случаев ($1 \leqslant T \leqslant 100$).

Каждый тестовый случай состоит из двух строк. В первой строке содержится количество магазинов N ($1 \leqslant N \leqslant 20$). Во второй строке записаны через пробел N целых чисел x_i ($0 \leqslant x_i \leqslant 99$) - координаты магазинов.

Формат выходных данных

Для каждого тестового случая выведите в отдельной строке единственное число — минимальный путь, который нужно пройти, чтобы обойти все магазины и вернуться в начальную точку, при оптимальном выборе начальной точки.

Примеры

стандартный ввод	стандартный вывод
2 4 24 13 89 37 6 7 30 41 14 39 42	152 70

Задача D. Пазл из подстрок

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Вам будут даны две строки S и T, состоящие только из маленьких букв английского алфавита. Вы можете разрезать строку S на несколько фрагментов произвольной длины и попытаться сложить из этих фрагментов строку T. При этом не обязательно использовать все полученные при разрезании фрагменты. Например, если $S =$ "$abcdef$", а $T =$ "de", можно просто вырезать из строки S нужный фрагмент, он и будет строкой T. В этом примере для того, чтобы получить строку T, понадобился один фрагмент.

Если $S =$ "$abcdef$", а $T =$ "fbc", то можно воспользоваться двумя фрагментами строки S, вырезав из нее подстроки "bc" и "f" и сложив их в нужном порядке.

Очевидно, что в общем случае сложить строку T из фрагментов строки S можно разными способами, используя различное количество вырезанных фрагментов. Например, если $S =$ "$tesatestcast$", а $T =$ "$test$", то сложить строку T из фрагментов строки S можно, например, такими способами: "t"+"e"+"s"+"t", просто вырезав нужные буквы, или "tes"+"t", или "te"+"st", или даже просто вырезав "$test$".

В этой задаче вам необходимо найти минимальное количество фрагментов строки S, из которых можно сложить строку T, или выяснить, что строку T сложить из S невозможно.

Формат входных данных

В первой строке входных данных записана непустая строка S, фрагменты которой можно использовать для того, чтобы сложить строку T. Длина строки S не превосходит 13 символов.

Во второй строке входных данных записана непустая строка T, которую нужно сложить из фрагментов строки S. Длина строки T не превосходит 13 символов.

Формат выходных данных

Если строку T можно сложить из фрагментов строки S, выведите единственное число - минимальное количество фрагментов строки S, которое потребуется, чтобы сложить строку T. Если строку T сложить из фрагментов строки S невозможно, выведите слово *impossible*.

Примеры

стандартный ввод	стандартный вывод
abcdef fbc	2
abc def	impossible

Задача E. Вирус

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Рассмотрим следующую модель распространения вируса. Рассмотрим связный граф с N узлами и M ребрами. Узлы графа графа - это города, между которыми может распространяться вирус. Если два узла соединены ребром, это означает, что есть канал распространения вируса из одного города в другой. Узлы графа нумеруются, начиная с 1.

Вам необходимо написать систему мониторинга распространения вируса, которая должна обрабатывать запросы трех типов:

- 1 v - этот запрос означает, что в городе с номером v обнаружен вирус. Необходимо пометить соответствующий узел графа.
- 2 t - этот запрос означает, что произошло t шагов распространения инфекции. Если на некотором шаге распространения инфекции город v заражен, то на следующем шаге будут заражены все смежные с ним города. Если город заразился, то он уже остается таким все время.
- 3 v - необходимо проверить статус города v. Если город заражен, система должна вывести YES, а иначе NO.

Формат входных данных

Первая строка входных данных содержит три целых числа N ($1 \leqslant N \leqslant 10^5$) - количество узлов графа, M ($1 \leqslant M \leqslant 3 \cdot 10^5$) - количество ребер и Q ($1 \leqslant Q \leqslant 10^5$) - количество запросов.

Далее следуют M строк, i-я строка содержит два целых числа u_i и v_i ($1 \leqslant u_i, v_i \leqslant N$), означающих, что узлы u_i и v_i соединены ребром.

Затем следуют Q строк, j-я из которых содержит два целых числа t_j и x_j - тип запроса и аргумент запроса. Если тип запроса равен 1 или 3, аргумент x_j является номером узла и лежит в диапазоне $1 \leqslant x_j \leqslant N$. Если тип запроса равен 2, то аргумент задает количество шагов распространения вируса и лежит в диапазоне $1 \leqslant x_j \leqslant 10^9$.

Формат выходных данных

Для каждого запроса третьего типа выведите YES, если соответствующий город заражен вирусом, иначе выведите NO.

Примеры

стандартный ввод	стандартный вывод
6 6 6 1 2 2 3 4 1 5 4 6 1 6 5 1 1 3 1 1 5 2 1 3 4 3 3	YES YES NO

Задача F. Робот-доставщик

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Робот-доставщик перемещается между N пунктами доставки, соединенными специальными дорогами. Робот оснащен электродвигателем, поэтому ему приходится периодически заряжать свою аккумуляторную батарею. Станции зарядки есть только в пунктах доставки, причем эти стации сильно разные и заряжают батарею робота с разной скоростью. Например, если скорость зарядки составляет 5 минут на единицу заряда, то для зарядки батареи до величины 10 единиц заряда потребуется 50 минут. Известно, что на единицу расстояния робот тратит единицу заряда. То есть если начальный заряд робота был 20 единиц, а робот проехал расстояние 15, то заряд его батареи станет равен 5 единицам. Если заряд батареи закончится, когда робот будет в дороге между пунктами доставки, он остановится и не сможет доставить товар.

Вам будет дана информация о дорогах, связывающих пункты доставки, и о скорости зарядки аккумулятора на каждом из этих пунктов. Очень важно, чтобы робот простаивал на зарядке как можно меньше, поэтому вам необходимо написать программу, которая определит минимально необходимое суммарное время простоя робота на зарядке в различных пунктах при перемещении из заданного начального пункта доставки в заданный конечный пункт.

Формат входных данных

Первая строка входных данных содержит два целых числа N ($1 \leqslant N \leqslant 1000$) и M ($0 \leqslant M \leqslant 7000$) - количество пунктов доставки и количество связывающих их дорог.

Далее следует строка, содержащая N целых чисел t_i - скорость зарядки робота на зарядной станции, расположенной в i-ом пункте доставки.

Далее следуют M строк, каждая из которых содержит три целых числа u, v и d ($0 \leqslant u, v < N; 1 \leqslant d \leqslant 100$), означающие, что между пунктами доставки с номерами u и v есть дорога длины d. Обратите внимание, что пункты доставки пронумерованы от 0 до $N - 1$.

Далее следует строка, содержащая единственное число Q ($1 \leqslant Q \leqslant 100$) - количество запросов.

Затем следуют Q строк, в каждой из которых записаны три числа, разделенные пробелами - емкость батареи робота ($1 \leqslant C \leqslant 100$), то есть максимальное количество единиц заряда, на которое можно зарядить батарею, начальный пункт S ($0 \leqslant S < N$) и конечный пункт T ($0 \leqslant T < N$). Сумма C по всем запросам не превосходит 3200.

Будем считать, что перед началом движения батарея робота полностью разряжена, то есть ему необходимо будет сначала подзарядить батарею до некоторого значения в начальном пункте S.

Формат выходных данных

Для каждого запроса выведите в отдельной строке минимальное время, которое робот должен провести, подзаряжая батарею, чтобы доехать из пункта S в пункт T, если максимальный заряд, на который можно зарядить батарею, равен C. Если робот не сможет довезти товар из пункта S в пункт T, выведите слово *impossible*.

Примеры

стандартный ввод	стандартный вывод
5 5 10 10 20 12 13 0 1 9 0 2 8 1 2 1 1 3 11 2 3 7 2 10 0 3 20 1 4	170 impossible

Задача G. Матрёшки

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

У вас есть набор из N пустых матрёшек. Известен размер каждой пустой матрешки - ее ширина W и высота H. Как известно, матрёшку с шириной W_1 и высотой H_1 можно вложить внутрь матрёшки шириной W_2 и высотой H_2 только тогда, когда размеры первой матрёшки стого меньше размеров второй матрёшки, то есть выполняются неравенства $W_1 < W_2$ и $H_1 < H_2$. Вам необходимо определить, какое минимальное количество сложенных матрёшек можно получить, вкладывая матрёшки одну в другую до тех пор, пока это возможно.

Формат входных данных

В первой строке входных данных содержится единственное целое число T — количество тестовых случаев ($1 \leqslant T \leqslant 20$).

Каждый тестовый случай состоит из двух строк. В первой строке содержиться единственное целое число N ($1 \leqslant N \leqslant 20000$) - количество матрёшек. В следующей строке записаны через пробел $2N$ положительных целых чисел $W_1, H_1, W_2, H_2, ..., W_N, H_N$, где W_i - это ширина, а H_i - это высота матрёшки с номером i. $1 \leqslant W_i, H_i \leqslant 10000$ для всех i.

Формат выходных данных

Для каждого тестового случая выведите в отдельную сроку единственное число - минимальное количество матрёшек, которое можно получить, вкладывая заданные матрёшки одну в другую, пока это возможно.

Примеры

стандартный ввод	стандартный вывод
4 3 20 30 40 50 30 40 4 20 30 10 10 30 20 40 50 3 10 30 20 20 30 10 4 10 10 20 30 40 50 39 51	1 2 3 2

Задача H. Замечательная скидка

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Вы зашли в магазин, в котором действует замечательная скидка - если вы покупаете несколько товаров, то каждый третий из купленных товаров вы получаете бесплатно. Естественно, продавец отдает бесплатно самые дешевые из выбранных вами товаров.

Например, если вы выбрали товары стоимостью 250, 300, 400, 350, 200, 100, and 150 рублей, всего семь товаров, продавец отдаст вам бесплатно два самых дешевых товара стоимостью 100 и 150. В результате вы получите скидку в 250 рублей, заплатив за оставшиеся товары 1500 рублей.

Но есть небольшая хитрость. Если вы купите эти же самые товары, но в несколько заходов, вы можете получить большую скидку. Например, вы сначала можете купить товары стоимостью 250, 300 и 400 рублей, получив скидку 250 рублей. Затем можно купить товар стоимостью 150 рублей, не получив при этом скидку. И, наконец, в третий заход можно купить товары стоимостью 350, 200 и 100 рублей, получив при этом скидку 100 рублей. Итак, вы купили те же самые товары, но теперь уже со скидкой 350 рублей.

Зная стоимости товаров, найдите максимальную скидку, с которой вы можете купить все эти товары.

Формат входных данных

В первой строке входных данных содержится единственное целое число T — количество тестовых случаев ($1 \leqslant T \leqslant 20000$).

Каждый тестовый случай состоит из двух строк. В первой строке записано количество товаров, которые необходимо купить N ($1 \leqslant N \leqslant 20000$). Во второй строке записаны через пробел N целых чисел p_i - цены товаров ($1 \leqslant p_i \leqslant 20000$).

Сумма N по всем тестовым случаям не превосходит 200000.

Формат выходных данных

Для каждого тестового случая выведите в отдельной строке единственное число — максимальную скидку, с которой можно купить товары из этого тестового случая.

Примеры

стандартный ввод	стандартный вывод
1 6 400 100 200 350 300 250	400