Задача 1. Обработка сигналов

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Группа космических исследований много лет занимается поисками внеземного разума с целью установить с этим разумом контакт. Для того, чтобы какой-либо разум обнаружить, группа предпринимала такие меры, как наблюдение за космическими телами в телескопы, прослушивание радиоэфира на различных частотах, передача всевозможных сообщений по радио, отправка в космос беспилотных летательных аппаратов, несущих на себе множество датчиков. Впрочем, каждый раз поиски не давали результатов, которые могли бы гарантированно показать присутствие внеземного разума. Однако некоторые сигналы, принятые исследовательской аппаратурой, показались ряду членов группы подозрительными, и решено было подвергнуть их тщательному анализу. Эти сигналы аппаратура записала в виде последовательностей нулей и единиц. Возникла загвоздка: ведь если сигналы были посланы нам внеземной цивилизацией, то неизвестно, какой способ кодирования в них применяется, и как именно нужно эти бинарные последовательности интерпретировать.

Появилась мысль начать с простого и в первую очередь перекодировать сигналы из двоичного представления в шестнадцатеричное. Результат такой обработки для сигнала, являющегося последовательностью длины L, должен содержать $\lceil \frac{L}{4} \rceil$ шестнадцатеричных цифр. Цифры от нуля до девяти представляются символами '`0`'-'`9`', а цифры от десяти по пятнадцати представляются большими латинскими буквами '`A`', '`B`', '`C`', '`D`', '`E`' и '`F`' соответственно. Будем считать, что младшие разряды находятся справа как у необработанной последовательности, так и у результата обработки.

Формат входных данных

В первой строке содержится целое число: N — количество сигналов, показавшихся подозрительными ($1 \leqslant N \leqslant 100$).

Каждая из следующих N строк содержит один такой сигнал, представленный в виде строки, состоящей из нулей и единиц. Каждая такая строка непуста и имеет длину не более 250.

Формат выходных данных

Для каждого сигнала выведите результат его обработки на отдельной строке.

Пример

input.txt	output.txt
3 100 00001 01110100101011011000000100 0	4 01 3A56C08

Задача 2. Инновационная клавиатура

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Мультиклавиатура — это обычная клавиатура с множеством выходов, каждый из которых можно подключить к одному персональному компьютеру. Создатели устройства утверждают, что использование мультиклавиатуры значительно упрощает жизнь каждого, у кого она есть на рабочем месте.

Недавно в серверную одной очень большой и очень секретной конторы установили мультиклавиатуру. Последняя была подключена к N компьютерам. Таким образом, при нажатии клавиши, символ печатается сразу на всех машинах. Но вот незадача: кабели мультиклавиатуры оказались чрезвычайно хлипкими, и если отсоединить кабель от компьютера, то подключить его вновь не удастся.

Рабочий день уже в самом разгаре, и срочно требуется напечатать на каждом компьютере по заданной строке. При этом разрешено пользоваться только мультиклавиатурой: можно нажимать исключительно клавиши с буквами латинского алфавита и отключать от компьютеров провода-соединители с мультиклавиатурой. В конце набора на каждом компьютере должна быть набрана в точности заданная строка.

Определите, за какое минимальное количество нажатий клавиш мультиклавиатуры возможно напечатать все заданные строки на компьютерах.

Формат входных данных

В первой строке входного файла задано целое число T — количество тестов ($1 \leqslant T \leqslant 100$). Далее следует T блоков.

В первой строке блока задано целое число N — количество компьютеров, к которым подключена мультиклавиатура ($1 \leqslant N \leqslant 10^5$). В каждой i-ой из следующих строк задана непустая строка s_i, которую требуется напечатать на i-ом компьютере.

Суммарная длина всех строк по всем тестам не превышает 10^5. Строки могут содержать только строчные буквы латинского алфавита.

Формат выходных данных

В выходной файл выведите T строк, в i-й строке — ответ на i-й тестовый пример. Если напечатать заданные строки невозможно, ответом должно быть слово `Impossible`. В противном случае требуется вывести одно целое число — минимальное количество нажатий клавиш мультиклавиатуры, с помощью которых можно напечатать заданные строки.

Пример

input.txt	output.txt
2 3 aba abacaba abca 2 hello hell	Impossible 5

Задача 3. Ракета

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	5 секунд
	10 секунд (для Java)
Ограничение по памяти:	512 мегабайт

Дима играет в новую игру: Kerbal Space Program. В игре нужно строить большие и интересные ракеты и выводить их в космос. Диме нравится ставить рекорды, поэтому он хочет строить настолько лёгкие ракеты, насколько это возможно.

Ракета состоит из последовательности из N ступеней и полезного груза на конце. Полезный груз имеет заданную массу M_p. Каждая ступень — это либо один твердотопливный двигатель, либо набор из нескольких одинаковых твердотопливных двигателей, установленных параллельно. Каждый двигатель характеризуется параметрами: масса M, тяга T и удельный импульс топлива I_{sp}. Если K одинаковых двигателей стоят параллельно в одной ступени, то они работают синхронно, выдавая тягу $K \cdot T$, имея массу $K \cdot M$ и общий удельный импульс топлива I_{sp}. Будем считать, что все двигатели идеальные, то есть масса каждого двигателя пренебрежимо мала по сравнению с массой заложенного в него топлива. Иными словами, в двигателе массы M есть топливо общей массы M и сам двигатель нулевой массы.

Процесс запуска ракеты происходит следующим образом. В любой момент в течение разгона работают двигатели ровно одной ступени: первой ступени среди тех, которые ещё не отброшены. Когда в двигателях ступени выгорает всё топливо, ступень отбрасывают от ракеты и включают двигатели следующей ступени. Сначала «зажигается» первая ступень, то есть начинают работать все её двигатели. Когда топливо в двигателях первой ступени заканчивается, её отбрасывают и зажигают вторую ступень. Когда вырабатывается вся масса топлива во второй ступени, она отбрасывается и включается третья ступень. И так далее до тех пор, пока не будут отброшены все ступени.

Прирост скорости ракеты за время выработки всего топлива в одной ступени определяется по формуле Циолковского:

$$\Delta V = I_{sp} \ln \frac{M_{start}}{M_{end}}$$

Здесь: ΔV — приращение скорости ракеты, I_{sp} — удельный импульс топлива для двигателей работающей ступени, M_{start} — начальная масса всей ракеты в момент зажигания ступени, M_{end} — конечная масса всей ракеты в момент отброса ступени, ln — функция натурального логарифма, то есть логарифма по основанию e. Конечная скорость ракеты равна сумме этих приращений для всех ступеней ракеты. Изначально скорость ракеты нулевая.

От ракеты требуется разогнать полезный груз до скорости не меньше первой космической V_*. Кроме того, отношение тяги работающих двигателей к текущей массе ракеты **не** должно опускаться ниже постоянной $c = 8/5$. Поскольку ступени зажигаются последовательно, достаточно проверить выполнение этого условия в момент зажигания каждой ступени: при работе двигателей их топливо выгорает, тяга при этом остаётся постоянной, а масса постепенно уменьшается. Диму заведомо не интересуют варианты ракеты с начальной массой больше M_{max} — слишком плохо, чтобы даже браться.

В игровом магазине доступно для покупки T типов двигателей. У Димы достаточно денег, чтобы покупать любые двигатели в любом желаемом количестве. Требуется узнать, какова минимально возможная начальная масса ракеты, при которой можно разогнать груз с выполнением всех условий.

Рассмотрим пример трёхступенчатой ракеты. Каждая i-ая ступень имеет параметры $M^{(i)}$, $T^{(i)}$, $I_{sp}^{(i)}$. Тогда общий прирост скорости за всё время разгона равен:

$$\Delta V = I_{sp}^{(1)} \ln \frac{M^{(1)} + M^{(2)} + M^{(3)} + M_p}{M^{(2)} + M^{(3)} + M_p} + I_{sp}^{(2)} \ln \frac{M^{(2)} + M^{(3)} + M_p}{M^{(3)} + M_p} + I_{sp}^{(3)} \ln \frac{M^{(3)} + M_p}{M_p}$$

Не стоит забывать про ограничение снизу на тягу:

$$\frac{T^{(1)}}{M^{(1)} + M^{(2)} + M^{(3)} + M_p}, \ \frac{T^{(2)}}{M^{(2)} + M^{(3)} + M_p}, \ \frac{T^{(3)}}{M^{(3)} + M_p} \ \geqslant \ \frac{8}{5} = c$$

Формат входных данных

В первой строке записано два целых числа: T — количество типов двигателей в магазине ($1 \leqslant T \leqslant 3 \cdot 10^2$) и M_{max} — максимально допустимая масса всей ракеты ($1 \leqslant M_{max} \leqslant 10^5$). Во второй строке записано ещё два целых числа: M_p — масса полезного груза ракеты ($1 \leqslant M_p \leqslant M_{max}$) и V_* — первая космическая скорость ($1 \leqslant V_* \leqslant 10^9$).

В остальных T строках описываются типы двигателей в магазине, по одному в строке. Для каждого двигателя записано три целых числа: M — масса одного двигателя ($1 \leqslant M \leqslant 10^5$), T — тяга одного двигателя ($1 \leqslant T \leqslant 2 \cdot 10^5$) и I_{sp} — удельный импульс топлива ($1 \leqslant I_{sp} \leqslant 10^5$).

Формат выходных данных

Если невозможно достичь первой космической скорости при начальной массе ракеты M_{max} или меньше, выведите одно число -1.

В противном случае выведите в первой строке два целых числа: M_{start} — начальная масса ракеты ($M_p \leqslant M_{start} \leqslant M_{max}$) и N — количество ступеней в ракете ($N \geqslant 1$). В каждой из следующих N строк требуется описать одну ступень ракеты. Ступени описываются в порядке зажигания, от первой до последней. Описание ступени должно состоять из двух целых чисел: j — номер типа двигателя, который стоит в ступени ($1 \leqslant j \leqslant T$) и k — количество двигателей, установленных в ступени параллельно ($k \geqslant 1$). Типы двигателя нумеруются в порядке их описания.

Жюри гарантирует, что во всех тестах при изменении первой космической скорости на тысячную долю процента ответ на задачу (т.е. минимальная начальная масса ракеты) **не** изменяется.

Пример

input.txt	output.txt
2 100000 3 408 5 13 200 8 20 200	24 2 2 2 1 1

Комментарий

Заметим, что в данной задаче игнорируется сила притяжения Земли в формуле приращения скорости. Кроме того, считается, что ракету нужно разгонять строго по прямой, что в реальности не совсем верно.

Основание натурального логарифма: $e \approx 2.718281828459045$.

Задача 4. Поступление в ВУЗ

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В некотором царстве, в некотором государстве для выпускников школ ввели Единый Государственный Экзамен. По результатам экзамена выпускники могут поступать в вузы. Однако из-за того, что каждый выпускник может подавать документы в несколько вузов, до самого конца представители вузов точно не знают, кто в них поступит.

В начале очередного периода поступления имеется N выпускников и M вузов. Будем считать, что в каждом вузе неограниченное количество мест, и поэтому нет ограничения на проходной балл.

Каждый выпускник за период поступления может подать документы в любое количество вузов, и для каждого вуза он может сделать это в произвольный день в течение периода зачисления. Благодаря опросу в социальных сетях, про каждого выпускника точно известно, в какие вузы он подаст документы, и в какой день он это сделает для каждого вуза. Все документы подаются утром, в самом начале дня.

Каждый вуз обязан устроить процедуру зачисления в один из дней в течение периода поступления. Когда происходит процедура зачисления в вуз, все свободные на этот момент выпускники, которые уже подали в него документы, окончательно и бесповоротно засчисляются в этот вуз. Выпускник является свободным, если он **не** был зачислен в какой-либо другой вуз в один из предыдущих дней. Процедура зачисления устраивается после обеда, поэтому выпускника могут зачислить, даже если он подал документы ровно в день процедуры зачисления. Зачисленные выпускники в дальнейшем **не** могут попасть ни в какой другой вуз, а зачисливший их вуз более **не** может никого зачислить.

Если процедуру зачисления одновременно устроили несколько вузов, в которые выпускник подал документы, то возникает неопределённость в том, в какой вуз он будет зачислен. В таком случае выпускник может самостоятельно выбрать, в какой из этих вузов он попадёт. На практике выпускник всегда выбирает наиболее престижный вуз среди нескольких возможных вариантов.

Представители Неизвестного Государственного Университета (НГУ) продумывают стратегию на текущий период поступления. Каждому выпускнику специалисты НГУ присвоили некоторую ценность в баллах. Имеются разведданные о том, когда другие вузы планируют устроить процедуру зачисления. К сожалению, это не точные данные, а лишь вероятностные оценки: для каждого вуза и каждого дня известно, с какой вероятностью данный вуз устроит процедуру зачисления в данный день. Будем считать, что вузы выбирают процедуру зачисления абсолютно независимо друг от друга.

Требуется определить, в какой день нужно устроить процедуру зачисления в НГУ, чтобы сумма ценностей всех зачисленных в него выпускников в среднем была максимальной. Учтите, что НГУ — самый престижный вуз.

Формат входных данных

В первой строке дано четыре целых числа: N — количество выпускников ($1 \leqslant N \leqslant 3\,000$), M — количество вузов ($1 \leqslant M \leqslant 50$), K — количество заявлений на подачу документов ($1 \leqslant K \leqslant 30\,000$), T — длина периода поступления в днях ($1 \leqslant T \leqslant 10\,000$). Выпускники и вузы пронумерованы по порядку, начиная с единицы. Вуз НГУ имеет номер M.

Во второй строке содержится N целых чисел V_i — ценности всех выпускников ($1 \leqslant V_i \leqslant 10^3$).

В каждой l-ой из следующих K строк описывается по одному заявлению на подачу документов. Каждое описание содержит три целых числа: P_l — номер выпускника, который подал документы ($1 \leqslant P_l \leqslant N$), Q_l — номер вуза, в который он подал документы ($1 \leqslant Q_l \leqslant M$), R_l — номер дня, в который это произошло ($1 \leqslant R_l \leqslant T$). Гарантируется, что каждый выпускник в каждый вуз подаёт документы не больше одного раза. Выпускник может подать документы в несколько вузов сразу в один день.

В оставшихся T строках описаны развед. данные о других вузах, по $M-1$ вещественных чисел в каждой строке. В t-ой из этих строк j-ое число P_{tj} определяет, с какой вероятностью j-ый вуз устроит процедуру зачисления в t-ый день. Гарантируется, что сумма заданных вероятностей для каждого вуза в точности равна единице. Вещественные числа $0 \leqslant P_{tj} \leqslant 1$ задаются в виде десятичной дроби с не более чем четырьмя знаками после точки.

Формат выходных данных

В первой строке требуется вывести два числа: X — максимальное среднее значение суммы ценностей зачисленных в НГУ выпускников (вещественное) и t^* — номер дня, в который нужно устроить процедуру зачисления для этого (целое, $1 \leqslant t^* \leqslant T$).

Относительная ошибка вашего ответа X не должна превышать 10^{-8}.

Пример

input.txt	output.txt
4 4 7 6 4 666 5 3 1 1 2 3 1 5 4 4 6 1 4 4 4 2 4 3 4 3 3 2 2 0.2 0 0.01 0.2 0.7 0.17 0.2 0 0.03 0.2 0 0.25 0.1 0.3 0.09 0.1 0 0.45	3.9 4

Пояснение к примеру

Следует заметить, что особо ценный второй выпускник уехал за границу, а в третий вуз никто не хочет поступать.

Задача 5. Папка с нотами

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Хранение нот — одна из повседневных забот рядового музыканта. К счастью, сегодня в любом канцелярском магазине можно легко купить папку со встроенными мультифорами («файлами»). В таких папках ноты можно хранить долго, регулярно играя по ним. А вот перекладывать ноты в папке с мультифорами — это не самое приятное занятие.

Имеется папка с мультифорами, все мультифоры в папке пронумерованы по порядку от 1 до M. Каждая мультифора может быть либо пустой, либо в неё может быть вставлен один бумажный лист с нотами какого-нибудь музыкального произведения. В одной мультифоре **не** может находиться несколько листов, потому что так можно легко их потерять.

Всего в оркестре играют N произведений, которые для удобства пронумерованы числами от 1 до N. Для каждого произведения ноты занимают один или несколько листов. Музыкант хранит все ноты всех произведений в одной папке, причём каждый лист нот каждого произведения присутствует в папке ровно в одном экземпляре. Чтобы проще было разбираться с нотами, на каждом листе с нотами написан номер произведения и номер этого листа в произведении. Например, надпись на листе «7: Passacaglia, 2/3» означает, что это второй лист из трёх произведения номер семь, которое называется «Пассакалия».

Если ноты произведения занимают несколько листов, то музыканту в процессе его игры приходится «переворачивать страницы». Времени на размышления и поиски при этом нет совсем, поэтому любой музыкант все листы с нотами одного произведения непременно хранит в папке подряд (в соседних мультифорах). Например, если ноты произведения «Пассакалия» занимают три листа, и второй лист с нотами Пассакалии лежит в 6-ой мультифоре, то первый лист Пассакалии должен лежать в 5-ой мультифоре, а третий — в 7-ой. Если в папке с нотами это правило нарушено, то на очередном выступлении может случиться непоправимое!

№	Ноты упорядочены	Начальное состояние
(1)	—	—
(2)	1: March 1/2	1: March 1/2
(3)	1: March 2/2	1: March 2/2
(4)	—	—
(5)	2: Passacaglia 1/3	—
(6)	2: Passacaglia 2/3	4: Nocturne 1/1
(7)	2: Passacaglia 3/3	—
(8)	3: Waltz 1/2	2: Passacaglia 1/3
(9)	3: Waltz 2/2	2: Passacaglia 2/3
(10)	—	2: Passacaglia 3/3
(11)	4: Nocturne 1/1	3: Waltz 1/2
(12)	—	3: Waltz 2/2

Чтобы быстро искать ноты определённого произведения в папке, музыканты используют алгоритм бинарного или интерполяционного поиска (правда они обычно не знают таких названий). Естественно, эти подходы можно применить только в том случае, если ноты произведений в папке идут в порядке увеличения номера произведения. Однако иногда приходится раскладывать произведения в другом порядке, например для ответственных выступлений.

Естественно, при этом тоже непременно соблюдается правило, что ноты каждого произведения идут подряд. В данной задаче требуется найти оптимальный вариант перекладывания нот в папке так, чтобы в конце они шли в порядке увеличения номера произведения.

Например, рассмотрим два состояния папки с двенадцатью мультифорами (см. таблицу на предыдущей странице). Справа изображено начальное состояние нот, расставленных не в порядке увеличения номера. А слева те же ноты переставлены так, что они идут в порядке увеличения номера произведения. Возможны и другие варианты расстановки листов, при которых произведения идут в правильном порядке.

Чтобы сильно не возиться с кучей бумаги вне папки, музыкант переставляет листы в папке, выполняя цепочки перекладываний. Цепочка перекладываний (i_1, i_2, i_3, ..., i_k) при $k \geqslant 2$ выполняется следующим образом:

1. Извлекаем лист из мультифоры с номером i_1.
2. Лист вкладываем в мультифору с номером i_2, из которой предварительно вытаскиваем находящийся там лист.
3. Освободившийся лист вставляем в мультифору i_3, из которой предварительно извлекаем имеющийся там лист.

...

k. Освободившийся на предыдущем шаге лист вставляем в пустую на текущий момент мультифору i_k.

Мультифора может входить в цепочку несколько раз.

Каждый раз, когда музыкант извлекает или вставляет лист в мультифору, есть риск того, что край листа разрежет край мультифоры. Поэтому требуется найти способ перестановки нот, при котором количество операций извлечения и вставки минимально возможно.

Например, в приведённом выше примере можно получить упорядоченную папку из начальной при помощи последовательности цепочек:

- (6 9 6)
- (8 5)
- (10 7)
- (9 11 8)
- (12 9)

Общее количество операций вставки в этом случае равно 7, хотя можно достигнуть того же результата за 6 операций вставки.

Формат входных данных

В первой строке дано два целых числа: M — количество мультифор в папке ($1 \leqslant M \leqslant 2 \cdot 10^5$), N — количество произведений ($1 \leqslant N \leqslant 10^5$).

В остальных M строках описывается начальное состояние папки, по одной мультифоре в строке в порядке их нумерации. В каждой j-ой из этих строк указано два целых числа, описывающих лист с нотами, лежащий в j-ой мультифоре: C_j — номер произведения ($1 \leqslant C_j \leqslant N$), и P_j — номер листа в нотах произведения ($1 \leqslant P_j \leqslant M$). Если мультифора пустая, то оба числа равны -1.

Гарантируется, что в папке присутствуют ноты всех произведений от 1 до N, что для каждого j-ого произведения есть все листы с нотами с номерами от 1 до L_j, что каждый лист присутствует в папке в одном экземпляре. Кроме того, для каждого произведения все листы с его нотами идут в папке подряд.

Формат выходных данных

В первой строке требуется вывести два целых числа: A — минимально возможное количество операций вставки листа в мультифору и Q — количество цепочек в оптимальном

решении.

В каждой из следующих Q строк должно быть описано по одной цепочке перекладываний в порядке их выполнения. Описание цепочки перекладываний начинается с целого числа k — её длины ($k \geqslant 2$), после которого приводится k целых чисел $i_1, i_2, \ldots, i_k$ — номера мультифор в цепочке в порядке выполнения операций над ними ($1 \leqslant i_1, i_2, \ldots, i_k \leqslant M$).

Пример

input.txt	output.txt
12 4 -1 -1 1 1 1 2 -1 -1 -1 -1 4 1 -1 -1 2 1 2 2 2 3 3 1 3 2	6 4 4 6 11 8 5 2 10 7 2 9 6 2 12 9

Задача 6. Деление бутерброда

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Перед детьми Поволжья стоит непростая задача. Незадолго до инспекции великого комбинатора Альхен выдал им бутерброд, намазанный маслом, и пошел затем угощать пройдоху чем бог послал. Братовья же очень хотят разделить бутерброд одним разрезом, поделив при этом ровно пополам и хлеб и масло. Да вот беда — гимназиев они не кончали, и с задачей этой ну никак не справятся. Надо бы им помочь.

Бутерброд сей представляет собой вертикальный цилиндр $x^2+y^2 \leqslant 1$, рассеченный тремя плоскостями $z = 0$, $z = a_1x+b_1y+c_1$ и $z = a_2x+b_2y+c_2$. Между первой и второй плоскостями — слой хлеба, между второй и третьей — слой масла.

Требуется вычислить три вещественных коэффициента уравнения вертикального разреза: $ax + by = c$.

Формат входных данных

Во входном файле записано шесть вещественных чисел $a_1, b_1, c_1, a_2, b_2, c_2$, задающих коэффициенты уравнений плоскостей, ограничивающих слой хлеба и слой масла. Все числа по модулю не превышают 100 и заданы с не более чем шестью знаками после десятичной точки.

Гарантируется, что всюду внутри единичного цилиндра бутерброда вторая плоскость лежит строго выше первой, а третья — строго выше второй, т.е.: $0 < a_1x+b_1y+c_1 < a_2x+b_2y+c_2$ для всех $x^2 + y^2 \leqslant 1$.

Формат выходных данных

В выходной файл необходимо вывести три вещественных числа a, b и c, задающих искомое положение разреза — вертикальной плоскости $ax+by = c$, делящей и хлеб, и масло пополам. При этом должно выполняться соотношение $\frac{1}{10} \leqslant a^2 + b^2 \leqslant 10$.

Выведенная плоскость будет засчитана, если отношение объемов двух частей, на которые она делит каждую субстанцию бутерброда (и хлеб, и масло), отличается от единицы не более, чем на 10^{-8}.

Если такое деление невозможно, то нужно вывести лишь одно число -1.

Примеры

input.txt
1 0 3 -1 0 20
output.txt
0.000000000000 1.000000000000 0.000000000000

input.txt
10 10 50 -10 2 100
output.txt
-0.514495755428 0.857492925713 0.022850559544

Задача 7. Поиск коллизий

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	10 секунд
Ограничение по памяти:	256 мегабайт

Степан участвует в разработке игрового мода на движке idTech4. В игре требуется искать коллизии мгновенно перемещающихся объектов (вроде пуль), а методы поиска таких коллизий в движке работают медленно. Вам предлагается помочь Степану реализовать эффективный алгоритм поиска коллизий.

Имеется G геометрических моделей, пронумерованных числами от 1 до G. Каждая модель состоит из набора треугольников в трёхмерном пространстве. Гарантируется, что все эти треугольники невырождены. Других гарантий, например гарантий топологической корректности, замкнутости, отсутствия самопересечений, нет. Размер всех геометрических моделей примерно единичный, то есть если рассмотреть для любой геометрической модели минимальный прямоугольный параллелепипед, параллельный осям координат и содержащий модель, то самая длинная его сторона будет иметь длину $\frac{1}{2} \leqslant L \leqslant 1$.

В трёхмерном пространстве выделена игровая область с координатами $0 \leqslant x \leqslant M$, $0 \leqslant y \leqslant N$, $0 \leqslant z \leqslant K$, в которой расположено $S = (M \cdot N \cdot K)$ игровых объектов, пронумерованных числами от 1 до S. Каждый объект задаётся номером его геометрической модели и координатами его центра. Геометрическая модель объекта сдвинута таким образом, что начало координат в системе модели попадает в центр объекта. Иными словами, объект состоит в точности из тех треугольников, которые присутствуют в его геометрической модели, но все вершины этих треугольников сдвинуты на вектор v, направленный из начала координат в центр объекта. Игровые объекты могут пересекаться.

Требуется обработать Q запросов на поиск коллизии. В каждом запросе приводится луч, заданный координатами начальной точки и единичным вектором направления. Нужно пересечь луч со всеми треугольниками всех игровых объектов, и выдать среди всех полученных точек пересечения ближайшую к началу луча.

Все тесты жюри, за исключением теста из условия, составлены следующим образом:

1. Используется один и тот же набор из 10 геометрических моделей, изображённых на следующей странице. Все модели соответствуют реальным игровыми объектам. Их можно скачать во время тура.
2. Центры объектов сгенерированы псевдослучайным образом с использованием равномерного распределения по игровой области.
3. Начала лучей в запросах сгенерированы псевдослучайным образом с использованием равномерного распределения по игровой области.
4. Векторы направления лучей в запросах сгенерированы псевдослучайным образом с использованием равномерного распределения по единичной сфере.

Формат входных данных

В первой строке дано одно целое число G — количество заданных геометрических моделей ($1 \leqslant G \leqslant 10$). Далее описываются G геометрических моделей в порядке их нумерации.

Описание каждой модели начинается с двух целых чисел: V — количество заданных точек ($3 \leqslant V \leqslant 700$), T — количество треугольников в модели ($1 \leqslant T \leqslant 1\,000$). Далее следует V строк, в которых заданы координаты V точек. Точки нумеруются в порядке описания, начиная с единицы. В i-ой из этих строк даны координаты i-ой точки как вещественные числа P_x^i, P_y^i и P_z^i ($0 \leqslant P_x^i, P_y^i, P_z^i \leqslant 1$). Затем следует T строк, в которых описываются треугольники

модели. Каждая j-ая из этих строк содержит три целых числа a_j, b_j и c_j, задающие номера трёх точек, являющихся вершинами j-ого треугольника ($1 \leqslant a_j < b_j < c_j \leqslant V$).

В следующей строке входных данных задаётся три целых числа M, N и K, определяющие размеры игровой области, а также количество игровых объектов ($1 \leqslant M, N, K \leqslant 50$). В следующих $(M \cdot N \cdot K)$ строках описываются игровые объекты в порядке нумерации, по одному в строке. Каждая из этих строк содержит целое число q_l — номер геометрической модели объекта ($1 \leqslant q_l \leqslant G$), и три вещественных числа C_x^l, C_y^l, C_z^l — координаты центра объекта ($0 \leqslant C_x^l \leqslant M$, $0 \leqslant C_y^l \leqslant N$, $0 \leqslant C_z^l \leqslant K$).

В следующей строке записано целое число Q — количество запросов, которые нужно обработать ($1 \leqslant Q \leqslant 10\,000$). Каждая из следующих Q строк описывает один запрос и содержит шесть вещественных чисел $O_x^q, O_y^q, O_z^q, D_x^q, D_y^q, D_z^q$. Первые три числа — координаты начала луча ($0 \leqslant O_x^q \leqslant M$, $0 \leqslant O_y^q \leqslant N$, $0 \leqslant O_z^q \leqslant K$), а остальные три числа — компоненты единичного вектора направления ($-1 \leqslant D_x^q, D_y^q, D_z^q \leqslant 1$, $(D_x^q)^2 + (D_y^q)^2 + (D_z^q)^2 = 1$).

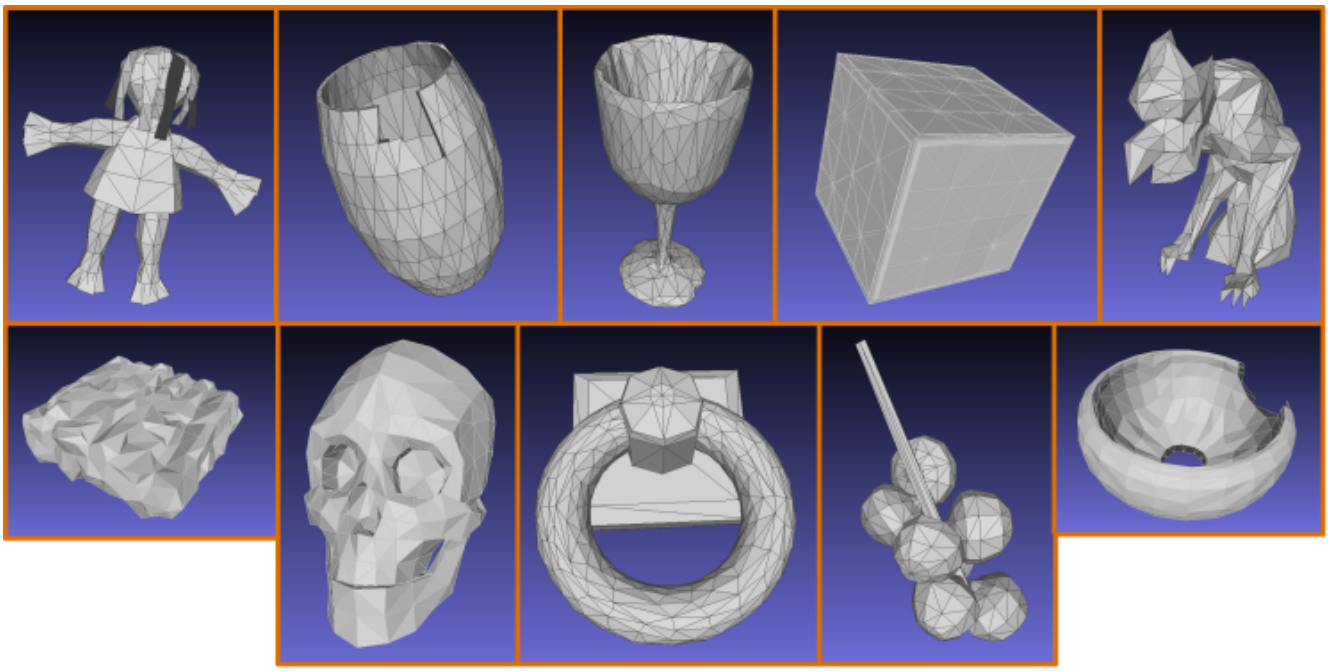
Все вещественные числа записаны с не более чем 8-ю знаками после десятичной точки.

Формат выходных данных

Требуется вывести Q строк с ответами на запросы в порядке их описания во входных данных. Ответ на каждый запрос — это координаты искомой точки пересечения I_x^q, I_y^q и I_z^q, заданные как вещественные числа. Точка пересечения может находиться вне игровой области. Если луч не пересекает ни одного объекта, выведите три числа -1.

Ответ на запрос считается правильным, если абсолютная или относительная погрешность координат точки пересечения не превышает 10^{-5}. Полный ответ на тесте засчитывается, если количество неправильных ответов на запросы не превышает одной тысячной доли от количества запросов (с округлением вверх).

Ниже изображены используемые в тестах жюри модели. Можно скачать архив с этими моделями на вкладке «Новости». В файле `all.txt` описаны все модели точно согласно формату входных данных (до чисел M, N, K).



Пример

input.txt

```
2
8 12
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1
1 2 3
2 3 4
5 6 7
6 7 8
1 2 5
2 5 6
3 4 7
4 7 8
1 3 5
3 5 7
2 4 6
4 6 8
3 1
0.5 0 0
0 0.5 0
0 0 0.5
1 2 3
3 1 1
2 0 0 0.5
1 0.5 0.5 0.00
2 1 0 0.5
5
0 0 0.5 0.66666667 0.33333333 0.66666667
0 0.1 0.6 1 0 0
1 0.1 0.6 1 0 0
1.2 1 0.7 0 -1 0
0 0 0 -0.48 -0.6 -0.64
```

output.txt

```
0.2 0.1 0.70
0.3 0.1 0.60
1.3 0.1 0.60
1.2 0.5 0.70
-1 -1 -1
```

Задача 8. Поиск стульев

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Спасаясь от землетрясения, концессионеры в панике побросали стулья на обширной площади. Теперь стоит задача найти все нажитое непосильным трудом.

Первое, что знают Остап и Киса — это количество стульев n. Кроме того, известно, что координаты всех стульев целые, и по модулю не превосходят 10^6. Никакие два стула **не** находятся в одной точке.

Все, что могут делать незадачливые кладоискатели — это по произвольно заданным координатам (x_q, y_q) узнавать количество стульев, для координат которых (x_i, y_i) выполняются условия $x_i \leqslant x_q$, $y_i \leqslant y_q$.

Протокол взаимодействия

Это интерактивная задача, и в ней вам предстоит работать не с файловым вводом-выводом, а со специальной программой — интерактором. Взаимодействие с ней осуществляется через стандартные потоки ввода-вывода.

Сначала ваша программа считывает одно целое число n — количество стульев ($1 \leqslant n \leqslant 100$). Затем программа может выводить запросы на количество стульев в формате (без кавычек):

- "? x_q y_q", где x_q и y_q — целые числа, по модулю не превосходящие 10^7.

В ответ на запрос на поток ввода приходит одно целое число k — количество стульев, для координат которых (x_i, y_i) выполняются условия $x_i \leqslant x_q$, $y_i \leqslant y_q$.

В конце работы ваша программа должна вывести ответ для задачи в следующем формате (без кавычек):

- "! x_1 y_1 ... x_n y_n", где x_i и y_i — целые числа, задающие координаты i-ого стула.

Порядок стульев при выводе ответа значения **не** имеет.

Если координаты стульев в ответе неправильные, то решение получит вердикт Wrong Answer. Количество запросов на подсчёт стульев не должно превышать $50n$, иначе решение получит вердикт Wrong Answer.

Убедитесь, что вы выводите символ перевода строки и очищаете буфер потока вывода (команда `flush` языка) после каждого выведенного запроса и после ответа на задачу. Иначе решение может получить вердикт Timeout.

Пример

stdin	stdout
2	? 1 1
2	? 1 0
1	? 0 1
1	? 0 0
1	? -1 0
0	? 0 -1
0	! 0 0 1 1

Задача 9. Vim

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	6 секунд
	12 секунд (для Java)
Ограничение по памяти:	256 мегабайт

Описанные в данной задаче команды могут отличаться от команд настоящего редактора Vim. Пожалуйста, внимательно читайте условие.

В редакторе Vim текст отображается построчно, при этом строки текста могут быть разной длины. Текст состоит из строчных букв латинского алфавита и пробелов.

Для навигации по тексту используется курсор. В редакторе Vim курсор находится **не** между символами, а на символе. Его положение в тексте задаётся парой координат: номером строки и позицией в строке. Положение курсора всегда соответствует реальному символу в тексте: номер строки не может быть меньше 1 или больше количества строк в тексте, а позиция не может быть меньше 1 или больше длины текущей строки.

Положение в тексте A считается расположенным раньше положения B, если:

- номер строки A меньше номера строки B;
- оба положения имеют одинаковый номер строки, но позиция A меньше позиции B.

Соответственно положение A считается расположенным позже положения B, если B расположено раньше A.

Словом в тексте называется последовательность букв латинского алфавита. Слова разделяются одним или несколькими пробелами и переводами строки. Началом слова называется первый символ слова, а концом — последний символ слова.

Vim имеет несколько различных режимов работы, один из которых — режим команд. В режиме команд можно использовать следующие клавиши для навигации по тексту:

- Клавиша '`l`' сдвигает курсор на одну позицию вправо. Если курсор стоял на последнем символе строки, то он никуда не перемещается.
- Клавиша '`h`' сдвигает курсор на одну позицию влево. Если курсор стоял на первом символе строки, то он никуда не перемещается.
- Клавиша '`w`' перемещает курсор в начало следующего слова в тексте: наименьшее положение, соответствующее началу какого-либо слова, и которое идёт позже текущего положения курсора. Если такого начала слова нет, то курсор никуда не перемещается.
- Клавиша '`b`' перемещает курсор в начало текущего слова: наибольшее положение, соответствующее началу какого-либо слова и которое идёт раньше текущего положения курсора. Если такого начала слова нет, то курсор никуда не перемещается.
- Клавиша '`$`' перемещает курсор на последнюю позицию текущей строки.
- Клавиша '`0`' (символ нуля) перемещает курсор на первую позицию текущей строки.
- Клавиша '`j`' перемещает курсор на строчку вниз. Если строка последняя, то курсор не перемещается.
- Клавиша '`k`' перемещает курсор на строчку вверх. Если строка первая, то курсор не перемещается.

При перемещении вверх и вниз курсор запоминает, с какой позиции было начато перемещение между строками — назовём эту позицию стартовой. Если курсор перемещается на строку, длина которой больше или равна номеру стартовой позиции, то в новой строке положение курсора будет иметь стартовую позицию. Если курсор перемещается на строку, длина которой меньше стартовой позиции, то курсор перемещается на последнюю позицию новой строки. При дальнейшем нажатии на клавиши '`k`' или '`j`' курсор так же выбирает позицию

в новой строке исходя **не** из текущей позиции, а из стартовой позиции, с которой началось перемещение между строками. Память о стартовой позиции стирается после нажатия на клавишу, отличную от '`k`' или '`j`'.

Рассмотрим работу клавиши '`k`' на примере. Пусть есть три строки длиной 16, 6 и 11, и курсор находится в третьей строке в позиции 9. Если первый раз нажать на «вверх», то курсор переместится из положения (3, 9) в положение (2, 6). Если второй раз нажать на «вверх», то курсор переместится из положения (2, 6) в положение (1, 9), т.к. движение между строками началось с позиции 9. Если же между первым и вторым нажатием сделать перемещение курсора влево, то перещения курсора будут: $(3,9) \to (2,6) \to (2,5) \to (1,5)$.

Вы — эффективный программист, потому в качестве основного редактора выбрали Vim. Поскольку вы эффективны, то хотите все действия выполнять за минимальный объём приложенных усилий. Перед вами, как перед программистом, часто стоит задача переместить курсор из одного положения в другое и, конечно, вы хотите это сделать за минимальное количество нажатий на клавиши. Помогите себе: напишите программу, которая по заданному тексту и заданным начальному и конечному положениям курсора найдёт кратчайшую последовательность нажатий на клавиши, перемещающую курсор из начального положения в конечное.

Формат входных данных

В первой строке содержится единственное целое число N ($N \geqslant 1$) — количество строк в тексте.

Следующие N строк содержат текст. Для удобства все символы пробела в тексте заменены на символ точки (ASCII 46). Текст состоит из строчных букв латинского алфавита и пробелов (заменённых на точки). Каждая строка текста содержит как минимум один символ, однако в ней могут отсутствовать буквы. Длина текста (суммарная длина всех строк) не превосходит 10^5.

Следующая строка содержит число Q ($1 \leqslant Q \leqslant 10^3$) — количество запросов на перемещение курсора.

Каждый запрос описан на отдельной строке. Запрос состоит из 4 чисел, описывающих начальное и конечное положение курсора: row_s, col_s, row_f и col_f — соответственно номер строки и позиция в этой строке начального положения курсора и номер строки и позиция в этой строке конечного положения курсора. Нумерация строк и позиций начинается с 1. Гарантируется, что данные координаты корректны и соответствуют реальным позициям в тексте. Начальное и конечное положения курсора не совпадают.

Произведение длины текста на количество запросов не превышает 10^7.

Формат выходных данных

Ответом для каждого запроса является кратчайшая последовательность нажатий клавиш, перемещающая курсор из начального положения в конечное. Для каждого запроса выведите ответ на него на отдельной строке. Если существует несколько кратчайших последовательностей, выведите любую из них.

Пример

input.txt	output.txt
3 abc.def q qewrewqr 4 3 7 3 5 1 2 1 5 1 2 1 6 1 1 3 5	hh w $h wjj

Задача 10. Ускорение

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Виктор Михайлович Полесов убегает от разъяренной толпы на своем мотоцикле. Для спасения необходимо проехать расстояние в L километров как можно быстрее.

Двигателями этого чудо-агрегата являются огнетушители, которые включаются строго по одному в произвольном порядке. Огнетушитель работает в течение t микросекунд и все это время сообщает мотоциклу ускорение a метров на секунду в квадрате. Если же ни один огнетушитель не работает, то мотоцикл двигается с постоянной скоростью.

Изначально скорость мотоцикла равна нулю.

Формат входных данных

В первой строке входного файла записано одно целое число M — количество тестов ($1 \leqslant M \leqslant 10^4$).

Далее следует M блоков. В каждом из них в первой строке записано два целых числа N и L — соответственно, количество огнетушителей и длина участка пути в километрах ($1 \leqslant N \leqslant 10^5, 1 \leqslant L \leqslant 10^9$). Затем идут N строк с описаниями огнетушителей. В каждой из них по два целых числа a_i и t_i — соответственно, ускорение в метрах на секунду в квадрате, которое сообщает i-ый огнетушитель, и время его работы в микросекундах ($1 \leqslant a_i \leqslant 10^5, 1 \leqslant t_i \leqslant 10^5$).

Суммарное количество огнетушителей по всем тестам не превышает 10^5.

Формат выходных данных

В выходной файл необходимо вывести M вещественных чисел, по одному числу в строке. Каждое k-ое число — ответ на k-ый тест: минимальное время в секундах, за которое можно преодолеть участок заданной длины.

Ответы требуется выводить с абсолютной или относительной погрешностью, не превышающей 10^{-8}.

Пример

input.txt	output.txt
1 1 1 100000 100000	0.15

Пояснение к примеру

Слесарь-интеллигент сразу включает единственный огнетушитель, и за одну десятую долю секунды пролетает половину пути, разогнавшись до поистине космической скорости десять километров в секунду. Сохраняя эту скорость, он пролетает вторую половину пути за одну двадцатую секунды.

Задача 11. Остров невезения

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

На острове Невезения живет племя дикарей. У каждого обитателя острова есть круг *знакомых* ему обитателей. При этом каждый знакомый обитатель может быть либо *другом*, либо *врагом*. Дикари относятся друг к другу симметрично: если обитатель A знаком с обитателем B, то B знаком с A; если A — друг B, то B — друг A; если A — враг B, то B — враг A.

Когда два дикаря A и B встречаются, они должны понять, как они относятся друг к другу. Если встретившиеся дикари знакомы, то они сразу понимают по типу этого знакомства, друзья они друг другу или враги. Иначе выполняется следующий ритуал приветствия.

Сначала определяется, кто из дикарей выше по росту. Будем считать, что это дикарь A. Далее дикарь A начинает перечислять всех своих знакомых в любом порядке, а дикарь B слушает до тех пор, пока не услышит среди них своего знакомого. Как только дикарь A называет общего знакомого C обоих дикарей, перечисление останавливается и отношения определяются по чётности связей. Здесь действуют первобытные принципы вроде «враг моего врага — мой друг»: дикари A и B становятся друзьями, если C — их общий друг или их общий враг. В противном случае они становятся врагами.

Если дикарь A перечислил всех своих знакомых, а дикарь B никого среди них не узнал, тогда дикари в шоке расходятся, как будто никогда не встречались, так и не поняв, кем они друг другу являются.

Знаменитый антрополог Геннадий Козодоев приехал на остров изучать обычаи аборигенов. Он опросил всех жителей и составил карту отношений между ними. Самого высокого дикаря на острове зовут Уэф, и он работает у антрополога переводчиком. Уэф хочет жить дружно со своими соплеменниками, и просит у Геннадия помощи. Геннадий откликнулся на его просьбу. Он хочет составить такой порядок перечисления знакомых Уэфа для ритуала приветствия, чтобы вероятность того, что случайно встреченный Уэфом дикарь станет ему другом, была максимальной.

Формат входных данных

В первой строке входного файла дано два целых числа: N — количество дикарей на острове ($2 \leqslant N \leqslant 42$), и M — количество знакомств на острове ($1 \leqslant M \leqslant \frac{1}{2}N(N-1)$).

В каждой из следующих M строк описано одно знакомство тремя целыми числами u, v и t, где u и v — номера двух знакомых дикарей ($1 \leqslant u < v \leqslant N$), а t — тип знакомства: $t = 0$ если они друзья, и $t = 1$ если они враги. Все дикари пронумерованы по порядку от 1 до N, причём Уэф имеет номер 1.

Гарантируется, что для каждой пары дикарей u и v в файле описано не более одного знакомства между ними.

Формат выходных данных

В первую строку требуется вывести максимально возможную вероятность того, что если Уэф встретит случайного другого дикаря на острове, то они станут друзьями. Заметим, что Уэф может встретить любого из $N-1$ остальных дикарей с равной вероятностью. Вероятность нужно выводить как вещественное число между 0 и 1, его абсолютная или относительная погрешность не должна превышать 10^{-8}.

Во вторую строку нужно вывести порядок перечисления знакомых Уэфа, при котором получается максимальная вероятность дружбы.

Пример

input.txt	output.txt
7 8 1 2 0 1 3 0 2 4 0 4 5 0 3 4 1 2 5 1 5 7 1 1 7 1	0.666666666666667 7 2 3

Пояснение к примеру

В силу знакомства, встреча дикарей 2 и 3 автоматически заканчивается дружбой, а дикаря 7 — враждой. Встреча с дикарём 6 ни к чему не приводит, потому что он никого не знает. В результате ритуала приветствия дружбой заканчивается также встреча с незнакомыми дикарями 4 и 5: дикарь 5 становится другом после называния дикаря 7 по правилу «враг моего врага — мой друг», в дикарь 4 — после называния дикаря 2 согласно правилу «друг моего друга — мой друг».

Задача 12. UTF-8

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Хакер Вася исследует файлы сохранения игры, которую он пытается взломать. Файлы могут содержать произвольные бинарные данные, но Вася предполагает, что в них встречаются подстроки символов UTF-8.

Кодировка UTF-8 используется для представления произвольной последовательности 31-битных значений (символов) в виде последовательности байтов. Обычно в виде UTF-8 кодируются только символы Unicode, но Вася уверен в том, что в игре используется весь диапазон значений от 0 до $2^{31} - 1$, поскольку все названия в игре записываются на процедурно генерируемых языках. Поэтому Вася сомневается, что сможет использовать для декодирования данных функции из стандартной библиотеки своего языка программирования.

UTF-8 — префиксная самосинхронизующаяся кодировка. Чтобы закодировать последовательность значений в UTF-8, нужно представить каждое значение в виде последовательности от одного до шести байтов. Далее нужно записать все полученные последовательности байтов подряд друг за другом в порядке записи значений в исходной последовательности. Каждое отдельное значение кодируется согласно таблице:

длина	шаблон	битов
1	`0xxxxxxx`	7
2	`110xxxxx 10xxxxxx`	11
3	`1110xxxx 10xxxxxx 10xxxxxx`	16
4	`11110xxx 10xxxxxx 10xxxxxx 10xxxxxx`	21
5	`111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx`	26
6	`1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx`	31

Здесь в первом столбце записано количество байтов в представлении, во втором столбце — шаблон, а в третьем — количество букв ‘x’ в шаблоне. В шаблоне записаны в двоичном виде байты, которые получаются при его использовании для представления значения: цифры ‘1’ и ‘0’ обозначают фиксированные значения битов, а буква ‘x’ обозначает бит, состояние которого определяется по кодируемому значению.

Чтобы закодировать значение V в виде последовательности из k байтов, нужно:

1. Записать число V в двоичной системе исчисления.
2. Найти в таблице k-ую строку и зафиксировать её для дальнейших шагов.
3. Если требуется, дополнить V ведущими нулями так, чтобы количество битов в V совпало с числом в последнем столбце таблицы.
4. В буквы ‘x’ шаблона вставить биты числа V. Биты вставляются слева направо, в порядке от старшего бита к младшему.
5. В результате вместо шаблона получилось k байтов, по восемь бит в каждом из них — результат кодирования V.

Заметим, что если число V в двоичном виде имеет больше битов, чем букв ‘x’ в шаблоне длины k, то закодировать его в k байтов нельзя. С другой стороны, для конкретного значения V нередко можно выбрать количество байтов k несколькими способами, и все эти варианты возможны.

Например, буква ‘с’ русского алфавита представляется значением 1089 в Unicode, что в двоичном виде выглядит как 10001000001. В UTF-8 это значение можно записать пятью способами:

- 11010001 10000001
- 11100000 10010001 10000001
- 11110000 10000000 10010001 10000001
- 11111000 10000000 10000000 10010001 10000001
- 11111100 10000000 10000000 10000000 10010001 10000001

У Васи имеется файл сохранения, являющийся последовательностью байтов. Он хочет найти и декодировать все участки файла, которые закодированы при помощи UTF-8. Помогите Васе и напишите программу для этой цели.

Требуется найти в последовательности все максимальные по включению наборы подряд идущих байтов, которые являются закодированными в UTF-8 последовательностями значений. Следует заметить, что никакие два таких набора не могут перекрываться, то есть у них не может быть общих байтов. Для каждого набора нужно вывести значения, которые в нём закодированы. Наборы, в которых закодировано меньше трёх значений, выводить **не** нужно.

Формат входных данных

Во входных данных приводится содержимое файла сохранения игры: непустая последовательность байтов, разделённых пробелами и/или переводами строк. Значение каждого байта задаётся шестнадцатиричным числом, состоящим из двух цифр. Каждая цифра может быть от 0 до 15, причём цифры 10, 11, 12, 13, 14 и 15 кодируются заглавными буквами ‘A’, ‘B’, ‘C’, ‘D’, ‘E’ и ‘F’ соответственно.

Количество байтов лежит в пределах от 1 до $2 \cdot 10^5$ включительно.

Формат выходных данных

Для каждого максимального декодируемого участка файла сохранения с тремя или более значениями, нужно вывести закодированные в нём значения в отдельной строке. Каждое значение выводится в виде шестнадцатиричного числа без лидирущих нулей, значения разделяются пробелом.

Примеры

input.txt	output.txt
C1 B3 E0 81 B3 F0 80 81 B3 F8 80 80 81 B3 FC 80 80 80 81 B3 80 C1 B3 E0 81 B3 F0 80 81 B3 F8 80 80 81 B3 FC 80 80 80 81 B3 E0 AF B5	73 73 73 73 73 73 73 73 73 73 BF5
FF 00 FF D1 81 F0 80 91 81 00 D1 81 FF F0 80 91 81 FF FF 00 D1 81 F0 80 91 81	441 441 0 441 0 441 441

Комментарий

В данной задаче **не** требуется, чтобы любая последовательность значений в кодировке UTF-8 непременно представлялась минимальным количеством байтов, как того требует настоящая кодировка UTF-8.