

Задача А. Сталкер

Имя входного файла: `stalker.in`
Имя выходного файла: `stalker.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

В городе N при невыясненных обстоятельствах территория одного из заводов превратилась в аномальную зону. Все подъезды к территории были перекрыты, а сама она получила название промзоны. В промзоне находятся N зданий, некоторые из них соединены дорогами. По любой дороге можно перемещаться в обоих направлениях.

Начинающий сталкер получил задание добраться до склада в промзоне. Он нашел в электронном архиве несколько карт территории промзоны. Так как карты составлялись разными людьми, то на каждой из них есть информация только о некоторых дорогах промзоны. Одна и та же дорога может присутствовать на нескольких картах.

В пути сталкер может загружать из архива на мобильный телефон по одной карте. При загрузке новой карты предыдущая в памяти телефона не сохраняется. Сталкер может перемещаться лишь по дорогам, отмеченным на карте, загруженной на данный момент. Каждая загрузка карты стоит 1 рубль. Для минимизации расходов сталкеру нужно выбрать такой маршрут, чтобы как можно меньшее число раз загружать карты. Сталкер может загружать одну и ту же карту несколько раз, при этом придется заплатить за каждую загрузку. Изначально в памяти мобильного телефона нет никакой карты.

Требуется написать программу, которая вычисляет минимальную сумму расходов, необходимую сталкеру, чтобы добраться от входа в промзону до склада.

Формат входного файла

В первой строке входного файла находятся два натуральных числа N и K ($2 \leq N \leq 2000$; $1 \leq K \leq 2000$) — количество зданий промзоны и количество карт соответственно. Вход в промзону находится в здании с номером 1, а склад — в здании с номером N . В последующих строках находится информация об имеющихся картах. Первая строка описания i -ой карты содержит число r_i — количество дорог, обозначенных на i -ой карте. Затем идут r_i строк, содержащие по два натуральных числа a и b ($1 \leq a, b \leq N$; $a \leq b$), означающих наличие на i -ой карте дороги, соединяющей здания a и b . Суммарное количество дорог, обозначенных на всех картах, не превышает 300000 ($r_1 + r_2 + \dots + r_K \leq 300000$).

Формат выходного файла

В выходной файл необходимо вывести одно число — минимальную сумму

расходов сталкера. В случае, если до склада добраться невозможно, выведите число -1 .

Пример

<code>stalker.in</code>	<code>stalker.out</code>
5 3 1 3 4 3 1 2 1 3 2 4 1 4 5	2
5 3 2 3 2 4 5 1 2 1 2 1 3 5 4	-1

Задача В. Файловый менеджер

Имя входного файла: `fur.in`
Имя выходного файла: `fur.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

Петя работает над очень большим проектом. Проект содержит N файлов. В процессе работы Пете часто приходится просматривать и редактировать файлы. Для ускорения работы Петя использует файловый менеджер Fur Manager, который отображает список имен файлов проекта в некотором порядке.

В текущей версии Fur Manager'a для перемещения по списку имен файлов есть следующие возможности:

- 1) можно нажать клавишу вниз, при этом курсор перемещается на следующий файл в списке, для последнего файла следующим считается первый;
- 2) можно нажать клавишу вверх, при этом курсор перемещается на предыдущий

файл в списке, для первого файла предыдущим считается последний;

3) можно нажать клавишу Alt и, удерживая ее, набрать последовательность латинских букв. После этого клавишу Alt следует отпустить, и в этот момент курсор переместится на ближайший файл, имя которого начинается с заданной последовательности символов. Ближайший файл — это файл, на который можно переместиться за наименьшее количество нажатий клавиши вниз. Если заданная последовательность является началом имени текущего файла, или файла, имя которого начинается с этой последовательности, не существует, то курсор останется на месте.

Первая и вторая из описанных возможностей файлового менеджера требуют по одному нажатию клавиши, а третья — одного нажатия (нажатие клавиши Alt) плюс количество нажатий, равное длине набранной последовательности латинских букв. Файлы пронумерованы от 1 до N в порядке их следования. После загрузки File Manager'a курсор находится на первом файле. Петя знает, что ему сначала придется редактировать файл с номером a_1 , затем с номером a_2 и так далее, а последним — файл с номером a_k . В последовательности a_1, a_2, \dots, a_k один и тот же номер может встречаться несколько раз. При каждом перемещении от одного файла к другому Петя хочет нажимать как можно меньше клавиш. Требуется написать программу, которая выдает искомую последовательность нажатий клавиш.

Формат входного файла

В первой строке входного файла записано целое число N ($1 \leq N \leq 1000$) — количество файлов в проекте.

В следующих N строках записаны имена файлов, по одному в каждой строке. Файлы перечислены в том порядке, в котором они отображаются файловым менеджером. Имена состоят только из строчных латинских букв. Длина каждого имени не превосходит 2000 символов. Все имена файлов различны. Далее в следующей строке записано целое число k ($1 \leq k \leq 10$). Последняя строка входного файла содержит k целых чисел a_1, a_2, \dots, a_k ($1 \leq a_i \leq N$) — номера редактируемых файлов. Редактирование файлов выполняется в том порядке, в котором они встречаются в последовательности a_1, a_2, \dots, a_k .

Формат выходного файла

Выходной файл должен содержать описание искомой последовательности нажатий клавиш в виде k блоков информации:

- первый блок информации описывает перемещение от файла с номером 1 к файлу с номером a_1 ;

- второй блок информации описывает перемещение от файла с номером a_1 к файлу с номером a_2 ;

- ...

- k -ый блок информации описывает перемещение от файла с номером a_{k-1} к файлу с номером a_k .

Каждый блок информации выглядит следующим образом.

В первой строке блока записывается число L — наименьшее количество нажатий клавиш, необходимое для перемещения от очередного файла последовательности к следующему. Следующие L строк блока описывают нажимаемые клавиши. Каждая из строк содержит описание одной клавиши:

- если нажимается клавиша вниз, то в строке записывается слово down;

- если нажимается клавиша вверх, то в строке записывается слово up;

- если нажимается клавиша Alt, то в строке записывается слово Alt;

- при нажатии клавиши с латинской буквой выводится соответствующая ей латинская буква.

Если существует несколько оптимальных способов перемещения, то требуется вывести любой из них.

Пример

fur.in	fur.out
6 submit monitor monitorx monyator subversion sub 5 6 3 3 5 2	1 up 3 Alt m down 0 2 down down 2 Alt m
8 abc abv abba auto test auvto ioi olympiad 2 4 6	3 Alt a u 2 down down

Задача С. Внеземные цивилизации у тебя дома

Имя входного файла: `seti.in`
Имя выходного файла: `seti.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

Петя занимается расшифровкой сигналов из космоса. Для анализа сигнала он применяет следующий метод. Сигнал из космоса записывается в виде последовательности символов t . Для поиска сообщений от внеземных цивилизаций Петя использует следующий алгоритм.

Исходно у Пети есть пустая строка p . В процессе расшифровки Петя

попеременно выполняет следующие действия:

- дописать некоторый символ c в конец строки p ;
- выяснить, сколько раз строка p встречается в t в качестве подстроки.

Петя написал программу, но она почему-то работает очень долго. Помогите ему — напишите программу, которая будет работать не больше двух секунд.

Формат входного файла

Первая строка входного файла содержит строку t , состоящую только из символов латинского алфавита, ее длина не превышает 200 000 символов. Вторая строка содержит последовательность Петиних действий. Дописывание символа в конец строки p обозначается этим символом, запрос обозначается вопросительным знаком. Количество действий не превышает 200 000.

Формат выходного файла

Для каждого запроса выведите одно целое число — сколько раз текущая строка p встречается в t в качестве подстроки.

Примеры

seti.in	seti.out
HelloWorld	3
l?o?v?e?	1
	0
	0

Задача D. Метро

Имя входного файла: `metro.in`
Имя выходного файла: `metro.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

Метрополитен состоит из нескольких линий метро. Все станции метро в городе пронумерованы натуральными числами от 1 до N . На каждой линии расположено несколько станций. Если одна и та же станция расположена сразу на нескольких линиях, то она является станцией пересадки и на этой станции можно пересест с любой линии, которая через нее проходит, на любую другую (опять же проходящую через нее).

Напишите программу, которая по данному вам описанию метрополитена определит, с каким минимальным числом пересадок можно добраться со станции A на станцию B . Если данный метрополитен не соединяет все линии в одну систему,

то может так получиться, что со станции А на станцию В добраться невозможно, в этом случае ваша программа должна это определить.

Формат входного файла

Сначала вводится число N — количество станций метро в городе ($2 \leq N \leq 100$). Далее следует число M — количество линий метро ($1 \leq M \leq 20$). Далее идет описание M линий. Описание каждой линии состоит из числа P_i — количество станций на этой линии ($2 \leq P_i \leq 50$) и P_i чисел, задающих номера станций, через которые проходит линия (ни через какую станцию линия не проходит дважды).

Затем вводятся два различных числа: А — номер начальной станции, и В — номер станции, на которую нам нужно попасть. При этом если через станцию А проходит несколько линий, то мы можем спуститься на любую из них. Так же если через станцию В проходит несколько линий, то нам не важно, по какой линии мы приедем.

Формат выходного файла

Выведите минимальное количество пересадок, которое нам понадобится. Если добраться со станции А на станцию В невозможно, программа должна вывести одно число -1 (минус один).

Пример

metro.in	metro.out
5 2 4 1 2 3 4 2 5 3 3 1	0
5 5 2 1 2 2 1 3 2 2 3 2 3 4 2 4 5 1 5	2
10 2 6 1 3 5 7 4 9 6 2 4 6 8 10 7 3 8	1
4 2 2 1 2 2 3 4 1 3	-1

Задача Е. Монополия

Имя входного файла: `monopoly.in`
Имя выходного файла: `monopoly.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

В Тридесятом государстве есть N фирм, занимающихся разработкой программного обеспечения. Однажды известный олигарх Тридесятого государства Иванушка решил монополизировать эту отрасль. Для этого он хочет купить максимальное число программистских фирм Тридесятого государства.

Он разослал предложения всем N компаниям и через некоторое время получил от каждой из них согласие или отказ. Однако он знает, что в бизнесе очень многое

зависит от взаимного доверия партнеров.

В результате небольшого исследования Иванушка установил, между какими компаниями существует взаимное доверие (причем всегда если компания А доверяет компании В, то компания В доверяет компании А).

Теперь, при желании, Иванушка может повторно разослать предложения всем компаниям, включив в письма список компаний, давших согласие участвовать в его проекте. При этом каждая компания, независимо от своего первоначального мнения дает согласие, если в списке есть хотя бы одна компания, которой она доверяет, и отказ в противном случае. Таким образом, некоторые компании, которые изначально не согласились участвовать в проекте, могут теперь дать свое согласие, а некоторые из давших согласие — наоборот отказаться. В результате этого у Иванушки формируется новый список, который он опять может разослать фирмам. Он может сколь угодно долго повторять операцию, каждый раз рассылая текущий список. Иванушка может остановить процесс в любой момент и заключить договор с теми, кто после последней рассылки дал согласие.

Напишите программу, которая определит, какое максимальное число компаний может объединить Иванушка под своим началом.

Будем считать, что Иванушка — честный предприниматель и он никогда не подтасовывает рассылаемые им списки.

Формат входного файла

В первой строке входных данных содержится число N — количество фирм ($1 \leq N \leq 2000$). $N, (1^-, 0^-).$ M ($0 \leq M \leq 200000$) — количество пар компаний, между которыми существует доверие. Далее следуют M пар чисел, задающих номера фирм, между которыми существует взаимное доверие (числа в паре не могут быть одинаковыми). Любая пара компаний упоминается в этом списке не более одного раза.

Формат выходного файла

Выведите одно число — максимальное число фирм, которое сможет купить Иванушка.

Пример

monopoly.in	monopoly.out
7 1 0 0 0 0 0 1 6 1 2 1 3 1 4 4 5 5 6 2 5	4
3 0 0 0 0	0

Задача F. Домой на электричках

Имя входного файла: rail.in
Имя выходного файла: rail.out
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

Одна из команд-участниц олимпиады решила вернуться домой на электричках. При этом ребята хотят попасть домой как можно раньше. К сожалению, не все электрички идут от города, где проводится олимпиада, до станции, на которой живут ребята. И, что еще более обидно, не все электрички, которые идут мимо их станции, останавливаются на ней (равно как вообще, электрички останавливаются далеко не на всех станциях, мимо которых они идут).

Все станции на линии пронумерованы числами от 1 до N . При этом станция номер 1 находится в городе, где проводится олимпиада, и в момент времени 0 ребята приходят на станцию. Станция, на которую нужно попасть ребятам, имеет номер E .

Напишите программу, которая по данному расписанию движения электричек вычисляет минимальное время, когда ребята могут оказаться дома.

Формат входного файла

Во входном файле записаны сначала числа N ($2 \leq N \leq 100$) и E ($2 \leq E \leq N$). Затем записано число M ($0 \leq M \leq 100$), обозначающее число рейсов электричек. Далее идет описание M рейсов электричек. Описание каждого рейса электрички начинается с числа K_i ($2 \leq K_i \leq N$) — количества станций, на которых она

останавливается, а далее следует K_i пар чисел, первое число каждой пары задает номер станции, второе — время, когда электричка останавливается на этой станции (время выражается целым числом из диапазона от 0 до 10^9). Станции внутри одного рейса упорядочены в порядке возрастания времени. В течение одного рейса электричка все время движется в одном направлении — либо от города, либо к городу.

Формат выходного файла

В выходной файл выведите одно число — минимальное время, когда ребята смогут оказаться на своей станции. Если существующими рейсами электричек они добраться не смогут, выведите -1 .

Пример

rail.in	rail.out
5 3 4 2 1 5 2 10 2 2 10 4 15 4 5 0 4 17 3 20 2 35 3 1 2 3 40 4 45	20