**Вариант 1**

На территории некоторого государства с сильно пересеченной, горной местностью идут военные маневры между двумя противоборствующими сторонами - "Синими" и "Зелеными". Особенности ландшафта и сложные климатические условия вынуждают подразделения обеих сторон размещаться только на территории некоторых населенных пунктов. Общее количество населенных пунктов в этом государстве равно N.   
Тактика ведения боевых действий "Синих" рассчитана на нанесение противнику быстрых и внезапных ударов, что возможно лишь в том случае, если в операциях используются моторизованные части, а их передвижение происходит только по дорогам.   
Разнообразие используемой боевой техники приводит к тому, что время перемещения различных боевых частей из одного пункта в другой оказывается различным и определяется величиной Vj - скоростью движения подразделений боевой части, расквартированной в j-ом населенном пункте.   
Используя подавляющий перевес в технике одна из сторон, под кодовым названием "Синие", планирует организовать ночной налет на базы противника, под кодовым названием "Зеленые", и полностью их разгромить. Все боевые подразделения "Синих" приступают к выполнению операции одновременно. Если боевая часть "Синих" врывается в населенный пункт, занятый "Зелеными", то, учитывая фактор внезапности, им удается полностью разгромить эту группировку.   
К сожалению, блестящему проведению этой операции помешало то обстоятельство, что спустя время T после начала операции "Зелеными" был осуществлен радиоперехват сообщения о начавшейся операции. После радиоперехвата группировки "Зеленых" мгновенно рассеиваются в окрестных горах, и остаются невредимыми.   
Выясните какое количество группировок "Зеленых" и в каких населенных пунктах удастся все-таки разгромить "Синим"?   
Предполагается, что в начальный момент времени группировки "Зеленых" и "Синих" не могут находиться в одном и том же населенном пункте. Если сигнал тревоги поступает в тот момент, когда боевая часть "Синих" только врывается в населенный пункт занятый "Зелеными", то используя превосходное знание местности, "Зеленым" все-таки удается скрыться в горах. Подавляющее превосходство в технике и живой силе позволяет боевым частям "Синих" организовать из каждой части любое количество экспедиций для разгрома "Зеленых". Ничто не мешает одной экспедиции за время проведения операции уничтожить несколько группировок.   
Описание формата входных данных   
Исходные данные задачи содержатся в файлах MAP.IN и TROOPS.IN Структура файла MAP.IN описывает карту местности. В первой строке этого файла содержатся два целых числа: N - количество населенных пунктов (0<N<256) и K - количество дорог, соединяющих эти населенные пункты (0<=K<=1000). Дороги нигде не пересекаются. В последующих K строках файла содержится схема дорог. В каждой строке записана пара двух натуральных чисел i и j и одно положительное вещественное число Lij, означающее, что между населенными пунктами i и j существует дорога длиной Lij километров (Lij < 1000).   
Содержимое файла TROOPS.IN отражает размещение боевых частей воюющих сторон. Первая строка файла содержит число MF - количество боевых частей "Синих". Каждая из последующих MF строк содержат по два числа. Первое - целое число j - номер населенного пункта, в котором размещается часть, второе - вещественное неотрицательное число Vj - скорость движения боевых колонн этой части в км/час (Vj < 110). Далее в отдельной строке файла записано число MB - количество боевых группировок "Зеленых", за которым перечислены MB чисел - номера населенных пунктов, в которых эти группировки находятся. И, наконец, в последней строке файла хранится положительное вещественное число T, измеренное в часах (T < 24). Все числа в строках файлов разделены пробелами.   
Описание формата выходных данных   
Результат решения задачи необходимо вывести в текстовый файл VICTORY.OUT. В первую строку файла выводится количество разгромленных группировок, а во вторую - в порядке возрастания номера населенных пунктов, в которых эти группировки базировались.

**Вариант 2**

Внутри лабиринта есть N комнат, в которых установлены 2М модулей телепортации, составляющих М устройств каждое устройство состоит из двух модулей, которые располагаются в разных комнатах, и предназначено для перемещения между парой комнат, в которых установлены эти модули. Перемещение происходит за 0.5 условных единицы времени. В начальный момент времени модули всех устройств переходят в "подготовительный режим". Каждый из модулей имеет некоторый свой целочисленный период времени, в течение которого он находится в "подготовительном режиме". По истечении этого времени модуль мгновенно "срабатывает" после чего опять переходит в "подготовительный режим". Устройством можно воспользоваться только в тот момент, когда одновременно "срабатывают" оба его модуля.   
Археолог сумел проникнуть в лабиринт (комната 1). Обследовав ее, он включил устройства и собрался уходить, но в этот момент проснулся хранитель лабиринта. Теперь археологу необходимо как можно быстрее попасть в комнату N, в которой находится выход из лабиринта. При этом из комнаты в комнату он может попадать только при помощи устройств, так как проснувшийся хранитель закрыл все двери в комнатах лабиринта.   
Необходимо написать программу, которая получает на входе описания расположения устройств и их характеристик (смотри описание формата ввода), а выдает значение оптимального времени и последовательность устройств, которыми надо воспользоваться, что бы попасть из комнаты 1 в комнату N за это время.

Входной файл: input2.txt Выходной файл: output2.txt   
Формат ввода:   
N   
M   
R11 T11 R12 T12   
RM1 TM1 RM2 TM2   
где N (2<=N<=100) - количество комнат   
M (M<=100) - количество устройств   
Ri1 и Ri2 - номера комнат, в которых располагаются модули устройства i Ti1, Ti2 (Ti1,Ti2<=1000) - периоды времени, через которые срабатывают эти модули Все числа - натуральные.   
Пример   
4   
5   
1 5 3 2   
1 1 2 1   
2 5 3 5   
4 4 3 2   
3 5 4 5   
Оптимальное время: 8.5 искомая последовательность: 2 3 4

**Вариант 3**

Имеется N городов, которые пронумерованы от 1 до N (где N - натуральное, 1<N<=100). Некоторые из них соединены двухсторонними дорогами, которые пересекаются только в городах. Имеется два типа дорог - шоссейные и железные. Для каждой дороги известна базовая стоимость проезда по ней.   
Необходимо проехать из города А в город В, уплатив минимальную сумму за проезд. Стоимость проезда зависит от набора проезжаемых дорог и от способа проезда. Так, если вы подъехали к городу С по шоссейной (железной) дороге X->C и хотите ехать дальше по дороге C->Y того же типа, то вы должны уплатить только базовую стоимость проезда по дороге C->Y. Если тип дороги C->Y отличен от типа дороги Х->C, то вы должны уплатить базовую стоимость проезда по дороге C->Y плюс 10% от базовой стоимости проезда по этой дороге (страховой взнос). При выезде из города А страховой взнос платится всегда. Написать программу, которая находит самый дешевый маршрут проезда в виде последовательности городов и вычисляет стоимость проезда по этому маршруту.   
Спецификация входных данных   
Входные данные находятся в текстовом файле с именем TOUR.IN и имеют следующий формат:   
- в первой строке находятся число N   
- во второй строке - число M (количество дорог, натуральное M<=1000);   
- в каждой из следующих M строк находятся 4 числа x, y, t, p, разделенные пробелом, где x и y - номера городов, которые соединяет дорога, t - тип дороги (0 - шоссейная, 1 - железная), p - базовая стоимость проезда по ней (p - вещественное, 0<p<=1000).   
- в последней строке задается номера начального и конечного городов A и B.   
Спецификация выходных данных   
Выходные данные должны быть записаны в текстовый файл с именем TOUR.OUT и иметь следующий формат:   
- в первой строке находится число S - стоимость проезда по самому дешевому маршруту, с точностью 2 знака после запятой;   
- в каждой из последующих строк, кроме последней, находится два числа - номер очередного города в маршруте (начиная с города А) и тип дороги, по которой он выезжает из этого города (0 или 1), разделенные пробелом.   
- в последней строке находится единственное число - номер города B.

----------------------------------------------------------------------------------------------------------------------------------  
**Вариант 4**

Дана система односторонних дорог, определяемая набором пар городов. Каждая такая пара (i,j) указывает, что из города i можно проехать в город j, но это не значит, что можно проехать в обратном направлении.   
Необходимо определить, можно ли проехать из заданного города А в заданный город В таким образом, чтобы посетить город С и не проезжать ни по какой дороге более одного раза.   
Входные данные задаются в файле с именем PATH.IN следующим образом. В первой строке находится натуральное N(N<=50) - количество городов (города нумеруются от 1 до N). Во второй строке находится натуральное M(M<=100) - количество дорог. Далее в каждой строке находится пара номеров городов, которые связывает дорога. В последней (M+3)-й строке находятся номера городов А, В и С.   
Ответом является последовательность городов, начинающаяся городом А и заканчивающаяся городом В, удовлетворяющая условиям задачи, который должен быть записан в файл PATH.OUT в виде последовательности номеров городов по одному номеру в строке. Первая строка файла должна содержать количество городов в последовательности. При отсутствии пути записать в первую строку файла число -1.

**Вариант 5**

Даны декартовы координаты N перекрестков города, которые пронумерованы от 1 до N. На каждом перекрестке имеется светофор. Некоторые из перекрестков соединены дорогами с двухсторонним (правосторонним) движением, которые пересекаются только на перекрестках. Для каждой дороги известно время, которое требуется для проезда по ней от одного перекрестка до другого.   
Необходимо проехать от перекрестка с номером А до перекрестка с номером В за минимальное время.   
Время проезда зависит от набора проезжаемых дорог и от времени ожидания на перекрестках. Так, если вы подъехали от перекрестка X к перекрестку C по дороге Х->C и хотите ехать дальше по дороге C->У, то время ожидания на перекрестке C эависит от того, поворачиваете ли вы налево или нет. Если вы поворачиваете налево, то время ожидания равно D\*К, где D равно количеству дорог, пересекающихся на перекрестке C, а К - некоторая константа. Если вы не поворачиваете налево, то время ожидания равно нулю.   
Написать программу, которая определяет самый быстрый маршрут.   
Спецификация входных данных.   
Входные данные находятся в текстовом файле с именем PER.IN и имеют следующую структуру:   
- в первой строке находится число N (натуральное, <=1000);   
- во второй строке - количество дорог M (натуральное, <=1000);   
- в третьей строке - константа K (натуральное число, <=1000);   
- в каждой из N следующих стpок находится паpа чисел х и у, разделенных пробелом, где x, y - кооpдинаты перекрестка (целые числа, не пpевышающие по модулю 1000);   
- в каждой из M следующих строк находится 3 числа p, r, t, разделенные пробелом, где p и r - номера перекрестков, которые соединяет дорога, а t (натуральное, <=1000) - время проезда по ней;   
- в последней строке находятся номера начального А и конечного В перекрестков.   
Спецификация выходных данных.   
Выходные данные должны быть записаны в текстовый файл с именем PER.OUT и иметь следующий формат:   
- в первой строке находится натуральное число T - время проезда по самому быстрому маршруту;   
- в каждой из следующих строк находится одно число - номер очередного перекрестка в маршруте (начиная с перекрестка с номером А и кончая В).

**Вариант 6**

Скрудж Мак-Дак решил сделать прибор для управления самолетом. Как известно, положение штурвала зависит от состояния входных датчиков, но эта функция довольно сложна.   
Его механик сделал устройство, вычисляющее эту функцию в несколько этапов с использование промежуточной памяти и вспомогательных функций. Для вычисления каждой из функций требуется, чтобы в ячейках памяти уже находились вычисленные параметры (которые являются значениями вычисленных функций), необходимые для ее вычисления. Вычисление функции без параметров может производится в любое время.   
После вычисления функции ячейки могут быть использованы повторно (хотя бы для записи результата вычисленной функции). Структура вызова функций такова, что каждая функция вычисляется не более одного раза и любой параметр используется не более одного раза. Любой параметр есть имя функции. Так как Скрудж не хочет тратить лишних денег на микросхемы, он поставил задачу минимизировать память прибора. По заданной структуре вызовов функций необходимо определить минимальный возможный размер памяти прибора и указать последовательность вычисления функций.   
Входной файл INPUT.TXT   
Формат ввода:   
1 строка> <общее количество функций N>   
2 строка> <имя функции, которую небходимо вычислить>   
3 строка> <имя функции> <кол-во параметров>[<список имен параметров>]   
...   
N+2 строка> <имя функции> <кол-во параметров>[<список имен параметров>]   
Выходной файл OUTPUT.TXT   
Формат вывода:   
Размер памяти (в ячейках)   
имя 1-й вычисляемой функции   
имя 2-й вычисляемой функции   
....   
имя функции, которую необходимо вычислить   
Примечание: имя функции есть натуральное число от 1 до N.

**Вариант 7**

1. Есть N карточек. На каждой из них черными чернилами написан ее уникальный номер - число от 1 до N. Также на каждой карточке красными чернилами написано еще одно целое число, лежащее в промежутке от 1 до N (некоторыми одинаковыми "красными" числами могут помечаться несколько карточек).   
Например, N=5, 5 карточек помечены следующим образом:   
---------------   
"черное" число ¦ 1¦ 2¦ 3¦ 4¦ 5¦   
---------------   
"красное" число ¦ 3¦ 3¦ 2¦ 4¦ 2¦   
---------------   
Необходимо выбрать из данных N карточек максимальное число карточек таким образом, чтобы множества "красных" и "черных" чисел на них совпадали.   
Для примера выше это будут карточки с "черными" номерами 2, 3, 4 (множество красных номеров, как и требуется в задаче, то же - {2,3,4}).

Входной файл: число N, N<=50   
Черный1 красный1   
......   
ЧерныйN красныйN

Числа разделены табуляцией.  
ВЫВОД:   
<Число карточек> S   
<"Черные" номера выбранных карточек> в столбик.   
  
Пример ввода Пример вывода   
5 3   
1 3 2  
2 3 3   
3 2 4   
4 4   
5 2

**Вариант 8**

Некоторые школы связаны компьютерной сетью. Между школами заключены соглашения: каждая школа имеет список школ получателей, которым она рассылает программное обеспечение всякий раз, получив новое бесплатное программное обеспечение (извне сети или из другой школы). При этом, если школа В есть в списке получателей школы А, то школа А может и не быть в списке получателей школы В.   
Требуется написать програму, определяющую минимальное количество школ, которым надо передать по экземпляру нового программного обеспечения, что бы распространить его по всем школам сети в соответствии с соглашениями (подзадача А).   
Кроме того, надо обеспечить возможность рассылки нового программного обеспечения из любой школы по всем остальным школам. Для этого расширить списки получателей некоторых школ, добавляя в них новые школы. Требуется найти минимальное суммарное количество расширений списков, при которых программное обеспечение из любой школы достигло бы остальных школ (подзадача В). Одно расширение означает добавление одной новой школы-получателя в список получателей одной из школ.   
Входные данные   
Первая строка файла INPUT.TXT содержит целое число N - количество школ в сети (2<=N<=100). Школы нумеруются первыми N положительными целыми числами. Каждая из следующих N строк задает список получателей. Строка i+1 содержит номера получателей i-й школы. Каждый такой список заканчивается нулем. Пустой список содержит только 0.   
Выходные данные   
Ваша программа должна записать 2 строки в файл OUTPUT.TXT. Его первая строка должна содержать одно положительное целое число - решение подзадачи А. Вторая строка должна содержать решение подзадачи B.   
Пример ввода и вывода   
INPUT.TXT   
5   
2 4 3 0   
4 5 0   
0   
0   
1 0   
OUTPUT.TXT   
1   
2

**Вариант 9**

Винни-Пух и Пятачок нанялись защищать компьютерную сеть от хакеров, которые выкачивали из компьютеров секретную информацию. Компьютерная сеть Винни-Пуха и Пятачка состояла из связанных между собой больших ЭВМ, к каждой из которых подключалось несколько терминалов. Подключение к одной из больших ЭВМ позволяло получить информацию, содержащуюся в памяти этой ЭВМ, а также всю информацию, доступную для ЭВМ, к которым данная ЭВМ могла направлять запросы. Хаккеры и раньше нападали на подобные компьютерные сети и их тактика была известна. Поэтому Винни-Пух и Пятачок разработали специальную программу, которая помогла принять меры против готовившегося нападения.   
Тактика хакеров.   
При нападениях хакеры всегда получают доступ к информации всех ЭВМ сети. Добиваются они этого, захватывая некоторые ЭВМ сети, так чтобы от них можно было запросить информацию у оставшихся ЭВМ. Вариантов захвата существует множество.   
Например, захватить все ЭВМ. Но хакеры всегда выбирают такой вариант, при котором суммарное количество терминалов у захваченных ЭВМ минимально.   
Примечание.   
В сети Винни-Пуха и Пятачка ни у каких 2-х ЭВМ количество терминалов не совпадает.   
Техническое задание.   
Вам необходимо написать программу, входными данными которой было бы описание сети, а выходными - список номеров ЭВМ, которые могут быть выбраны хакерами для захвата сети согласно их тактике. Ввод осуществляется из файла с именем INPUT.TXT.

Формат ввода.   
Количество ЭВМ в сети : N   
ЭВМ #1 имеет терминалов : T[1]   
ЭВМ #2 имеет терминалов : T[2]   
...   
ЭВМ #N имеет терминалов : T[N]   
Права на запрос :   
A[1] B[1]   
A[2] B[2]   
...   
A[K] B[K]   
0 0   
A[i] и В[i] - номера ЭВМ, последняя строка '0 0' обозначает конец списка прав на запрос, каждая пара A[i] B[i] обозначает, что ЭВМ с номеров A[i] имеет право запрашивать информацию у ЭВМ с номером B[i] (A[i] не равно B[i]).   
При вводе числа N и T[i] - натуральные, T[i] <=1000, N<=50, K<=2450.   
Входные данные соответствуют приведенным условиям.   
Формат вывода.   
Номера захватываемых ЭВМ : С[1] C[2] ... С[M].   
Количество захватываемых ЭВМ : <M>

|  |  |
| --- | --- |
| Input.txt  5  100  2  1  3  10  1 3  3 2  4 3  4 5  5 4  0 0 | Output.txt  1 4  2 |