Condense. Конденсация графа

Имя входного файла: `condense.in`
Имя выходного файла: `condense.out`

Дан связный ориентированный граф. Требуется найти компоненты сильной связности.

Формат входного файла

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и дуг графа соответственно ($1 \le n \le 20\,000$, $1 \le m \le 200\,000$).

Следующие m строк содержат описание дуг по одной на строке. Дуга номер i описывается двумя натуральными числами b_i, e_i — номерами концов дуги ($1 \le b_i, e_i \le n$).

Формат выходного файла

Первая строка выходного файла должна содержать одно натуральное число k — количество компонент сильной связности в заданном графе.

На следующей строке выведите n чисел — для каждой вершины выведите номер компоненты сильной связности, которой принадлежит эта вершина. Компоненты сильной связности должны быть занумерованы таким образом, чтобы для любого ребра номер компоненты сильной связности его начала не превышал номера компоненты сильной связности его конца.

Пример

condense.in	condense.out
6 7 1 2 2 3 3 1 4 5 5 6 6 4 2 4	2 1 1 1 2 2 2

Bridges. Мосты

Имя входного файла: `bridges.in`
Имя выходного файла: `bridges.out`

Дан неориентированный граф. Требуется найти все мосты в нем.

Формат входного файла

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($n \le 20\,000$, $m \le 200\,000$).

Следующие m строк содержат описание ребер по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i — номерами концов ребра ($1 \le b_i, e_i \le n$).

Формат выходного файла

Первая строка выходного файла должна содержать одно натуральное число b — количество мостов в заданном графе. На следующей строке выведите b целых чисел — номера ребер, которые являются мостами, в возрастающем порядке. Ребра нумеруются с единицы в том порядке, в котором они заданы во входном файле.

Пример

bridges.in	bridges.out
6 7 1 2 2 3 3 4 1 3 4 5 4 6 5 6	1 3

Post. Почтальон (*)

Имя входного файла: `post.in`
Имя выходного файла: `post.out`

В городе есть n площадей, соединенных улицами. При этом количество улиц не превышает ста тысяч и существует не более трех площадей, на которые выходит нечетное число улиц. Для каждой улицы известна ее длина. По улицам разрешено движение в обе стороны. В городе есть хотя бы одна улица. От любой площади до любой можно дойти по улицам.

Почтальону требуется пройти хотя бы один раз по каждой улице так, что бы длина его пути была наименьшей. Он может начать движение на любой площади и закончить также на любой (в том числе и на начальной).

Формат входного файла

Первая строка входного файла содержит натуральное число n — количество площадей в городе ($1 \le n \le 1000$). Далее следуют n строк, задающих улицы. В i-ой из этих строк находится число m_i — количество улиц, выходящих из площади i. Далее следуют m_i пар положительных чисел. В j-ой паре первое число — номер площади, в которую идет j-ая улица с i-ой площади, а второе число — длина этой улицы.

Между двумя площадями может быть несколько улиц, но не может быть улиц с площади на нее саму.

Все числа во входном файле не превосходят 10^5.

Формат выходного файла

Если решение существует, то в первую строку выходного файла выведите одно число — количество улиц в искомом маршруте (считая первую и последнюю), а во вторую — номера площадей в порядке их посещения.

Если решений нет, выведите в выходной файл одно число `-1`.

Если решений несколько, выведите любое.

Пример

post.in	post.out
4 2 2 1 2 2 4 1 2 4 4 3 5 1 1 2 2 5 4 8 2 3 8 2 4	5 1 2 3 4 2 1

Optimal. Оптимальный параметр

Имя входного файла: `optimal.in`
Имя выходного файла: `optimal.out`

Как известно, для хранения расстояний в алгоритме Дейкстры можно использовать структуру данных «k-ичная куча». Время работы алгоритма Дейкстры в таком случае составляет $O(Vk\log_k V + E\log_k V)$, где V — количество вершин в графе, а E — количество ребер.

Для простоты будем считать, что время работы алгоритма составляет ровно $f(V, E, k) = Vk\log_k V + E\log_k V$.

В этом случае для данных V и E существует такое вещественное число k, что $f(V, E, k)$ минимально. При этом рассматриваются только значения $k \geq 2$.

Также среди всех натуральных k, больших 1, существует такое, на котором достигается наименьшее значение функции $f(V, E, k)$.

По заданным V и E найдите оптимальное вещественное и оптимальное натуральное значения k.

Примечание. Значение $\log_k V$ можно вычислить следующим способом: `ln(V) / ln(k)`.

Формат входного файла

Входной файл содержит два натуральных числа V и E ($2 \leq V \leq 10^9$, $0 \leq E \leq \frac{V(V-1)}{2}$, $E \leq 10^9$).

Формат выходного файла

В выходной файл выведите два числа — оптимальное вещественное и натуральное значения k. Вещественное значение следует вывести с точностью четыре знака после десятичной точки.

Пример

optimal.in	optimal.out
100 4950	23.122248253 23
3 2	3.322319048 3

Pref. Преферанс (*)

Имя входного файла: `pref.in`
Имя выходного файла: `pref.out`

В новой колоде 32 карты для преферанса расположены в следующем порядке (сверху вниз): червы, бубны, трефы, пики. В каждой масти сначала лежит семерка, под ней — восьмерка, затем девятка, десятка, валет, дама, король, туз. Тасовка карт осуществляется так: 16 карт, составляющих верхнюю половину колоды, распределяются между картами нижней половины колоды. Каждая карта верхней половины вставляется в нижнюю колоду так, что в получившейся колоде карты верхней половины идут в том же порядке, в котором они были изначально. Любое число карт верхней половины можно располагать как над верхней, так и под нижней картой второй половины колоды, а также между любыми двумя соседними картами нижней половины колоды. Такие действия повторяются не более пяти раз.

Требуется написать программу, которая указывает, как надо осуществить тасовку, чтобы в итоге получить заранее заданное расположение карт.

Формат входного файла

Единственная строка входного файла содержит информацию о порядке карт, в котором они должны оказаться после тасовки. Карты перечислены сверху вниз.

Каждая карта обозначается латинской буквой, указывающей масть (пики — `S`, трефы — `C`, бубны — `D`, червы — `H`), и номиналом (туз — `A`, король — `K`, дама — `Q`, валет — `J`, десятка — `0`, остальные — в соответствии со своим значением: `9`, `8`, `7`).

Формат выходного файла

В первую строку выходного файла необходимо вывести целое число N ($0 \leq N \leq 5$) — количество шагов тасовки. Следующие N строк должны содержать информацию о каждом шаге тасовки. Каждая строка при этом должна содержать 16 чисел, указывающих номера позиций, на которых оказываются снятые карты. Номера позиций выводятся в порядке возрастания и разделены пробелами. Нумерация позиций производится сверху вниз от 1 до 32.

Пример

В примере входная строка для удобства разбита на две. Во входных файлах при тестировании задачи будет ровно одна строка, завершенная переводом строки.

pref.in
C7 H7 C8 H8 C9 H9 C0 H0 S7 D7 S8 D8 S9 D9 S0 D0 SJ DJ SQ DQ SK DK SA DA CJ HJ CQ HQ CK HK CA HA
pref.out
2 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 1 2 3 4 5 6 7 8 25 26 27 28 29 30 31 32