Содержание

Пример работы с файлами.

Если вы не умеете читать/выводить данные, или открывать файлы, воспользуйтесь примерами. `http://acm.math.spbu.ru/~sk1/algo/sum/`

Пример работы с файлами.

В некоторых задачах большой ввод и вывод. Про ввод-вывод в C++:
`http://acm.math.spbu.ru/~sk1/algo/input-output/cpp_common.html`
Имеет смысл пользоваться супер быстрым вводом-выводом. Две версии:
`http://acm.math.spbu.ru/~sk1/algo/input-output/io_export.cpp.html`
`http://acm.math.spbu.ru/~sk1/algo/input-output/fread_write_export.cpp.html`

Выделение памяти.

В некоторых задачах нужен STL, который активно использует динамическую память (set-ы, map-ы) переопределение стандартного аллокатора ускорит вашу программу:
`http://acm.math.spbu.ru/~sk1/algo/memory.cpp.html`

Обязательные задачи

Задача 13A. Unionday. День Объединения [0.5 sec, 256 mb]

В Байтландии есть целых n городов, но нет ни одной дороги. Король решил исправить эту ситуацию и соединить некоторые города дорогами так, чтобы по этим дорогам можно было бы добраться от любого города до любого другого. Когда строительство будет завершено, Король планирует отпраздновать День Объединения. К сожалению, казна Байтландии почти пуста, поэтому Король требует сэкономить деньги, минимизировав суммарную длину всех построенных дорог.

Формат входных данных

Первая строка входного файла содержит натуральное число n ($1 \leqslant n \leqslant 5\,000$) — количество городов в Байтландии. Каждая из следующих n строк содержит два целых числа x_i, y_i — координаты i-го города ($-10\,000 \leqslant x_i, y_i \leqslant 10\,000$). Никакие два города не расположены в одной точке.

Формат выходных данных

Первая строка выходного файла должна содержать минимальную суммарную длину дорог. Выведите число с точностью не менее 10^{-3}.

Примеры

unionday.in	unionday.out
6 1 1 7 1 2 2 6 2 1 3 7 3	9.65685

Замечание

Нужно решение за $\mathcal{O}(n^2)$.

Задача 13B. Остовное дерево 2 [0.5 sec, 256 mb]

Требуется найти в связном графе остовное дерево минимального веса.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно. Следующие m строк содержат описание ребер по одному на строке. Ребро номер i описывается тремя натуральными числами b_i, e_i и w_i — номера концов ребра и его вес соответственно ($1 \leqslant b_i, e_i \leqslant n$, $0 \leqslant w_i \leqslant 100\,000$). $n \leqslant 20\,000, m \leqslant 100\,000$.

Граф является связным.

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число — вес минимального остовного дерева.

Примеры

spantree2.in	spantree2.out
4 4 1 2 1 2 3 2 3 4 5 4 1 4	7

Замечание

Можно написать Прима с кучей или Краскала...

Задача 13C. Разрезание графа [0.5 sec, 256 mb]

Дан неориентированный граф. Над ним в заданном порядке производят операции следующих двух типов:

- `cut` — разрезать граф, то есть удалить из него ребро;
- `ask` — проверить, лежат ли две вершины графа в одной компоненте связности.

Известно, что после выполнения всех операций типа `cut` рёбер в графе не осталось. Найдите результат выполнения каждой из операций типа `ask`.

Формат входных данных

Первая строка входного файла содержит три целых числа, разделённые пробелами — количество вершин графа n, количество рёбер m и количество операций k ($1 \leqslant n \leqslant 50\,000$, $0 \leqslant m \leqslant 100\,000$, $m \leqslant k \leqslant 150\,000$).

Следующие m строк задают рёбра графа; i-ая из этих строк содержит два числа u_i и v_i ($1 \leqslant u_i, v_i \leqslant n$), разделённые пробелами — номера концов i-го ребра. Вершины нумеруются с единицы; граф не содержит петель и кратных рёбер.

Далее следуют k строк, описывающих операции. Операция типа `cut` задаётся строкой "`cut` u v" ($1 \leqslant u, v \leqslant n$), которая означает, что из графа удаляют ребро между вершинами u и v. Операция типа `ask` задаётся строкой "`ask` u v" ($1 \leqslant u, v \leqslant n$), которая означает, что необходимо узнать, лежат ли в данный момент вершины u и v в одной компоненте связности. Гарантируется, что каждое ребро графа встретится в операциях типа `cut` ровно один раз.

Формат выходных данных

Для каждой операции `ask` во входном файле выведите на отдельной строке слово "`YES`", если две указанные вершины лежат в одной компоненте связности, и "`NO`" в противном случае. Порядок ответов должен соответствовать порядку операций `ask` во входном файле.

Пример

cutting.in	cutting.out
3 3 7 1 2 2 3 3 1 ask 3 3 cut 1 2 ask 1 2 cut 1 3 ask 2 1 cut 2 3 ask 3 1	YES YES NO NO

Замечание

Это простая задача на `DSU`.

Задача 13D. Ребра добавляются, граф растет [0.5 sec, 256 mb]

В неориентированный граф последовательно добавляются новые ребра. Изначально граф пустой. После каждого добавления нужно говорить, является ли текущий граф двудольным.

Формат входных данных

На первой строке n — количество вершин, m — количество операций «добавить ребро». Следующие m строк содержат пары чисел от 1 до n — описание добавляемых ребер.

Формат выходных данных

Выведите в строчку m нулей и единиц. i-й символ должен быть равен единице, если граф, состоящий из первых i ребер, является двудольным.

Ограничения: $1 \leqslant n, m \leqslant 300\,000$.

Примеры

addedge.in	addedge.out
3 3 1 2 2 3 3 1	110

Замечание

Решение с бинарным поиском скорее всего не зайдёт по времени. Используйте `DSU`.

Задача 13E. Цикл отрицательного веса [0.5 sec, 256 mb]

Дан ориентированный граф. Определите, есть ли в нем цикл отрицательного веса, и если да, то выведите его.

Формат входных данных

Во входном файле в первой строке число N ($1 \leqslant N \leqslant 100$) — количество вершин графа. В следующих N строках находится по N чисел — матрица смежности графа. Все веса ребер не превышают по модулю 10 000. Если ребра нет, то соответствующее число равно 100 000.

Формат выходных данных

В первой строке выходного файла выведите «YES», если цикл существует или «NO» в противном случае. При его наличии выведите во второй строке количество вершин в искомом цикле и в третьей строке — вершины входящие в этот цикл в порядке обхода.

Пример

negcycle.in	negcycle.out
2 0 -1 -1 0	YES 2 1 2

Замечание

Можно написать Флойда или Форд-Беллмана... Следите аккуратно за переполнениями.

Задача 13F. Отрицательный цикл [0.5 sec, 256 mb]

Дан взвешенный ориентированный граф. Требуется определить, содержит ли он цикл отрицательного веса. Гарантируется, что все вершины графа достижимы из первой.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($n \leqslant 1111$, $m \leqslant 11111$). Следующие m строк содержат описание ребер по одному на строке. Ребро номер i описывается тремя числами b_i, e_i и w_i — номера концов ребра и его вес соответственно ($1 \leqslant b_i, e_i \leqslant n, -100\,000 \leqslant w_i \leqslant 100\,000$). Обратите внимание, что в графе могут быть кратные ребра и петли.

Формат выходных данных

Первая строка выходного файла должна содержать `yes`, если граф содержит цикл отрицательного веса и `no` в противном случае.

Пример

negcycle.in	negcycle.out
4 4 2 1 -4 1 2 1 3 4 2 2 3 3	yes
4 6 2 1 4 1 2 1 3 4 2 2 3 3 1 1 2 1 2 2	no

Замечание

Можно написать Флойда или Форд-Беллмана... Флойд получит `TL`. Следите аккуратно за переполнениями.

Задача 13G. Зависимости между функциями [0.5 sec, 256 mb]

Даны названия функций и зависимости между ними. Нужно выписать названия функций по одному разу в таком порядке, что:

- Если функция A зависит от функции B, функция B должна быть выписана раньше функции A.
- Если предыдущий критерий позволяет в какой-то момент выписать более чем одну функцию, то выписана должна быть та из них, название которой лексикографически минимально.

Известно, что между функциями нет циклических зависимостей.

Формат входных данных

В первой строке входного файла задано целое число n — количество функций ($1 \leqslant n \leqslant 50$). Следующие n строк содержат описания функций. Каждая из этих строк имеет вид $\texttt{name}_i\ d_{i,1}\ d_{i,2}\ \ldots\ d_{i,k_i}$; здесь $\texttt{name}_i$ — имя функции, а $d_{i,l}$ — номера функций (считая с нуля), от которых зависит данная. Имя функции непусто и состоит из не более чем 50 заглавных букв латинского алфавита; имена всех функций различны. Соседние элементы строки отделены друг от друга одним пробелом. Длина каждой строки не превосходит 200 символов.

Формат выходных данных

Выведите в выходной файл n строк. В каждой строке выведите название одной из функций. Функции должны быть перечислены в требуемом порядке.

Примеры

functions.in	functions.out
3 B 1 C A 1	C A B
3 B 1 C A 0	C B A
10 K A B 1 1 C 2 D 3 E 4 F 5 G 6 H 7 I 8	A B C D E F G H I K

Замечание

У вас есть два алгоритма топологической сортировки. Второй подойдёт.

Дополнительные задачи

Задача 13H. Возьми себе за правило: летай всегда GraphAero! [1 sec, 256 mb]

Наконец авиаперевозки стали доступны всем и каждому! Однако, из-за жёсткой конкуренции в сфере пассажироперевозок осталось только две авиакомпании: «GraphAero Airlines» и «Aerofloat».

Авиакомпания «GraphAero Airlines» активно развивается. Ведь для получения большей прибыли... простите, для удобства пассажиров каждый месяц компания добавляет один новый рейс. Компании «Aerofloat» остаётся довольствоваться тем, что остаётся. А именно, единственная возможность удержаться на плаву — добавлять рейсы, дублирующие самые загруженные рейсы компании «GraphAero Airlines». Рейс является самым загруженным, если существует такая пара городов, что можно долететь (возможно, с пересадками) из одного города в другой, используя рейсы авиакомпании, но если этот рейс отменить — то долететь будет невозможно. Аналитикам компании «Aerofloat» необходимо постоянно контролировать ситуацию — сколько в данный момент существует самых загруженных рейсов.

Поскольку вы уже давно мечтаете летать по льготным ценам (скидка $10^{-5}\%$), вы решили оказать посильную помощь. Помните: самолёты летают по всему миру! Между двумя крупными городами может быть более одного рейса, а города бывают настолько большими, что самолёты могут летать в пределах одного города. Рейсами можно пользоваться как в одну, так и в другую сторону.

Формат входных данных

Первая строка входного файла содержит целое число N ($1 \leqslant N \leqslant 100\,000$) — количество городов и M ($0 \leqslant M \leqslant 100\,000$) — изначальное число рейсов компании «GraphAero Airlines». Далее следует M строк, в каждой содержится описание очередного рейса — номера двух городов, между которыми осуществляется рейс. В следующей строке содержится число K ($1 \leqslant K \leqslant 100\,000$) — количество добавленных рейсов. Далее содержится описание добавленных рейсов в таком же формате.

Формат выходных данных

После каждого добавления нового рейса выведите на отдельной строке одно число — количество самых загруженных рейсов.

Примеры

bridges.in	bridges.out
4 0 4 1 2 2 3 3 4 1 4	1 2 3 0
4 3 1 2 2 3 3 4 4 1 1 1 2 1 3 1 4	3 2 1 0

Замечание

Это несложная задача на тему DSU. Её можно решать в offline. Хранить скорее всего придётся целых два DSU...

Задача 13I. MST случайных точек [2 sec, 256 mb]

Даны n различных точек на плоскости. Координаты точек — целые числа от 0 до 30 000 включительно. Точки выбраны *случайно* в следующем смысле: рассмотрим все возможные наборы из n различных точек на плоскости с заданными ограничениями на координаты и выберем из них случайно и равновероятно один набор.

Вы можете провести отрезок между любыми двумя заданными точками. Длина отрезка между точками с координатами (x_1, y_1) и (x_2, y_2) равна $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Будем говорить, что точки a и b *связаны*, если они соединены отрезком, или же существует точка d, которая связана и с a, и с b. Ваша задача — провести отрезки минимальной суммарной длины так, чтобы все точки были связаны.

Формат входных данных

В первой строке ввода задано целое число n ($2 \leqslant n \leqslant 50\,000$). Следующие n строк содержат координаты точек. Гарантируется, что все точки различны. Кроме того, во всех тестах, кроме примера, гарантируется, что точки выбраны случайно, как описано в условии.

Формат выходных данных

В первой строке выведите вещественное число w — суммарную длину отрезков. В следующих $(n - 1)$ строках выведите отрезки, по одному на строке. Каждый отрезок следует выводить как два числа от 1 до n, обозначающие номера точек, являющихся концами этого отрезка.

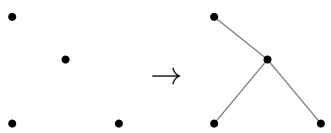
Пусть на самом деле суммарная длина выведенных вами отрезков равна w^*, а суммарная длина отрезков в оптимальном ответе равна w_{opt}. Тогда ваш ответ будет считаться верным, если

$$\max\left(\left|\frac{w}{w^*} - 1\right|, \left|\frac{w^*}{w_{\text{opt}}} - 1\right|\right) < 10^{-12}.$$

Пример

randommst.in	randommst.out
4 0 10 5 6 10 0 0 0	22.02362358924615 1 2 2 3 4 2

Иллюстрация



Замечание

Вы уже знаете из последней теорпрактики, каким алгоритмом воспользоваться. Вопрос в том, как найти ближайшие $\mathcal{O}(1)$ точек? Воспользуйтесь методом “проекция на случайную прямую”. Приветствуются попытки решить задачу любым другим способом =)

Задача 13J. Лифтостроитель Эдуард [0.5 sec, 256 mb]

Эдуард работает инженером в компании «Нетривиальные лифты». Его очередное задание — разработать новый лифт для небоскрёба из h этажей.

У Эдуарда есть идея-фикс: он считает, что четырёх кнопок должно хватать каждому. Его последнее конструктивное предложение предполагает следующие кнопки:

- Подняться на a этажей вверх
- Подняться на b этажей вверх
- Подняться на c этажей вверх
- Вернуться на первый этаж

Исходно лифт находится на первом этаже. Пассажир использует три первые кнопки, чтобы попасть на нужный этаж. Если пассажир пытается переместиться на этаж, которого не существует, то есть нажать одну из первых трёх кнопок на этаже со слишком большим номером, лифт не перемещается.

Чтобы доказать, что план достоин реализации, Эдуард хочет подсчитать количество этажей, до которых возможно доехать с его помощью.

Формат входных данных

В первой строке записано целое число h — количество этажей небоскрёба ($1 \leqslant h \leqslant 10^{18}$).

Во второй строке записаны целые числа a, b и c — параметры лифта ($1 \leqslant a, b, c \leqslant 100\,000$).

Формат выходных данных

Выведите одно целое число — количество этажей, доступных с первого с помощью лифта.

Пример

elevator.in	elevator.out
15 4 7 9	9

Замечание

Это идейная задача. Реализация предельно простая.