# Содержание

**Общая информация:**

Вход в контест: `http://contest.yandex.ru/contest/1757/`
Дедлайн на задачи: 10 дней, до 7-го ноября 23:59.
К каждой главе есть более простые задачи (base), посложнее (advanced), и сложные (hard).
В скобках к каждой главе написано сколько любых задач из этой главы нужно сдать.
Сайт курса: `https://compscicenter.ru/courses/algorithms-1/2015-autumn/`
Семинары ведут Сергей Копелиович (burunduk30@gmail.com, vk.com/burunduk1) и Глеб Леонов (gleb.leonov@gmail.com, vk.com/id1509292)

В каждом условии указан таймлимит для `C/C++`.
Таймлиминт для `Java` примерно в 2-3 раза больше.
Таймлиминт для `Python` примерно в 5 раз больше.

**C++:**

Быстрый ввод-вывод.
`http://acm.math.spbu.ru/~sk1/algo/input-output/cpp_common.html`

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`

**Java:**

Быстрый ввод-вывод.
`http://acm.math.spbu.ru/~sk1/algo/input-output/java/java_common.html`

# 08.Base [4/4]

## Задача 08A. Светофоры [0.1 sec, 256 mb]

В подземелье $M$ тоннелей и $N$ перекрестков, каждый тоннель соединяет какие-то два перекрестка. Мышиный король решил поставить по светофору в каждом тоннеле перед каждым перекрестком. Напишите программу, которая посчитает, сколько светофоров должно быть установлено на каждом из перекрестков. Перекрестки пронумерованы числами от 1 до $N$.

### Формат входных данных

Во входном файлее записано два числа $N$ и $M$ ($0 < N \leqslant 100$), $0 \leqslant M \leqslant \frac{N(N-1)}{2}$). В следующих $M$ строках записаны по два числа $i$ и $j$ ($1 \leqslant i, j \leqslant N$), которые означают, что перекрестки $i$ и $j$ соединены тоннелем.

### Формат выходных данных

В выходной файл вывести $N$ чисел: $k$-е число означает количество светофоров на $k$-м перекрестке.

### Пример

| lights.in | lights.out |
|---|---|
| 7 10<br>5 1<br>3 2<br>7 1<br>5 2<br>7 4<br>6 5<br>6 4<br>7 5<br>2 1<br>5 3 | 3 3 2 2 5 2 3 |

### Задача 08B. От списка ребер к матрице смежности [0.1 sec, 256 mb]

Простой неориентированный граф задан списком ребер, выведите его представление в виде матрицы смежности.

## Формат входных данных

Входной файл содержит числа $N$ ($1 \leqslant N \leqslant 100$) — число вершин в графе и $M$ ($1 \leqslant M \leqslant \frac{n(n-1)}{2}$) — число ребер. Затем следует $M$ пар чисел — ребра графа.

## Формат выходных данных

Выведите в выходной файл матрицу смежности заданного графа.

## Пример

| e2m.in | e2m.out |
|---|---|
| 3 3<br>1 2<br>2 3<br>1 3 | 0 1 1<br>1 0 1<br>1 1 0 |

## Задача 08C. Связность [0.1 sec, 256 mb]

В этой задаче требуется проверить, что граф является *связным*, то есть что из любой вершины можно по рёбрам этого графа попасть в любую другую. В графе могут существовать петли и кратные ребра.

## Формат входных данных

В первой строке входного файла заданы числа $N$ и $M$ через пробел — количество вершин и рёбер в графе, соответственно ($1 \leqslant N \leqslant 100$, $0 \leqslant M \leqslant 10\,000$). Следующие $M$ строк содержат по два числа $u_i$ и $v_i$ через пробел ($1 \leqslant u_i, v_i \leqslant N$); каждая такая строка означает, что в графе существует ребро между вершинами $u_i$ и $v_i$.

## Формат выходных данных

Выведите "`YES`", если граф является связным, и "`NO`" в противном случае.

## Примеры

| connect.in | connect.out |
|---|---|
| 3 2<br>1 2<br>3 2 | YES |
| 3 1<br>1 3 | NO |

### Задача 08D. Дерево [0.1 sec, 256 mb]

Дан неориентированный граф. Проверьте, является ли он деревом.

## Формат входных данных

В первой строке входного файла заданы через пробел два целых числа $n$ и $m$ — количество вершин и рёбер в графе, соответственно ($1 \leqslant n \leqslant 100$). В следующих $m$ строках заданы рёбра; $i$-я из этих строк содержит два целых числа $u_i$ и $v_i$ через пробел — номера концов $i$-го ребра ($1 \leqslant u_i, v_i \leqslant n$). Граф не содержит петель и кратных рёбер.

## Формат выходных данных

В первой строке выходного файла выведите "`YES`", если граф является деревом, и "`NO`" в противном случае.

## Примеры

| tree.in | tree.out |
|---|---|
| 3 2<br>1 2<br>1 3 | YES |
| 3 3<br>1 2<br>2 3<br>3 1 | NO |

# 08.Advanced [5/8]

### Задача 08E. Любители Кошек [0.16 sec, 256 mb]

В университетском клубе любителей кошек зарегистрировано $n$ членов. Естественно, что некоторые из членов клуба знакомы друг с другом. Нужно сосчитать, сколькими способами можно выбрать из них троих, которые могли бы свободно общаться (то есть, любые два из которых знакомы между собой).

## Формат входных данных

В первой строке входного файла заданы числа $n$ и $m$ ($1 \leqslant n \leqslant 1\,000$, $1 \leqslant m \leqslant 30\,000$), где $m$ обозначает общее число знакомств. В последующих $m$ строках идут пары чисел $a_i$ $b_i$, обозначающие, что $a_i$ знаком с $b_i$. Информация об одном знакомстве может быть записана несколько раз, причем даже в разном порядке (как $(x, y)$, так и $(y, x)$).

## Формат выходных данных

В выходной файл необходимо вывести количество способов выбрать троих попарно знакомых друг с другом людей из клуба.

## Пример

| catlover.in | catlover.out |
|---|---|
| 3 3<br>1 2<br>2 3<br>3 1 | 1 |

## Замечание

Заметьте, что у задачи есть два решения – за $\mathcal{O}(n^3)$ и $\mathcal{O}(nm)$, второе быстрее... В этой задаче достаточно матрицы смежности.

### Задача 08F. Компоненты связности [0.2 sec, 256 mb]

Вам задан неориентированный граф с $N$ вершинами и $M$ ребрами ($1 \leqslant N \leqslant 20\,000$, $1 \leqslant M \leqslant 200\,000$). В графе отсутствуют петли и кратные ребра.

Определите компоненты связности заданного графа.

## Формат входных данных

Граф задан во входном файле следующим образом: первая строка содержит числа $N$ и $M$. Каждая из следующих $M$ строк содержит описание ребра — два целых числа из диапазона от 1 до $N$ — номера концов ребра.

## Формат выходных данных

На первой строке выходного файла выведите число $L$ — количество компонент связности заданного графа. На следующей строке выведите $N$ чисел из диапазона от 1 до $L$ — номера компонент связности, которым принадлежат соответствующие вершины. Компоненты связности следует занумеровать от 1 до $L$ произвольным образом.

## Пример

| connect.in | connect.out |
|---|---|
| 4 2<br>1 2<br>3 4 | 2<br>1 1 2 2 |

### Задача 08G. Поиск пути на гриде [0.8 sec, 256 mb]

Дано прямоугольное поле $W \times H$. Некоторые клетки проходимы, через некоторые ходить нельзя. Из клетки можно ходить в соседние по ребру (слева, справа, сверху, снизу).

Нужно из клетки $(x_1, y_1)$ найти любой (не обязательно кратчайший, даже не обязательно простой) путь в клетку $(x_2, y_2)$.

## Формат входных данных

На первой строке $W, H, x_1, y_1, x_2, y_2$ ($1 \leqslant x_1, x_2 \leqslant W \leqslant 1000$, $1 \leqslant y_1, y_2 \leqslant H \leqslant 1000$). Далее $H$ строк, в каждой из которых по $W$ символов. Символ "." означает, что клетка проходима, а символ "*" означает, что по ней ходить нельзя.

Клетки $(x_1, y_1)$ и $(x_2, y_2)$ не совпадают и обе проходимы.

## Формат выходных данных

Если пути не существует, выведите `NO`.

Иначе выведите `YES` и последовательность клеток $(x_i, y_i)$, в которой первая совпадает с клеткой $(x_1, y_1)$, а последняя с клеткой $(x_2, y_2)$.

## Пример

| dfsongrid.in | dfsongrid.out |
|---|---|
| 4 2 1 1 4 2<br>....<br>.... | YES<br>1 1 2 1 3 1 4 1 3 1 3 2<br>4 2 |
| 4 2 1 1 4 2<br>..*.<br>.*.. | NO |
| 4 2 1 1 4 2<br>..*.<br>*... | YES<br>1 1 2 1 2 2 3 2 4 2 |

## Замечание

В этой задаче не обязательно строить граф в явном виде. Более того из-за лишних манипуляций можно легко получить `TL` или `ML`.

### Задача 08H. TopSort. Топологическая сортировка [0.2 sec, 256 mb]

Дан ориентированный невзвешенный граф. Необходимо его топологически отсортировать.

## Формат входных данных

В первой строке входного файла даны два натуральных числа $N$ и $M$ ($1 \leqslant N \leqslant 100\,000, M \leqslant 100\,000$) — количество вершин и рёбер в графе соответственно. Далее в $M$ строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

## Формат выходных данных

Вывести любую топологическую сортировку графа в виде последовательности номеров вершин. Если граф невозможно топологически отсортировать, вывести -1.

## Пример

| topsort.in | topsort.out |
|---|---|
| 6 6<br>1 2<br>3 2<br>4 2<br>2 5<br>6 5<br>4 6 | 4 6 3 1 2 5 |
| 3 3<br>1 2<br>2 3<br>3 1 | -1 |

### Задача 08I. Поиск цикла [0.3 sec, 256 mb]

Дан ориентированный невзвешенный граф без петель и кратных рёбер. Необходимо определить есть ли в нём циклы, и если есть, то вывести любой из них.

## Формат входных данных

В первой строке входного файла находятся два натуральных числа $N$ и $M$ ($1 \leqslant N \leqslant 100\,000$, $M \leqslant 100\,000$) — количество вершин и рёбер в графе соответственно. Далее в $M$ строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

## Формат выходных данных

Если в графе нет цикла, то вывести «NO», иначе — «YES» и затем перечислить все вершины в порядке обхода цикла.

## Примеры

| cycle.in | cycle.out |
|---|---|
| 2 2<br>1 2<br>2 1 | YES<br>1 2 |
| 3 3<br>1 2<br>2 3<br>1 3 | NO |

### Задача 08J. Condense 2. Конденсация графа [0.4 sec, 256 mb]

Требуется найти количество ребер в конденсации ориентированного графа. Примечание: конденсация графа не содержит кратных ребер.

## Формат входных данных

Первая строка входного файла содержит два натуральных числа $n$ и $m$ — количество вершин и ребер графа соответственно ($n \leqslant 10\,000,\ m \leqslant 100\,000$). Следующие $m$ строк содержат описание ребер, по одному на строке. Ребро номер $i$ описывается двумя натуральными числами $b_i$, $e_i$ — началом и концом ребра соответственно ($1 \leqslant b_i,\ e_i \leqslant n$). В графе могут присутствовать кратные ребра и петли.

## Формат выходных данных

Первая строка выходного файла должна содержать одно число — количество ребер в конденсации графа.

## Пример

| condense2.in | condense2.out |
|---|---|
| 4 4<br>2 1<br>3 2<br>2 3<br>4 3 | 2 |

### Задача 08K. Longpath. Длиннейший путь [0.4 sec, 256 mb]

Дан ориентированный граф без циклов. Требуется найти в нем длиннейший путь.

## Формат входных данных

Первая строка входного файла содержит два натуральных числа $n$ и $m$ — количество вершин и дуг графа соответственно. Следующие $m$ строк содержат описания дуг по одной на строке. Ребро номер $i$ описывается двумя натуральными числами $b_i$ и $e_i$ — началом и концом дуги соответственно ($1 \leqslant b_i, e_i \leqslant n$).

Входной граф не содержит циклов и петель.

$n \leqslant 10\,000$, $m \leqslant 100\,000$.

## Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число — количество дуг в длиннейшем пути.

## Пример

| longpath.in | longpath.out |
|---|---|
| 5 5<br>1 2<br>2 3<br>3 4<br>3 5<br>1 5 | 3 |

### Задача 08L. Unique Topsort [0.4 sec, 256 mb]

Дан ориентированный ацикличный граф $G$. Проверить, что существует единственный топологический порядок вершин графа.

## Формат входных данных

Первая строка входных данных содержит число вершин графа $n$ ($1 \leqslant n \leqslant 100\,000$) и число ребер графа $m$ ($0 \leqslant m \leqslant 100\,000$). Следующие $m$ строк содержат пары чисел от 1 до $n$, задающие начало и конец соответствующего ребра. Гарантируется, что граф не содержит циклов.

## Формат выходных данных

Если топологический порядок единственный, выведите на первой строке `YES`, а на второй номера вершин в топологическом порядке, иначе выведите `NO`.

## Примеры

| unitopsort.in | unitopsort.out |
|---|---|
| 1 0 | YES<br>1 |
| 2 1<br>2 1 | YES<br>2 1 |
| 4 2<br>1 2<br>4 3 | NO |

# 08.Hard [0/2]

### Задача 08M. Avia. Авиаперелеты [1.5 sec, 256 mb]

Главного конструктора Петю попросили разработать новую модель самолета для компании «Air Бубундия». Оказалось, что самая сложная часть заключается в подборе оптимального размера топливного бака.

Главный картограф «Air Бубундия» Вася составил подробную карту Бубундии. На этой карте он отметил расход топлива для перелета между каждой парой городов.

Петя хочет сделать размер бака минимально возможным, для которого самолет сможет долететь от любого города в любой другой (возможно, с дозаправками в пути).

## Формат входных данных

Первая строка входного файла содержит натуральное число $n$ ($1 \leqslant n \leqslant 1000$) — число городов в Бубундии. Далее идут $n$ строк по $n$ чисел каждая. $j$-ое число в $i$-ой строке равно расходу топлива при перелете из $i$-ого города в $j$-ый. Все числа не меньше нуля и меньше $10^9$. Гарантируется, что для любого $i$ в $i$-ой строчке $i$-ое число равно нулю.

## Формат выходных данных

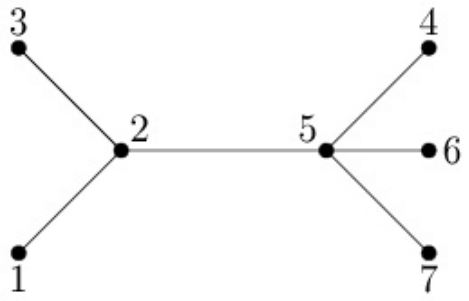
Первая строка выходного файла должна содержать одно число — оптимальный размер бака.

## Пример

| avia.in | avia.out |
|---|---|
| 4<br>0 10 12 16<br>11 0 8 9<br>10 13 0 22<br>13 10 17 0 | 10 |

### Задача 08N. Autotourism [1.5 sec, 256 mb]

В Байтландии существуют $n$ городов, соединённых $n-1$ дорогами с двусторонним движением таким образом, что из каждого города можно проехать в любой другой по сети дорог. Длина каждой дороги равна 1 километру.




Бензобак автомобиля позволяет проехать без заправки $m$ километров. Требуется выбрать маршрут, позволяющий посетить наибольшее количество различных городов без дозаправки. При этом начинать и заканчивать маршрут можно в произвольных городах.

## Формат входных данных

В первой строке входного файла заданы два целых числа $n$ и $m$ ($2 \le n \le 500\,000$, $1 \le m \le 200\,000\,000$) — количество городов в стране и количество километров, которое автомобиль может проехать без дозаправки. В последующих $n-1$ строках описаны дороги. Каждая дорога задаётся двумя целыми числами $a$ и $b$ ($1 \leqslant a, b \leqslant n$) — номерами городов, которые она соединяет. Длина каждой дороги равна 1 км.

## Формат выходных данных

Выведите одно число — максимальное количество городов, которое можно посетить без дозаправки.

## Пример

| autotourism.in | autotourism.out |
|---|---|
| 7 6<br>1 2<br>2 3<br>2 5<br>5 6<br>5 7<br>5 4 | 5 |

## Пояснение к примеру

5 городов можно посетить, например, по схеме $4 \to 5 \to 7 \to 5 \to 6 \to 5 \to 2$ или по схеме $3 \to 2 \to 1 \to 2 \to 5 \to 6 \to 5$.