Задание к лабораторной работе №3 (семестр 2).
Графы

В данной лабораторной работе изучаются графы, поиск в глубину, поиск в ширину, алгоритм Дейкстры, алгоритм Беллмана-Форда. Аналогично предыдущим лабораторным работам, есть **два** способа ее выполнения и защиты, однако присутствуют изменения:

1 **Базовый уровень.** ***Решается 3 задачи по вариантам***. Варианты в табличке внизу, номер вашего варианта соответствует вашему номеру в списке группы. **Посмотреть свой номер нужно, например, в журнале успеваемости по дисциплине**. Вариант в течении семестра менять нальзя!

Условия базового уровня.

- **Всего баллов.** За базовый уровень максимум за защиту и отчет **в сумме** можно получить 9 баллов, если сдаете в срок (1 месяц, до **25 мая 2022** года включительно).
- **Замена.** ***Одну*** задачу можно заменить без потерь **внутри** уровня сложности, или на более сложную. На уровень ниже - нельзя заменить.
- **Сложность.** У каждой задачи подписано количество баллов, которые можно за нее получить, и по этим баллам задачи распределены на 4 уровня сложности:
  * 1 уровень, задачи по 1-2 баллов: с 1 по 10;
  * 2 уровень, задачи по 2-3 балла: с 11 по 17;
  * Задачи №18 и 19 не входят ни в один вариант (дополнительная тема по MST).
- **Замена на задачу ниже уровнем**. Вы можете заменить одну задачу на уровень ниже, но тогда вы потеряете разницу в стоимости между ними.
- Замена второй задачи влечет за собой штраф в дополнительные 2 балла.
- **Отчет.** На защиту ***без отчета***, оформленного по правилам и загруженного в гитхаб, можно ***не приходить***!
- Правила оформления отчета:
  * Формат файла - pdf;
  * Титульный лист с названием учреждения и факультета, в котором вы учитесь, курса, названия работы, а также ваши ФИО, группа; ФИО преподавателя; город, дата выполнения;
  * Описание задания к задаче;
  * Описание вашего решения и исходный код;
  * Описание проведенных тестов;
  * Выводы по каждой задаче и в целом по проделанной работе.
- Хорошо оформленный отчет может добавить до 1 балла к защите, на усмотрение преподавателя. Т.е. можно получить 10 баллов в итоге.
- За плохо оформленный отчет может сниматься до 2 баллов, также по усмотрению преподавателя. Т.е. можно получить только 7 баллов даже с учетом правильности всех задач.
- **Дедлайн**. Если вы сдаете лабораторную работу *позже* срока (начиная с 25.05.2022), то суммарная базовая стоимость понижается на 2 балла, т.е. до 7 баллов. Эти 2 балла можно компенсировать решением дополнительных задач.

2 **Продвинутый уровень.** Решаются ваши задачи по вариантам, причем принципы, описанные выше для базового случая тоже учитываются. До максимальных 15 баллов вы можете набрать, решая другие задачи не из вашего варинта, выбирая произвольно с учетом стоимости задач. На ваше усмотрение.

Условие на дедлайн так же действует для продвинутого уровня (-2 балла).

P.s. Решать все задачи вообще не обязательно. Но если хочется, то каждая решенная **до дедлайна** задача 2 уровня сложности (а также 18 и 19), которая превышает лимит в максимальные 15 баллов, при условии, что на 15 баллов уже сданы задачи, – может быть дополнительно учтена по количеству баллов в конце семестра в счет, например, экзамена.

Варианты

Вариант	Номера задач	Вариант	Номера задач
1	1,5,17	16	1,9,16
2	2,7,13	17	2,8,17
3	3,8,14	18	3,9,13
4	4,9,15	19	4,5,14
5	5,10,16	20	6,8,15
6	5,6,15	21	1,7,11
7	1,10,14	22	2,7,16
8	3,10,13	23	3,9,15
9	4,9,17	24	4,10,14
10	5,8,11	25	5,10,17
11	2,7,16	26	6,9,13
12	4,6,11	27	2,8,11
13	3,12,13	28	3,7,16
14	4,7,12	29	4,10,15
15	5,10,15	30	5,9,14

Содержание

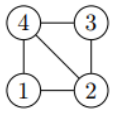
1 Формат представления графов в некоторых заданиях

В задачах 1,2... графы даются на ввод следующим образом. В первой строке заданы целые неотрицательные числа n и m – количество вершин и количество ребер соответственно. Вершины всегда нумеруются от 1 до n. Каждая из следующих m строк определяет ребро в формате u v через пробел, где $1 \leq u, v \leq n$ – инцидентные вершины ребра. Если в задаче дан неориентированный граф, то такая запись задает неориентированное ребро между вершинами u и v. В случае ориентированного графа – определяет направленное ребро от u к v. Если в задаче дан взвешенный граф, то каждое ребро задается как u v w, где u и v – вершины, а w – вес ребра.

Гарантируется, что данный граф является простым. То есть он не содержит петель (ребер, идущих от вершины к самой себе) и кратных ребер. Примеры:

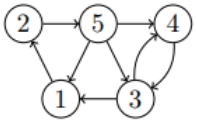
- Неориентированный граф с 4-мя вершинами и 5-ю ребрами:

input
4 5
2 1
4 3
1 4
2 4
3 2



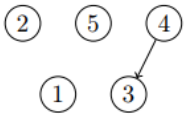
- Ориентированный граф с 5 вершинами и 8 ребрами:

input
5 8
4 3
1 2
3 1
3 4
2 5
5 1
5 4
5 3



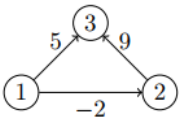
- Ориентированный граф с 5 вершинами и одним ребром:

input
5 1
4 3



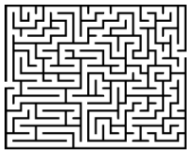
- Взвешенный ориентированный граф с 3 вершинами и 3 ребрами:

input
3 3
2 3 9
1 3 5
1 2 -2



1 Задача. Лабиринт [5 s, 512 Mb, 1 балл]

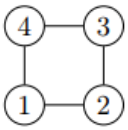
Лабиринт представляет собой прямоугольную сетку ячеек со стенками между некоторыми соседними ячейками. Вы хотите проверить, существует ли путь от данной ячейки к данному выходу из лабиринта, где выходом также является ячейка, лежащая на границе лабиринта (в примере, показанном на рисунке, есть два выхода: один на левой границе и один на правой границе). Для этого вы представляете лабиринт в виде неориентированного графа: вершины графа являются ячейками лабиринта, две вершины соединены неориентированным ребром, если они смежные и между ними нет стены. Тогда, чтобы проверить, существует ли путь между двумя заданными ячейками лабиринта, достаточно проверить, что существует путь между соответствующими двумя вершинами в графе.



Вам дан неориентированный граф и две различные вершины u и v. Проверьте, есть ли путь между u и v.

- **Формат ввода / входного файла (input.txt).** Неориентированный граф с n вершинами и m ребрами по формату 1. Следующая строка после ввода всго графа содержит две вершины u и v.
- **Ограничения на входные данные.** $2 \leq n \leq 10^3$, $1 \leq m \leq 10^3$, $1 \leq u, v \leq n$, $u \neq v$.
- **Формат вывода / выходного файла (output.txt).** Выведите 1, если есть путь между вершинами u и v; выведите 0, если пути нет.
- Ограничение по времени. 5 сек.
- Ограничение по памяти. 512 мб.
- Пример 1:

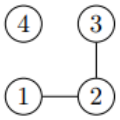
input	output
4 4 1 2 3 2 4 3 1 4 1 4	1



В этом графе между вершинами 1 и 4 есть два пути: 1-4 и 1-2-3-4.

- Пример 2:

input	output
4 2 1 2 3 2 1 4	0



Во втором примере пути между вершинами 1 и 4 нет.

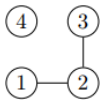
2 Задача. Компоненты [5 s, 512 Mb, 1 балл]

Теперь вы решаете сделать так, чтобы в лабиринте не было мертвых зон, то есть чтобы из каждой клетки был доступен хотя бы один выход. Для этого вы находите связные компоненты соответствующего неориентированного графа и следите за тем, чтобы каждый компонент содержал выходную ячейку.

Дан неориентированный граф с n вершинами и m ребрами. Нужно посчитать количество компонент связности в нем.

- **Формат ввода / входного файла (input.txt).** Неориентированный граф с n вершинами и m ребрами по формату 1.
- **Ограничения на входные данные.** $1 \leq n \leq 10^3$, $0 \leq m \leq 10^3$.
- **Формат вывода / выходного файла (output.txt).** Выведите количество компонент связности.
- Ограничение по времени. 5 сек.
- Ограничение по памяти. 512 мб.
- Пример:

input	output
4 2 1 2 3 2	2



В этом графе есть два компонента связности: 1, 2, 3 и 4.

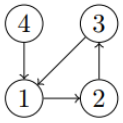
3 Задача. Циклы [5 s, 512 Mb, 1 балл]

Учебная программа по инфокоммуникационным технологиям определяет пререквизиты для каждого курса в виде списка курсов, которые необходимо пройти перед тем, как начать этот курс. Вы хотите выполнить проверку согласованности учебного плана, то есть проверить отсутствие циклических зависимостей. Для этого строится следующий ориентированный граф: вершины соответствуют курсам, есть направленное ребро (u, v) – курс u следует пройти перед курсом v. Затем достаточно проверить, содержит ли полученный граф цикл.

Проверьте, содержит ли данный граф циклы.

- **Формат ввода / входного файла (input.txt).** Ориентированный граф с n вершинами и m ребрами по формату 1.
- **Ограничения на входные данные.** $1 \leq n \leq 10^3$, $0 \leq m \leq 10^3$.
- **Формат вывода / выходного файла (output.txt).** Выведите 1, если данный граф содержит цикл; выведите 0, если не содержит.
- Ограничение по времени. 5 сек.
- Ограничение по памяти. 512 мб.
- Пример 1:

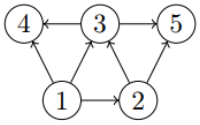
input	output
4 4 1 2 4 1 2 3 3 1	1



Этот граф содержит цикл: $3 \to 1 \to 2 \to 3$.

- Пример 2:

input	output
5 7 1 2 2 3 1 3 3 4 1 4 2 5 3 5	0



В этом графе нет циклов. Это можно увидеть, например, отметив, что все ребра в этом графе идут от вершины с меньшим номером к вершине с большим номером.

4 Задача. Порядок курсов [10 s, 512 Mb, 1 балл]

Теперь, когда вы уверены, что в данном учебном плане нет циклических зависимостей, вам нужно найти порядок всех курсов, соответствующий всем зависимостям. Для этого нужно сделать топологическую сортировку соответствующего ориентированного графа.

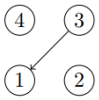
Дан ориентированный ациклический граф (DAG) с n вершинами и m ребрами. Выполните топологическую сортировку.

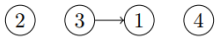
- **Формат ввода / входного файла (input.txt).** Ориентированный ациклический граф с n вершинами и m ребрами по формату 1.
- **Ограничения на входные данные.** $1 \leq n \leq 10^5$, $0 \leq m \leq 10^5$. Графы во входных файлах гарантированно ациклические.
- **Формат вывода / выходного файла (output.txt).** Выведите ***любое*** линейное упорядочение данного графа (Многие ациклические графы имеют более одного варианта упорядочения, вы можете вывести любой из них).
- Ограничение по времени. 10 сек.
- Ограничение по памяти. 512 мб.
- Пример 1:

input	output
4 3 1 2 4 1 3 1	4 3 1 2

- Пример 2:

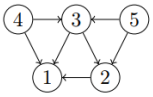
input	output
4 1 3 1	2 3 1 4

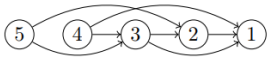




- Пример 3:

input	output
5 7 2 1 3 2 3 1 4 3 4 1 5 2 5 3	5 4 3 2 1





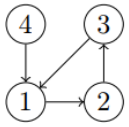
5 Задача. Город с односторонним движением [5 s, 512 Mb, 1.5 балла]

Департамент полиции города сделал все улицы односторонними. Вы хотели бы проверить, можно ли законно проехать с любого перекрестка на какой-либо другой перекресток. Для этого строится ориентированный граф: вершины – это перекрестки, существует ребро (u, v) всякий раз, когда в городе есть улица (с односторонним движением) из u в v. Тогда достаточно проверить, все ли вершины графа лежат в одном компоненте сильной связности.

Нужно вычислить количество компонентов сильной связности заданного ориентированного графа с n вершинами и m ребрами.

- **Формат ввода / входного файла (input.txt).** Ориентированный граф с n вершинами и m ребрами по формату 1.
- **Ограничения на входные данные.** $1 \le n \le 10^4$, $0 \le m \le 10^4$.
- **Формат вывода / выходного файла (output.txt).** Выведите число – количество компонентов сильной связности.
- Ограничение по времени. 5 сек.
- Ограничение по памяти. 512 мб.
- Пример 1:

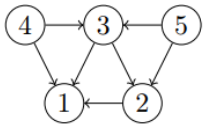
input	output
4 4 1 2 4 1 2 3 3 1	2



В этом графе есть два компонента сильной связности: $\{1, 3, 2\}$ и $\{4\}$.

- Пример 2:

input	output
5 7 2 1 3 2 3 1 4 3 4 1 5 2 5 3	5



В этом графе пять компонента сильной связности: $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$ и $\{5\}$.

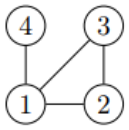
6 Задача. Количество пересадок [10 s, 512 Mb, 1 балл]

Вы хотите вычислить минимальное количество сегментов полета, чтобы добраться из одного города в другой. Для этого вы строите следующий неориентированный граф: вершины представляют города, между двумя вершинами есть ребро всякий раз, когда между соответствующими двумя городами есть перелет. Тогда достаточно найти кратчайший путь из одного из заданных городов в другой.

Дан *неориентироанный* граф с n вершинами и m ребрами, а также две вершины u и v, нужно посчитать длину кратчайшего пути между u и v (то есть, минимальное количество ребер в пути из u в v).

- **Формат ввода / входного файла (input.txt).** Нериентированный граф задан по формату 1. Следующая строка содержит две вершины u и v.
- **Ограничения на входные данные.** $2 \leq n \leq 10^5$, $0 \leq m \leq 10^5$, $1 \leq u, v \leq n$, $u \neq v$.
- **Формат вывода / выходного файла (output.txt).** Выведите минимальное количество ребер в пути из u в v. Выведите -1, если пути нет.
- Ограничение по времени. 10 сек.
- Ограничение по памяти. 512 мб.
- Пример 1:

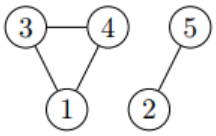
input	output
4 4 1 2 4 1 2 3 3 1 2 4	2



В этом графе существует единственный кратчайший путь между вершинами 2 и 4: 2 – 1 – 4.

- Пример 2:

input	output.txt
5 4 5 2 1 3 3 4 1 4 3 5	-1



В этом графе нет пути между вершинами 3 и 5.

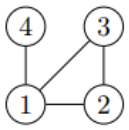
7 Задача. Двудольный граф [10 s, 512 Mb, 1.5 балла]

Неориентированный граф называется **двудольным**, если его вершины можно разбить на две части так, что каждое ребро графа соединяет вершины из разных частей, то есть не существует рёбер между вершинами одной и той же части графа. Двудольные графы естественным образом возникают в задачах, где граф используется для моделирования связей между объектами двух разных типов (например, мальчиками и девочками, или студентами и общежитиями). Альтернативное определение таково: граф двудольный, если его вершины можно раскрасить двумя цветами (например, черным и белым) так, что концы каждого ребра окрашены в разные цвета.

Дан неориентированный граф с n вершинами и m ребрами, проверьте, является ли он двудольным.

- **Формат ввода / входного файла (input.txt).** Неориентированный граф задан по формату 1.
- **Ограничения на входные данные.** $1 \leq n \leq 10^5$, $0 \leq m \leq 10^5$.
- **Формат вывода / выходного файла (output.txt).** Выведите 1, если граф двудольный; и 0 в противном случае.
- Ограничение по времени. 10 сек.
- Ограничение по памяти. 512 мб.
- Пример 1:

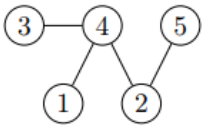
input	output
4 4 1 2 4 1 2 3 3 1	0



Этот граф не является двудольным. Чтобы убедиться в этом, предположим, что вершина 1 окрашена в белый цвет. Тогда вершины 2 и 3 нужно покрасить в черный цвет, так как граф содержит ребра 1, 2 и 1, 3. Но тогда ребро 2, 3 имеет оба конца одного цвета.

- Пример 2:

input	output
5 4 5 2 4 2 3 4 1 4	1



Этот граф двудольный: вершины 4 и 5 покрасим в белый цвет, все остальные вершины – в черный цвет.

- Что делать? Адаптируйте поиск в ширину (BFS), чтобы решить эту проблему.

8 Задача. Стоимость полета [10 s, 512 Mb, 1.5 балла]

Теперь вас интересует минимизация не количества пересадок, а общей стоимости полета. Для этого строится взвешенный граф: вес ребра из одного города в другой – это стоимость соответствующего перелета.

Дан ориентированный граф с положительными весами ребер, n - количество вершин и m - количество ребер, а также даны две вершины u и v. Вычислить вес кратчайшего пути между u и v (то есть минимальный общий вес пути из u в v).

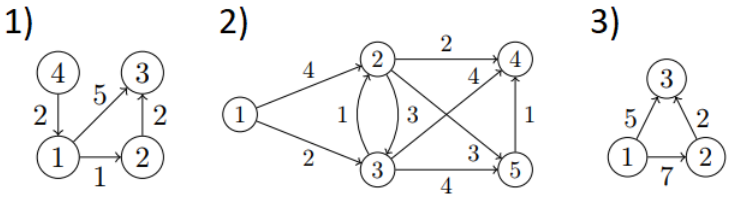
- **Формат ввода / входного файла (input.txt).** Ориентированный взвешенный граф задан по формату 1. Следующая строка содержит две вершины u и v.
- **Ограничения на входные данные.** $1 \leq n \leq 10^4$, $0 \leq m \leq 10^5$, $1 \leq u, v \leq n$, $u \neq v$, вес каждого ребра – неотрицательное целое число, не превосходящее 10^8.
- **Формат вывода / выходного файла (output.txt).** Выведите минимальный вес пути из u в v. Веведите -1, если пути нет.
- Ограничение по времени. 10 сек.
- Ограничение по памяти. 512 мб.
- Примеры:

input	output
4 4 1 2 1 4 1 2 2 3 2 1 3 5 1 3	3

input	output
5 9 1 2 4 1 3 2 2 3 2 3 2 1 2 4 2 3 5 4 5 4 1 2 5 3 3 4 4 1 5	6

input	output
3 3 1 2 7 1 3 5 2 3 2 3 2	-1

- Объяснения:



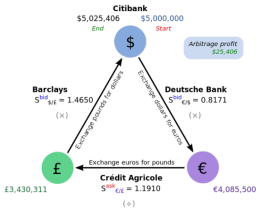
Пример 1 – В этом графе существует единственный кратчайший путь из вершины 1 в вершину 3 ($1 \to 2 \to 3$), и он имеет вес 3. Пример 2 – Есть два пути от 1 до 5 общего веса 6: $1 \to 3 \to 5$ и $1 \to 3 \to 2 \to 5$. Пример 3 – Нет пути от вершины 3 до 2.

9 Задача. Аномалии курсов валют [10 s, 512 Mb, 2 балла]

Вам дан список валют $c_1, c_2, \ldots, c_n$ вместе со списком обменных курсов: r_{ij} – количество единиц валюты c_j, которое можно получить за одну единицу c_i. Вы хотите проверить, можно ли начать делать обмен с одной единицы какой-либо валюты, выполнить последовательность обменов и получить более одной единицы той же валюты, с которой вы начали обмен. Другими словами, вы хотите найти валюты $c_{i_1}, c_{i_2}, \ldots, c_{i_k}$ такие, что $r_{i_1,i_2} \cdot r_{i_2,i_3} \cdot \cdots \cdot r_{i_{k-1},i_k} \cdot r_{i_k,i_1} > 1$.

Для этого построим следующий граф: вершинами являются валюты $c_1, c_2, \ldots, c_n$, вес ребра из c_i в c_j равен $-\log r_{ij}$. Тогда достаточно проверить, есть ли в этом графе отрицательный цикл. Пусть цикл $c_i \to c_j \to c_k \to c_i$ имеет отрицательный вес. Это означает, что $-(\log c_{ij} + \log c_{jk} + \log c_{ki}) < 0$ и, следовательно, $\log c_{ij} + \log c_{jk} + \log c_{ki} > 0$. Это, в свою очередь, означает, что

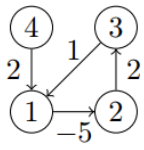
$$r_{ij} r_{jk} r_{ki} = 2^{\log c_{ij}} 2^{\log c_{jk}} 2^{\log c_{ki}} = 2^{\log c_{ij} + \log c_{jk} + \log c_{ki}} > 1.$$



Для заданного ориентированного графа с возможными отрицательными весами ребер, у которого n вершин и m ребер, проверьте, содержит ли он цикл с отрицательным суммарным весом.

- **Формат ввода / входного файла (input.txt).** Ориентированный взвешенный граф задан по формату 1.
- **Ограничения на входные данные.** $1 \le n \le 10^3$, $0 \le m \le 10^4$, вес каждого ребра – целое число, не превосходящее ***по модулю*** 10^4.
- **Формат вывода / выходного файла (output.txt).** Выведите 1, если граф содержит цикл с отрицательным суммарным весом. Выведите 0 в противном случае.
- Ограничение по времени. 10 сек.
- Ограничение по памяти. 512 мб.
- Пример:

input	output
4 4 1 2 -5 4 1 2 2 3 2 3 1 1	1



Вес цикла $1 \to 2 \to 3$ равен -2, то есть отрицателен.

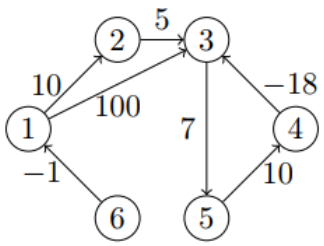
10 Задача. Оптимальный обмен валюты [10 s, 512 Mb, 2 балла]

Теперь вы хотите вычислить оптимальный способ обмена данной вам валюты c_i на *все* другие валюты. Для этого вы находите кратчайшие пути из вершины c_i во все остальные вершины.

Дан ориентированный граф с возможными отрицательными весами ребер, у которого n вершин и m ребер, а также задана одна его вершина s. Вычислите длину кратчайших путей из s во все остальные вершины графа.

- **Формат ввода / входного файла (input.txt).** Ориентированный взвешенный граф задан по формату 1.
- **Ограничения на входные данные.** $1 \le n \le 10^3$, $0 \le m \le 10^4$, $1 \le s \le n$, вес каждого ребра – целое число, не превосходящее ***по модулю*** 10^9.
- **Формат вывода / выходного файла (output.txt).** Для каждой вершины i графа от 1 до n выведите в каждой отдельной строке следующее:
  - «*», если пути из s в i нет;
  - «-», если существует путь из s в i, но нет кратчайшего пути из s в i (то есть расстояние от s до i равно $-\infty$);
  - длину кратчайшего пути в остальных случаях.
- Ограничение по времени. 10 сек.
- Ограничение по памяти. 512 мб.
- Пример 1:

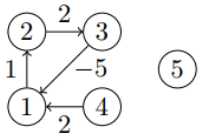
input	output
6 7	0
1 2 10	10
2 3 5	-
1 3 100	-
3 5 7	-
5 4 10	*
4 3 -18	
6 1 -1	
1	



Первая строка вывода утверждает, что расстояние от вершины 1 до 1 равно 0. Вторая показывает, что расстояние от 1 до 2 равно 10 (соответствующий путь $1 \to 2$). Следующие три строки показывают, что расстояние от 1 до вершин 3, 4 и 5 равно $-\infty$: действительно, сначала можно достичь вершины 3 через ребра $1 \to 2 \to 3$, а затем сделать длину пути произвольно малой, совершая достаточно много обходов по циклу $3 \to 5 \to 4 \to 3$ с отрицательным весом. Последняя строка вывода показывает, что в этом графе нет пути от 1 до 6.

- Пример 2:

input	output
5 4	-
1 2 1	-
4 1 2	-
2 3 2	0
3 1 -5	*
4	



В этом случае расстояние от 4 до вершин 1, 2 и 3 равно $-\infty$, поскольку существует отрицательный цикл $1 \to 2 \to 3 \to 1$, достижимый из 4. Расстояние от 4 до 4 равно нулю. Пути от 4 к 5 нет.

11 Задача. Алхимия [1 s, 16 Mb, 3 балла]

Алхимики средневековья владели знаниями о превращении различных химических веществ друг в друга. Это подтверждают и недавние исследования археологов.

В ходе археологических раскопок было обнаружено m глиняных табличек, каждая из которых была покрыта непонятными на первый взгляд символами. В результате расшифровки выяснилось, что каждая из табличек описывает одну алхимическую реакцию, которую умели проводить алхимики.

Результатом алхимической реакции является превращение одного вещества в другое. Задан набор алхимических реакций, описанных на найденных глиняных табличках, исходное вещество и требуемое вещество. Необходимо выяснить: возможно ли преобразовать исходное вещество в требуемое с помощью этого набора реакций, а в случае положительного ответа на этот вопрос – найти минимальное количество реакций, необходимое для осуществления такого преобразования.

- **Формат входных данных (input.txt) и ограничения.** Первая строка входного файла INPUT.TXT содержит целое число m ($0 \leq m \leq 1000$) – количество записей в книге. Каждая из последующих m строк описывает одну алхимическую реакцию и имеет формат «вещество1 -> вещество2», где «вещество1» – название исходного вещества, «вещество2» – название продукта алхимической реакции. $m + 2$-ая строка входного файла содержит название вещества, которое имеется исходно, $m + 3$-ая – название вещества, которое требуется получить.

  Во входном файле упоминается не более 100 различных веществ. Название каждого из веществ состоит из строчных и заглавных английских букв и имеет длину не более 20 символов. Строчные и заглавные буквы различаются.

- **Формат выходных данных (output.txt).** В выходной файл OUTPUT.TXT выведите минимальное количество алхимических реакций, которое требуется для получения требуемого вещества из исходного, или -1, если требуемое вещество невозможно получить.

- Ограничение по времени. 1 сек.

- Ограничение по памяти. 16 мб.

- Примеры:

input.txt	output.txt
5 Aqua -> AquaVita AquaVita -> PhilosopherStone AquaVita -> Argentum Argentum -> Aurum AquaVita -> Aurum Aqua Aurum	2

input.txt	output.txt
5 Aqua -> AquaVita AquaVita -> PhilosopherStone AquaVita -> Argentum Argentum -> Aurum AquaVita -> Aurum Aqua Osmium	-1

- Проверяем **обязательно** – на acmp.

12 Задача. Цветной лабиринт [1 s, 16 Mb, 2 балла]

В одном из парков одного большого города недавно был организован новый аттракцион Цветной лабиринт. Он состоит из n комнат, соединенных m двунаправленными коридорами. Каждый из коридоров покрашен в один из ста цветов, при этом от каждой комнаты отходит не более одного коридора каждого цвета. При этом две комнаты могут быть соединены любым количеством коридоров.

Человек, купивший билет на аттракцион, оказывается в комнате номер один. Кроме билета, он также получает описание пути, по которому он может выбраться из лабиринта. Это описание представляет собой последовательность цветов $c_1 \dots c_k$. Пользоваться ей надо так: находясь в комнате, надо посмотреть на очередной цвет в этой последовательности, выбрать коридор такого цвета и пойти по нему. При этом если из комнаты нельзя пойти по коридору соответствующего цвета, то человеку приходится дальше самому выбирать, куда идти.

В последнее время в администрацию парка стали часто поступать жалобы от заблудившихся в лабиринте людей. В связи с этим, возникла необходимость написания программы, проверяющей корректность описания и пути, и, в случае ее корректности, сообщающей номер комнаты, в которую ведет путь.

Описание пути некорректно, если на пути, который оно описывает, возникает ситуация, когда из комнаты нельзя пойти по коридору соответствующего цвета.

- **Формат входных данных (input.txt) и ограничения.** Первая строка входного файла INPUT.TXT содержит два целых числа n ($1 \le n \le 10000$) и m ($1 \le m \le 100000$) - соответственно количество комнат и коридоров в лабиринте. Следующие m строк содержат описания коридоров. Каждое описание содержит три числа u ($1 \le u \le n$), v ($1 \le v \le n$), c ($1 \le c \le 100$) - соответственно номера комнат, соединенных этим коридором, и цвет коридора. Следующая, $(m+2)$-ая строка входного файла содержит длину описания пути - целое число k ($0 \le k \le 100000$). Последняя строка входного файла содержит k целых чисел, разделенных пробелами, - описание пути по лабиринту.
- **Формат выходных данных (output.txt).** В выходной файл OUTPUT.TXT выведите строку INCORRECT, если описание пути некорректно, иначе выведите номер комнаты, в которую ведет описанный путь. Помните, что путь начинается в комнате номер один.
- Ограничение по времени. 1 сек.
- Ограничение по памяти. 16 мб.
- Примеры:

input.txt	output.txt
3 2 1 2 10 1 3 5 5 10 10 10 10 5	3

input.txt	output.txt
3 2 1 2 10 2 3 5 5 5 10 10 10 10	INCORRECT

input.txt	output.txt
3 2 1 2 10 1 3 5 4 10 10 10 5	INCORRECT

- Проверяем **обязательно** – на acmp.

13 Задача. Грядки [1 s, 16 Mb, 3 балла]

Прямоугольный садовый участок шириной N и длиной M метров разбит на квадраты со стороной 1 метр. На этом участке вскопаны грядки. Грядкой называется совокупность квадратов, удовлетворяющая таким условиям:

- из любого квадрата этой грядки можно попасть в любой другой квадрат этой же грядки, последовательно переходя по грядке из квадрата в квадрат через их общую сторону;
- никакие две грядки не пересекаются и не касаются друг друга ни по вертикальной, ни по горизонтальной сторонам квадратов (касание грядок углами квадратов допускается).

Подсчитайте количество грядок на садовом участке.

- **Формат входных данных (input.txt) и ограничения.** В первой строке входного файла INPUT.TXT находятся числа N и M через пробел, далее идут N строк по M символов. Символ # обозначает территорию грядки, точка соответствует незанятой территории. Других символов в исходном файле нет $(1 \le N, M \le 200)$.
- **Формат выходных данных (output.txt).** В выходной файл OUTPUT.TXT выведите количество грядок на садовом участке.
- Ограничение по времени. 1 сек.
- Ограничение по памяти. 16 мб.
- Примеры:

input.txt	output.txt
5 10 ##.....#. .#..#...#. .###....#. ..##....#. ....#.	3

input.txt	output.txt
5 10 ##..#####. .#.#.#.... ###..##.#. ..##.....# .###.#####	5

- Проверяем **обязательно** – на acmp.

14 Задача. Автобусы [1 s, 16 Mb, 3 балла]

Между некоторыми деревнями края Власюки ходят автобусы. Поскольку пассажиропотоки здесь не очень большие, то автобусы ходят всего несколько раз в день.

Марии Ивановне требуется добраться из деревни d в деревню v как можно быстрее (считается, что в момент времени 0 она находится в деревне d).

- **Формат входных данных (input.txt) и ограничения.** Во входном файле INPUT.TXT записано число N - общее число деревень ($1 \leq N \leq 100$), номера деревень d и v, затем количество автобусных рейсов R ($0 \leq R \leq 10000$). Затем идут описания автобусных рейсов. Каждый рейс задается номером деревни отправления, временем отправления, деревней назначения и временем прибытия (все времена - целые от 0 до 10000). Если в момент t пассажир приезжает в деревню, то уехать из нее он может в любой момент времени, начиная с t.
- **Формат выходных данных (output.txt).** В выходной файл OUTPUT.TXT вывести минимальное время, когда Мария Ивановна может оказаться в деревне v. Если она не сможет с помощью указанных автобусных рейсов добраться из d в v, вывести -1.
- Ограничение по времени. 1 сек.
- Ограничение по памяти. 16 мб.
- Пример:

input.txt	output.txt
3 1 3 4 1 0 2 5 1 1 2 3 2 3 3 5 1 1 3 10	5

- Проверяем **обязательно** – на acmp.

15 Задача. Герои [1 s, 16 Mb, 3 балла]

Коварный кардинал Ришелье вновь организовал похищение подвесок королевы Анны; вновь спасать королеву приходится героическим мушкетерам. Атос, Портос, Арамис и д'Артаньян уже перехватили агентов кардинала и вернули украденное; осталось лишь передать подвески королеве Анне. Королева ждет мушкетеров в дворцовом саду. Дворцовый сад имеет форму прямоугольника и разбит на участки, представляющие собой небольшие садики, содержащие коллекции растений из разных климатических зон. К сожалению, на некоторых участках, в том числе на всех участках, расположенных на границах сада, уже притаились в засаде гвардейцы кардинала; на бой с ними времени у мушкетеров нет. Мушкетерам удалось добыть карту сада с отмеченными местами засад; теперь им предстоит выбрать наиболее оптимальные пути к королеве. Для надежности друзья разделили между собой спасенные подвески и проникли в сад поодиночке, поэтому начинают свой путь к королеве с разных участков сада. Двигаются герои по максимально короткой возможной траектории.

Марлезонский балет вот-вот начнется; королева не в состоянии ждать героев больше L минут; ровно в начале $L+1$-ой минуты королева покинет парк, и те мушкетеры, что не успеют к этому времени до нее добраться, не смогут передать ей подвески. На преодоление одного участка у мушкетеров уйдет ровно по минуте. С каждого участка мушкетеры могут перейти на 4 соседние. Требуется выяснить, сколько подвесок будет красоваться на платье королевы, когда она придет на бал.

- **Формат входных данных (input.txt) и ограничения.** Первая строка входного файла INPUT.TXT содержит целые числа N и M ($1 \le N, M \le 20$) – размеры сада. Далее идут N строк по M символов в каждом; символы '0' соответствуют участкам, на которых нет засады, символы '1' – участкам, на которых разместились гвардейцы. В $N+2$-ой строке теста записано три целых числа: координаты участка, на котором королева будет ждать мушкетёров (Q_x, Q_y) ($1 < Q_x < N$, $1 < Q_y < M$) и время в минутах до начала балета ($1 \le L \le 1000$). В $N+3$-ей строке записаны через пробел целые числа координаты участка, с которого стартует Атос (A_x, A_y) ($1 < A_x < N$, $1 < A_y < M$) и количество подвесок, хранящихся у него ($1 \le P_a \le 1000$). В $N+4$, $N+5$ и $N+6$-ой строках аналогично записаны стартовые координаты и количество подвесок у Портоса, Арамиса и д'Артаньяна.
- **Формат выходных данных (output.txt).** В выходной файл OUTPUT.TXT выведите количество подвесок, которое королева успеет получить у мушкетеров до начала балета.
- Ограничение по времени. 1 сек.
- Ограничение по памяти. 16 мб.
- Примеры:

input.txt	output.txt
5 5 11111 10001 10001 10001 11111 4 4 10 2 2 1 2 3 2 3 2 3 3 3 4	10

input.txt	output.txt
5 5 11111 10001 10111 10101 11111 4 4 10 2 2 1 2 2 2 2 2 3 2 2 4	0

- Проверяем **обязательно** – на acmp.

16 Задача. Рекурсия [1 s, 16 Mb, 3 балла]

Одним из важных понятий, используемых в теории алгоритмов, является рекурсия. Неформально ее можно определить как использование в описании объекта самого себя. Если речь идет о процедуре, то в процессе исполнении эта процедура напрямую или косвенно (через другие процедуры) вызывает сама себя.

Рекурсия является очень «мощным» методом построения алгоритмов, но таит в себе некоторые опасности. Например, неаккуратно написанная рекурсивная процедура может войти в бесконечную рекурсию, то есть, никогда не закончить свое выполнение (на самом деле, выполнение закончится с переполнением стека).

Поскольку рекурсия может быть косвенной (процедура вызывает сама себя через другие процедуры), то задача определения того факта, является ли данная процедура рекурсивной, достаточно сложна. Попробуем решить более простую задачу.

Рассмотрим программу, состоящую из n процедур $P_1, P_2, \ldots, P_n$. Пусть для каждой процедуры известны процедуры, которые она может вызывать. Процедура P называется потенциально рекурсивной, если существует такая последовательность процедур $Q_0, Q_1, \ldots, Q_k$, что $Q_0 = Q_k = P$ и для $i = 1...k$ процедура Q_{i-1} может вызвать процедуру Q_i. В этом случае задача будет заключаться в определении для каждой из заданных процедур, является ли она потенциально рекурсивной.

Требуется написать программу, которая позволит решить названную задачу.

- **Формат входных данных (input.txt) и ограничения.** Первая строка входного файла INPUT.TXT содержит целое число n – количество процедур в программе ($1 \leq n \leq 100$). Далее следуют n блоков, описывающих процедуры. После каждого блока следует строка, которая содержит 5 символов «*».

  Описание процедуры начинается со строки, содержащий ее идентификатор, состоящий только из маленьких букв английского алфавита и цифр. Идентификатор непуст, и его длина не превосходит 100 символов. Далее идет строка, содержащая число k ($k \leq n$) – количество процедур, которые могут быть вызваны описываемой процедурой. Последующие k строк содержат идентификаторы этих процедур – по одному идентификатору на строке.

  Различные процедуры имеют различные идентификаторы. При этом ни одна процедура не может вызвать процедуру, которая не описана во входном файле.

- **Формат выходных данных (output.txt).** В выходной файл OUTPUT.TXT для каждой процедуры, присутствующей во входных данных, необходимо вывести слово YES, если она является потенциально рекурсивной, и слово NO – в противном случае, в том же порядке, в каком они перечислены во входных данных.

- Ограничение по времени. 1 сек.

- Ограничение по памяти. 16 мб.

- Пример:

input.txt	output.txt
3 p1 2 p1 p2 ***** p2 1 p1 ***** p3 1 p1 *****	YES YES NO

- Проверяем **обязательно** – на acmp.

17 Задача. Слабая K-связность [1 s, 16 Mb, 4 балла]

Ане, как будущей чемпионке мира по программированию, поручили очень ответственное задание. Правительство вручает ей план постройки дорог между N городами. По плану все дороги односторонние, но между двумя городами может быть больше одной дороги, возможно, в разных направлениях. Ане необходимо вычислить минимальное такое K, что данный ей план является слабо K-связным.

Правительство называет план слабо K-связным, если выполнено следующее условие: для любых двух различных городов можно проехать от одного до другого, нарушая правила движения не более K раз. Нарушение правил - это проезд по существующей дороге в обратном направлении. Гарантируется, что между любыми двумя городами можно проехать, возможно, несколько раз нарушив правила.

- **Формат входных данных (input.txt) и ограничения.** В первой строке входного файла INPUT.TXT записаны два числа $2 \leq N \leq 300$ и $1 \leq M \leq 10^5$ - количество городов и дорог в плане. В последующих M строках даны по два числа - номера городов, в которых начинается и заканчивается соответствующая дорога.
- **Формат выходных данных (output.txt).** В выходной файл OUTPUT.TXT выведите минимальное K, такое, что данный во входном файле план является слабо K-связным.
- Ограничение по времени. 1 сек.
- Ограничение по памяти. 16 мб.
- Примеры:

input.txt	output.txt
3 2 1 2 1 3	1

input.txt	output.txt
4 4 2 4 1 3 4 1 3 2	0

- Проверяем **обязательно** – на acmp.

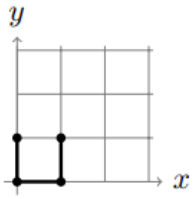
18 Задача*. Построение дорог [10 s, 512 Mb, 4 балла]

В этой задаче цель состоит в том, чтобы построить дороги между некоторыми парами заданных городов так, чтобы между любыми двумя городами существовал путь и общая длина дорог была минимальна.

Даны n точек на плоскости, соедините их отрезками минимальной *общей* длины так, чтобы между любыми двумя точками был путь. Напомним, что длина отрезка с концами (x_1, y_1) и (x_2, y_2) равна $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

- **Формат ввода / входного файла (input.txt).** Первая строка содержит количество n точек на плоскости. Каждая из следующих n строк определяет координаты точки на плоскости (x_i, y_i) через пробел.
- **Ограничения на входные данные.** $1 \le n \le 200$, $-10^3 \le x_i, y_i \le 10^3$ – целые числа. Все точки попарно различны, никакие три точки не лежат на одной прямой.
- **Формат вывода / выходного файла (output.txt).** Выведите минимальную общую длину отрезков. Абсолютная погрешность между ответом вашей программы и оптимальным значением должно быть не более 10^{-6}. Чтобы убедиться в этом, выведите свой ответ как минимум с семью знаками после запятой.
- Ограничение по времени. 10 сек.
- Ограничение по памяти. 512 мб.
- Пример 1:

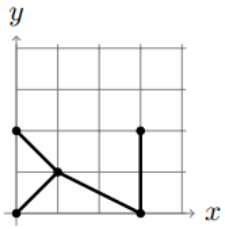
input	output
4 0 0 0 1 1 0 1 1	3.000000000



Оптимальный способ соединения этих четырех точек показан выше. Заметим, что существуют и другие способы соединения этих точек отрезками суммарного веса 3.

- Пример 2:

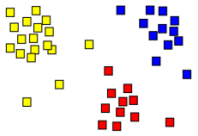
input	output
5 0 0 0 2 1 1 3 0 3 2	7.064495102



Оптимальный способ соединения этих пяти точек показан выше. Общая длина здесь равна $2\sqrt{2} + \sqrt{5} + 2$.

- Что делать? Вам нужен алгоритм Крускала (задача поиска минимального остовного дерева, MST).

19 Задача*. Кластеризация [10 s, 512 Mb, 4 балла]

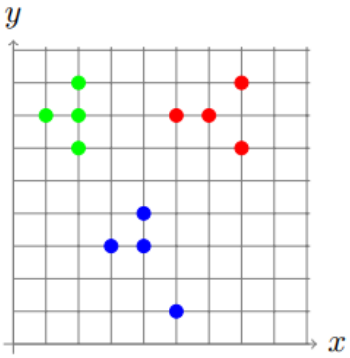


Кластеризация является фундаментальной задачей интеллектуального анализа данных. Цель заключается в том, чтобы разбить данный набор объектов на подмножества (или кластеры) таким образом, чтобы любые два объекта из одного и того же подмножества были близки (или похожи) друг к другу, а любые два объекта из разных подмножеств находились далеко друг от друга.

Даны n точек на плоскости и целое число k, вычислите максимально возможное значение d, чтобы данные точки можно было разбить на непустые подмножества таким образом, чтобы расстояние между любыми двумя точками из разных подмножеств было не менее d.

- **Формат ввода / входного файла (input.txt).** Первая строка содержит число n точек. Далее, следующие n строк определяют координаты точек (x_i, y_i) через пробел. Последняя строка содержит число k – количество кластеров.
- **Ограничения на входные данные.** $2 \le k \le n \le 200$, $-10^3 \le x_i, y_i \le 10^3$ – целые числа. Все точки попарно различны.
- **Формат вывода / выходного файла (output.txt).** Выведите наибольшее значение d. Абсолютная погрешность между ответом вашей программы и оптимальным значением должно быть не более 10^{-6}. Чтобы убедиться в этом, выведите свой ответ как минимум с семью знаками после запятой.
- Ограничение по времени. 10 сек.
- Ограничение по памяти. 512 мб.
- Примеры:

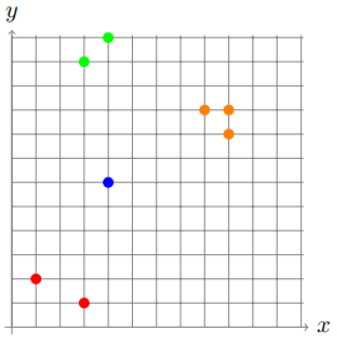
input	output
12 7 6 4 3 5 1 1 7 2 7 5 7 3 3 7 8 2 8 4 4 6 7 2 6 3	2.828427124746



Ответ – $\sqrt{8}$. Соответствующее разбиение набора точек на три кластера показано выше.

-

input	output
8 3 1 1 2 4 6 9 8 9 9 8 9 3 11 4 12 4	5.000000000



Ответ равен 5. Соответствующее разбиение набора точек на 4 кластера показано выше.

- Что делать? Подумайте, как применить алгоритм Крускала для решения этой задачи.