Дисциплина «Алгоритмы решения прикладных задач»
Рабочая тетрадь 5.
Основы теории графов

Теоретический материал

Граф $G = (V, E)$ состоит из множества V, чьи элементы называют **вершинами** графа, и множества E его ребер, соединяющих некоторые пары вершин.

Вершины u и v графа называют **смежными**, если они соединены каким-то ребром e, про которое говорят, что оно инцидентно вершинам u и v.

Степенью вершины v считают число $\delta(v)$ ребер графа, инцидентных v.

Граф, в котором существует маршрут (называемый **эйлеровым**), начинающийся и заканчивающийся в одной и той же вершине и проходящий по каждому ребру графа ровно один раз, называется **Эйлеровым** графом. Связный граф с двумя или более вершинами является эйлеровым тогда и только тогда, когда каждая его вершина имеет четную степень.

Лемма об эстафете утверждает, что сумма степеней вершин произвольного графа $G = (V, E)$ равна удвоенному числу его ребер.

Простым принято называть граф $G = (V, E)$ с конечным множеством вершин V и конечным множеством ребер E, в котором нет петель и кратных ребер.

Логическая матрица отношения на множестве вершин простого графа G, которое задается его ребрами, называется **матрицей смежности**.

Подграфом графа $G = (V, E)$ называют граф $G' = (V', E')$, в котором $E' \subset E$ и $V' \subset V$.

Маршрутом длины k в графе называют такую последовательность различных вершин $v_0, v_1, \ldots, v_k$, в которой каждая пара соседних вершин $v_{i-1}v_i$ соединена ребром.

Циклом в графе является замкнутый маршрут $v_0, v_1, \ldots, v_0$, у которого все вершины, кроме первой и последней, различны.

Граф, не содержащий циклов, называют **ацикличным**.

Связным является тот граф, в котором каждая пара вершин соединена маршрутом.

Количество компонент связности графа можно подсчитать с помощью **алгоритма связности**.

Гамильтоновым называют такой цикл в графе, который проходит через каждую вершину графа, причем только один раз. Граф, в котором существует гамильтонов цикл, называют **гамильтоновым**.

Задача коммивояжера состоит в поиске гамильтонова цикла минимального общего веса в нагруженном графе. **Алгоритм ближайшего соседа** позволяет найти субоптимальное решение задачи коммивояжера.

Связный ацикличный граф $G = (V, E)$ является **деревом**. Следующие утверждения о связном графе $G = (V, E)$ с n вершинами и m ребрами эквивалентны:

(а) G — дерево;

(б) любую пару вершин G связывает единственный путь;

(в) G связен и $m = n - 1$;

(г) G связен, а удаление любого его ребра нарушает это свойство;

(д) G ацикличен, но соединяя любую пару вершин новым ребром, мы получаем цикл.

Остовным деревом графа G называют такой его подграф, который является деревом и содержит все вершины графа G. **Алгоритм поиска минимального остовного дерева** позволяет найти остовное дерево минимального общего веса в нагруженном графе и может быть использован для решения **задачи поиска кратчайшего соединения**.

Алгоритм Прима построения минимального остовного дерева взвешенного связного неориентированного графа

В качестве примера моделирования рассмотрим следующую задачу:

Расстояние (в милях) между шестью шотландскими городами: Абердин, Эдинбург, Форт Уильям, Глазго, Инвернесс и Перт дано в табл. 1.1. Требуется найти дорожную сеть минимальной длины, связывающую все шесть городов.

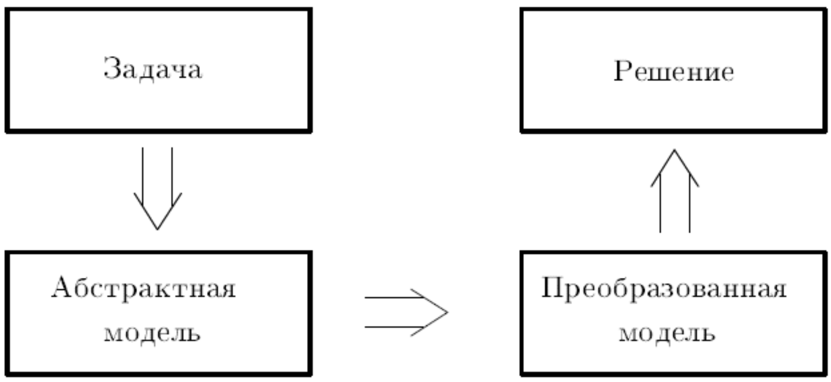


Рисунок 1.1. Схема моделирования

Сама таблица является абстрактной моделью реальной задачи. Однако для нашего решения мы преобразуем ее в геометрическую модель.

Таблица 1.1

	Абердин	Эдинбург	Форт Уильям	Глазго	Инвернесс	Перт
Абердин	—	120	147	142	107	81
Эдинбург	120	—	132	42	157	45
Форт Уильям	147	132	—	108	66	105
Глазго	142	42	108	—	168	61
Инвернесс	107	157	66	168	—	112
Перт	81	45	105	61	112	—

Мы нарисуем *граф* , чьи *вершины* обозначают города, а *ребра* — дороги их связывающие. Более подробно о графах рассказано в главе 7. Каждое ребро нашего графа, изображенного на рис. 1.2, снабжено *весом*, который означает расстояние между соответствующими городами согласно табл. 1.1.

Для решения поставленной задачи с помощью подходящего *алгоритма* (последовательности однозначных инструкций, выполнение которых влечет решение за конечное время), мы построим новый граф, имеющий минимальный общий вес, в котором все шесть городов будут соединены дорогами.

Алгоритм Прима

Шаг 1 Выберите произвольную вершину и ребро, соединяющее ее с ближайшим (по весу) соседом.

Шаг 2 Найдите не присоединенную (еще) вершину, ближе всего лежащую к одной из присоединенных, и соедините с ней.

Шаг 3 Повторяйте шаг 2 до тех пор пока все вершины не будут присоединены.

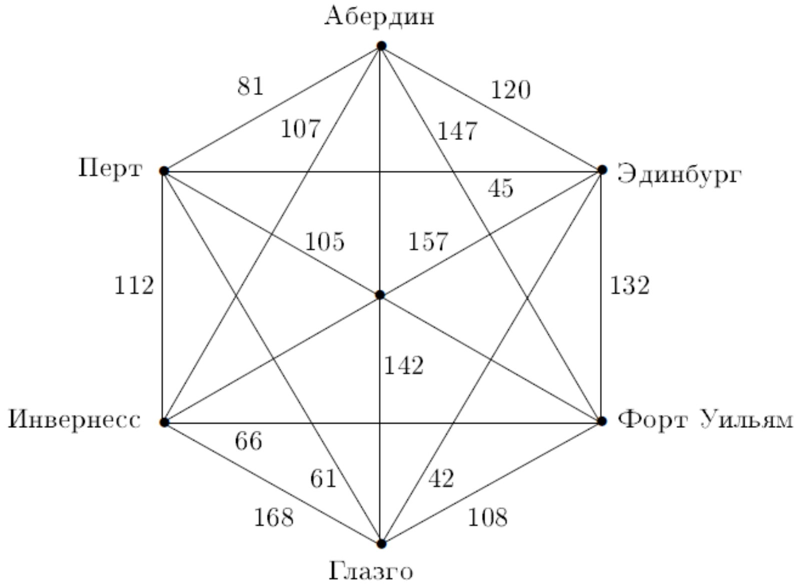


Рисунок 1.2.

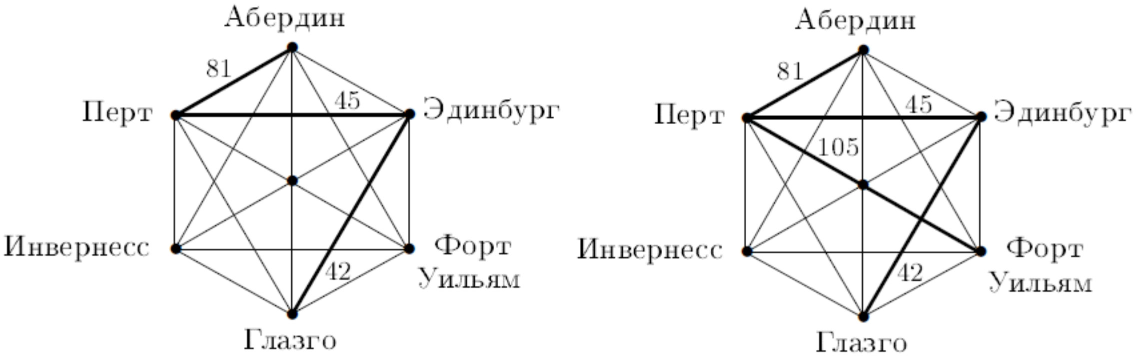
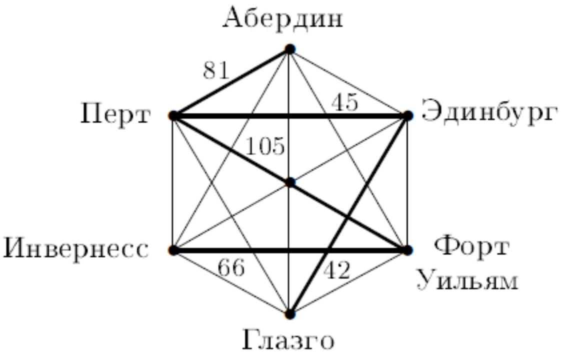


Рисунок 1.4.



На рисунках 1.3, 1.4 и 1.5 изображена последовательность графов, которая получается в результате применения алгоритма Прима, если начинать с вершины Перт. Последний граф (с общим весом 339) представляет собой минимальную сеть дорог, охватывающую все шесть городов.

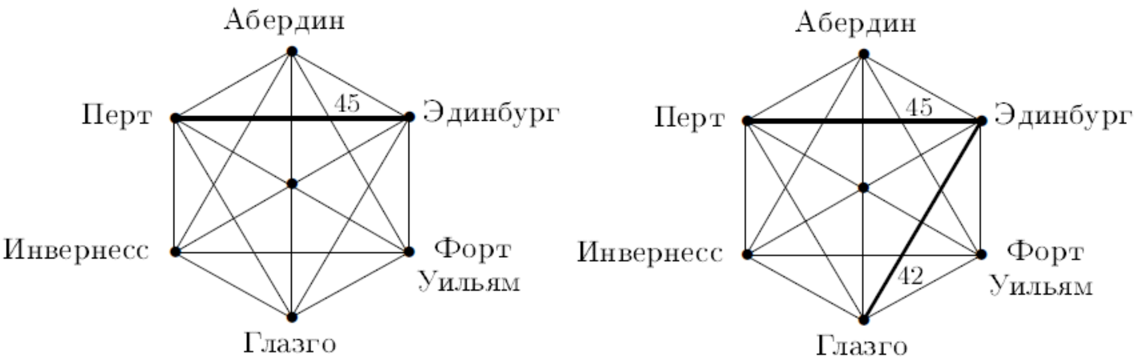


Рисунок 1.3.

Псевдокод алгоритма выглядит так:

```
begin
    v := произвольная вершина;
    u := ближайшая соседняя вершина;
    связать v и u;
    while остаются неприсоединенные вершины do
        begin
            u := неприсоединенная вершина, ближайшая
                к одной из присоединенных вершин;
            соединить u с ближайшей
            из присоединенных вершин;
        end
end
```

Хранение графа в памяти компьютера

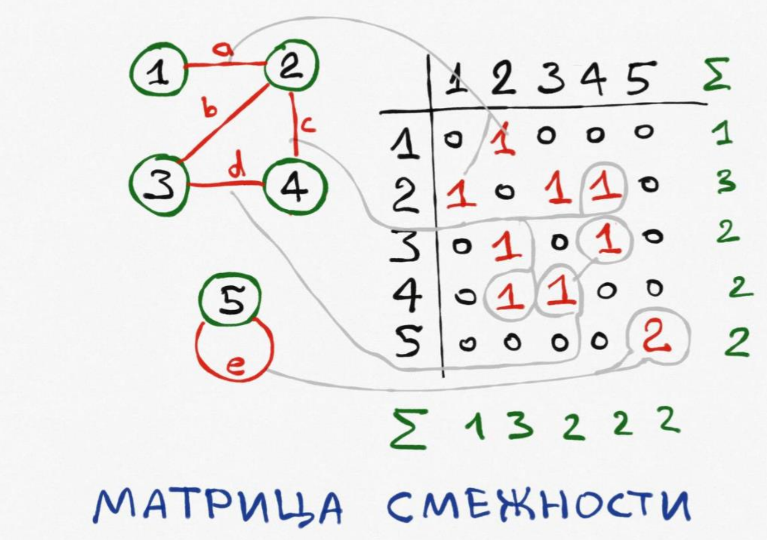
Для хранения графа в памяти компьютера используются различные подходы. Наиболее распространенные – матрица смежности и матрица инцидентности

1. *Матрица смежности*

Это самый популярный и расточительный способ представления графа в памяти. Его уместно использовать, если количество рёбер велико, порядка V2.

Для хранения рёбер используется двумерная матрица размерности [V, V], каждый [a, b] элемент которой равен 1, если вершины a и b являются смежными и 0 в противном случае.

В случае неориентированного графа матрица является симметричной относительно главной диагонали, а сумма каждой строчки и каждого столбца равна степени вершины. В связи с этим, при записи рёбер-петель в матрицу необходимо записывать число 2.



- ✓ Сложность по памяти: O(V2).
- ✓ Сложность перечисления всех рёбер: O(V2).
- ✓ Сложность перечисления всех вершин, смежных с а: O(V).
- ✓ Сложность проверки смежности вершин а и b: O(1).

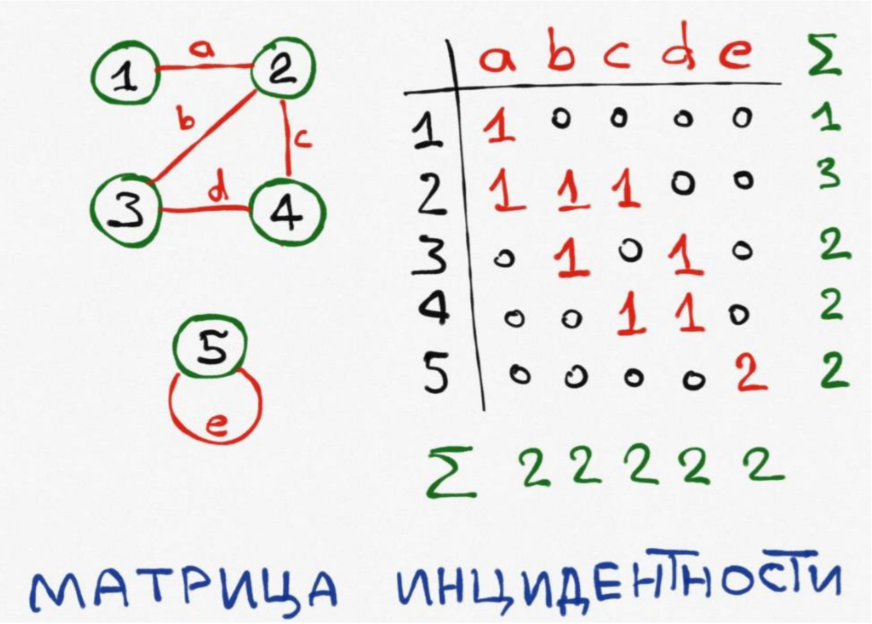
Если граф взвешенный, то элементом матрицы смежности является вес ребра (вместо единицы пишется вес соответствующего ребра)

2. *Матрица инцидентности*

Это самый расточительный способ хранения графа, его уместно использовать, если количество рёбер невелико.

Для хранения используется двумерная матрица размера [V, E], в каждом столбце которой записано одно ребро таким образом: напротив вершин, инцидентных этому ребру, записаны 1, в остальных случаях 0.

Таким образом, сумма чисел в каждом столбце равна 2, а сумма чисел в строчке а равна степени вершины а.



- ✓ Сложность по памяти: O(V x E).
- ✓ Сложность перечисления всех рёбер: O(V x E) - хоть каждое ребро и хранится в отдельном столбце, но для получения информации об инцидентных ему вершинах нужно перебрать все числа в столбце.
- ✓ Сложность перечисления всех вершин, смежных с а: O(V x E).
- ✓ Сложность проверки смежности вершин а и b: O(E) - достаточно пройтись по строчкам а и b в поисках двух единиц.

Алгоритм связности

Граф называют *связным*, если любую пару его вершин соединяет какой-нибудь маршрут. Любой общий граф можно разбить на подграфы, каждый из которых окажется связным. Минимальное число таких связных компонент называется *числом связности* графа и обозначается через $c(G)$. Вопросы связности имеют важное значение в приложениях теории графов к компьютерным сетям. Следующий алгоритм применяется для определения числа связности графа.

Алгоритм связности.

Пусть $G = (V, E)$ — граф. Алгоритм предназначен для вычисления значения $c = c(G)$, т. е. числа компонент связности данного графа G.

begin
    $V' := V$;
    $c := 0$;
    while $V' \neq \varnothing$ **do**
        begin
            Выбрать $y \in V'$;
            Найти все вершины, соединенные маршрутом с y;
            Удалить вершину y *из* V' *и*
            соответствующие ребра из E;
            $c := c + 1$;
        end
end

Пример 7.4. Проследите за работой алгоритма связности на графе, изображенном на рис. 7.6.

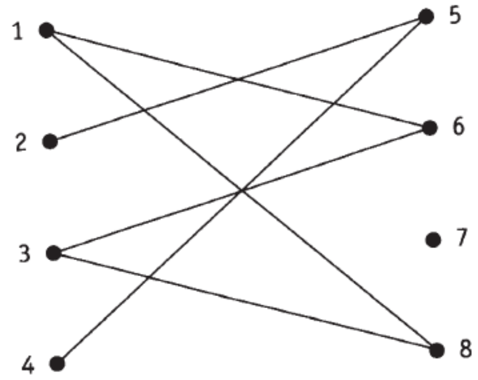


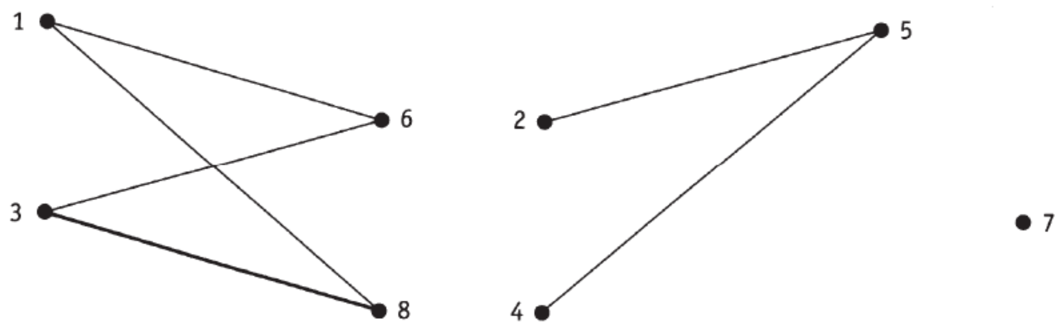
Рисунок 7.6.

Решение. Смотри табл. 7.1.

Таблица 7.1

	V'	c
Исходные значения	$\{1, 2, 3, 4, 5, 6, 7, 8\}$	0
Выбор $y = 1$	$\{2, 4, 5, 7\}$	1
Выбор $y = 2$	$\{7\}$	2
Выбор $y = 7$	$\varnothing$	3

Итак, $c(G) = 3$. Соответствующие компоненты связности приведены на рис. 7.7.



Алгоритм ближайшего соседа для решения задачи коммивояжера

Если в графе существует замкнутый маршрут, проходящий ровно один раз через каждую вершину графа, то такой граф называется гамильтоновым, а маршрут – гамильтоновым циклом

Коммивояжер должен совершить поездку по городам и вернуться обратно, побывав в каждом городе ровно один раз, сведя при этом затраты на передвижения к минимуму.

Графическая модель задачи коммивояжера состоит из гамильтонова графа, вершины которого изображают города, а ребра — связывающие их дороги. Кроме того, каждое ребро оснащено *весом*, обозначающим транспортные затраты, необходимые для путешествия по соответствующей дороге, такие, как, например, расстояние между городами или время движения по дороге[2]. Для решения задачи нам необходимо найти гамильтонов цикл минимального общего веса.

К сожалению, эффективный алгоритм решения данной задачи пока не известен. Для сложных сетей число гамильтоновых циклов, которые необходимо просмотреть для выделения минимального, непомерно огромно. Однако существуют алгоритмы поиска *субоптимального решения*. Субоптимальное решение необязательно даст цикл минимального общего веса, но найденный цикл будет, как правило, значительно меньшего веса, чем большинство произвольных гамильтоновых циклов! Один из таких алгоритмов мы сейчас и изучим.

Алгоритм ближайшего соседа.

Этот алгоритм выдает субоптимальное решение задачи коммивояжера, генерируя гамильтоновы циклы в нагруженном графе с множеством вершин V. Цикл, полученный в результате работы алгоритма, будет совпадать с конечным значением переменной *маршрут*, а его общая длина — конечное значение переменной w.

```
begin
    Выбрать v ∈ V;
    маршрут := v;
    w := 0;
    v' := v;
    Отметить v';
    while остаются неотмеченные вершины do
        begin
            Выбрать неотмеченную вершину u,
            ближайшую к v';
            маршрут := маршрут u;
            w := w + вес ребра v'u;
            v' := u;
            Отметить v';
        end
    маршрут := маршрут v;
    w := w + вес ребра v'v;
end
```

Пример 7.6. Примените алгоритм ближайшего соседа к графу, изображенному на рис. 7.11. За исходную вершину возьмите вершину D.

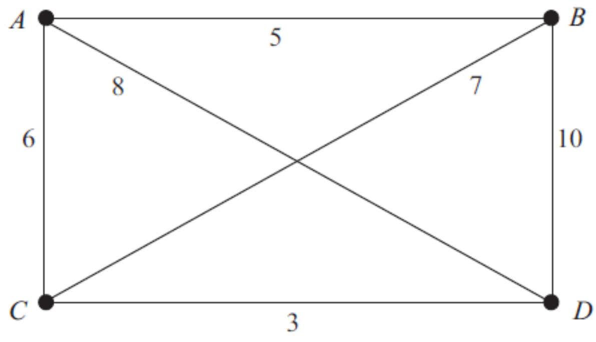


Рисунок 7.11.

Решение. Смотри табл. 7.2.

Таблица 7.2

	u	*маршрут*	w	v'
Исходные значения	—	D	0	D
	C	DC	3	C
	A	DCA	9	A
	B	$DCAB$	14	B
Последний проход	B	$DCABD$	24	B

В результате работы алгоритма был найден гамильтонов цикл $DCABD$ общего веса 24. Делая полный перебор всех циклов в этом маленьком графе, можно обнаружить еще два других гамильтоновых цикла: $ABCDA$ общего веса 23 и $ACBDA$ общего веса 31. В полном графе с двадцатью вершинами существует приблизительно $6,1 \cdot 10^{16}$ гамильтоновых циклов, перечисление которых требует чрезвычайно много машинной памяти и времени.

Задание 1

В таблице приведено расстояние между кампусами РТУ МИРЭА

	Пр-т Вернадского, 78	Пр-т Вернадского, 86	Стромынка, 20	Малая Пироговская, 1	5-я улица Соколиной горы, 22	Усачева, 7
Пр-т Вернадского, 78	-	1.97	21.6	10.7	22.3	10.4
Пр-т Вернадского, 86	1.97	-	22.3	11.4	23	11.1
Стромынка, 20	21.6	22.3	-	11.5	5.2	12
Малая Пироговская, 1	10.7	1.4	11.5	-	13.4	0.68
5-я улица Соколиной горы, 22	22.3	23	5.2	13.4	-	13.8
Усачева, 7	10.4	11.1	12	0.68	13.8	-

С помощью алгоритма Прима выделить маршрут с минимальной общей длиной, связывающий все кампусы РТУ МИРЭА

Решение:

Ответ:

Задание 2

a) Задать в программе граф из не менее 7 вершин с помощью матрицы смежности. Определить для этого графа число связности (с помощью алгоритма связности)

b) Задать в программе граф из не менее 7 вершин с помощью матрицы инцидентности. Определить для этого графа число связности (с помощью алгоритма связности)

Решение:

Ответ:

Задание 3

Задача:

Задать в программе взвешенный граф с помощью матрицы смежности (число вершин не менее 7, вес в диапазоне от 0 до 10). Применить к графу алгоритм ближайшего соседа для поиска субоптимального решения задачи коммивояжера.

Решение:

Ответ:

Задание 4*

Задача:

Создать класс для работы с графами. Наполнить класс методами для печати структуры графа, добавления и удаления ребер, сохранения графа в текстовом файле в виде матрицы смежности. Реализовать конструкторы:

А) Вершины графа идентифицируются целыми числами от 0 до N-1, а конфигурация графа задана в текстовом файле в виде последовательности строк:

<количество вершин>

<номер вершины> <номер соседа>…<номер соседа>

…

Б) Вершины графа идентифицируются целыми числами от 0 до N-1, вес ребра есть вещественное число, а конфигурация графа задана в текстовом файле в виде последовательности строк:

<количество вершин>

<номер вершины> <номер соседа> <вес ребра>

…

В) Вершины графа идентифицируются целыми числами от 0 до N-1, вес ребра есть вещественное число, а конфигурация графа задана в текстовом файле в виде строк матрицы смежности:

<количество вершин>

<вес ребра> … <вес ребра>

…

Решение:

Ответ: