Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Информатика»

Т. В. Тихоненко, М. В. Задорожнюк

МЕТОДЫ И АЛГОРИТМЫ ТЕОРИИ ГРАФОВ

ПРАКТИКУМ
по дисциплине «Дискретная математика
и математическая логика»
для студентов специальности
1-40 04 01 «Информатика
и технологии программирования»
дневной формы обучения

Электронный аналог печатного издания

Гомель 2019

УДК 519.1(075.8)
ББК 22.176я73
Т46

Рекомендовано к изданию научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 11 от 04.06.2018 г.*)

Рецензент: доц. каф. информационно-вычислительных систем УО БТЭУ ПК
канд. физ.-мат. наук, доц. *Л. А. Воробей*

Тихоненко, Т. В.

Т46 Методы и алгоритмы теории графов : практикум по дисциплине «Дискретная математика и математическая логика» для студентов специальности 1-40 04 01 «Информатика и технологии программирования» днев. формы обучения / Т. В. Тихоненко, М. В. Задорожнюк. – Гомель : ГГТУ им. П. О. Сухого, 2019. – 52 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

ISBN 978-985-535-413-1.

Кратко изложен основной теоретический материал по теории графов и ее приложениям. Приведены примеры, иллюстрирующие основные положения, формулы и определения, а также алгоритмы решения наиболее часто встречающихся на практике оптимизационных задач на графах. Содержит задания для практических занятий и 30 вариантов заданий для индивидуальной работы студентов.

Для студентов специальности 1-40 04 01 «Информатика и технологии программирования» дневной формы обучения.

УДК 519.1(075.8)
ББК 22.176я73

ISBN 978-985-535-413-1

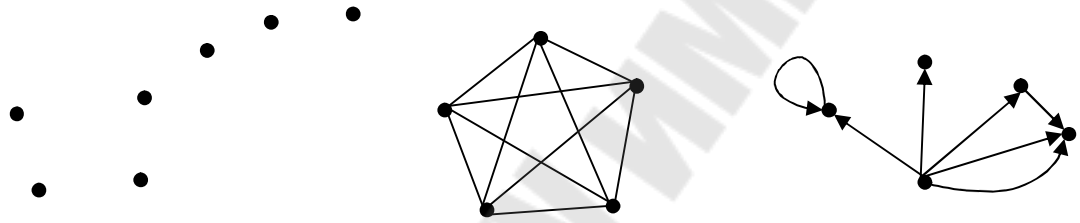
СОДЕРЖАНИЕ

1. ОСНОВНЫЕ ПОНЯТИЯ ТЕОРИИ ГРАФОВ

1.1. Ориентированный и неориентированный графы. Виды графов

Определение 1.1. *Граф* представляет собой непустое конечное множество вершин V и набор E неупорядоченных или упорядоченных пар вершин; обозначается граф через G, (V, E). Неупорядоченная пара вершин называется *ребром*, упорядоченная – *дугой*. Граф, содержащий только ребра, называется *неориентированным* (неографом); граф, содержащий только дуги, – *ориентированным* (орграфом).

Графы удобно изображать в виде рисунков, состоящих из точек и линий, соединяющих некоторые из этих точек.



а) *б*) *в*)

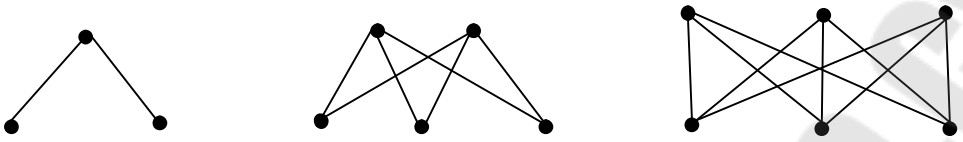
Рис. 1

Пара вершин может соединяться двумя или более ребрами (дугами одного направления). Такие ребра (дуги) называют *кратными*. Если ребро (дуга) начинается и заканчивается в одной и той же точке, то оно называется *петлей*. Граф, в котором допускаются кратные ребра, назовем *мультиграфом*, а в котором допускаются кратные ребра и петли – *псевдографом* (рис. 1, *в*). Далее под графом будем понимать граф без петель и кратных ребер (такой граф называют *скелетным* графом).

Вершины, соединенные ребром или дугой, называются *смежными*. Ребра, имеющие общую вершину, также называются *смежными*. Ребро (дуга) и любая из двух его вершин называются *инцидентными*. Вершина, не принадлежащая ни одному ребру, называется *изолированной*. Граф, состоящий только из изолированных вершин, называют *пустым*, или *нуль-графом* (рис. 1, *а*).

Граф называется *полным*, если любые две его вершины соединены одним и только одним ребром (рис. 1, *б*). Полный граф с n вершинами обозначается через K_n. Если множество V вершин графа можно

разбить на два подмножества V_1 и V_2 так, что в каждом подмножестве нет смежных вершин, а любые две вершины $u \in V_1$ и $v \in V_2$ смежны, то граф называется *полным двудольным* и обозначается $K_{m,n}$, где m и n – число элементов множеств V_1 и V_2 соответственно (рис. 2).



$K_{1,2}$ $K_{2,3}$ $K_{3,3}$

Рис. 2

1.2. Степени вершин графа

Определение 1.2. *Степенью* вершины v (обозначается $dg(v)$) в неориентированном графе называется количество ребер, инцидентных этой вершине. Вершина v называется четной, если $dg(v)$ – четное число, и нечетной, если $dg(v)$ – нечетное число. Граф называется *однородным* (*регулярным*) *степени* n, если степень любой его вершины равна n. В частности, любой полный граф является регулярным. В ориентированном графе различают полустепень захода и полустепень выхода. *Полустепенью захода* вершины v в ориентированном графе называют число $dg^-(v)$ заходящих в нее дуг, а *полустепенью выхода* вершины v – число $dg^+(v)$ исходящих из нее дуг. *Степень вершины* определяется как сумма полустепеней захода и выхода:

$$dg(v) = dg^+(v) + dg^-(v).$$

У регулярных орграфов полустепени выходов и полустепени заходов у всех вершин одинаковы.

Теорема 1.1 («*лемма о рукопожатиях*»). В графе сумма степеней всех вершин есть число четное, равное удвоенному количеству ребер графа.

Следствие. Количество нечетных вершин графа четно.

Теорема 1.2. В любом графе с n вершинами ($n \geq 2$) всегда найдутся по меньшей мере две вершины с одинаковыми степенями.

Теорема 1.3. Если в графе с n вершинами ($n \geq 3$) в точности две вершины имеют одинаковую степень, то в этом графе всегда найдется в точности одна вершина степени 0 либо в точности одна вершина степени $n-1$.

1.3. Способы задания графов

До сих пор мы задавали графы с помощью рисунков. Однако более распространенными способами описания графа, пригодными для практических вычислений, являются матрицы.

Граф (неориентированный или ориентированный) может быть представлен в виде матрицы B размерности $n \times m$, где n – число вершин, m – число ребер (или дуг). Для неориентированного графа элементы этой матрицы задаются следующим образом:

$$b_{ij} = \begin{cases} 1, \textit{ если i-я вершина инцидентна j-му ребру}, \\ 0, \textit{ иначе.} \end{cases}$$

Для ориентированного графа элементы матрицы задаются так:

$$b_{ij} = \begin{cases} 1, \textit{ если для i-й вершины j-я дуга выходящая}, \\ -1, \textit{ если для i-й вершины j-я дуга заходящая}, \\ 0, \textit{ иначе.} \end{cases}$$

Матрицу (b_{ij}), определенную указанным образом, называют *матрицей инцидентности* (или *матрицей инциденций*).

Пример 1.1. Для неориентированного графа, изображенного на рис. 3 и орграфа на рис. 4, матрицы инцидентности будут иметь вид $\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ и $\begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix}$ соответственно.

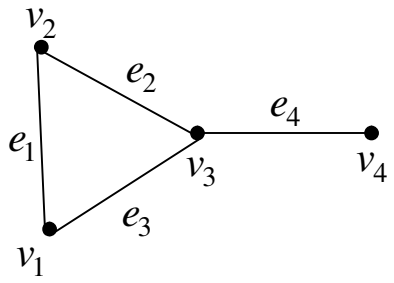


Рис. 3

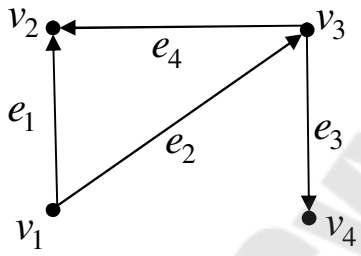


Рис. 4

Если граф содержит петли, то элементы матрицы инцидентности, соответствующие дугам, образующим петли, одновременно равны 1 и –1, что приводит к неоднозначности матрицы инцидентности. Кроме того, в каждом столбце матрицы находятся только два ненулевых элемента, что делает такой способ представления неэкономным при большом количестве вершин.

Другой матричной структурой, представляющей граф, является *матрица смежности* вершин. Это квадратная матрица A порядка n, элементы которой определяют следующим образом:

для неориентированного графа

$$a_{ij} = \begin{cases} 1, \textit{ если } i\textit{-я и } j\textit{-я вершины смежные}, \\ 0, \textit{ иначе}; \end{cases}$$

для ориентированного графа

$$a_{ij} = \begin{cases} 1, \textit{ если из } i\textit{-й вершины в } j\textit{-ю ведет дуга}, \\ 0, \textit{ иначе}. \end{cases}$$

Пример 1.2. Матрицы смежности графов, изображенных на рис. 3 и 4, имеют вид $\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ и $\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, соответственно.

Матрица смежности является достаточно эффективным способом представления графа, но эту матрицу удобно строить по графу, уже заданному каким-либо образом, например, рисунком. Во многих задачах граф создается динамически, т. е. в процессе решения меняется множество вершин и множество ребер. В этом случае эффектив-

ным способом машинного представления графа являются *списки смежности*. Задать для любой вершины v ее список смежности $L(v)$ означает поместить в произвольном порядке в данные элементов списка номера тех вершин u, для которых в графах есть дуга (ребро) из v в u. Вершину u в этом случае называют образом вершины v, а вершину v – прообразом вершины u.

Пример 1.3. Граф на рис. 3 может быть задан списками смежности $L(v_1)=\{v_2, v_3\}$, $L(v_2)=\{v_1, v_3\}$, $L(v_3)=\{v_1, v_4, v_2\}$, $L(v_4)=\{v_3\}$. Для графа на рис. 4 списки смежности имеют вид $L(v_1)=\{v_2, v_3\}$, $L(v_2)=\varnothing$, $L(v_3)=\{v_2, v_4\}$, $L(v_4)=\varnothing$.

1.4. Части графа. Операции над графами

Язык теории множеств очень удобен как для выделения частей графа, так и для ввода операций над ними.

Определение 1.3. Граф $G'(V', E')$ называется *подграфом* графа $G(V, E)$, если $V' \subseteq V$, $E' \subseteq E$ и множество ребер E' графа G' образовано теми и только теми ребрами графа G, обе концевые вершины каждого из которых принадлежат множеству V'. Подграф G' называется *собственным*, если он отличен от самого графа G. Подграф G' называется *остовным*, если $V' = V$, $E' \subseteq E$.

На рис. 5 изображен граф G, его собственный подграф G' и остовной подграф G''.

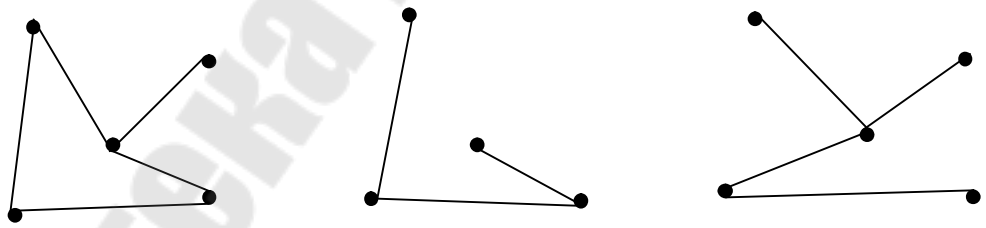


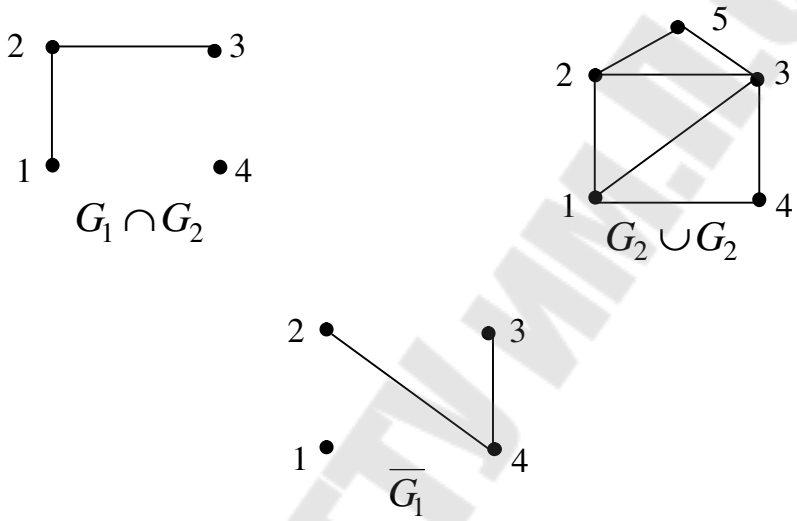
Рис. 5

На графах выполняются основные теоретико-множественные операции: объединение, пересечение, разность, дополнение, разностная сумма.

Пример 1.4. Для графов G_1 и G_2, изображенных на рис. 6, построить пересечение $G_1 \cap G_2$, объединение $G_1 \cup G_2$, дополнение $\overline{G_1}$.

Рис. 6

Решение



На графах определяют и некоторые другие операции, используемые при разработке алгоритмов: композицию, удаление вершины, удаление ребра, стягивание ребра. Так, операция *удаления вершины* v из графа G приводит к построению графа $G'(V', E') = G \setminus \{v\}$, где $V' = V \setminus \{v\}$, а множество ребер E содержит те и только те ребра, которые не инцидентны вершине v.

Операция *стягивания ребра* состоит в удалении ребра между вершинами $x, y \in V$ и отождествлении этих вершин в одну, например, в вершину v; множество ребер, инцидентных вершинам x и y, в новом графе полагают инцидентными вершине v; остальные вершины и ребра в результирующем графе остаются такими же, как в исходном графе.

1.5. Связность графов

Определение 1.4. *Маршрутом* M в графе G называется чередующаяся последовательность вершин и ребер

$$M = \{v_1, e_1, v_2, e_2, \text{K}, v_k, e_k, v_{k+1}\}$$

такая, что $e_i = (v_i, v_{i+1})$. Иногда маршрут задают либо только последовательностью составляющих его ребер, либо только последовательностью вершин, через которые он проходит. Одно и то же ребро может встречаться в маршруте несколько раз.

Длиной маршрута называется число ребер, входящих в маршрут, причем каждое ребро считается столько раз, сколько оно входит в данный маршрут.

Если в маршруте нет совпадающих ребер, то он называется *цепью*, а если различны как ребра, так и вершины, то *простой цепью*. Замкнутая цепь ненулевой длины, в которой совпадают начальная и конечная вершины, называется *циклом.* В ориентированном графе цепь называют также *путем*, а цикл – *контуром*.

Например, для графа, изображенного на рис. 7, маршрут $M = \{e_1, e_4, e_2, e_3, e_5, e_6\}$ является цепью, маршрут $M_1 = \{e_1, e_4, e_6\}$ – простой цепью, а $M_2 = \{e_1, e_4, e_5, e_3\}$ – циклом.

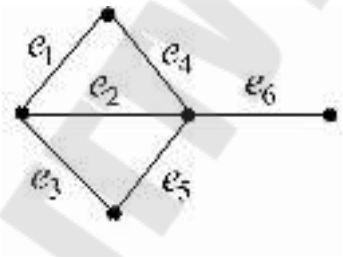
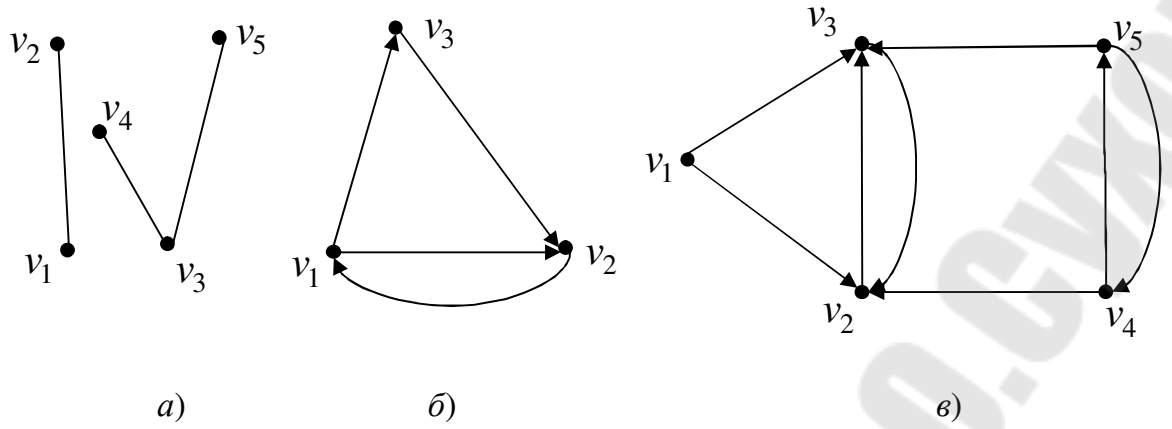


Рис. 7

Определение 1.5. Граф $G(V, E)$ называется *связным*, если между любыми двумя его различными вершинами существует маршрут. *Компонентой связности* графа G называется любой его максимальный связный подграф, т. е. подграф, который сам является связным и не содержится ни в каком связном собственном подграфе графа G.

Компоненты неориентированного графа не пересекаются, а компоненты орграфа могут пересекаться.

Пример 1.5. На рис. 8, *а* изображен несвязный неориентированный граф, имеющий две компоненты связности $K_1 = \{v_1, v_2\}$ и $K_2 = \{v_3, v_4, v_5\}$, $K_1 \cap K_2 = \varnothing$.



а) *б*) *в*)

Рис. 8

Граф, изображенный на рис. 8, *б*, является связным орграфом и состоит из одной компоненты $K = \{v_1, v_2, v_3\}$. Ориентированный граф на рис. 8, *в* не является связным, так как из вершины v_1 нельзя попасть, например, в вершину v_4. Этот граф имеет две компоненты связности, $K_1 = \{v_1, v_2, v_3\}$ и $K_2 = \{v_2, v_3, v_4, v_5\}$, причем эти компоненты пересекаются, $K_1 \cap K_2 = \{v_2, v_3\}$.

Интересна следующая задача: сколько ребер может иметь k-связный граф (т. е. граф, имеющий k-связных компонент) порядка n?

Теорема 1.4. Если $k(G) = k$, то для n-вершинного графа G число ребер m удовлетворяет неравенству

$$n - k \le m(G) \le \frac{(n-k)(n-k+1)}{2}.$$

Ребро (v_i, v_j) называется *мостом* графа G, если граф, получившийся из G после удаления этого ребра, содержит больше компонент связности, чем граф G. Например, любое ребро графа, изображенного на рис. 8, *а*, является мостом. Граф на рис. 8, *б* мостов не имеет.

Для ориентированного графа можно определить понятия сильной и слабой связности.

Определение 1.6. Ориентированный граф G называется *сильно связным*, если между любыми двумя различными вершинами u и v существуют пути из u в v и из v в u. *Бикомпонентой* орграфа G называется его максимальный сильно связный подграф.

Определение 1.7. Неориентированный граф $G'(V', E')$ называется *ассоциированным* с орграфом $G(V, E)$, если $V' = V$, а пара (u, v)

образует ребро графа G' тогда и только тогда, когда $u \neq v$ и в графе G существует дуга (u, v) или (v, u).

Определение 1.8. Орграф называется *слабо связным*, если ассоциированный с ним граф связный. Так, граф на рис. 8, *б* является сильно связным, а граф на рис. 8, *в* – слабо связным. Он содержит три бикомпоненты: $B_1 = \{v_1\}$, $B_2 = \{v_2, v_3\}$ и $B_3 = \{v_4, v_5\}$.

1.6. Деревья

Определение 1.9. *Неориентированным деревом* называется связный граф, не имеющий циклов. Известно, что на n вершинах можно построить n^{n-2} различных вариантов деревьев.

Свойства деревьев:

1. Дерево с n вершинами содержит $n-1$ ребро.

2. Каждое ребро дерева является мостом, т. е. дерево – минимально связный граф.

3. Любые две вершины дерева можно соединить цепью, причем эта цепь единственна.

4. Добавление в дерево ребра, инцидентного двум различным вершинам, ведет к образованию цикла, причем этот цикл единственен и обязательно содержит добавленное ребро.

Ориентированным (*корневым*) *деревом* называется бесконтурный граф, у которого полустепень захода любой вершины не больше 1 и существует ровно одна вершина, называемая *корнем*, полустепень захода которой равна нулю. Вершины, полустепень выхода которых равна 0, называют *листьями*.

Ориентированное дерево называется *бинарным*, если полустепень выхода любой его вершины не больше 2.

Граф на рис. 9 является неориентированным деревом, граф на рис. 10 – корневое бинарное дерево, вершина 1 – корень, вершины 5, 8, 10, 11 – листья.

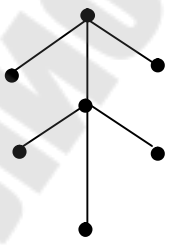


Рис. 9

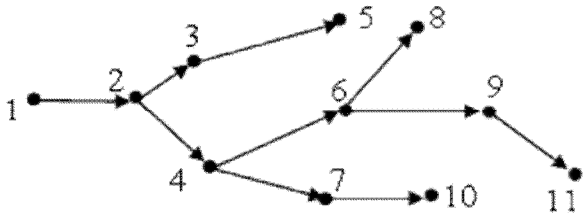


Рис. 10

При исследовании корневых деревьев используют следующие характеристики вершин: глубина, высота и уровень вершины.

Глубина вершины определяется длиной максимального пути, идущего от корня до данной вершины. Например, глубина вершины 6 на рис. 10 равна 3.

Высота вершины определяется длиной максимального пути от данной вершины до одного их листьев этого дерева. Для графа на рисунке 10 высота вершины 6 равна 2. Высота всего дерева определяется высотой его корня (на рис. 10 изображено дерево высоты 5).

Уровень вершины определяется как разность между высотой дерева и глубиной этой вершины (уровень вершины 6 равен 2).

1.7. Изоморфизм графов

Определение 1.10. Два графа $G_1(V_1, E_1)$ и $G_2(V_2, E_2)$ называются *изоморфными* (т. е. тождественными по структуре), если задано такое взаимно однозначное отображение $f : V_1 \to V_2$, устанавливающее соответствие между вершинами первого графа V_1 и вершинами второго графа V_2, при котором любые две вершины v_i и v_j смежны в графе G_1 тогда и только тогда, когда смежны их образы $f(v_i)$ и $f(v_j)$ в графе G_2. Обозначается $G_1 \cong G_2$.

Рассмотрим необходимые условия, которые должны выполняться для графов G_1 и G_2 прежде, чем приступить к поиску отображения.

Очевидно, что, во-первых, оба графа должны содержать одинаковое количество вершин, во-вторых, одинаковое количество ребер (дуг), и, в-третьих, одинаковое количество вершин, имеющих равные степени (равные полустепени выходов и заходов, если графы ориентированные).

Выясним, изоморфны ли графы G_1 и G_2 на рис. 11.

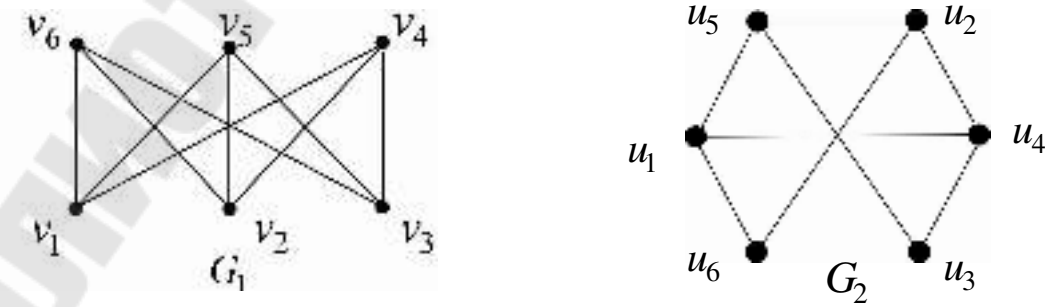


Рис. 11

Они имеют одинаковое количество вершин – 6 и одинаковое количество ребер – 9, степени всех вершин в этих графах одинаковы и равны 3. Следовательно, необходимые условия изоморфизма графов G_1 и G_2 выполняются.

Найдем отображение f, устанавливающее соответствие между множествами вершинам этих графов V и U. Если пронумеровать вершины графа G_2 так, как указано на рис. 11, то отображение $f: v_i \to u_i$ переводит смежные вершины графа G_1 в смежные вершины графа G_2, т. е. является искомым изоморфизмом. Таким образом, $G_1 \cong G_2$.

1.8. Плоские и планарные графы

Определение 1.11. *Плоским* называется граф, который изображен на плоскости без пересечения и самопересечения его ребер (дуг). Если граф изображен на плоскости с пересечением и (или) самопересечением ребер (дуг), но существует плоский изоморфный ему граф, то такой граф называется *планарным*. На рис. 12 изображен полный планарный граф K_4 и его плоское представление.

Рис. 12

При решении задачи плоской укладки графа используется понятие грани. *Гранью* плоского графа называется максимальный участок плоскости, такой, что любые две точки этого участка могут быть соединены кривой, не пересекающей ребро графа. Внешняя часть графа также является гранью (ее называют *бесконечной* гранью). Граф на рис. 12 имеет 4 грани (с учетом бесконечной).

Теорема 1.5 (формула Эйлера). Если G – связный плоский граф, содержащий B вершин, P ребер и Γ граней, то справедливо равенство: $B - P + \Gamma = 2$.

Следствие. Для связного планарного графа с количеством вершин $B \geq 3$ справедливо неравенство: $P \leq 3B - 6$.

Теорема 1.6 (*критерий планарности Понтрягина–Куратовского*). Неориентированный граф G планарен тогда и только тогда, когда в нем нет подграфов, изоморфных графам $K_{3,3}$ или K_5.

1.9. Экстремальные числа графа

Экстремальное число графа – это оценка решения некоторой оптимизационной задачи для данного графа. Рассмотрим подробнее некоторые из этих чисел, наиболее часто встречающиеся на практике.

Цикломатическое число графа $\sigma(G)$ указывает на количество ребер, которое надо удалить из графа, чтобы получить его остовное дерево (остовный лес). Для графа с n вершинами, m ребрами и k компонентами связности оно определяется формулой

$$\sigma(G) = m - n + k .$$

Например, цикломатическое число графа K_4, изображенного на рис. 13, равно 3, так как для того, чтобы получить остовное дерево, состоящее из выделенных ребер, необходимо удалить ребра (1–2), (2–3) и (2–4).

Теорема 1.7 (об основном свойстве цикломатического числа). Цикломатическое число графа определяет максимальное количество независимых циклов в нем.

Число внутренней устойчивости графа α_0 (число независимости графа) – это максимальное количество несмежных между собой вершин в графе.

Для всякого полного графа $\alpha_0 = 1$, для пустого графа с n вершинами $\alpha_0 = n$. Для графа, изображенного на рис. 14, $\alpha_0 = 4$ (вершины 1, 5, 2, 6 несмежны между собой).

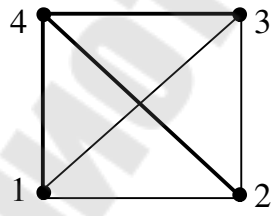


Рис. 13

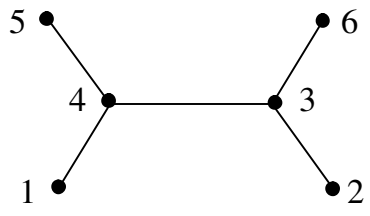


Рис. 14

Число внутренней устойчивости используется при решении задачи поиска наибольших пустых подграфов в графе.

Кликовое число графа φ – максимальное количество вершин в графе, смежных между собой. Это число связано с оптимизационной задачей поиска наибольших полных подграфов в графе.

Клика – это максимальный полный подграф в графе, т. е. такой полный подграф, который нельзя расширить за счет других вершин без нарушения свойства его полноты. Количество вершин в клике с наибольшим количеством вершин и определяет число φ.

Граф на рис. 15 содержит две клики: первая – это полный подграф, порожденный вершинами 3, 4, 5, вторая – полный подграф на четырех вершинах 1, 2, 3, 4. Так как наибольший полный подграф содержит четыре вершины, то кликовое число $\varphi = 4$.

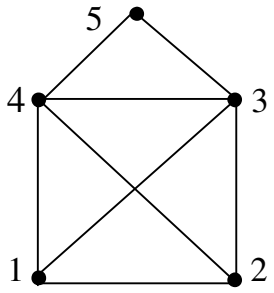


Рис. 15

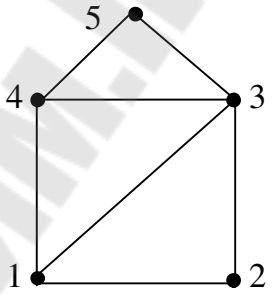


Рис. 16

Хроматическое число графа χ – это минимальное количество цветов для вершинной раскраски графа.

Вершинная раскраска графа – это операция, в результате выполнения которой любые две смежные вершины в графе раскрашиваются в разные цвета. Очевидно, что для полного графа с n вершинами $\chi = n$, для пустого – $\chi = 1$. Хроматическое число графа на рис. 16 равно 3 (окрасим вершины 1 и 5 в красный цвет, вершины 4 и 2 – в синий, вершину 3 – в желтый).

Свойства хроматического числа:

1. Для любого связного графа с n вершинами справедлива оценка:

$$\chi(G) \le dg^*(G) + 1,$$

где $dg^*(G) = \max\limits_{1 \le i \le n} \{dg(v_i)\}$ – максимальная степень у вершин этого графа. В случае полного графа неравенство превращается в равенство.

2. Хроматическое число и число внутренней устойчивости графа связаны соотношением

$$\alpha_0(G) \cdot \chi(G) \ge n.$$

Хроматическое число любого дерева равно 2.

Число внешней устойчивости графа (*число вершинного покрытия*) $\beta_0(G)$ – это минимальное количество вершин, образами которых являются все остальные вершины этого графа.

Графы, содержащие изолированные вершины, не имеют вершинного покрытия, поэтому для таких графов число внешней устойчивости не определяется. Если в графе с n вершинами есть хотя бы одна вершина степени $n-1$, то $\beta_0(G)=1$.

При определении числа внешней устойчивости используется понятие опоры. *Опора* графа – это такое подмножество R вершин графа, для которого выполняется следующее условие: при объединении этого подмножества с множеством образов вершин, входящих в R получается исходное множество вершин этого графа. Количество вершин в наименьшей опоре и определяет число внешней устойчивости β_0. Например, для графа, изображенного на рис. 14, $\beta_0 = 2$, так как наименьшую опору образуют две вершины – 4 и 3.

Число паросочетания графа α_1 – это максимальное количество несмежных между собой ребер. В связном графе с n вершинами $\alpha_1(G) \le \frac{n}{2}$.

Рассмотрим граф на рис. 17. Выпишем все его максимальные паросочетания (паросочетания, которые нельзя расширить за счет добавления новых ребер): $\{e_1, e_3, e_4, e_7\}, \{e_1, e_4, e_7\}, \{e_1, e_4, e_6\}, \{e_2, e_5, e_7\}$. По количеству ребер наибольшим будет первое паросочетание – оно содержит четыре ребра, следовательно, $\alpha_1 = 4$.

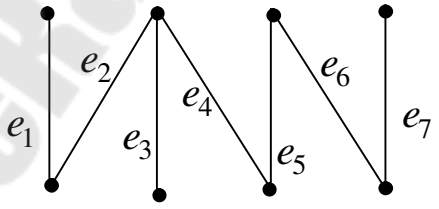


Рис. 17

Число реберного покрытия β_1 – это минимальное количество ребер для покрытия всех вершин графа. Для графов с изолированными вершинами это число не определяется. Для связного графа с n вершинами и m ребрами справедливо соотношение

$$\frac{n}{2} \le \beta_1(G) \le m.$$

Реберное покрытие графа – это любое подмножество его ребер для покрытия всех вершин графа (т. е. каждая вершина графа должна быть инцидентна хотя бы одному ребру из этого подмножества). Количество ребер в наименьшем покрытии и определяет число реберного покрытия β_1.

Для графа на рис. 17 справедлива оценка:

$$4 = \frac{8}{2} \le \beta_1(G) \le 7.$$

Так как в графе существует покрытие из четырех ребер $\{e_1, e_3, e_4, e_7\}$, соответствующее нижней границе, то $\beta_1(G) = 4$.

2. НЕКОТОРЫЕ АЛГОРИТМЫ, РЕАЛИЗУЕМЫЕ НА ГРАФАХ

2.1. Методы систематического обхода вершин в графе

Важными задачами теории графов являются задачи глобального анализа (поиск циклов, контуров, нахождение путей между парами вершин и т. п.). Базой для их решения являются алгоритмы обхода всех вершин графа таким образом, чтобы каждая вершина графа была отмечена один раз. Обычно такое «путешествие» сопровождается маркировкой пройденных ребер и нумерацией вершин графа в том порядке, в котором они посещались. Существуют две основных стратегии обхода вершин: поиск «в глубину» и поиск «в ширину».

Идея *поиска «в глубину»* состоит в следующем: отправляемся из некоторой вершины v_0. Предположим, мы достигли некоторой вершины v. Просматриваем список смежности $L(v)$ и, если там есть хоть одна неотмеченная вершина w, то двигаемся к ней, просматриваем ее список смежности $L(w)$, оставляя остальные вершины из $L(v)$ на потом.

Если в $L(v)$ неотмеченных вершин нет, то возвращаемся в ту вершину, из которой мы туда попали, и проверяем ее список смежности. Процесс прекратится, когда окажется, что все вершины отмечены или когда окажется, что из вершины v_0 больше пойти некуда.

Алгоритм *поиска «в ширину»* заключается в следующем: достигнув вершины v, отмечаем ее, затем просматриваем ее список

смежности $L(v)$ и отмечаем все ранее не отмеченные вершины из этого списка. Таким образом, вершина v считается полностью обработанной, и продолжается обработка вершин из списка $L(v)$ по очереди по указанным правилам.

И в том, и в другом случае результат работы алгоритма зависит от выбора начальной вершины, а также от порядка вершин в списках смежности.

2.2. Алгоритмы поиска минимальных маршрутов в графе

Многие проблемы, решаемые на практике, так или иначе связаны с поиском минимальных маршрутов в графе. В теории графов термин «минимальный маршрут» используется в двух аспектах. Мы говорим о минимальных маршрутах, когда минимизируем их длину или их вес. С этим связаны две различные задачи. Первая задача – поиск *минимального по длине* маршрута – сводится к следующему. В некотором заданном графе указывается пара вершин – начало и конец маршрута. Надо найти такой маршрут для данной пары вершин, длина которого будет минимальной. Вторая задача – поиск *минимального по весу* маршрута – сводится к следующему: в некотором заданном взвешенном графе указывается пара вершин – начало и конец маршрута. Надо найти такой маршрут для данной пары вершин, вес которого будет минимальным. Вес маршрута складывается из весов его ребер (дуг).

2.2.1. Волновой алгоритм

Волновой алгоритм позволяет найти минимальный путь между парой вершин в графе с ребрами единичной длины. В основе его лежит алгоритм поиска «в ширину».

Рассмотрим связный непустой граф $G(V, E)$. Требуется найти путь $\mu^* = (s, \text{K}, t)$ между вершинами s (источник) и t (приемник) графа (s не совпадает с t), содержащий минимальное количество промежуточных вершин (ребер).

Волновой алгоритм решает задачу построения минимального маршрута в два этапа. На первом этапе от источника к приемнику распространяется волна. На втором выполняется обратный ход, в процессе которого из ячеек волны формируется путь.

Приведем волновой алгоритм по шагам. Введем обозначения:

$k(v_i)$ – волновая метка i-й вершины; N – номер волны; *OLD* и *NEW* – соответственно старый и новый фронт волны.

Перед началом работы алгоритма положим $k(s)=0$, $k(v_i)=-1$ для всех $v_i \neq s$, $N=0$, $OLD=\{s\}$, $NEW=\{\}$.

I этап

Шаг 1. Для каждой вершины v_i, входящей в *OLD*, просматриваются все смежные с ней вершины u_j, и если $k(u_j)=-1$, то $NEW := NEW+\{u_j\}$, $k(u_j) := k(u_j)+1$, $N := N+1$.

Шаг 2. а) Если $NEW=\{\}$, то граф не является связным и не существует маршрута, соединяющего *s* и *t*;

б) если $t \in NEW$ (т. е. $u_j = t$ для некоторого *j*), то кратчайший путь между вершинами *s* и *t* найден. Он содержит N ребер.

Шаг 3. $OLD := NEW$, $NEW := \{\}$ и к шагу 1.

II этап

Если на N-м шаге была достигнута вершина *t*, то можно восстановить кратчайший путь следующим образом: среди соседей вершины *t* найдем любую вершину с волновой меткой $N-1$, среди соседей последней – вершину с меткой $N-2$, и т. д., пока не достигнем *s*. Найденная последовательность вершин определяет один из кратчайших путей из *s* в *t*. На практике выгодно сохранять на шаге 1 информацию о том, из какой вершины «волна» пришла в вершину u_j – тогда восстановление пути осуществляется быстрее.

Пример 2.1. Дан связный граф G. Найти в нем минимальный по длине маршрут μ^* от вершины v_1 к вершине v_7.

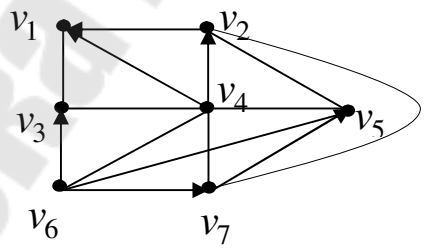
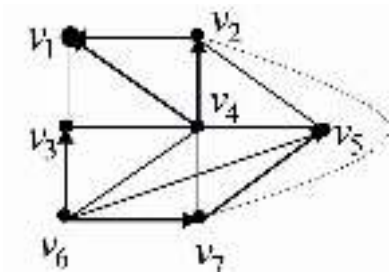


Рис. 18

Решение

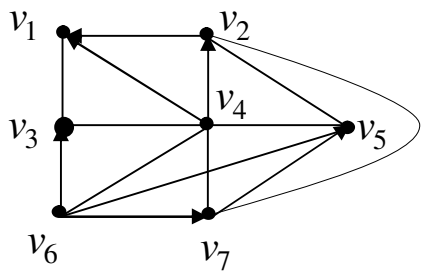
Проиллюстрируем работу алгоритма на рисунке и на матрице смежности заданного графа.

1. Строим фронт волны W_0. Включим в него вершину v_1 – источник волны и установим метку вершины v_1 по номеру фронта волны: $r(v_1)=0$.



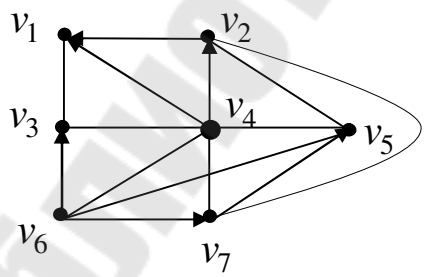
	1	2	3	4	5	6	7
1	0	0	1	0	0	0	0
2	1	0	0	0	1	0	1
3	1	0	0	1	0	0	0
4	1	1	1	0	1	1	1
5	0	1	0	1	0	0	0
6	0	0	1	1	1	0	1
7	0	0	0	1	1	0	0

2. Строим следующий фронт W_1, куда включаем образы вершины v_1 (вершины из списка смежности вершины v_1): $W_1 = \{v_3\}$. Установим метку этой вершины по номеру фронта волны: $r(v_3) = 1$. Так как W_1 не содержит вершину конца маршрута v_7, то продолжим распространение цифровой волны.



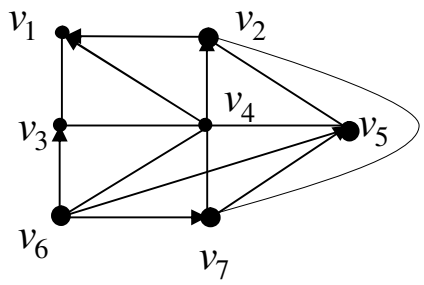
	1	2	3	4	5	6	7
1	0	0	1	0	0	0	0
2	1	0	0	0	1	0	1
3	1	0	0	1	0	0	0
4	1	1	1	0	1	1	1
5	0	1	0	1	0	0	0
6	0	0	1	1	1	0	1
7	0	0	0	1	1	0	0

3. Для построения следующего фронта волны W_2 найдем образы вершины, помеченной на предыдущем шаге: $L(v_3) = \{v_1, v_4\}$. Так как вершина v_4 еще не имеет метки, то включим ее в следующий фронт волны – $W_2 = \{v_4\}$ – и установим метку этой вершины по номеру фронта волны: $r(v_4) = 2$. Так как W_2 не содержит вершину конца маршрута v_7, то продолжим распространение волны по графу.



	1	2	3	4	5	6	7
1	0	0	1	0	0	0	0
2	1	0	0	0	1	0	1
3	1	0	0	1	0	0	0
4	1	1	1	0	1	1	1
5	0	1	0	1	0	0	0
6	0	0	1	1	1	0	1
7	0	0	0	1	1	0	0

4. Найдем образы вершины x_4: $L(v_4)=\{v_1,v_2,v_3,v_5,v_6,v_7\}$. Не имеющие меток вершины v_2,v_5,v_6 и v_7 включим в следующий фронт волны – $W_3=\{v_2,v_5,v_6,v_7\}$ и установим им метку $r(v_2)=r(v_5)= \ =r(v_6)=r(v_7)=3$. Так как фронт волны W_3 содержит конец маршрута v_7, то установим $l=3$ и перейдем ко второму этапу алгоритма.



	1	2	3	4	5	6	7
1	0	0	1	0	0	0	0
2	1	0	0	0	1	0	1
3	1	0	0	1	0	0	0
4	1	1	1	0	1	1	1
5	0	1	0	1	0	0	0
6	0	0	1	1	1	0	1
7	0	0	0	1	1	0	0

1. При обратном ходе будем включать в искомый маршрут по одной вершине из каждого фронта волны: это вершина $v_7\in W_3$, вершина $v_4\in W_2$, вершина $v_3\in W_1$ и вершина $v_1\in W_0$. Окончательно имеем маршрут длины 3: $v_1\to v_3\to v_4\to v_7$. На рис. 19 он выделен жирной линией.

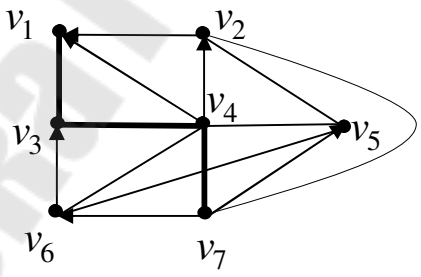


Рис. 19

2.2.2. Поиск кратчайшего пути между заданной парой вершин во взвешенном графе

Пусть $G(V,E)$ – связный взвешенный граф с матрицей весов $C=(c_{ij})$. Задача построения кратчайшего пути между заданной парой вершин $s\in V$ и $t\in V$ заключается в том, чтобы из множества путей, соединяющих указанные вершины, найти путь с наименьшим весом.

Для решения задачи можно воспользоваться ***алгоритмом Дейкстры***. Он основан на приписывании каждой вершине временной пометки (верхней границы). На каждой итерации ровно одна временная пометка становится постоянной (соответствующая вершина становится «окрашенной»), что означает, что пометка перестала быть верхней границей и дает уже точную длину кратчайшего пути от s к рассматриваемой вершине.

Приведем описание алгоритма по шагам.

Перед началом выполнения алгоритма все вершины не окрашены. Каждой вершине $v \in V$ в ходе выполнения алгоритма присваивается число $d(v)$, равное длине кратчайшего пути из s в v, включающего только окрашенные вершины.

Шаг 1. Положить $d(s) = 0, d(v) = \infty$ для всех v, отличных от s. Окрасить вершину s и положить $y = s$ (y – последняя из окрашенных вершин).

Шаг 2. Для каждой неокрашенной вершины v следующим образом пересчитать величину $d(v)$:

$$d(v) = \min\{d(v), d(y) + c(y,v)\}.$$

Если $d(v) = \infty$ для всех неокрашенных вершин v, закончить процедуру алгоритма: в заданном графе отсутствуют пути между указанной парой вершин. В противном случае окрасить ту из вершин v, для которой величина $d(v)$ является наименьшей. Положить $y = v$.

Шаг 3. Если $y = t$, закончить процедуру: кратчайший путь из вершины s в вершину t найден. Иначе перейти к шагу 2.

Пример 2.2. Пользуясь алгоритмом Дейкстры, найти кратчайший путь от вершины v_1 к вершине v_8 в графе, изображенном на рис. 20.

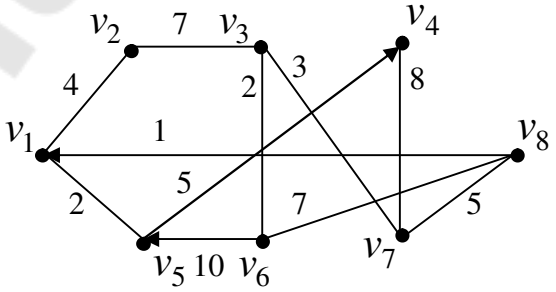


Рис. 20

Решение

Перед началом работы алгоритма окрасим вершину v_1 (припишем ей постоянную метку $d(v_1) = 0$). Остальным вершинам припишем временные метки $d(v_i) = \infty$, $i = 2,\text{K}\ ,8$.

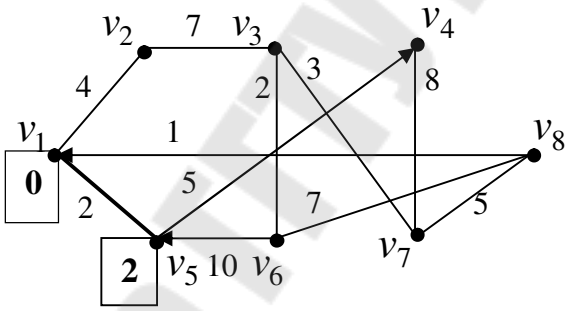
Первая итерация

Пересчитаем оценки вершин, смежных с окрашенной вершиной v_1 в соответствии с формулой, приведенной в шаге 2. Это вершины v_2 и v_5. Имеем:

$$d(v_2) = \min\{\infty, 0+4\} = 4,$$

$$d(v_5) = \min\{\infty, 0+2\} = 2.$$

Оценки вершин v_3, v_4, v_6, v_7, v_8 не изменились и по-прежнему равны бесконечности. Окрашиваем вершину с наименьшей оценкой – это вершина v_5. Она получает постоянную метку, равную 2. Присваиваем $y := v_5$. Так как $y \neq v_8$, алгоритм продолжает работу.



Вторая итерация

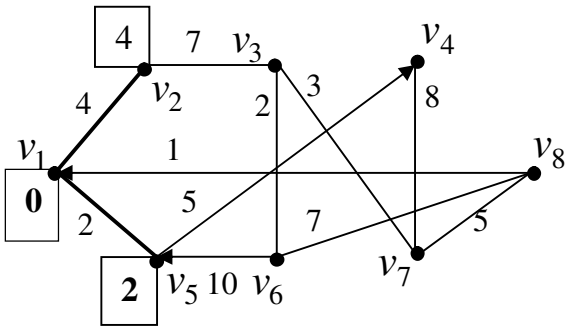
Пересчитаем метки $d(v_i)$ для всех неокрашенных вершин, смежных с v_5. Это вершина v_4. Имеем:

$$d(v_4) = \min\{\infty, 2+5\} = 7.$$

Выберем наименьшую среди меток всех неокрашенных вершин:

$$\min\{d(v_2), d(v_3), d(v_4), d(v_6), d(v_7), d(v_8)\} =$$
$$= \min\{4, \infty, 7, \infty, \infty, \infty\} = 4 = d(v_2),$$

следовательно, окрасим вершину v_2 и припишем ей постоянную метку, равную 4.



Третья итерация

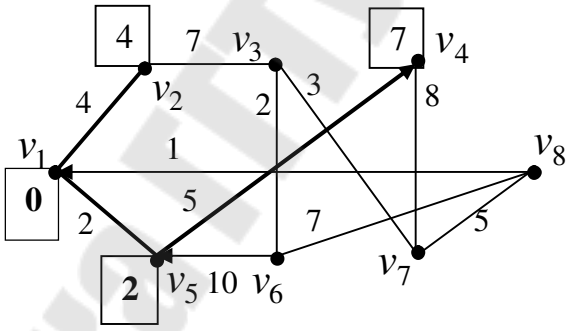
Пересчитаем метку $d(v_3)$ для единственной неокрашенной вершины v_3, смежной с v_2. Имеем:

$$d(v_3) = \min\{\infty, 4+7\} = 11.$$

Выберем наименьшую среди меток всех неокрашенных вершин:

$$\min\{d(v_3), d(v_4), d(v_6), d(v_7), d(v_8)\} = \min\{11, 7, \infty, \infty, \infty\} = 7 = d(v_4),$$

следовательно, вершина v_4 получает постоянную метку, равную 7, и становится окрашенной. Так как $y = v_4 \neq v_8$, то алгоритм продолжает работу.



Четвертая итерация

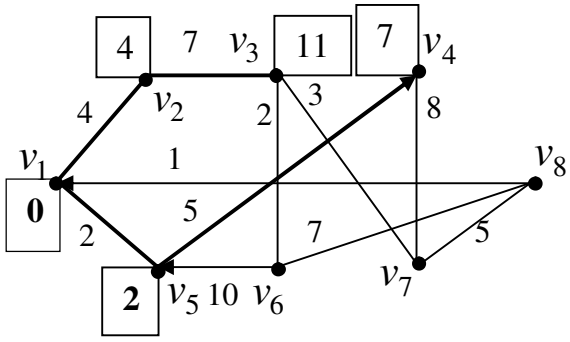
Пересчитаем оценку для неокрашенной вершины v_7, смежной с v_4. Имеем:

$$d(v_7) = \min\{\infty, 7+8\} = 15.$$

Выберем наименьшую среди меток всех неокрашенных вершин:

$$\min\{d(v_3), d(v_6), d(v_7), d(v_8)\} = \min\{11, \infty, 15, \infty\} = 11 = d(v_3).$$

Окрасим вершину v_3 и присвоим ей метку, равную 11.

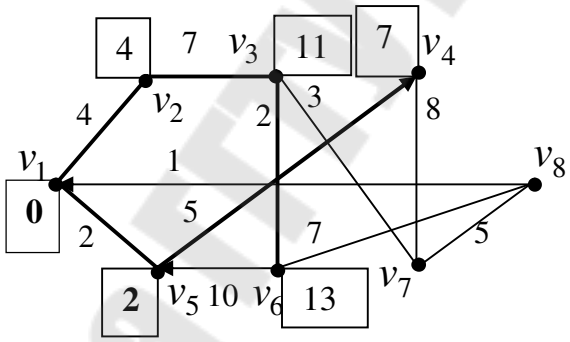


Пятая итерация

Пересчитаем оценку для неокрашенных вершин v_6 и v_7, смежных с v_3. Имеем:

$$d(v_6) = \min\{\infty, 11+2\} = 13,\ d(v_7) = \min\{15, 11+3\} = 14.$$

Как видим, оценка вершины v_7 улучшилась по сравнению с оценкой, полученной на предыдущей итерации, однако окрашиваем вершину v_6, так как ее оценка является наименьшей среди оценок неокрашенных пока вершин.

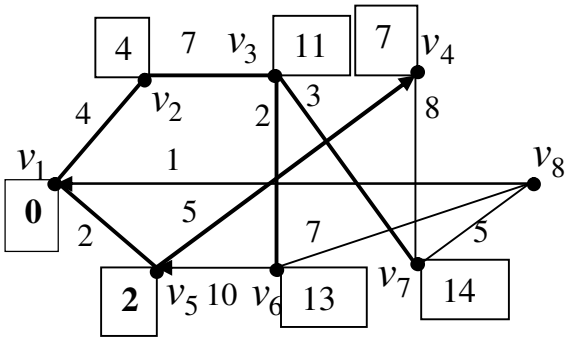


Шестая итерация

Остались две неокрашенных вершины – v_7 и v_8.

$$d(v_7) = 14,$$
$$d(v_8) = \min\{\infty, 13+7\} = 20.$$

Окрашиваем вершину v_7, присваиваем ей постоянную метку, равную 14.



Седьмая итерация

Осталась единственная неокрашенная вершина – v_8.

$$d(v_8) = \min\{20, 14+5\} = 19.$$

Вершина v_8 получает постоянную метку, равную 19, и алгоритм прекращает работу. Окончательно имеем: кратчайшее расстояние между вершинами v_1 и v_8 равно 19. Это маршрут проходит через вершины v_2, v_3 и v_7 и выделен жирной линией на рис. 21. Попутно в ходе работы алгоритма мы получили кратчайшие маршруты от вершины v_1 до всех остальных вершин графа. Эти маршруты также выделены на рис. 21, а постоянные метки, приписанные каждой вершине, и определяют длины этих маршрутов.

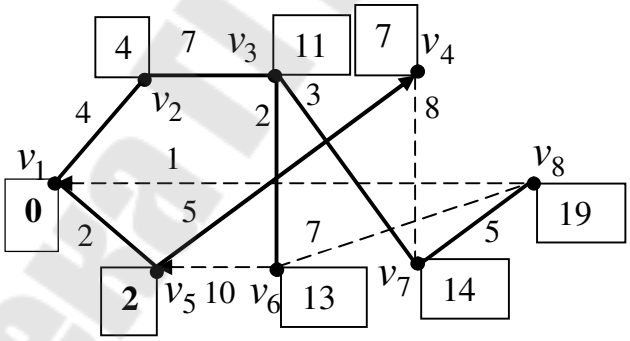


Рис. 21

Проиллюстрируем работу алгоритма Дейкстры для графа, заданного весовой матрицей. Запишем весовую матрицу графа, изображенного на рис. 20.

	1	2	3	4	5	6	7	8
1	–	4			2			
2	4	–	7					
3		7	–			2	3	
4				–			8	
5	2			5	–			
6			2		10	–		7
7			3	8			–	5
8	1					7	5	–

Найдем кратчайший путь от первой до восьмой вершины. Перед началом работы алгоритма окрасим начальную вершину v_1 и присвоим ей постоянную метку, равную 0.

Первая итерация

Просмотрим первую строку весовой матрицы и выберем в ней минимальный элемент. Это элемент $c_{15} = 2$. Следовательно, на первом шаге окрашиваем вершину v_5, которая будет иметь постоянную метку, равную 2.

		1	2	3	4	5	6	7	8
0+	1	–	4			2			
	2	4	–	7					
	3		7	–			2	3	
	4				–			8	
	5	2			5	–			
	6			2		10	–		7
	7			3	8			–	5
	8	1					7	5	–

Вторая итерация

Вычеркнем первый и пятый столбцы матрицы, чтобы исключить возврат в окрашенные вершины v_1 и v_5. Выделим пятую строку.

Просмотрим элементы выделенных строк (первой и пятой) и выберем $\min\{0 + c_{12}; 2 + c_{54}\} =$

$= \min\{0 + 4; 2 + 5\} = 4$.

Этот элемент находится во втором столбце, следовательно, вершина v_2 приобретает постоянную метку 4.

		1	2	3	4	5	6	7	8
0+	1	–	4			2			
	2	4	–	7					
	3		7	–			2	3	
	4				–			8	
2+	5	2			5	–			
	6			2		10	–		7
	7			3	8			–	5
	8	1					7	5	–

Третья итерация

Вычеркнем второй столбец весовой матрицы. Выделим вторую строку, так как вершина v_2 получила на предыдущем шаге постоянную метку. Просмотрим элементы выделенных строк (второй и пятой, так как в первой строке не осталось конечных элементов) и выберем

$$\min\{4+c_{23};2+c_{54}\} =$$
$$= \min\{4+7;2+5\} = 7.$$

Так как $7 = 2 + c_{54}$, то окрасим вершину v_4 с постоянной меткой, равной 7.

		1	2	3	4	5	6	7	8
0+	1	–	4			2			
4+	2	4	–	7					
	3		7	–			2	3	
	4				–			8	
2+	5	2			5	–			
	6			2		10	–		7
	7			3	8			–	5
	8	1					7	5	–

Четвертая итерация

Вычеркнем четвертый столбец весовой матрицы и выделим четвертую строку. Просмотрим элементы выделенных строк и выберем

$$\min\{4+c_{23};7+c_{47}\} =$$
$$= \min\{4+7;7+8\} = 11.$$

Так как $11 = 4 + c_{23}$, то постоянную метку, равную 11, получает вершина v_3.

		1	2	3	4	5	6	7	8
0+	1	–	4			2			
4+	2	4	–	7					
	3		7	–			2	3	
7+	4				–			8	
2+	5	2			5	–			
	6			2		10	–		7
	7			3	8			–	5
	8	1					7	5	–

Пятая итерация

Вычеркнем третий столбец и выделим третью строку. Просмотрим элементы выделенных строк (третьей и четвертой). Выберем

$$\min\{11+c_{36};11+c_{37};7+c_{47}\} =$$
$$= \min\{11+2;11+3;7+8\} = 13 = 11 + c_{36}.$$

Следовательно, постоянную метку, равную 13, присваиваем вершине v_6.

		1	2	3	4	5	6	7	8
0+	1	–	4			2			
4+	2	4	–	7					
11+	3		7	–			2	3	
7+	4				–			8	
2+	5	2			5	–			
	6			2		10	–		7
	7			3	8			–	5
	8	1					7	5	–

Шестая итерация

Вычеркнем шестой столбец и выделим шестую строку. Просмотрим элементы выделенных строк и выберем

$$\min\{11+c_{37};7+c_{47};13+c_{68}\}=$$
$$=\min\{11+3;7+8;13+7\}=$$
$$=14=11+c_{37}.$$

Присваиваем вершине v_7 постоянную метку, равную 14.

		1	2	3	4	5	6	7	8
0+	1	–	4			2			
4+	2	4	–	7					
11+	3		7	–			2	3	
7+	4				–			8	
2+	5	2			5	–			
13+	6			2		10	–		7
14+	7			3	8			–	5
	8	–					7	5	–

Седьмая итерация

Вычеркнем седьмой столбец и выделим седьмую строку. Просмотрим элементы выделенных строк и выберем

$$\min\{13+c_{68};14+c_{78}\}=$$
$$=\min\{13+7;14+5\}=19=14+c_{78}.$$

Вершина v_8 получает постоянную метку, равную 19. Следовательно, кратчайшее расстояние между вершинами v_1 и v_8 равно 19.

		1	2	3	4	5	6	7	8
0+	1	–	4			2			
4+	2	4	–	7					
11+	3		7	–			2	3	
7+	4				–			8	
2+	5	2			5	–			
13+	6			2		10	–		7
14+	7			3	8			–	5
	8	1					7	5	–

Так как вершина v_8 достигнута, то работа алгоритма прекращена, кратчайшее расстояние между вершинами v_1 и v_8 равно 19. Выписав элементы выделенных клеток весовой матрицы, можно восстановить кратчайшие маршруты от первой вершины до всех остальных:

$$c_{12}=4, c_{15}=5, c_{23}=7, c_{36}=2, c_{37}=3, c_{54}=5, c_{78}=5.$$

Результат работы алгоритма можно видеть на рис. 22.

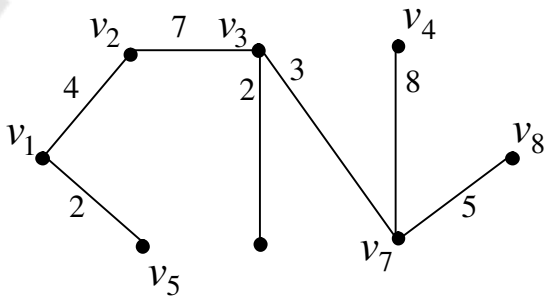


Рис. 22

2.3. Построение минимального остовного дерева во взвешенном графе

Граф, у которого каждому ребру (v_i, v_j) (дуге) сопоставлено некоторое действительное число c_{ij}, называется *взвешенным* (*размеченным*), а само число c_{ij} – *весом* ребра (дуги). Этот вес может описывать расстояние, стоимость или другие данные. ***Весом***, или ***стоимостью*** пути S называется число $l(S)$, равное сумме длин дуг, входящих в этот путь. Матрица $C = (c_{ij})$ называется ***матрицей весов***.

Остовным деревом (*покрывающим деревом*) связного графа G называется любой его подграф, содержащий все вершины графа G и являющийся деревом.

Минимальным остовным деревом (МОД) называется такое остовное дерево T графа G, что вес T меньше или равен весу любого другого остовного дерева графа G.

Наиболее популярными алгоритмами, решающими задачу построения минимального дерева-остова, являются *алгоритм Прима* и *алгоритм Краскала*.

Алгоритм Прима

Пусть n,m-граф $G(V,E)$ задан своей матрицей весов $C_{n\times n}$. Требуется построить минимальное остовное дерево $T(V,E^*)$. Алгоритм строит цепочку из n поддеревьев МОД $T' \to T'' \to \mathrm{L} \to T$, где T'– граф, состоящий из одной произвольно выбранной вершины графа $G(V,E)$, например, из вершины v_1. Наращивание МОД $T(V,E^*)$ происходит последовательно.

Введем следующие обозначения: g – индекс вершины графа$G(V,E)$; выбранной для построения очередного поддерева в цепочке; $l(S)$ – суммарная стоимость ребер очередного дерева в цепочке; l – количество ребер искомого дерева.

Шаг 1. Строим дерево $T' = \{v_1\}$.

Присваиваем переменным $g, l(S)$ и k следующие значения:

$$g := 1,\ l(S) := 0,\ k := 0.$$

Шаг 2. В матрице C вычеркиваем g-ю строку и выделяем g-й столбец. Просматриваем в матрице C все выделенные строки сверху

вниз и слева направо и выбираем минимальный элемент c_{ij}. Если таких элементов несколько, то выбираем первый по счету элемент; если все $c_{ij} = \infty$, то граф несвязный и построение остовного дерева невозможно.

Шаг 3. Строим очередное поддерево МОД в цепочке путем добавления новой вершины v_j и ребра $(v_i; v_j)$. Переменные $g, l(S)$, k примут следующие значения:

$$g := j,\ l(S) := l(S) + c_{ij},\ \ k := k + 1.$$

Шаг 4. Если $k < n - 1$, то к шагу 2.

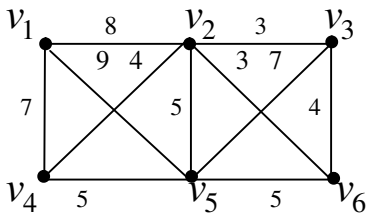
Шаг 5. Остовное дерево $T(V, E^*)$ построено, $l(S)$ – суммарный вес его ребер.

Пример 2.3. Дана матрица весов C взвешенного связного графа $G(V, E)$ с шестью вершинами. Для наглядности бесконечно большие веса отсутствующих ребер графа в матрице не указаны (пустые клетки). Построить остовное дерево наименьшего веса, используя алгоритм Прима.

	1	2	3	4	5	6
1	–	8		7	9	
2	8	–	3	4	5	3
3		3	–		7	4
4	7	4		–	5	
5	9	5	7	5	–	5
6		3	4		5	–

Решение

Для наглядности изобразим граф, заданный в условии.



Применим алгоритм Прима.

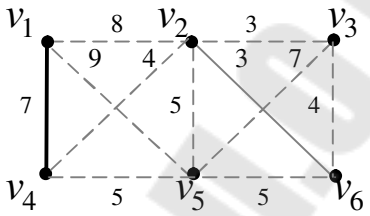
Поддерево T' состоит из одной вершины v_1;

$$g := 1,\ \ l(S) := 0,\ \ k := 0.$$

Первая итерация

Вычеркнем первый столбец и выделим первую строку в матрице. Проведем поиск минимального элемента в выделенной строке матрицы. Это элемент $c_{14} = 7$. Он указывает на новую вершину дерева – v_4 и ребро (v_1, v_4) для построения очередного поддерева T'' МОД из цепочки.

	1	2	3	4	5	6
1	–	8		7	9	
2	8	–	3	4	5	3
3		3	–		7	4
4	7	4		–	5	
5	9	5	7	5	–	5
6		3	4		5	–

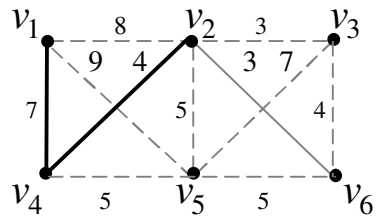


$$g := 4, l(S) := 7, k := 2.$$

Вторая итерация

Вычеркнем четвертый столбец и выделим четвертую строку в матрице. Найдем минимальные элементы в выделенных строках матрицы. Это элемент $c_{42} = 4$. Он указывает на новую вершину дерева – v_2 и ребро (v_4, v_2) для построения очередного поддерева T'''.

	1	2	3	4	5	6
1	–	8		7	9	
2	8	–	3	4	5	3
3		3	–		7	4
4	7	4		–	5	
5	9	5	7	5	–	5
6		3	4		5	–



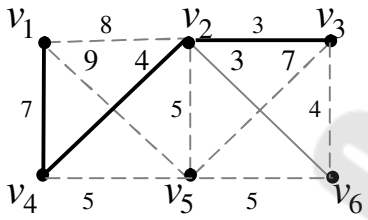
$$g := 2, l(S) := 7 + 4 = 11, k := 3.$$

Третья итерация

Вычеркнем второй столбец и выделим вторую строку в матрице. Найдем минимальные элементы в выделенных строках матрицы. Таких элементов с минимальным значением «3» – два. Элементы находятся в разных столбцах. Матрица просматривается сверху вниз, слева направо. Выберем первый по счету минимальный элемент $c_{23} = 3$.

Он указывает на новую вершину дерева – v_3 и ребро (v_2, v_3) для построения очередного поддерева МОД из цепочки.

	1	2	3	4	5	6
1	–	8		7	9	
2	8	–	3	4	5	3
3		3	–		7	4
4	7	4		–	5	
5	9	5	7	5	–	5
6		3	4		5	–

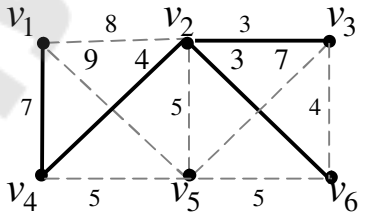


$$g := 3,\ l(S) := 11 + 3 = 14,\ k := 4.$$

Четвертая итерация

Вычеркнем третий столбец и выделим третью строку в матрице. Найдем минимальные элементы в выделенных строках матрицы. Это элемент $c_{26} = 3$. Он указывает на новую вершину дерева – v_6 и ребро (v_2, v_6) для построения очередного поддерева МОД из цепочки.

	1	2	3	4	5	6
1	–	8		7	9	
2	8	–	3	4	5	3
3		3	–		7	4
4	7	4		–	5	
5	9	5	7	5	–	5
6		3	4		5	–

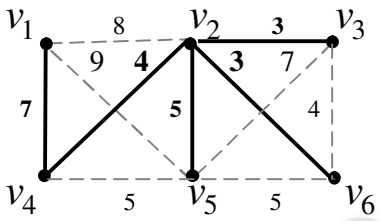


$$g := 6,\ l(S) := 14 + 3 = 17,\ k := 5.$$

Пятая итерация

Вычеркнем шестой столбец и выделим шестую строку в матрице. Найдем минимальные элементы в выделенных строках матрицы. Таких чисел три: $c_{25} = c_{45} = c_{65} = 5$. Выбираем первый по счету – c_{25}. Он и указывает на новую вершину дерева – v_5 и ребро (v_2, v_5) для построения очередного поддерева МОД из цепочки.

	1	2	3	4	5	6
1	–	8		7	9	
2	8	–	3	4	5	3
3		3	–		7	4
4	7	4		–	5	
5	9	5	7	5	–	5
6		3	4		5	–



$$g := 5, l(S) := 17 + 5 = 22, k := 6.$$

Цикл закончен. Минимальное остовное дерево с суммарным весом ребер $l(S) = 22$ построено.

Замечание. Очевидно, построенное остовное дерево с весом $l(S) = 22$ не единственное: если бы на пятой итерации мы выбрали ребро (v_4, v_5) или (v_6, v_5), то получили бы другие деревья с тем же весом, изображенные на рис. 23.

Рис. 23

Алгоритм Краскала

Алгоритм Краскала дает точное решение задачи построения минимального остовного дерева $T(V, E^*)$ связного взвешенного n, m-графа $G(V, E)$. Исходный граф задается в виде таблицы R_0, содержащий список ребер исходного графа $U = \{e_1, e_2, \text{K}, e_m\}$ и их веса $W = \{w_1, w_2, \text{K}, w_m\}$. Вначале полагается, что каждая вершина графа G образует отдельное поддерево МОД. Алгоритм Краскала наращивает дерево $T(V, E^*)$ из своих поддеревьев (компонент связности) параллельно-последовательно, используя следующее свойство деревьев: при добавлении нового ребра, инцидентного двум вершинам одного дерева, образуется цикл.

Шаг 1. Отсортируем ребра графа по неубыванию весов.

Шаг 2. Полагаем, что каждая вершина относится к своей компоненте связности.

Шаг 3. Проходим ребра в «отсортированном» порядке. Для каждого ребра выполняем следующую проверку:

а) если вершины, соединяемые данным ребром, лежат в разных компонентах связности, то объединяем эти компоненты в одну, а рассматриваемое ребро добавляем к минимальному дереву-остову;

б) если вершины, соединяемые данным ребром, лежат в одной компоненте связности, то исключаем ребро из рассмотрения, так как при включении данного ребра образуется цикл.

Шаг 4. Если есть еще нерассмотренные ребра и не все компоненты связности объединены в одну, то переходим к шагу 3, иначе алгоритм завершает работу:

а) если при этом просмотрены все ребра, но не все компоненты связности объединены в одну, то для исходного графа невозможно построить покрывающее дерево;

б) если просмотрены все ребра и все компоненты связности объединены в одну, то для исходного графа построено минимальное покрывающее дерево.

Пример 2.4. Построить остовное дерево наименьшего веса для графа из примера 2.3, пользуясь алгоритмом Краскала.

Решение

Составим таблицу весов ребер R_0 (для краткости для обозначения ребра, например, (v_1, v_2) будем использовать запись 1–2):

Номер	1	2	3	4	5	6	7	8	9	10	11
Ребро	1–2	1–4	1–5	2–3	2–4	2–5	2–6	3–5	3–6	4–5	5–6
Вес	8	7	9	3	4	5	3	7	4	5	5

Проведем в таблице R_0 сортировку ребер по возрастанию их весов и сформируем таблицу R:

Номер	1	2	3	4	5	6	7	8	9	10	11
Ребро	2–3	2–6	2–4	3–6	2–5	4–5	5–6	1–4	3–5	1–2	1–5
Вес	3	3	4	4	5	5	5	7	7	8	9

Обозначим через k – число ребер искомого дерева, множество X^* – множество его компонент связности.

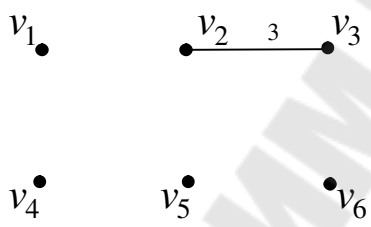
Так как перед началом работы алгоритма множество X^* содержит только изолированные вершины (шесть компонент связности), то

$$X^* = \{\{x_1\},\{x_2\},\{x_3\},\{x_4\},\{x_5\},\{x_6\}\},\ l(S) = 0, k = 0.$$

Первая итерация

Проверяем условие для первого ребра $e_1 = (2–3)$ в таблице R. Оба конца этого ребра принадлежат разным компонентам связности. Поэтому построим первое ребро МОД $T(V, E^*)$ $e_1 = (2–3)$ и установим следующие значения:

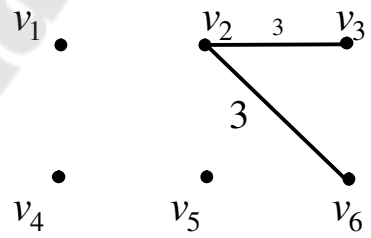
$$k = 1,\ l(S) = 3,\ X^* = \{\{x_1\},\{x_2, x_3\},\{x_4\},\{x_5\},\{x_6\}\}.$$



Вторая итерация

Проверяем условие для второго ребра $e_2 = (2–6)$ в таблице R. Оба конца этого ребра принадлежат разным компонентам связности. Поэтому построим второе ребро МОД $T(V, E^*)$ $e_2 = (2–6)$ и установим следующие значения:

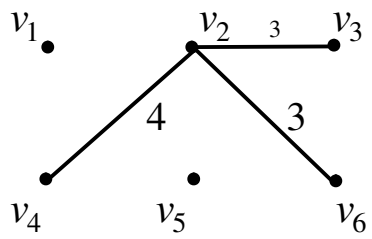
$$k = 2,\ l(S) = 3 + 3 = 6,\ X^* = \{\{x_1\},\{x_2, x_3, x_6\},\{x_4\},\{x_5\}\}.$$



Третья итерация

Проверяем условие для третьего ребра в таблице R. Оба конца этого ребра принадлежат разным компонентам связности. Поэтому построим третье ребро МОД $T(V, E^*)$ $e_3 = (2–4)$ и установим следующие значения:

$$k = 3,\ l(S) = 6 + 4 = 10,\ X^* = \{\{x_1\},\{x_2, x_3, x_4, x_6\},\{x_5\}\}.$$

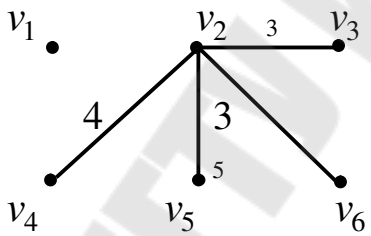


Четвертая итерация

Проверяем условие для четвертого ребра e = (3–6) в таблице R. Концы этого ребра принадлежат одной компоненте связности, поэтому это ребро пропускаем.

Берем следующее, пятое ребро e = (2–5) из таблицы R. Вершины v_2 и v_5 принадлежат разным компонентам связности, следовательно, четвертое ребро МОД $T(V, E^*)$ e_4 = (2–5) и установим следующие значения:

$$k = 4,\ l(S) = 10 + 5 = 15,\ X^* = \{\{x_1\}, \{x_2, x_3, x_4, x_5, x_6\}\}.$$



Пятая итерация

Следующие два ребра из таблицы R – (4–5) и (5–6) – не включаем в МОД $T(V, E^*)$, так как концы этих ребер принадлежат одной и той же компоненте связности.

Следующее по порядку ребро в таблице R – ребро (1–4). Вершины v_1 и v_4 принадлежат разным компонентам связности, следовательно, пятое ребро МОД $T(V, E^*)$ e_5 = (1–4). Установим следующие значения:

$$k = 5,\ l(S) = 15 + 7 = 22,\ X^* = \{x_1, x_2, x_3, x_4, x_5, x_6\}.$$

Так как все вершины объединены в одну компоненту, то остовное дерево построено, суммарный вес ребер $l(S) = 22$. Дерево изображено на рис. 24.

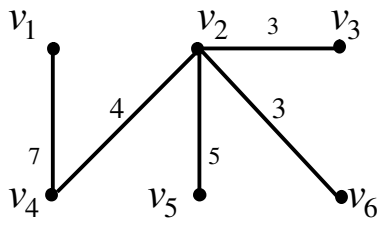


Рис. 24

2.4. Эйлеровы и гамильтоновы графы. Алгоритм Флери

В различных приложениях большое значение имеют понятия эйлерова и гамильтонова графа.

Определение 2.1. *Эйлеровой цепью* в графе называется цепь, содержащая все ребра графа. Замкнутая эйлерова цепь называется *эйлеровым циклом*. Связный граф, обладающий эйлеровым циклом, называют *эйлеровым графом*. Если в связном графе существует эйлерова цепь, но не существует эйлерова цикла, то он называется *полуэйлеровым.*

Следующие теоремы дают необходимые и достаточные условия существования эйлеровых путей и циклов.

Теорема 2.1. Неориентированный граф G является эйлеровым тогда и только тогда, когда он является связным и все его вершины четные.

Теорема 2.2. Неориентированный граф G является полуэйлеровым тогда и только тогда, когда он связный и ровно две его вершины имеют нечетную степень. При этом одна из этих вершин является началом эйлеровой цепи, а другая – концом.

Теорема 2.3. Ориентированный граф является эйлеровым тогда и только тогда, когда он связный и полустепень захода каждой его вершины равна ее полустепени выхода.

Теорема 2.4. Ориентированный граф является полуэйлеровым тогда и только тогда, когда он связный и удовлетворяет следующим условиям:

1) ровно в одной его вершине полустепень выхода на единицу больше полустепени захода (эта вершина – начало эйлерова пути);

2) ровно в одной его вершине полустепень захода на единицу больше полустепени выхода (эта вершина – конец эйлерова пути);

3) для остальных вершин полустепени захода и выхода равны.

В настоящее время существуют эффективные алгоритмы поиска эйлеровых путей в графе.

Алгоритм Флери

Чтобы построить эйлеров путь, нужно запустить алгоритм из вершины с нечетной степенью (если все степени четны, то из любой вершины), предварительно убедившись в существовании эйлерова пути.

Шаг 1. Положить текущий граф равным G, а текущую вершину – равной произвольной вершине $v \in V$.

Шаг 2. Выбрать произвольное, с учетом ограничения (см. ниже) ребро e текущего графа, инцидентное текущей вершине.

Шаг 3. Назначить текущей вторую вершину, инцидентную e.

Шаг 4. Удалить e из текущего графа и внести в список.

Шаг 5. Если в текущем графе еще остались ребра, вернуться на шаг 2.

Ограничение: если степень текущей вершины в текущем графе больше 1, нельзя выбирать ребро, удаление которого из текущего графа увеличит число компонент связности в нем.

Пример 2.5. Построить эйлеров цикл в графе, изображенном на рис. 25.

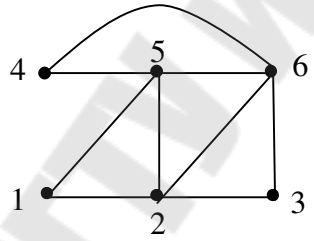


Рис. 25

Решение

Выпишем степени всех вершин графа:

$$dg(1) = dg(4) = dg(6) = 2,\ \ dg(2) = dg(3) = dg(5) = 4.$$

Так как степени всех вершин четны и граф связен, то в силу теоремы 2.1 в графе существует эйлеров цикл. Найдем его, используя алгоритм Флери.

Пусть на первом шаге выбрана вершина 1. Выберем одно из двух инцидентных ей ребер (на выборе на шаге 2 ограничение пока никак не сказывается). Пусть выбрано, например, ребро (1–5).

На двух следующих итерациях ограничений на выбор по-прежнему не возникает; пусть выбраны ребра (5–2) и (2–6). Тогда текущим графом становится граф, изображенный на рис. 26 (текущая вершина – 6).

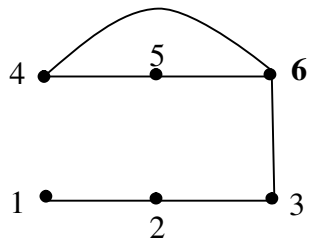


Рис. 26

На следующей итерации нельзя выбрать ребро (6–3) из-за ограничения (его удаление приведет к появлению двух компонент связности, в то время как степень вершины 6 больше единицы). Пусть выбрано ребро (6–5). Дальнейший выбор ребер определен однозначно (текущая вершина всегда будет иметь степень 1), так что в итоге будет построен следующий эйлеров цикл:

$$1 \to 5 \to 2 \to 6 \to 5 \to 4 \to 6 \to 3 \to 2 \to 1.$$

Определение 2.2. *Гамильтоновой цепью* в графе называется цепь, содержащая все вершины графа. Замкнутая гамильтонова цепь называется *гамильтоновым циклом*. Граф, обладающий гамильтоновым циклом, называется *гамильтоновым графом*. Если в связном графе существует гамильтонова цепь, но не существует гамильтонова цикла, то он называется *полугамильтоновым*.

В отличие от эйлеровых графов, простых критериев распознавания гамильтоновых графов не существует до сих пор. Все алгоритмы поиска гамильтоновых циклов в графе относятся к переборным алгоритмам. Очевидно лишь то, что гамильтонов граф всегда связен и что наличие петель и кратных ребер никак не влияет на гамильтоновость.

Следующие теоремы указывают некоторые достаточные условия существования гамильтоновых цепей в графе.

Теорема Оре. Пусть G – связный граф без петель и кратных ребер, содержащий n вершин $(n > 2)$. Если $dg(v) + dg(u) \geq n$ для любых двух различных вершин u и v графа G, то граф G гамильтонов.

Из теоремы Оре вытекает следующее, несколько более слабое, но очень простое для проверки достаточное условие гамильтоновости.

Теорема Дирака. Если в связном n, m-графе $(n \geq 3)$ степень каждой вершины не меньше $n/2$, то данный граф является гамильтоновым.

Задания для самостоятельного решения

1. Изобразите графы K_3, K_4, $K_{2,2}$, $K_{1,5}$.

2. В соревнованиях по круговой системе с двенадцатью участниками провели все встречи. Сколько встреч было сыграно?

3. Сколько ребер имеет полный граф с n вершинами?

4. Существует ли полный граф с 40 ребрами?

5. Что из приведенного ниже является путем в графе на рис. 27? Простым путем? Приведите длину каждого из путей.

1) 625341; 2) 62524351; 3) 6241624; 4) 624351634.

6. Что из приведенного ниже является циклом в графе на рис. 28? Простым циклом?

1) 1354351; 2) 12345651; 3) 534215345; 4) 6243516.

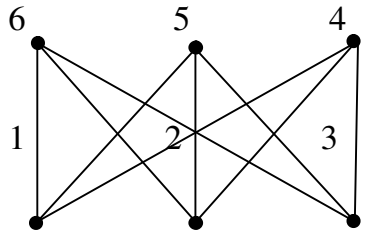


Рис. 27

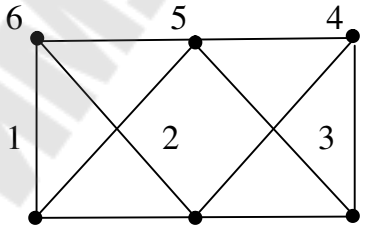


Рис. 28

7. Задать графы, изображенные на рис. 29, матрицами смежности и инцидентности, а также списками смежности:

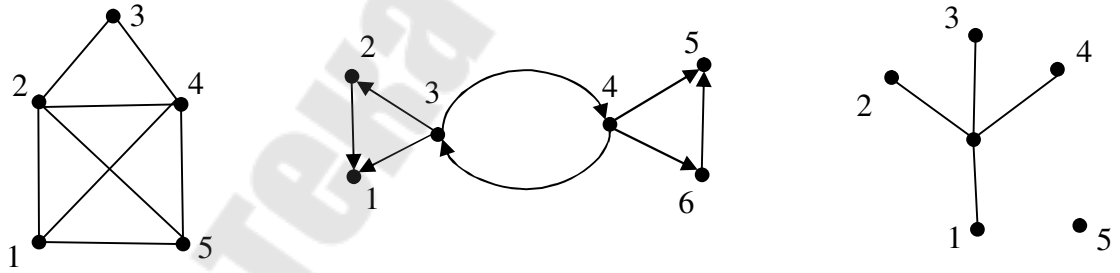


Рис. 29

8. По заданной матрице смежности изобразить граф:

а) $\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$; б) $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$.

9. Графы заданы своими матрицами инцидентности. Изобразить эти графы и записать их матрицы смежности.

а) $\begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$; б) $\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ -1 & 1 & 0 \\ 1 & -1 & 0 \end{pmatrix}$; в) $\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$; г) $\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 1 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix}$.

10. Для графа, заданного списками смежности, записать матрицу смежности и инцидентности:

а) $L(v_1)=\{v_2, v_3, v_4\}, L(v_2)=\{v_1\}, L(v_3)=\varnothing, L(v_4)=\varnothing$;

б) $L(v_1)=\{v_2, v_3\}, L(v_2)=\{v_1, v_3\}, L(v_3)=\{v_1, v_2, v_3, v_4\}$,

$L(v_4)=\{v_3, v_5\}, L(v_5)=\{v_3, v_4\}$.

11. Для графов G_1 и G_2, заданных матрицами смежности A и B соответственно, построить их объединение, пересечение, разность, дополнение, симметрическую разность и записать матрицы смежности полученных графов. Сделать вывод.

а) $A=\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$, $B=\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$;

б) $A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$, $B = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$.

12. Построить композицию графов $G_1 \circ G_2$ и $G_2 \circ G_1$ из № 11.

13. Изобразить однородный граф степени 3 с шестью вершинами.

14. Найдется ли граф с 5 вершинами, у которого одна вершина изолирована, а другие – степени 4?

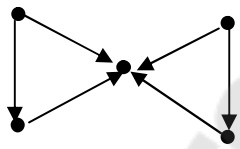
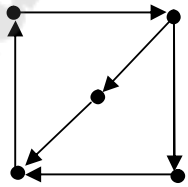
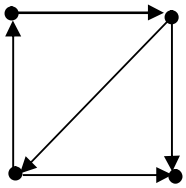
15. Существует ли граф, у которого есть вершины степеней 7, 4, 3, 2 и нет вершин других степеней? Какое наименьшее число вершин может быть в таком графе?

16. Расположить в пространстве пять одинаковых шаров так, чтобы каждый из них касался ровно трех других.

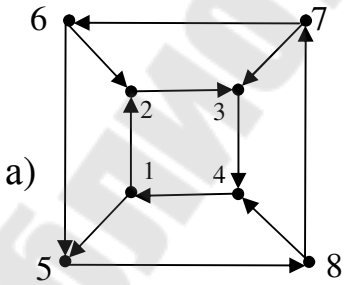
17. Может ли быть 50 дорог между населенными пунктами, если известно, что из каждого пункта выходит ровно три дороги?

18. Насыщенным углеводородом называется соединение углерода C, имеющего валентность 4, и водорода H, имеющего валентность 1, в котором при заданном числе атомов углерода содержится наибольшее число атомов водорода. Найдите формулу насыщенного углеводорода, содержащего n атомов углерода.

19. Какой из приведенных ниже орграфов является связным? Сильно связным?

а) 
б) 
в) 

20. Определить компоненты и бикомпоненты следующих графов:

а) 

б) $\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ в) $\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$

Найти все вершины, достижимые из вершины v_1 по пути длины 2; длины 3. Построить матрицу достижимости.

21. Для корневого ориентированного дерева на рис. 30 найдите потомков вершины 3, предков вершины 8, родителя вершины 5, сыновей вершины 3, листья дерева. Определить высоту дерева и уровень вершины 6. Является ли дерево бинарным?

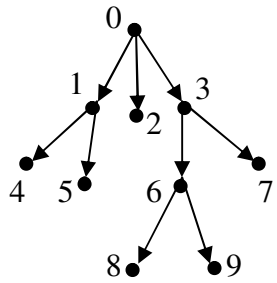


Рис. 30

22. Сколько ребер имеет дерево с n вершинами?

23. Какое максимальное число разрезов можно сделать в волейбольной сетке размером 20×10 так, чтобы она не распалась?

24. Спортсмен делает 5 выстрелов и за каждое попадание получает право еще на 2 выстрела. Определить число попаданий в цель, если известно, что спортсмен выстрелил 25 раз.

25. Пете дали конфету. Он ее съел, конфета ему понравилась, и ему дали еще две конфеты. Дальше события развивались следующим образом: если съеденная конфета нравилась Пете, то он получал еще две, а если нет – не получал ничего. Сорок третья конфета не понравилась Пете и на этом конфеты закончились. Сколько всего конфет не понравилось Пете?

26. Камень, падающий со скалы, разбивается о препятствие на три части. Упав на дно ущелья, осколки могут разбиться на 4 части, могут разбиться на две, а могут и не разбиться. Образовалось 102 камня. Определить число делений, если известно, что число камней, разбившихся на две части, в 6 раз больше числа камней, разбившихся на 4 части.

27. Провести, если это возможно, линию, которая пересекает каждый отрезок фигуры на рис. 31 ровно один раз.

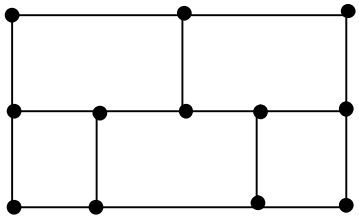


Рис. 31

28. На рис. 32 изображена схема зоопарка. Найти маршрут, по которому экскурсовод мог бы провести посетителей, показав им всех зверей и не проходя более одного раза ни одного участка цепи.

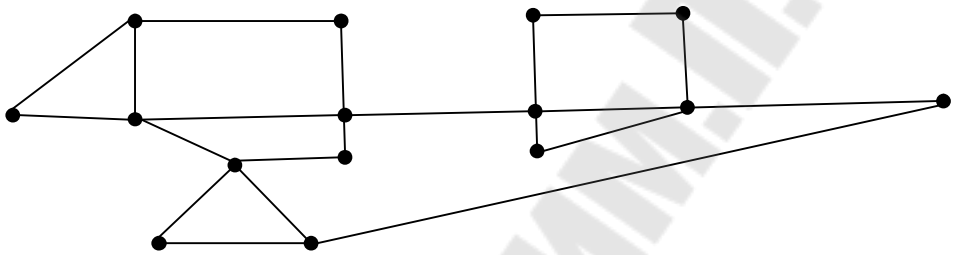


Рис. 32

29. На рис. 33 план подземелья, в одной из комнат которого скрыты сокровища. В завещании указано, что для отыскания сокровищ достаточно войти в одну из крайних комнат подземелья, пройти через все двери, причем в точности по одному разу через каждую. Сокровища скрыты за той дверью, которая будет пройдена последней. В какой комнате скрыты сокровища?

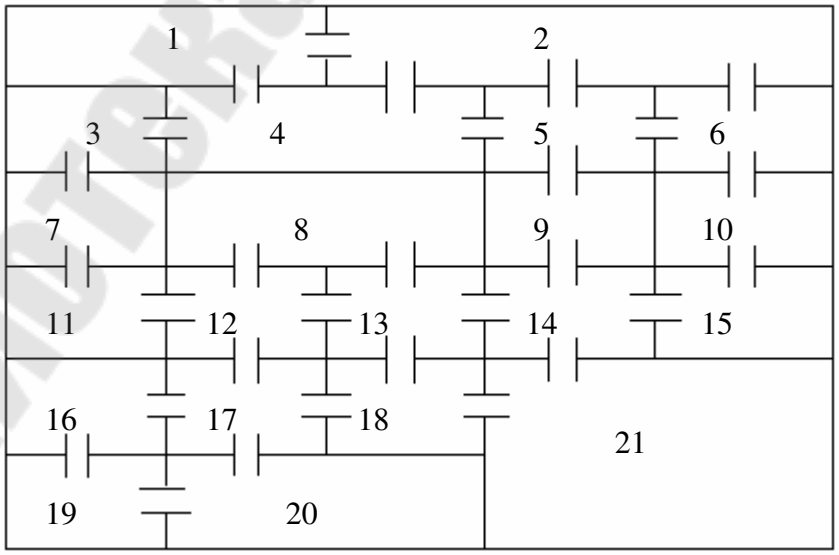


Рис. 33

30. Нарисовать граф с 8 вершинами, который:

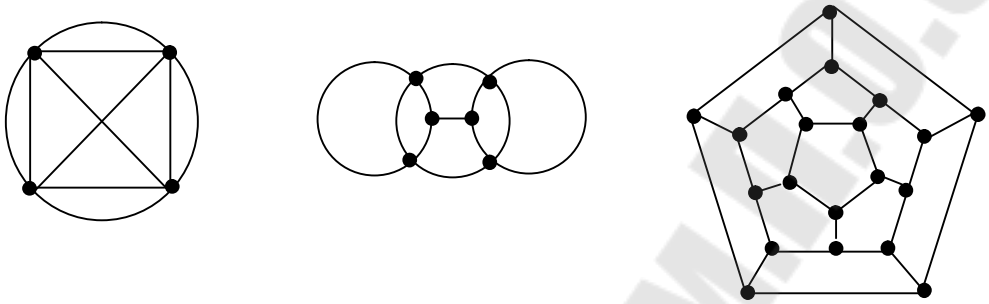
а) имеет эйлеров цикл;

б) имеет эйлерову цепь;

в) не имеет ни эйлерова цикла, ни эйлеровой цепи;

г) имеет простую цепь, содержащую все ребра графа.

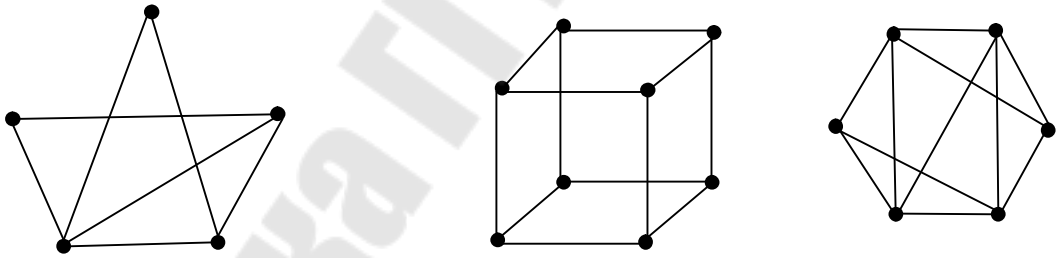
31. Указать, являются ли графы на рис. 34 эйлеровыми (гамильтоновыми), полуэйлеровыми (полугамильтоновыми).



а) *б*) *в*)

Рис. 34

32. Проверить формулу Эйлера для планарных графов на рис. 35 и изобразить их плоские представления:



а) *б*) *в*)

Рис. 35

33. Доказать непланарность графа K_5.

34. На участке три дома и три колодца. От каждого дома к каждому колодцу ведет тропинка. Когда владельцы домов поссорились, они задумали проложить от каждого дома к каждому колодцу тропинки так, чтобы не встречаться на пути. Показать, что их намерение неосуществимо.

35. Найти числовые характеристики графа на рис. 36.

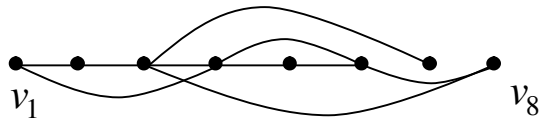
а) *б*)

Рис. 36

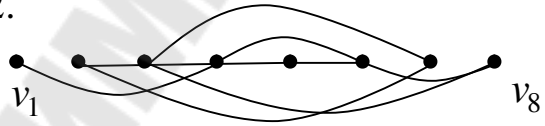
Индивидуальные задания

1. При помощи волнового алгоритма найти кратчайший маршрут в заданном неориентированном графе из вершины v_1 в вершину v_8. Записать матрицу смежности заданного графа.

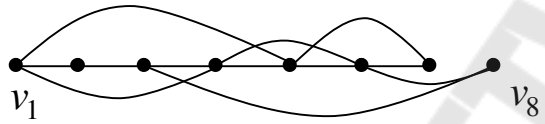
1.



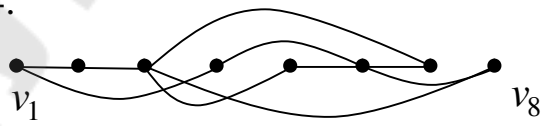
2.



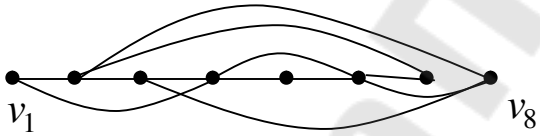
3.



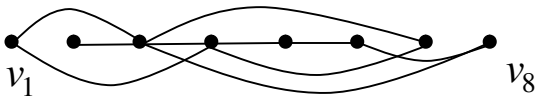
4.



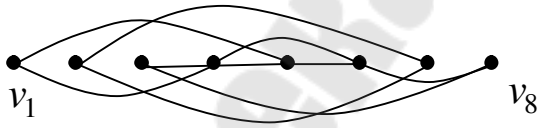
5.



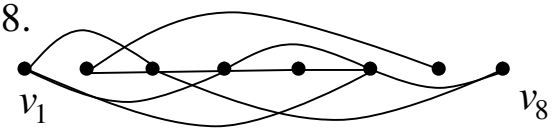
6.



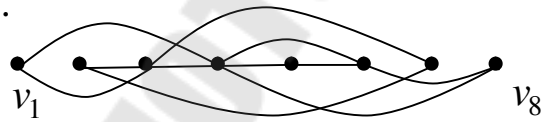
7.



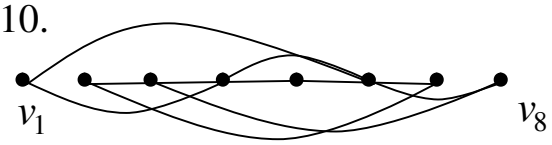
8.



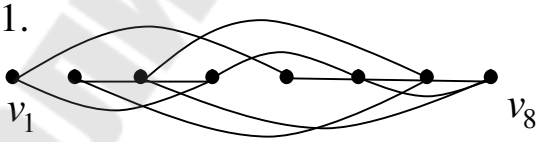
9.



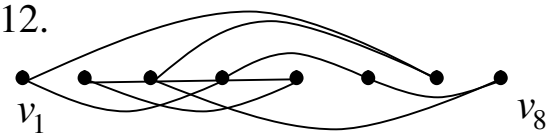
10.



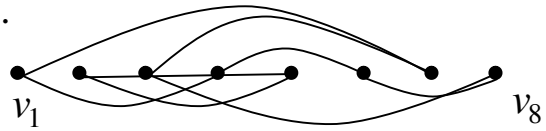
11.



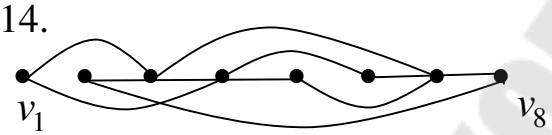
12.



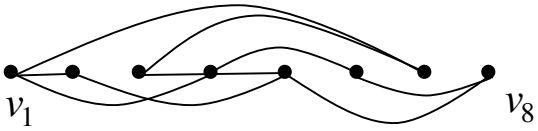
13.



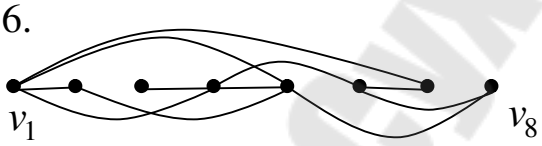
14.



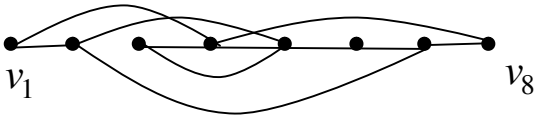
15.



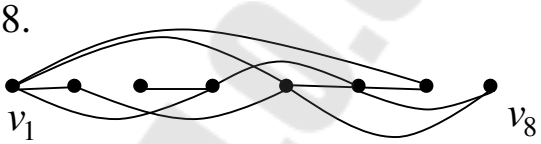
16.



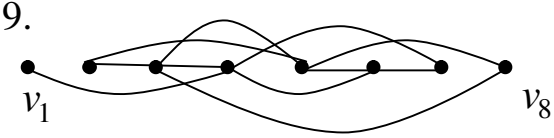
17.



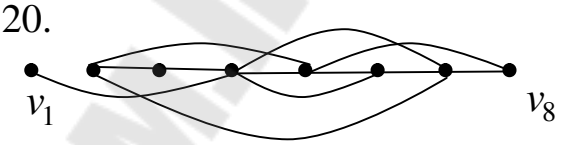
18.



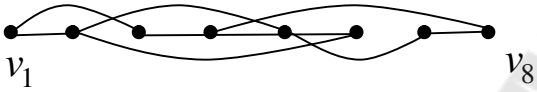
19.



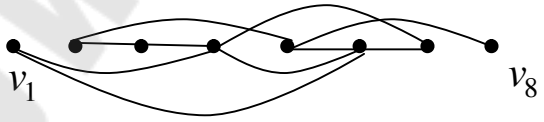
20.



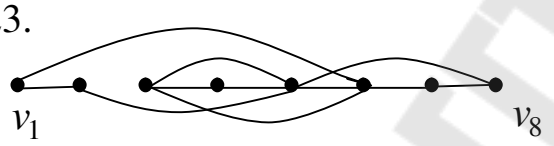
21.



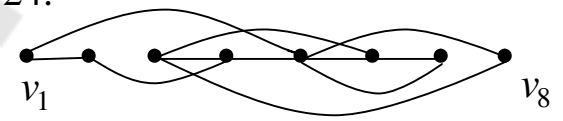
22.



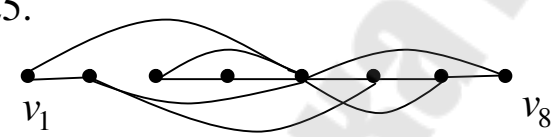
23.



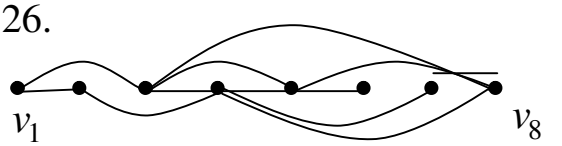
24.



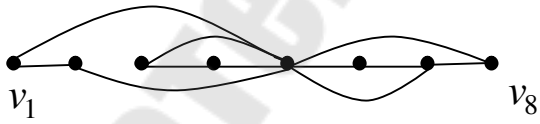
25.



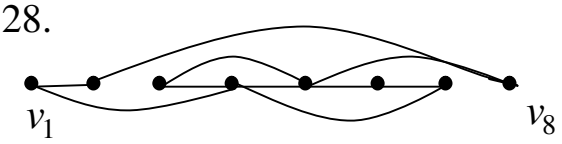
26.



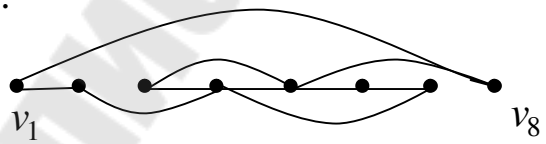
27.



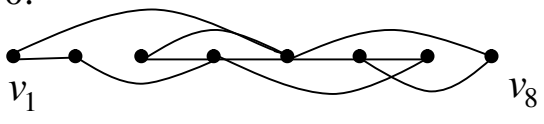
28.



29.



30.



2. Неориентированный граф G содержит десять вершин. Расстояния между вершинами заданы весовой матрицей (числовые зна-

чения элементов матрицы следует взять из таблицы). Построить его минимальное остовное дерево, используя:

а) алгоритм Прима;

б) алгоритм Краскала.

Изобразить исходный граф и построенное остовное дерево.

	1	2	3	4	5	6	7	8	9	10
1	0	***i***	***f***	–	***a***	–	***j***	***o***	***b***	–
2	***i***	0	***i***	***s***	–	***w***	***o***	***r***	***t***	***h***
3	***f***	***i***	0	–	***d***	***o***	***i***	***n***	***g***	–
4	–	***s***	–	0	***i***	***t***	–	***i***	***s***	–
5	***a***	–	***d***	***i***	0	***w***	***o***	***r***	***t***	***h***
6	–	***w***	***o***	***t***	***w***	0	***d***	***o***	***i***	***n***
7	***j***	***o***	***i***	–	***o***	***d***	0	***g***	–	***w***
8	***o***	***r***	***n***	***i***	***r***	***o***	***g***	0	***e***	***l***
9	***b***	***t***	***g***	***s***	***t***	***i***	–	***e***	0	***l***
10	–	***h***	–	–	***h***	***n***	***w***	***l***	***l***	0

3. Найти кратчайшие пути от вершины 1 до всех остальных вершин в графе, заданном весовой матрицей, используя алгоритм Дейкстры. Числовые значения элементов матрицы следует взять из таблицы. Изобразить исходный граф и указать кратчайшие пути.

.

	1	2	3	4	5	6	7	8	9	10
1	0	***a***	–	–	***r***	–	–	–	–	–
2	–	0	***b***	***c***	***g***	–	–	–	–	–
3	–	–	0	***e***	***f***	–	–	–	–	–
4	–	–	–	0	–	–	–	–	–	–
5	–	***l***	***h***	***w***	0	–	–	–	–	–
6	–	–	***d***	***i***	–	0	***j***	***o***	***t***	***g***
7	–	–	–	–	–	–	0	***s***	***v***	***m***
8	–	–	–	–	–	–	–	0	–	***n***
9	–	–	–	–	–	–	–	–	0	***k***
10	–	–	–	–	–	–	–	–	–	0

4. Выяснить, существует ли эйлерова цепь (цикл) в графе из № 1 и в случае утвердительного ответа найти ее, пользуясь алгоритмом Флери.

5. Найти гамильтонов путь (цепь) в графе из № 1, если такой путь существует.

6. Выяснить, является ли планарным граф из № 1. В случае утвердительного ответа построить его плоское представление.

7. Вычислить экстремальные числа графа из № 1 (числа внутренней и внешней устойчивости, кликовое, цикломатическое и хроматическое числа, числа реберного и вершинного покрытия).

№	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*r*	*s*	*t*	*v*	*w*	*x*	*y*	*z*
1	9	4	2	9	5	7	2	4	4	1	3	1	2	4	5	1	9	6	7	4	2	4	10
2	2	6	9	3	3	2	2	4	8	2	1	2	1	3	4	1	5	4	8	5	3	5	9
3	7	1	8	1	9	2	5	8	9	1	4	1	2	5	6	1	1	1	9	5	2	3	8
4	8	9	1	7	8	4	5	3	8	2	3	2	3	1	4	1	8	9	6	5	5	4	11
5	2	6	9	2	8	2	6	3	4	1	2	5	4	2	6	1	4	3	5	5	2	2	6
6	5	5	4	3	2	8	4	4	3	3	5	1	3	4	2	1	8	7	4	5	1	3	5
7	2	6	9	8	9	2	3	3	8	2	3	2	4	1	5	1	5	1	6	5	6	4	12
8	7	1	8	2	8	2	9	1	9	3	2	5	3	8	4	1	2	4	8	5	5	2	11
9	2	4	5	4	6	2	3	1	2	2	3	4	1	7	6	1	4	6	7	5	2	7	12
10	7	7	5	7	9	4	8	1	2	2	1	4	5	6	3	2	8	9	7	5	2	6	13
11	8	9	1	7	9	4	5	3	8	4	2	2	3	1	6	2	8	7	5	5	3	2	8
12	9	4	2	3	4	2	5	9	1	2	3	9	9	8	1	2	1	3	7	5	3	2	6
13	7	7	9	6	8	4	8	3	6	2	1	4	1	5	6	2	3	3	6	5	2	2	5
14	9	4	2	5	8	2	5	9	1	4	8	9	1	2	3	2	4	5	5	5	4	2	7
15	6	6	6	5	8	3	3	2	9	4	2	2	3	5	1	2	6	4	4	5	5	3	9
16	2	4	7	3	6	2	5	3	6	1	7	8	1	3	1	2	4	9	7	5	2	3	6
17	7	2	3	4	5	2	3	3	2	2	4	1	5	7	8	2	1	3	5	5	4	4	9
18	3	5	6	6	7	4	5	1	6	8	6	6	7	7	3	2	1	8	8	5	5	3	9
19	2	4	6	2	4	5	8	3	8	3	1	5	2	4	1	2	5	7	4	5	4	5	10
20	7	8	9	4	5	6	4	3	5	1	2	3	4	5	3	2	1	6	5	5	2	4	8
21	9	4	2	9	5	7	2	4	4	1	3	1	2	4	5	1	9	6	7	4	2	4	10
22	2	6	9	3	3	2	2	4	8	2	1	2	1	3	4	1	5	4	8	5	3	5	9
23	7	1	8	1	9	2	5	8	9	1	4	1	2	5	6	1	1	1	9	5	2	3	8
24	8	9	1	7	8	4	5	3	8	2	3	2	3	1	4	1	8	9	6	5	5	4	11
25	2	6	9	2	8	2	6	3	4	1	2	5	4	2	6	1	4	3	5	5	2	2	6
26	5	5	4	3	2	8	4	4	3	3	5	1	3	4	2	1	8	7	4	5	1	3	5
27	2	6	9	8	9	2	3	3	8	2	3	2	4	1	5	1	5	1	6	5	6	4	12
28	7	1	8	2	8	2	9	1	9	3	2	5	3	8	4	1	2	4	8	5	5	2	11
29	2	4	5	4	6	2	3	1	2	2	3	4	1	7	6	1	4	6	7	5	2	7	12
30	7	7	5	7	9	4	8	1	2	2	1	4	5	6	3	2	8	9	7	5	2	6	13

ЛИТЕРАТУРА

1. Андерсон, Джеймс А. Дискретная математика и комбинаторика : пер. с англ. / Джеймс А. Андерсон. – М. : Вильямс, 2004. – 960 с.

2. Белоусов, А. И. Дискретная математика : учеб. для вузов / А. И. Белоусов, С. Б. Ткачев ; под ред. В. С. Зарубина, А. П. Крищенко. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2002. – 744 с.

3. Галушкина, Ю. И. Конспект лекций по дискретной математике / Ю. И. Галушкина, А. Н. Марьямов. – М. : Айрис-пресс, 2007. – 176 с.

4. Плотников, А. Д. Дискретная математика : учеб. пособие / А. Д. Плотников. – М. : Новое знание, 2006. – 304 с.

5. Просветов, Г. И. Дискретная математика: задачи и решения : учеб. пособие / Г. И. Просветов. – М. : БИНОМ. Лаборатория знаний, 2008. – 222 с.

6. Соболева, Т. С. Дискретная математика : учеб. для студентов вузов / Т. С. Соболева, А. В. Чечкин ; под ред. А. В. Чечкина. – М. : Академия, 2006. – 256 с.

7. Хаггарти, Р. Дискретная математика для программистов : учеб. пособие для вузов / Р. Хаггарти ; пер. с англ. под ред. С. А. Кулешова с доп. А. А. Ковалева, В. А. Головешкина, М. В. Ульянова. – 2-е изд., испр. – М. : Техносфера, 2014. – 399 с.

Учебное электронное издание комбинированного распространения

Тихоненко Татьяна Владимировна
Задорожнюк Мария Викторовна

МЕТОДЫ И АЛГОРИТМЫ ТЕОРИИ ГРАФОВ

Практикум
по дисциплине «Дискретная математика и математическая логика» для студентов специальности 1-40 04 01 «Информатика и технологии программирования» дневной формы обучения

Электронный аналог печатного издания

Редактор *Н. В. Гладкова*
Компьютерная верстка *И. П. Минина*

Подписано в печать 12.06.19.
Формат 60x84/16. Бумага офсетная. Гарнитура «Таймс».
Ризография. Усл. печ. л. 3,25. Уч.-изд. л. 2,38.
Изд. № 29.
http://www.gstu.by

Издатель и полиграфическое исполнение
Гомельский государственный
технический университет имени П. О. Сухого.
Свидетельство о гос. регистрации в качестве издателя
печатных изданий за № 1/273 от 04.04.2014 г.
пр. Октября, 48, 246746, г. Гомель