Саратовский государственный университет
им. Н.Г.Чернышевского

М.Б.Абросимов, А.А.Долгов

ПРАКТИЧЕСКИЕ ЗАДАНИЯ ПО ГРАФАМ

Учебное пособие

Издание второе

ИЗДАТЕЛЬСТВО «НАУЧНАЯ КНИГА»
САРАТОВ 2009

УДК 519.17
ББК 22.176
А16

Абросимов М.Б., Долгов А.А.

А16 Практические задания по графам, 2-е издание: Учеб. пособие. – Саратов: Изд-во «Научная книга», 2009. – 76 с.

ISBN 978-5-9758-0905-6

Настоящее учебное пособие содержит теоретический материал и практические задания к курсу «Введение в теорию графов», читаемому в Саратовском государственном университете.

Для студентов и преподавателей математических факультетов университетов и технических вузов.

Рекомендует к печати
кафедра теоретических основ компьютерной безопасности
и криптографии
Саратовского государственного университета им. Н.Г.Чернышевского

Рецензент:
Доктор техн. наук, профессор Д.В. Сперанский

УДК 519.17
ББК 22.176

ISBN 978-5-9758-0905-6

Содержание

Предисловие

Теория графов – важный раздел современной математики с большим прикладным значением. Данное учебное пособие содержит необходимые теоретические сведения и ряд заданий, предназначенных для сопровождения теоретического курса «Теория графов». По сравнению с первым изданием увеличено количество заданий с 9 до 13 и расширен теоретический материал.

В пособии предлагается 13 заданий, которые включают теоретические задания (№ 1-11), практическое (№ 12) и аналитическое задание (№ 13). В начале каждого раздела приводится реферативное изложение теоретического материала необходимого для решения заданий.

Первые задания (№ 1-4) связаны с основными свойствами неориентированных графов: нахождение группы автоморфизмов графа, подобных вершин, минимальных расширений, построение реализаций вектора степеней и степенного множества.

Вторая группа заданий (№ 5, 6) связана с деревьями: нахождение центра и центроида, кодирование и визуализация деревьев.

В третьей группе заданий (№ 7) необходимо по заданной колоде реконструировать граф.

В четвертой группе заданий (№ 7, 8) рассматриваются ориентированные графы: требуется построить факторграф по заданной эквивалентности, конденсацию, базу орграфа и минимальный проверяющий тест.

Пятая группа содержит дополнительные задания. Задание № 12 – практическое задание, выполняя которое необходимо написать программу на любом современном языке программирования.

Заключительное задание №13 содержит утверждения, которые необходимо доказать или опровергнуть.

Основные свойства графов

Неориентированным графом (везде далее просто *графом*) называется пара $G = (V, a)$, где a – симметричное и антирефлексивное отношение на множестве вершин V, называемое *отношением смежности*. Если $(u,v) \in a$, то говорят, что вершины u и v *смежны* и эти вершины соединены *ребром* (u, v). При этом (u, v) и (v, u) это одно и то же ребро, которое обозначают $\{u, v\}$. Говорят, что ребро $\{u, v\}$ *инцидентно* каждой из вершин u и v. и эти вершины называются *концевыми вершинами* или *концами* ребра $\{u, v\}$. Два ребра называются *смежными*, если они имеют общую концевую вершину.

Граф, любые две вершины которого смежны, называется *полным* и обозначается $K_{|V|}$. По определению, $K_{|V|} = G(V, V \times V - \Delta)$, где через Δ обозначено тождественное отношение на множестве V. Граф с пустым отношением смежности a называется *вполне несвязным* или *нульграфом* и обозначается $O_{|V|}$. Граф называется *двудольным*, если множество его вершин V может быть разбито на два подмножества вершин V_1 и V_2, такие что концы любого ребра графа принадлежат разным подмножествам. Если граф содержит все ребра, соединяющие вершины из множеств V_1 и V_2, то он называется *полным двудольным графом* и обозначается $K_{m, n}$, где m и n – число вершин в множествах V_1 и V_2. Граф вида $K_{1, n}$ называется *звездой*.

Степенью вершины v в неориентированном графе G будем называть количество вершин в G, смежных с v, и обозначать через $d(v)$. Вершина, не смежная ни с одной другой вершиной, называется *изолированной*, а вершина, смежная со всеми остальными вершинами, называется *полной*. Вершина называется *четной* или *нечетной* в зависимости от четности или нечетности своей степени. Для степеней вершин имеет место

Теорема (Эйлер[1], 1736). *Во всяком графе* $G = (V, a)$ *с* n *вершинами и* m *ребрами*:

1. $\sum_{v \in V} d(v) = 2m$;
2. *количество нечетных вершин четно*;
3. *по крайней мере две вершины имеют одинаковую степень*.

[1] Леонард Эйлер (Leonhard Paul Euler, 1707 – 1783) – выдающийся швейцарский математик.

Множество D вершин графа называется *доминирующим*, если любая вершина графа или принадлежит D или смежна с подходящей вершиной из D. *Минимальным доминирующим множеством* называется такое доминирующее множество, что никакое его подмножество не обладает этим свойством.

Множество вершин графа называется *независимым*, если все вершины этого множества попарно несмежны. Независимое множество называется *максимальным независимым множеством*, если добавление любой вершины графа нарушает свойство независимости. Максимальное число вершин, составляющих независимое множество, называется числом *вершинной независимости*.

Теорема. *Независимое множество максимально тогда и только тогда, когда оно доминирующее.*

Вложением графа $G_1 = (V_1, a_1)$ в граф $G_2 = (V_2, a_2)$ называется такое взаимно однозначное отображение $f : V_1 \to V_2$, что для любых вершин $u, v \in V_1$ выполняется следующее условие: $(u, v) \in a_1 \Rightarrow (f(u), f(v)) \in a_2$.

Два графа $G_1 = (V_1, a_1)$ и $G_2 = (V_2, a_2)$ называются *изоморфными*, если можно установить взаимно однозначное соответствие $f : V_1 \to V_2$, сохраняющее отношение смежности: $(u, v) \in a_1 \Leftrightarrow (f(u), f(v)) \in a_2$, для любых $u, v \in V_1$. В этом случае пишут $G_1 \cong G_2$.

Граф, вершинам которого приписаны метки, называется *помеченным*. *Непомеченным* или *абстрактным* графом называется класс изоморфных графов. Количество непомеченных графов с ростом числа вершин быстро растет: 1, 2, 4, 11, 34, 156, 1044, 12346, 274668, 12005168[1].

Изоморфное отображение графа на себя называется *автоморфизмом*. Тождественное отображение $\Delta : V \to V$ является автоморфизмом для любого графа. Граф, имеющий только тождественный автоморфизм, называется *асимметричным* или *тождественным*. Минимальным тождественным графом является одновершинный граф. Тождественных графов с числом вершин 2, 3, 4, 5 не существует, а, начиная с 6 вершин, количество тождественных графов быстро увеличивается: 8, 152, 3696, 135004, 7971848, 805364776[2].

Две вершины u и v называются *подобными*, если существует автоморфизм f, такой что $f(u) = v$. Граф, все вершины которого подобны, называется *вершинно-симметрическим*. Два ребра $\{u_1, v_1\}$ и $\{u_2, v_2\}$ называются *подобными*, если существует автоморфизм, который переводит одно ребро в другое. Граф, все ребра которого подобны, называется *реберно-симметрическим*.

[1] См. последовательность A000088: http://www.research.att.com/~njas/sequences/A000088
[2] См. последовательность A003400: http://www.research.att.com/~njas/sequences/A003400

Пример. Для заданного графа найдите подобные вершины и ребра.

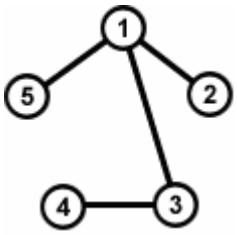


Рис. 1.

Очевидно, что подобные вершины должны иметь одинаковую степень, кроме того, должны совпадать и степени смежных вершин. Таким образом, единственными кандидатами в подобные вершины являются вершины 2 и 5, что кроме тождественного (1, 2, 3, 4, 5) дает еще один автоморфизм: (1, 5, 3, 4, 2). Таким образом, группа автоморфизмов заданного графа состоит из двух элементов. Подобные ребра, как можно заметить, должны соединять подобные вершины. В рассматриваемом графе есть лишь одна пара подобных ребер: {1, 2} и {1, 5}.•

Инвариантом графа G называется набор его характеристик, одинаковых для всех изоморфных ему графов. Инвариантами графа являются, например, число вершин графа, количество дуг или ребер. *Полным инвариантом* называется такой инвариант, который определяет граф однозначно с точностью до изоморфизма. Одним из известных числовых инвариантов является максимальный (минимальный) матричный код графа.

Матрица отношения смежности графа называется его *матрицей смежности*. Пусть $G = (V, a)$ – n-вершинный граф. Тогда его матрица смежности $A = M(a)$ имеет размерность $n \times n$, а на позиции (i, j) стоит 1, если есть дуга (v_i, v_j) и 0 в противном случае:

$$A = \begin{pmatrix} x_{1,1} & x_{1,2} & \mathbf{L} & x_{1,n} \\ x_{2,1} & x_{2,2} & \mathbf{L} & x_{2,n} \\ & & \mathbf{L} & \\ x_{n,1} & x_{n,2} & \mathbf{L} & x_{n,n} \end{pmatrix}$$

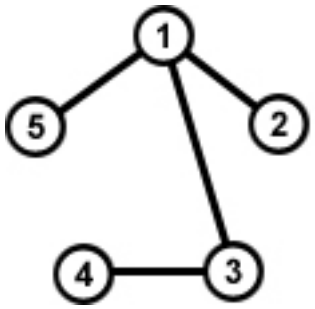
Если выписать элементы матрицы смежности по строкам, то получится некоторое двоичное число – *код* графа:

$$x_{1,1}, x_{1,2}, \ldots x_{1,n}, x_{2,1}, x_{2,2}, \ldots x_{2,n}, \ldots, x_{n,1}, x_{n,2}, \ldots x_{n,n}$$

Само по себе это число не является инвариантом, так как матрицы смежности у изоморфных графов могут различаться, однако если среди всех изо-

морфных графов выбрать матрицу смежности с максимальным (*максимальный код*) или минимальным значением (*минимальный код*) кода, то получится инвариант, причем полный. Очевидно, что для n-вершинного графа размер кода будет n^2 бит. Для некоторых классов графов, например для неориентированных графов, достаточно хранить только часть матрицы смежности, расположенную над главной диагональю. В этом случае размер кода будет составлять $n(n-1)/2$ бит.

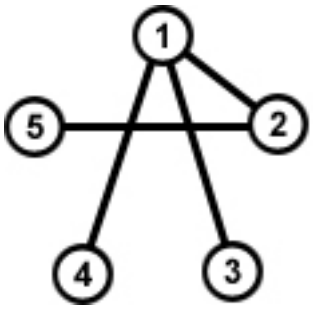
Пример. Рассмотрим 5-вершинный граф и соответствующую ему матрицу смежности (см. рис. 2)




```
(0 1 1 0 1)
(1 0 0 0 0)
(1 0 0 1 0)
(0 0 1 0 0)
(1 0 0 0 0)
```

Рис. 2.

Выпишем построчно элементы матрицы смежности, расположенные над главной диагональю, получим: $1101000100_2 = 836$. Очевидно, что для максимальности кода первая строка должна содержать максимальное количество единиц вначале строки и расположенных по порядку, то есть первой будет вершина с максимальной степенью, в нашем случае такая вершина одна – 5. Переберем все возможные перестановки оставшихся четырех вершин и найдем матрицу с максимальным кодом (см. рис. 3).

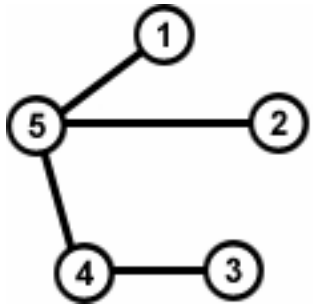



```
(0 1 1 1 0)
(1 0 0 0 1)
(1 0 0 0 0)
(1 0 0 0 0)
(0 1 0 0 0)
```

Рис. 3.

Таким образом, максимальный матричный код рассматриваемого графа равен $1110001000_2 = 904$.

Аналогичным образом находим минимальный матричный код (см. рис. 4): $0001001101_2 = 77$.




```
(0 0 0 0 1)
(0 0 0 0 1)
(0 0 0 1 0)
(0 0 1 0 1)
(1 1 0 1 0)
```

Рис. 4.

Матрица инцидентности для n-вершинного графа с m ребрами имеет n строк и m столбцов. Зафиксируем некоторый порядок вершин $v_1, \ldots, v_n$ и ребер $e_1, \ldots, e_m$. На позиции (i, j) матрицы инцидентности стоит 1, если вершина v_i инцидентна ребру e_j и 0 в противном случае.

Матрица смежности ребер для графа с m ребрами имеет m строк и m столбцов. Зафиксируем некоторый порядок ребер $e_1, \ldots, e_m$. На позиции (i, j) матрицы смежности ребер стоит 1, если ребро e_i смежно ребру e_j и 0 в противном случае.

Над графами можно рассмотреть несколько операций.

Дополнением графа $G = (V, a)$ называется граф $G' = (V, a')$, где $a' = (V \times V) \setminus (a \cup \Delta)$. Граф, изоморфный своему дополнению, называется *самодополнительным*.

Соединением двух графов $G_1 = (V_1, a_1)$ и $G_2 = (V_2, a_2)$, не имеющих общих вершин, называется граф $G_1 + G_2 := (V_1 \cup V_2, a_1 \cup a_2 \cup V_1 \times V_2 \cup V_2 \times V_1)$.

Объединением двух графов $G_1 = (V_1, a_1)$ и $G_2 = (V_2, a_2)$ называется граф $G_1 \cup G_2 := (V_1 \cup V_2, a_1 \cup a_2)$.

Набор чисел, являющихся степенями вершин графа G, называют его *степенным множеством*, а вектор, составленный из степеней вершин графа G в порядке убывания, – *вектором степеней*. Говорят, что граф является *реализацией* своего вектора степеней и степенного множества. Если вектор степеней имеет единственную реализацию, то говорят, что вектор степеней определяет *унограф*. Все графы с числом вершин меньше 5 являются унографами. *Однородным* или *регулярным* n-вершинным графом $R_{n,p}$ порядка p называют граф, в котором все степени вершин равны p.

Теорема (Капур[1], Полимени[2], Уолл[3], 1977). *Для любого множества натуральных чисел* $A = \{d_1, \ldots, d_n\}$, $k \geq 1$, *где* $d_1 < \ldots < d_n$ *существует граф с* $d_k + 1$ *вершинами, для которого* A *является степенным множеством.*

В доказательстве теоремы предлагается рекурсивный алгоритм построения графа с заданным степенным множеством.

При $n = 1$ графом со степенным множеством $A = \{d\}$ является полный граф K_{d+1}.

При $n = 2$ степенное множество $A = \{d_1, d_2\}$ имеет граф $K_{d_1} + O_{d_2 - d_1 + 1}$.

При $n > 2$ граф G со степенным множеством $A = \{d_1, \ldots, d_n\}$ строится следующим образом. Обозначим через G_0 граф со степенным множеством $\{d_2 - d_1, \ldots, d_{n-1} - d_1\}$. Тогда граф G имеет вид: $G = K_{d_1} + (O_{d_n - d_{n-1}} \cup G_0)$.

В отличие от множеств не любой вектор является *графичным*, то есть вектором степеней некоторого графа. Известно несколько критериев графичности векторов.

Теорема (Эрдёш[4], Галлаи[5], 1960). *Вектор* $d = (d_1, \ldots, d_n)$, *в котором* $d_1 \leq \ldots \leq d_n$, *тогда и только тогда является графическим, когда для каждого* $k = 1, \ldots, n - 1$ *выполняется неравенство*

$$\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min\{k, d_i\}.$$

Теорема (Гавел[6], 1955, Хакими[7], 1962). *Пусть дан вектор* $d = (d_1, \ldots, d_n)$, *в котором* $d_1 \leq \ldots \leq d_n$. *Если для какого-либо индекса* i, $1 \leq i \leq n$, *производный вектор* $d^i = (d_1 - 1, \ldots, d_{i-1} - 1, d_{i+1}, \ldots, d_n)$ *является графическим, то и вектор* d *является графическим. Если вектор* d *графический, то каждая последовательность* d^i $(i = 1, \ldots, n)$ *является графической.*

Из теоремы непосредственно следует процедура *layoff* построения реализации заданного вектора степеней $(d_1, \ldots, d_n)$:

Шаг 1. Возьмем n вершин и припишем им метки $d_1, \ldots, d_n$.

Шаг 2. Выберем в качестве ведущей вершину с максимальным значением метки d. Если $d = 0$, то процедура завершена.

Шаг 3. Ведущая вершина соединяется d ребрами с вершинам, имеющими максимальные значения меток. Ведущая вершина получает метку 0, а у вершин, с которыми она была соединена, метка уменьшается на 1. Переходим к шагу 2.

---

[1] Шаши Капур (Shashi. F. Kapoor) – индийский математик.
[2] Альберт Полимени (Albert D. Polimeni) – американский математик.
[3] Куртис Уолл (Curtiss E. Wall) – американский математик.
[4] Пол Эрдёш (Paul Erdős, 1913 – 1996) – венгерский математик.
[5] Тибор Галлаи (Tibor Gallai, 1912 – 1992) – венгерский математик.
[6] Иван Гавел (Ivan Havel, род. 1938) – чешский математик.
[7] Сейфола Хакими (Seifollah Louis Hakimi) – американский математик.

На рисунке 5 показан пример построения реализации вектора степеней (2,2,2,1,1) по описанному алгоритму. Черным цветом выделяется ведущая вершина:

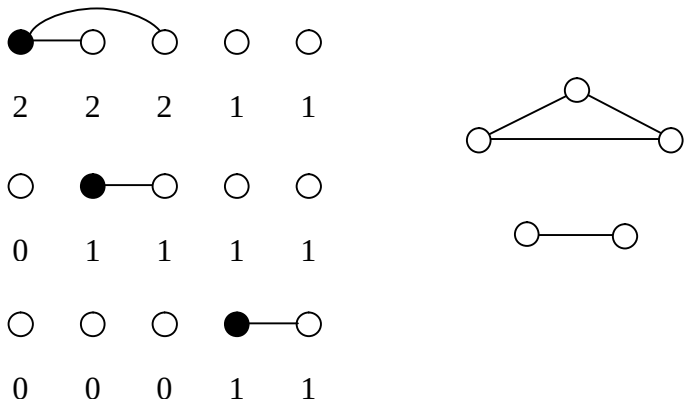


Рис. 5.

Различным образом выбирая ведущую вершину, можно получать реализации вектора степеней, обладающие нужными свойствами.

Теорема. *Вектор степеней* $d = (d_1, \ldots, d_n)$ *может быть реализован связным графом тогда и только тогда, когда* $d_n > 0$ *и верно неравенство*

$$\sum_{i=1}^{n} d_i \geq 2(n-1).$$

Если указанные условия выполняются, то процедура layoff, *на каждом шаге которой ведущей является вершина с минимальной положительной меткой, приводит к построению связного графа.*

Теорема. *Вектор степеней* $d = (d_1, \ldots, d_n)$ *может быть реализован деревом тогда и только тогда, когда* $d_n > 0$ *и верно неравенство*

$$\sum_{i=1}^{n} d_i = 2(n-1).$$

Если указанные условия выполняются, то процедура layoff, *на каждом шаге которой ведущей является вершина с минимальной положительной меткой, приводит к построению дерева.*

На рисунке 6 представлен пример построения связной реализации вектора степеней (2, 2, 2, 1, 1), которая является деревом.

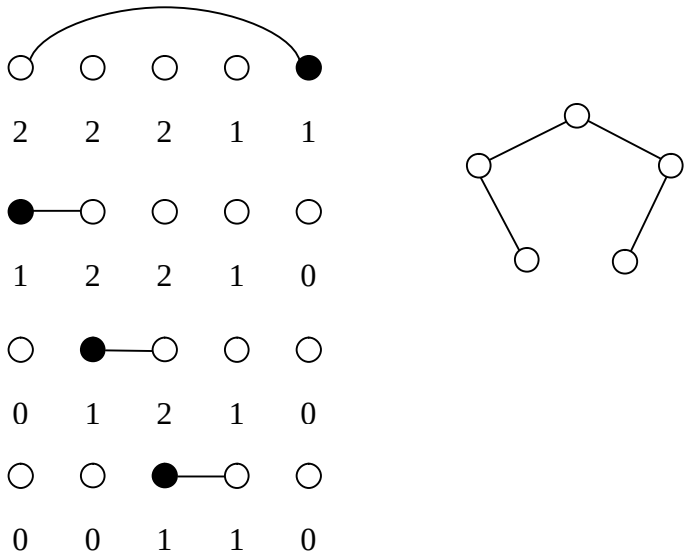


Рис. 6.

Теорема (Чангфейзен, 1978). *Если существует реализация вектора степеней* $d = (d_1, \ldots, d_n)$, *имеющая гамильтонову цепь с началом в вершине степени* d_i, *то к такой реализации приведет процедура* layoff, *в которой первый раз ведущей выбирается вершина степени* d_i, *а на каждом последующем – вершина с минимальной положительной степенью из вершин, соединенных с ведущей на предыдущем шаге*.

Различным образом выбирая вершины на шаге 3, можно получить все неизоморфные реализации вектора степеней. Другой способ получения всех неизоморфных реализаций заданного вектора степеней состоит в выполнении переключений ребер. Пусть в графе G есть ребра $\{u_1, u_2\}$ и $\{v_1, v_2\}$. Рассмотрим граф G', который получается из G удалением ребер $\{u_1, u_2\}$ и $\{v_1, v_2\}$ и добавлением ребер $\{u_1, v_1\}$ и $\{u_2, v_2\}$. Очевидно, что вектора степеней графов G и G' будут одинаковые. Имеет место

Теорема. *Всякая реализация графической последовательности может быть получена из любой другой ее реализации посредством применения подходящей цепочки переключений.*

Путем в графе $G = (V, a)$ называется последовательность вершин и ребер вида $v_0, \{v_0, v_1\}, v_1, \ldots, \{v_{n-1}, v_n\}, v_n$. При этом говорят, что v_0 – *начальная вершина* пути, а v_n – *конечная*. Говорят также, что путь соединяет вершины v_0 и v_n и вершина v_n *достижима* из v_0. Путь в графе можно задавать перечислением входящих в него вершин в порядке их прохождения: $v_0, \ldots, v_n$. Если начальная и конечная вершины совпадают, то путь называется *циклическим*. Путь, каждая

вершина которого принадлежит не более чем двум его ребрам, считается *простым*. Если начальная вершина простого пути совпадает с конечной, путь называют *циклом*, в противном случае – *цепью*.

Будем считать, что каждая вершина достижима из самой себя. Тогда отношение достижимости является отношением эквивалентности на множестве вершин графа. Классы этого отношения называются *компонентами связности* графа. Граф с универсальным отношением достижимости называется *связным*.

Вершина v в графе G называется *точкой сочленения*, если ее удаление увеличивает число компонент связности. Связный граф без точек сочленения называется *неразделимым*. Ребро $\{u, v\}$ в графе G называется *мостом*, если его удаление увеличивает число компонент связности.

Если после удаления любых k вершин вместе с инцидентными им ребрами граф остается связным, то говорят, что он *k-вершинно связный*. Наибольшее k, при котором граф G k-вершинно связный, называется *вершинной связностью* графа G. Граф с точкой сочленения имеет вершинную связность равную 1.

Если после удаления любых k ребер граф остается связным, то говорят, что он *k-реберно связный*. Наибольшее k, при котором граф G k-реберно связный, называется *реберной связностью* графа G. Граф с мостом имеет реберную связность равную 1.

В связном графе расстояние $d(u, v)$ между вершинами u и v есть длина кратчайшей цепи, соединяющей эти вершины. Расстояние от вершины u графа $G = (V, a)$ до наиболее удаленной от нее вершины называется *эксцентриситетом* $e(u)$ вершины u: $e(u) = \max\limits_{v \in V} d(u,v)$.

Наименьший из эксцентриситетов вершин называется *радиусом* $r(G)$ графа G, а наибольший – *диаметром* $d(G)$:

$$r(G) = \min_{u \in V} e(u) = \min_{u \in V} \max_{v \in V} d(u,v), \; d(G) = \max_{u \in V} e(u) = \max_{u \in V} \max_{v \in V} d(u,v).$$

Вершина, эксцентриситет которой равен радиусу, называется *центральной*. *Центр* графа – это множество его центральных вершин. Вершина, эксцентриситет которой равен диаметру, называется *периферийной*. *Окраина* графа – это множество его периферийных вершин.

Связный граф без циклов называется *деревом*. Дерево с одной вершиной называется *тривиальным*. В таблице приведено количество связных[1] n-вершинных графов и деревьев[2]:

[1] См. последовательность A001349: http://www.research.att.com/~njas/sequences/A001349
[2] См. последовательность A000055: http://www.research.att.com/~njas/sequences/A000055

n	1	2	3	4	5	6	7	8	9	10
Связные	1	1	2	6	21	112	853	11117	261080	11716571
Деревья	1	1	1	2	3	6	11	23	47	106

Цепью P_n называется граф $G = (V, a)$, где $V = \{v_1, v_2, \ldots, v_n\}$, и $a = \{(v_i, v_j): |i - j| = 1\}$, а *циклом* C_n – граф $G = (V, a)$, где $V = \{v_1, v_2, \ldots, v_n\}$, и $a = \{(v_i, v_j): |i - j| = 1\} \cup \{(v_1, v_n), (v_n, v_1)\}$.

Путь, который содержит все ребра графа, называется *эйлеровым*. Циклический путь, который содержит все ребра графа, называется *эйлеровым циклом*, а граф с таким путем – *эйлеровым*. Граф, содержащий эйлеров путь, но не цикл, называется *полуэйлеровым*. Следующие теоремы дают эффективные критерии для проверки эйлеровости и полуэйлеровости графа.

Теорема (Эйлер, 1736). *Связный граф тогда и только является эйлеровым, когда все его вершины четны.*

Теорема (Эйлер, 1736). *Связный граф G тогда и только тогда имеет эйлеров путь между вершинами u и v, когда эти вершины являются единственными нечетными вершинами в графе G.*

Цикл или цепь, содержащие все вершины графа, называются *гамильтоновыми*. Граф, содержащий гамильтонов цикл, также называется *гамильтоновым*. Для проверки гамильтоновости произвольного графа нет эффективных условий, приведем несколько достаточных условий гамильтоновости.

Теорема (Хватал[1], 1972). *Пусть G граф с вектором степеней* $(d_n, \ldots, d_1)$ *и* $n \geq 3$. *Если для любого k верна импликация*

$$(d_k \leq k < n / 2) \rightarrow (d_{n-k} \geq n - k),$$

то граф G гамильтонов.

Теорема (Оре[2], 1960). *Если в связном n-вершинном графе* $(n \geq 3)$ *для любых двух несмежных вершин u и v выполняется неравенство* $d(u) + d(v) \geq n$, *то этот граф гамильтонов.*

Граф, каждый максимальный подграф которого является гамильтоновым, называется *гипоциклическим*. Если гипоциклический граф сам не является гамильтоновым, то он называется *гипогамильтоновым*. Минимальным по числу вершин гипогамильтоновым графом является граф Петерсена[3], изображенный на рисунке 7.

[1] Вацлав Хватал (Václav Chvatal, род. 1946) – чешский математик.
[2] Ойстин Оре (Øystein Ore, 1899 – 1968) – норвежский математик.
[3] Юлиус Петерсен (Julius Peter Christian Petersen, 1839 – 1910) – датский математик.

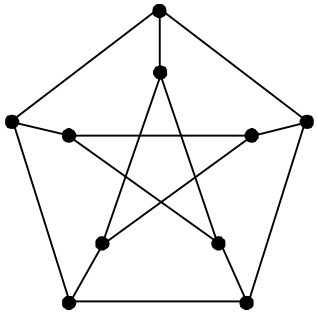
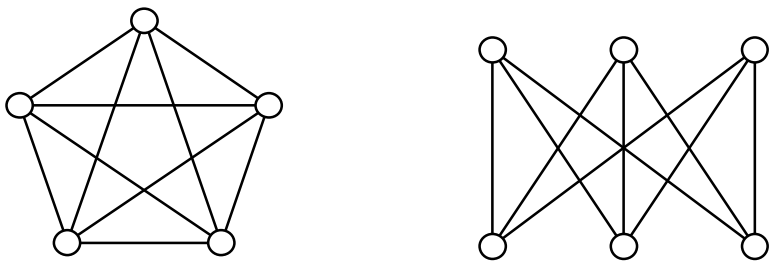


Рис. 7.

Подграфом графа $G = (V, \alpha)$ называется пара $G' = (V', \alpha')$, где $V' \subseteq V$ и $a' = (V' \times V') \cap a$. Подграф графа G называется *максимальным*, если он получается из G удалением одной вершины и всех связанных с нею ребер. Будем обозначать через $G - v$ максимальный подграф, получающийся из графа G удалением вершины v. Через $G - F$, где $F \subset V$, будем обозначать подграф, получающийся из графа G удалением всех вершин из F. Список максимальных подграфов графа G называют его *колодой*.

Изображение графа на плоскости без пересечения в ребрах, отличных от вершин, называется *плоским* изображением. Граф, который допускает плоское изображение, называется *планарным*. Не всякий граф является планарным, например, на рисунке изображены полный 5-вершинный граф K_5 и полный двудольный граф $K_{3,3}$ – эти графы не являются планарными.



Подразбиением ребра $\{u, v\}$ в графе называется операция, состоящая из удаления ребра $\{u, v\}$, добавления новой вершины w и двух ребер $\{u, w\}$ и $\{v, w\}$. Два графа называются *гомеоморфными*, если они могут быть получены из одного графа с помощью последовательности подразбиений ребер.

Теорема (Понтрягин[1], 1927, Куратовский[2], 1930). *Граф планарен тогда и только тогда, когда он не содержит частей гомеоморфных графу K_5 или графу $K_{3,3}$.*

[1] Лев Семенович Понтрягин (1908 – 1988) – русский математик.
[2] Казимеж Куратовский (Kazimierz Kuratowski, 1896 – 1980) – польский математик.

Раскраской графа называется такое приписывание цветов его вершинам, что никакие две смежные вершины не получают одинакового цвета. Если граф можно раскрасить в k цветов, то говорят, что он обладает k-раскраской и является *k-раскрашиваемым*. Наименьшее k, при котором граф G имеет k-раскраску, называется *хроматическим числом* графа G и обозначается $\chi(G)$. При этом говорят, что граф k-хроматический. Очевидно, что

1) 1-хроматические графы – это вполне несвязные графы и только они;
2) 2-хроматические графы – это непустые двудольные графы и только они;
3) $\chi(K_n) = n$;
4) $\chi(C_{2n}) = 2$, $\chi(C_{2n+1}) = 3$.

Для *планарных* графов, то есть графов, которые могут быть изображены на плоскости без пересечения ребер в точках, отличных от вершин, известна

Теорема (Хивуд[1], 1890). *Любой планарный граф может быть раскрашен в 5 цветов.*

В 1879 году английский математик Кели[2] сформулировал гипотезу о том, что любой планарный граф может быть раскрашен в 4 цвета. В 1976 году Хейкен[3] и Аппель[4] предложили компьютерное доказательство этой гипотезы, однако аналитического доказательства гипотезы 4-х красок пока нет.

Назовем граф $G_R = (V_R, a_R)$ *вершинным k-расширением* графа $G = (V, a)$, если граф G можно вложить в каждый подграф графа G_R, получающийся удалением любых его k вершин и всех связанных с ними ребер.

Граф $G_t = (V_t, a_t)$ называется *тривиальным k-расширением* графа $G = (V, a)$, если граф G_t получается из графа G добавлением k вершин, соединением их со всеми вершинами графа G и друг с другом, то есть граф G_t есть соединение графа G и полного графа $K_k = (V_k, a_k)$:

$$G_t = (V_t, a_t) = (V \cup V_k, a \cup a_k \cup V \times V_k \cup V_k \times V).$$

Очевидно, что тривиальное k-расширение графа является и его вершинным k-расширением.

Граф $G^* = (V^*, a^*)$ называется *минимальным вершинным k-расширением* n-вершинного графа $G = (V, a)$, если выполняются следующие условия:

---

[1] Перси Джон Хивуд (Percy John Heawood, 1861 – 1955) – английский математик.
[2] Артур Кели (Arthur Cayley, 1821 – 1895) – английский математик.
[3] Вольфганг Хейкен (Wolfgang Haken, род. 1928) – американский математик.
[4] Кеннет Аппель (Kenneth Appel, род. 1932) – американский математик.

1) G^* является вершинным k-расширением графа G, то есть граф G вложим в каждый подграф графа G^*, получающийся удалением любых его k вершин;
2) G^* содержит $n+k$ вершин, то есть $|V^*| = |V| + k$;
3) a^* имеет минимальную мощность при выполнении условий 1) и 2).

Заметим, что число ребер минимального k-расширения графа не более, чем у его тривиального k-расширения, то есть $|a^*| \le |a| + |V|k + \frac{k-1}{2}k$. Будем говорить, что минимальное k-расширение содержит $|a^*| - |a|$ *дополнительных* ребер. Из определения непосредственно следует и

Алгоритм. Построение всех минимальных k-расширений для заданного графа G, отличного от вполне несвязного:

1. $m := 0$.
2. $m := m + 1$.
3. Строим все графы, получающиеся из графа G добавлением k вершин и m дополнительных ребер.
4. Выбираем среди построенных на шаге 3 графов k-расширения графа G.
5. Если на шаге 4 не было найдено подходящих графов, то переходим на шаг 2.
6. Среди графов, выбранных на шаге 4, оставляем по одному представителю от изоморфных графов. Полученные графы будут являться минимальными k-расширениями графа G.

Для поиска вершинных расширений и доказательства их минимальности могут быть полезны следующие утверждения:

Лемма 1. *Минимальное вершинное k-расширение графа без изолированных вершин не содержит вершин со степенью ниже $k + 1$.*

Лемма 2. *Пусть наибольшая из степеней вершин графа G есть s и в точности m вершин имеют такую степень, тогда минимальное вершинное k-расширение графа G содержит, по крайней мере, $k + m$ вершин со степенью не ниже s.*

Лемма 3. *Если максимальная степень вершины графа G есть $d > 0$, то его минимальное вершинное k-расширение G^* содержит не менее kd дополнительных ребер.*

Лемма 4. *Если минимальная степень вершины графа G есть $d > 0$, то его минимальное вершинное k-расширение G^* не содержит вершин степени ниже $d + k$.*

Продемонстрируем технику доказательства минимальности расширения в следующем утверждении

Теорема (Хейз[1], 1976). *Единственным с точностью до изоморфизма минимальным вершинным 1-расширением цепи P_n является цикл C_{n+1}.*

Доказательство.

При удалении любой вершины цикла C_{n+1} получается цепь P_n, следовательно, цикл C_{n+1} является вершинным 1-расширением цепи P_n.

По лемме 1 минимальное вершинное 1-расширение цепи P_n не может содержать вершин со степенью ниже 2, следовательно, цикл C_{n+1} является и минимальным вершинным 1-расширением цепи P_n.

Пусть G^* – является минимальным вершинным расширением цепи P_n. Очевидно, G^* является связным графом, причем по лемме 1 степень любой его вершины не меньше двух. Как было установлено, цикл C_{n+1}, содержащий $n + 1$ ребро, является минимальным вершинным 1-расширением цепи P_n, следовательно, граф G^* также содержит $n + 1$ ребро и, значит, степень любой его вершины в точности равна двум. Заметим, что G^* не дерево и, следовательно, содержит цикл длины k. Если $k < n + 1$, то нарушается условие связности графа G^*, следовательно, $k = n + 1$, то есть G^* изоморфен C_{n+1}.•

Назовем граф $G_R = (V_R, a_R)$ *реберным k-расширением* графа $G = (V, a)$, если граф G можно вложить в каждый подграф графа G_R, получающийся удалением любых его k ребер (дуг).

Граф $G^* = (V^*, \alpha^*)$ называется *минимальным реберным k-расширением* n-вершинного графа $G = (V, a)$, если выполняются следующие условия:

1) G^* является реберным k-расширением графа G, то есть граф G вложим в каждый подграф графа G^*, получающийся удалением любых его k вершин;
2) G^* содержит $n+k$ вершин, то есть $|V^*| = |V| + k$;
3) a^* имеет минимальную мощность при выполнении условий 1) и 2).

Не все графы имеют минимальное реберное k-расширение. Например, полный граф K_n не имеет минимального реберного k-расширения ни при каком натуральном k.

По аналогии с леммами 1-4 для вершинных расширений полезными оказываются следующие утверждения о реберных расширениях.

Лемма 5. *Минимальное реберное k-расширение графа без изолированных вершин не содержит вершин со степенью ниже $k + 1$.*

[1] Джон Хейз (John Hayes) – американский математик.

Лемма 6. *Если минимальная степень вершины графа G есть $d > 0$, то его минимальное реберное k-расширение не содержит вершин степени ниже $d + k$.*

Пример. Построить все минимальные вершинные и реберные 1-расширения графа, изображенного на рисунке 8.

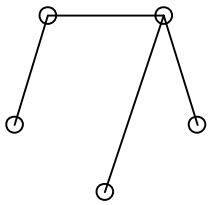


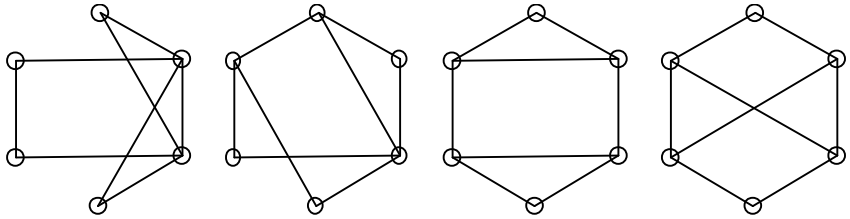
Рис. 8.

Решение. Обозначим через G заданный граф. Пусть G^* – минимальное вершинное 1-расширение G. Граф G не имеет изолированных вершин, а одна его вершина имеет степень 3, следовательно, по леммам 1 и 2 степень любой вершины G^* должна быть не ниже 2 и как минимум две вершины должны иметь степень не ниже 3. По лемме 3 G^* отличается от G не менее чем на 3 ребра.

Пусть G^* отличается от графа G на 3 дополнительных ребра. Из вышесказанного получается, что единственно возможным вектором степеней графа G^* может быть вектор (3,3,2,2,2,2). Предположим, что вершины со степенью 3 смежны. Тогда максимальный подграф, получающийся при удалении из G^* одной из этих вершин, не будет содержать ни одной вершины степени выше 2 и поэтому вложение в него графа G невозможно. Итак, вершины степени 3 в графе G^* несмежны. Каждая из них соединена с тремя вершинами степени 2. Всего таких вершин четыре и, значит, существует две вершины степени 2, каждая из которых является смежной с обеими вершинами степени 3. Тогда максимальный подграф, получающийся из G^* удалением одной из таких вершин, не будет содержать вершин степени выше 2 и, следовательно, граф G в него вкладываться не будет.

Таким образом, не существует графа G^* с 7 ребрами, который бы являлся минимальным вершинным 1-расширением графа G и, следовательно, минимальное вершинное 1-расширение графа G отличается от G не менее чем на 4 ребра.

Рассмотрим вектора 6-вершинных графов с 8 ребрами, у которых не менее двух вершин степени 3, а степени остальных вершин не менее 2: (4,4,2,2,2,2), (4,3,3,2,2,2) и (3,3,3,3,2,2). С помощью процедуры *layoff* и переключений построим все реализации указанных векторов.



а *б* *в* *г*

Рис. 9.

Вектор (4,4,2,2,2,2) имеет единственную реализацию G^*_{51}, изображенную на рисунке 9*а*.

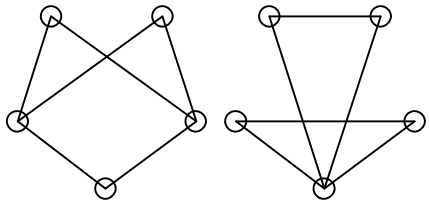
Вектор (4,3,3,2,2,2) имеет 4 реализаций, однако сразу можно отбросить те реализации, в которых вершина степени 4 смежна с двумя вершинами степени 3 (после удаления такой вершины, не останется вершин степени 3). В результате остается одна реализация G^*_{52}, изображенная на рисунке 9*б*.

Среди 4-х реализаций вектора (3,3,3,3,2,2) нас интересуют только такие, у которых нет вершины степени 3, смежной с остальными вершинами степени 3. Таких реализаций всего 2 – граф G^*_{53} и G^*_{54}, изображенные на рисунках 9*в* и 9*г*.

Непосредственной проверкой убеждаемся, что все четыре графа являются расширениями графа G. Рассмотрим для примера граф G^*_{51}. Нетрудно заметить, что вершины степени 4 и все вершины степени 2 подобны. Достаточно рассмотреть два максимальных подграфа графа G^*_{51}, получающиеся удалением вершины степени 4 и вершины степени 2, и убедиться, что они допускают вложение графа G.

Пусть теперь G^* – минимальное реберное 1-расширение G. По лемме 5 степень любой вершины G^* должна быть не ниже 2. Следовательно, G^* отличается от G не менее чем на два дополнительных ребра. Перебирая возможные способы добавления двух ребер, необходимо учесть, что в G^* должно получиться либо две несмежных вершины степени 3, либо вершина степени 4. В самом деле, если вершины степени 3 будут смежны, то удаление ребра между ними приведет к графу со степенями вершин не более 2. А если будет одна вершина степени 3, то удаление любого инцидентного ей ребра, снова приведет

к графу со степенями вершин не более 2. В итоге получается два графа, изображенные на рисунке 10.



а *б*

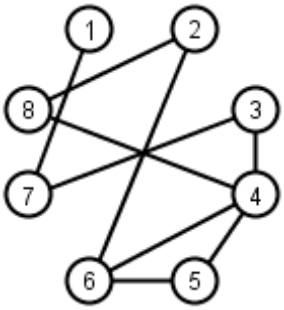
Рис. 10.

Непосредственной проверкой убеждаемся, что они являются реберными 1-расширениями графа *G*, а, следовательно, и минимальными реберными 1-расширениями.•

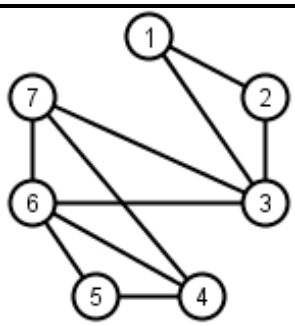
Задание 1. Характеристики графов

Для заданного графа найдите:

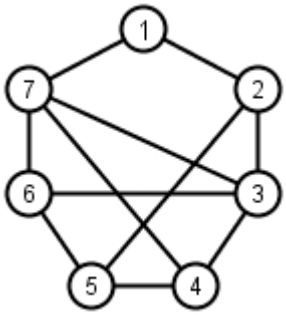
а) матрицы смежности вершин, ребер и инцидентности;
б) вектор степеней и степенное множество;
в) максимальное независимое множество вершин;
г) радиус и диаметр;
д) мосты и точки сочленения;
е) число вершинной и реберной связности;
ж) колоду;
з) плоское изображение, если граф планарный, или докажите его непланарность;
и) хроматическое число.



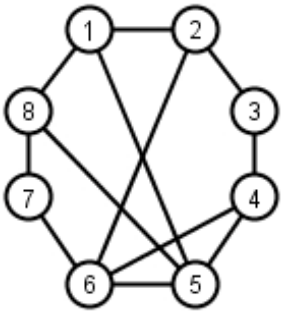
1.1



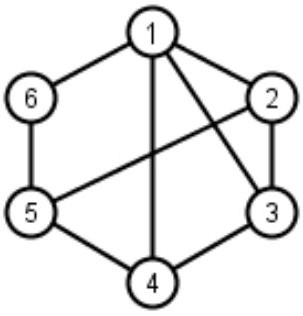
1.2



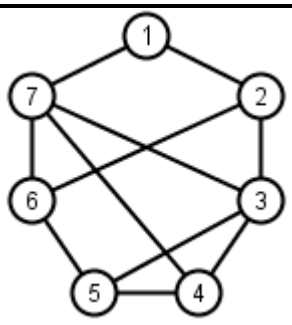
1.3



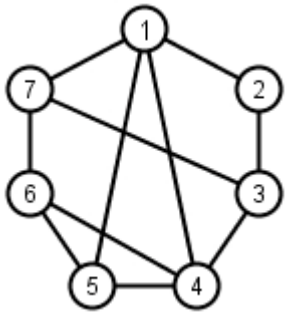
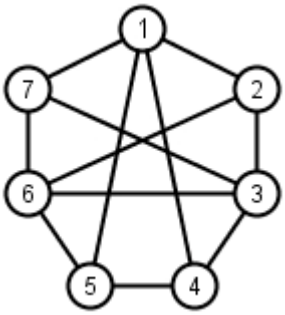
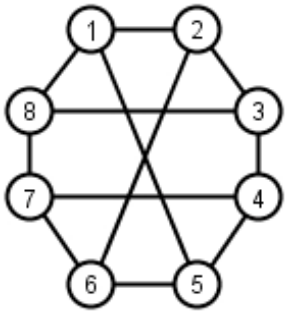
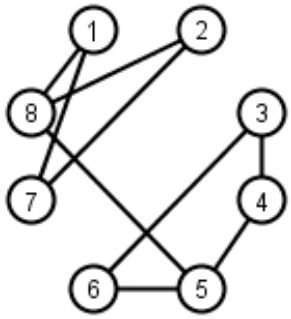
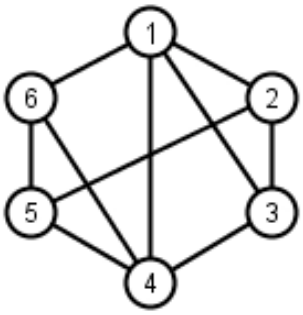
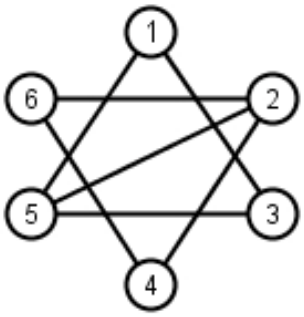
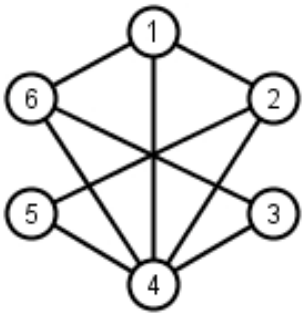
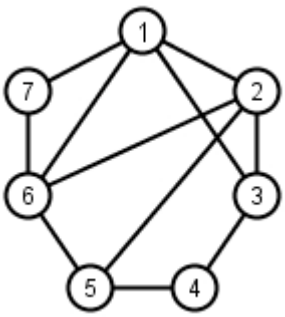
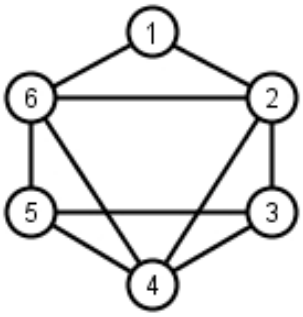
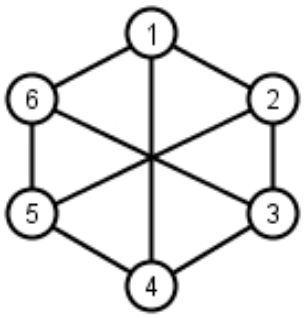
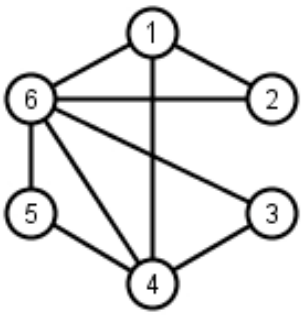
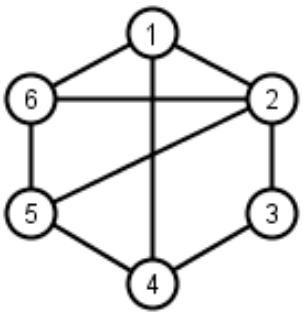
1.4

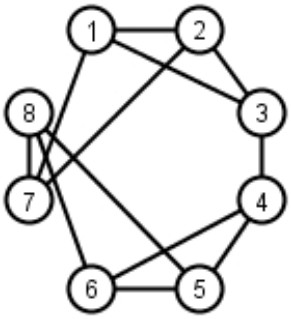
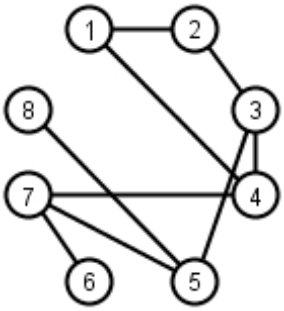
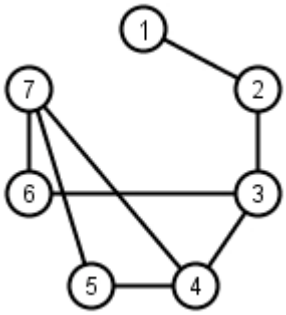
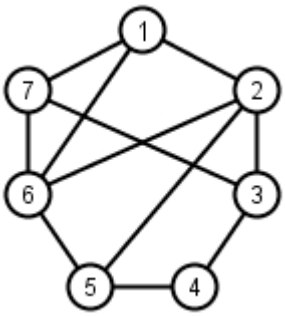
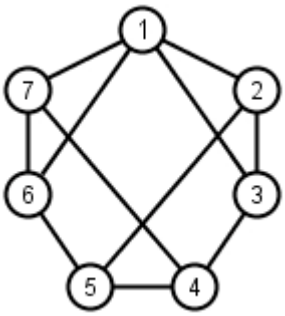
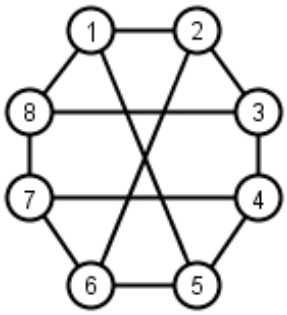
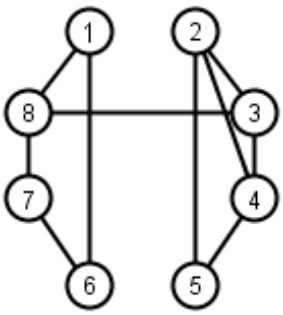
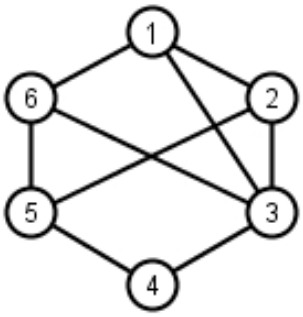
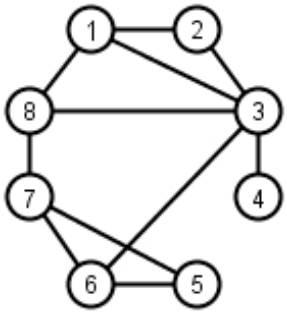


1.5



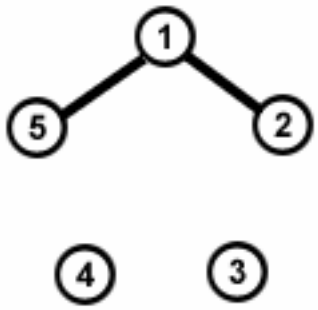
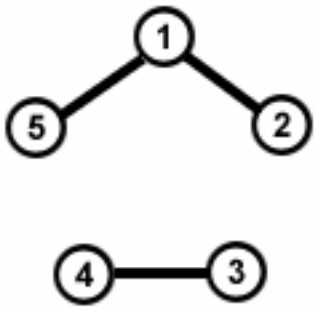
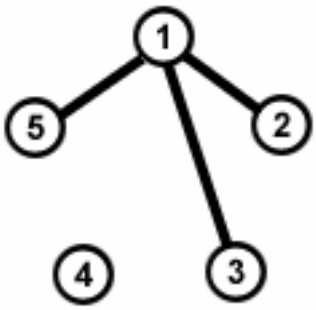
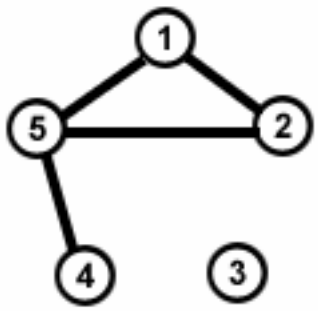
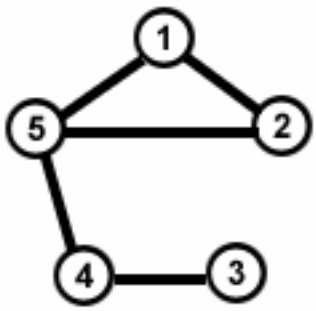
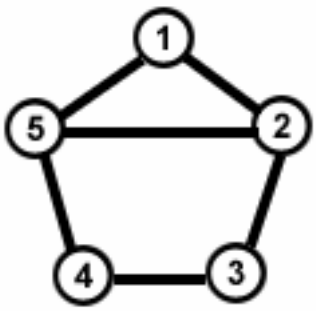
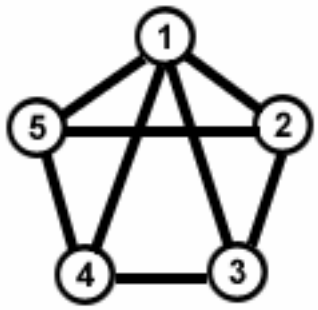
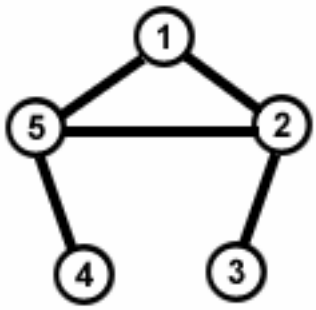
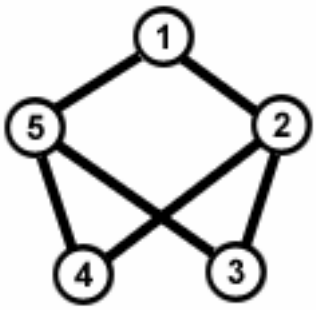
1.6

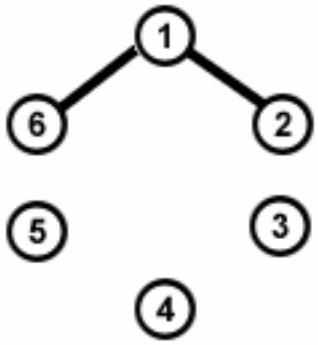
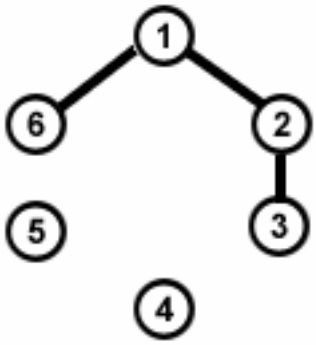
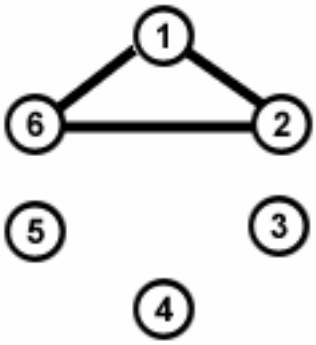
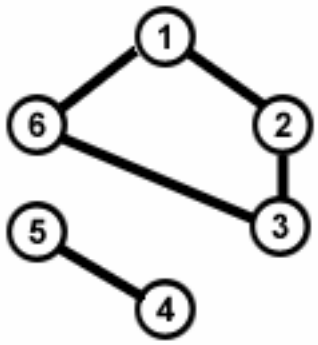
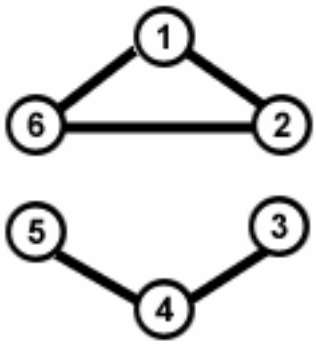
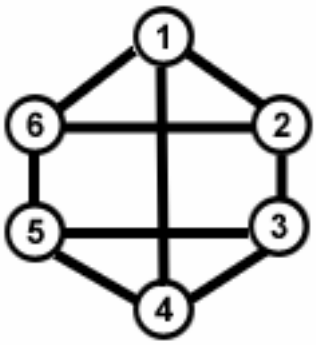
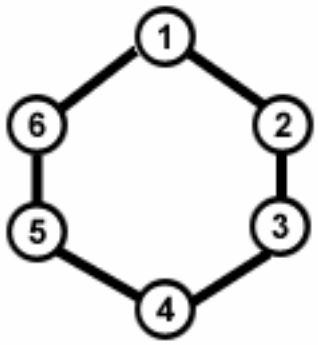
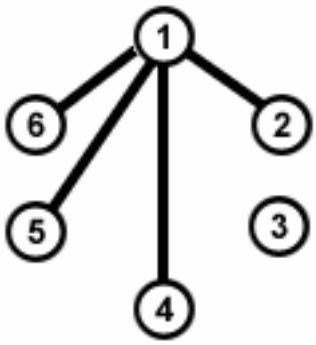
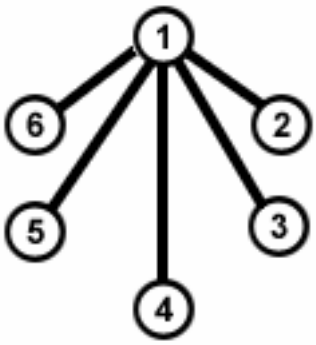
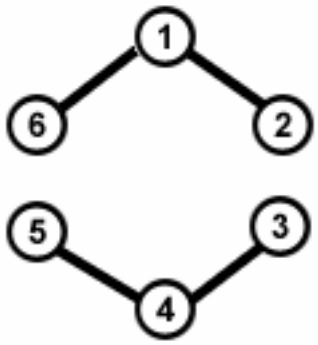
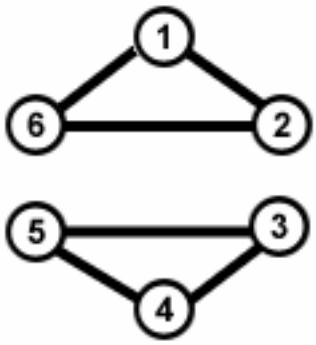
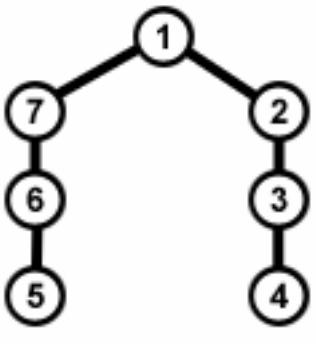
 1.7	 1.8	 1.9
 1.10	 1.11	 1.12
 1.13	 1.14	 1.15
 1.16	 1.17	 1.18

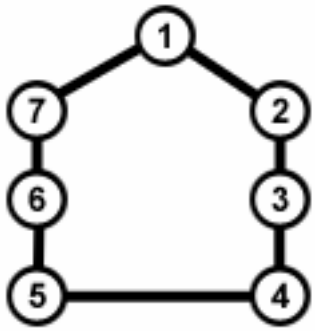
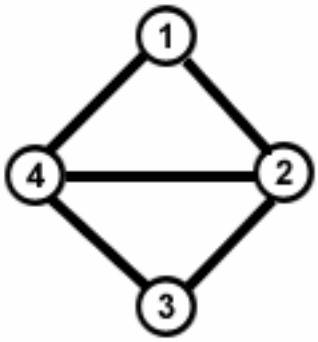
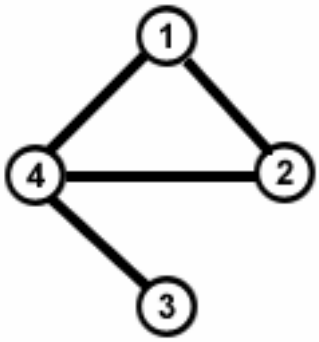
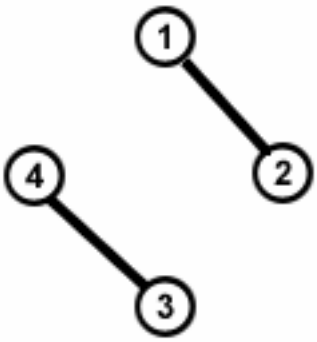
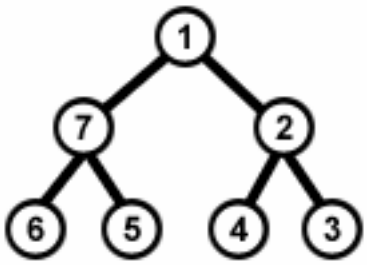
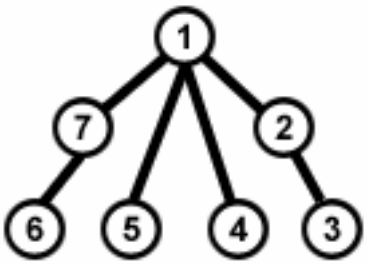
 1.19	 1.20	 1.21
 1.22	 1.23	 1.24
 1.25	 1.26	 1.27

Задание 2. Изоморфизмы и вложения графов

Для заданного графа найдите:
а) подобные вершины и ребра;
б) группу автоморфизмов;
в) максимальный матричный код;
г) минимальное вершинное и реберное 1-расширения.

 2.1	 2.2	 2.3
 2.4	 2.5	 2.6
 2.7	 2.8	 2.9

 2.10	 2.11	 2.12
 2.13	 2.14	 2.15
 2.16	 2.17	 2.18
 2.19	 2.20	 2.21

		
2.22	2.23	2.24
		
2.25	2.26	2.27

Задание 3. Степенной вектор

Для заданных векторов определите их графичность с помощью критериев Гавела-Хакими и Эрдёша-Галлаи и, если возможно, постройте реализацию по обычной процедуре layoff, связную реализацию и гамильтонову реализацию.

3.1. а) (4, 4, 4, 1, 1, 1, 1);
б) (3, 3, 3, 3, 3, 2, 2, 1);
в) (2, 2, 2, 2, 2, 1, 1).

3.2. а) (3, 3, 3, 3, 2, 2, 2);
б) (5, 4, 1, 1, 1, 1, 1);
в) (3, 2, 2, 2, 1, 1, 1).

3.3. а) (3, 3, 2, 1, 1, 1, 1);
б) (5, 4, 3, 1, 1, 1, 1);
в) (4, 4, 3, 2, 2, 2, 1, 1, 1).

3.4. а) (4, 2, 2, 1, 1, 1, 1);
б) (3, 3, 3, 3, 2, 2, 2);
в) (5, 4, 4, 2, 1, 1, 1).

3.5. а) (5, 4, 4, 4, 1, 1, 1);
б) (4, 4, 3, 3, 3, 1, 1, 1);
в) (2, 2, 2, 2, 2, 2, 1, 1).

3.6. а) (3, 2, 2, 2, 2, 1, 1, 1);
б) (5, 5, 2, 1, 1, 1, 1);
в) (8, 8, 7, 7, 7, 6, 5, 5, 3).

3.7. а) (3, 3, 2, 2, 1, 1, 1, 1);
б) (5, 4, 4, 3, 3, 3, 2);
в) (5, 5, 3, 2, 1, 1, 1).

3.8. а) (5, 5, 4, 1, 1, 1, 1);
б) (4, 3, 3, 2, 2, 2, 2, 1, 1);
в) (3, 3, 3, 1, 1, 1, 1, 1).

3.9. а) (4, 2, 2, 2, 1, 1, 1, 1);
б) (5, 5, 4, 2, 2, 1, 1);
в) (5, 4, 4, 4, 4, 3, 2).

3.10. а) (2, 2, 2, 2, 2, 2, 2, 1, 1);
б) (6, 5, 3, 3, 3, 3, 2, 1);
в) (5, 5, 4, 3, 1, 1, 1).

3.11. а) (3, 2, 2, 2, 2, 2, 1, 1, 1);
б) (5, 5, 4, 4, 2, 1, 1);
в) (5, 5, 4, 4, 4, 4, 2).

3.12. а) (5, 5, 5, 2, 1, 1, 1);
б) (7, 5, 5, 5, 5, 5, 4, 1, 1);
в) (3, 3, 2, 2, 2, 1, 1, 1, 1).

3.13. а) (3, 3, 3, 2, 1, 1, 1, 1, 1);
б) (6, 6, 5, 5, 5, 5, 5, 4, 3);
в) (5, 5, 5, 2, 2, 2, 1).

3.14. а) (4, 3, 2, 2, 2, 1, 1, 1);
б) (5, 5, 5, 3, 2, 1, 1);
в) (4, 2, 2, 2, 2, 1, 1, 1, 1).

3.15. а) (4, 3, 2, 2, 1, 1, 1, 1, 1);
б) (4, 4, 3, 3, 2, 1, 1);
в) (5, 5, 5, 4, 1, 1, 1).

3.16. а) (4, 4, 2, 1, 1, 1, 1, 1, 1);
б) (5, 5, 5, 4, 3, 1, 1);
в) (4, 4, 3, 3, 2, 2, 2).

3.17. а) (5, 5, 5, 5, 2, 1, 1);
б) (4, 4, 3, 3, 3, 1, 1, 1);
в) (5, 2, 2, 2, 1, 1, 1, 1, 1).

3.18. а) (5, 3, 2, 1, 1, 1, 1, 1, 1);
б) (5, 5, 5, 5, 2, 2, 2);
в) (6, 6, 5, 4, 3, 3, 3, 2).

3.19. а) (2, 2, 2, 2, 2, 2, 2, 2, 1, 1);
б) (5, 5, 5, 5, 3, 2, 1);
в) (4, 4, 3, 3, 3, 2, 1).

3.20. а) (5, 5, 5, 5, 4, 1, 1);
б) (5, 4, 3, 3, 3, 2, 2, 2);
в) (3, 2, 2, 2, 2, 2, 2, 1, 1, 1).

3.21. а) (5, 5, 5, 5, 5, 2, 1);
б) (3, 3, 2, 2, 2, 2, 1, 1, 1, 1);
в) (4, 4, 3, 3, 3, 3, 2).

3.22. а) (3, 3, 3, 2, 2, 1, 1, 1, 1, 1);
б) (5, 5, 4, 4, 3, 3, 2);
в) (6, 3, 1, 1, 1, 1, 1).

3.23. а) (7, 6, 5, 4, 4, 4, 4, 3, 3);
б) (6, 3, 3, 1, 1, 1, 1);
в) (3, 3, 3, 3, 1, 1, 1, 1, 1, 1).

3.24. а) (4, 2, 2, 2, 2, 2, 1, 1, 1, 1);
б) (5, 5, 4, 4, 4, 3, 3);
в) (6, 4, 2, 1, 1, 1, 1).

3.25. а) (5, 5, 4, 4, 4, 3, 3);
б) (6, 4, 3, 2, 1, 1, 1);
в) (4, 3, 2, 2, 2, 1, 1, 1, 1, 1).

3.26. а) (4, 3, 3, 2, 1, 1, 1, 1, 1, 1);
б) (6, 4, 4, 1, 1, 1, 1);
в) (6, 6, 6, 4, 4, 4, 4).

3.27. а) (4, 4, 2, 2, 1, 1, 1, 1, 1, 1);
б) (5, 3, 3, 3, 3, 3, 2, 2);
в) (6, 4, 4, 2, 2, 1, 1).

3.28. а) (6, 4, 4, 3, 1, 1, 1);
б) (4, 4, 3, 1, 1, 1, 1, 1, 1, 1);
в) (4, 3, 3, 3, 2, 2, 2, 2, 1).

Задание 4. Степенное множество

Постройте реализацию заданного степенного множества с минимально возможным числом вершин.

4.1. {3, 2, 1}.
4.2. {4, 3, 2}.
4.3. {4, 3, 1}.
4.4. {4, 2, 1}.
4.5. {4, 3, 0}.
4.6. {5, 4, 3}.
4.7. {5, 4, 2}.
4.8. {5, 4, 1}.
4.9. {5, 4, 0}.
4.10. {5, 3, 2}.
4.11. {5, 3, 1}.
4.12. {6, 5, 4}.
4.13. {6, 4, 3}.
4.14. {6, 4, 2}.
4.15. {6, 4, 1}.
4.16. {6, 4, 0}.
4.17. {6, 3, 2}.
4.18. {6, 3, 1}.
4.19. {6, 2, 1}.
4.20. {7, 6, 5}.
4.21. {7, 6, 4}.
4.22. {7, 6, 3}.
4.23. {7, 6, 2}.
4.24. {7, 5, 2}.
4.25. {7, 4, 2}.
4.26. {7, 3, 2}.
4.27. {7, 3, 1}.

Деревья

Деревом называется связный граф без циклов. Дерево с выделенной вершиной r называется *корневым*, а вершина r – *корнем*. Некорневое дерево называется *свободным*. Вершина степени 1 дерева называется *листом* или *висячей вершиной*.

Теорема (Жордан[1], 1869). *Центр дерева состоит или из одной или из двух смежных вершин.*

Если центр дерева состоит из двух вершин, то такое дерево называется *бицентральным*.

При решении многих задач удобно выбирать одну из центральных вершин в качестве корня. Легко заметить, что если удалить из дерева все листья, то центр не изменится, и каждая цепь максимальной длины проходит через центр.

Ветвь к вершине u дерева T – это максимальное поддерево, содержащее u в качестве листа. Число ветвей к вершине u очевидно равно $d(u)$. Вес вершины u дерева T определяется как наибольшее число ребер по всем ветвям к u. Вершина v называется *центроидной* вершиной дерева T, если она имеет наименьший вес, а множество всех центроидных вершин называется *центроидом*.

Теорема (Жордан, 1869). *Центроид дерева состоит или из одной или из двух смежных вершин.*

В общем случае центр дерева и его центроид не совпадают. На рисунке показаны примеры таких деревьев.

Рис. 11.

Рассмотрим несколько специфических способов кодирования свободных и корневых деревьев.

Код Прюфера

[1] Камиль Мари Эдмон Жордан (1838 – 1922) – французский математик.

Пусть T – дерево с множеством вершин $\{v_1, v_2, \ldots, v_n\}$. Будем считать, что номер вершины v_i равен i. Сопоставим дереву T вектор $(a_1, a_2, \ldots, a_{n-2})$ по следующему алгоритму.

1. Найти висячую вершину с минимальным номером i.
2. Записать очередным элементом кода Прюфера номер вершины смежной с i, а вершину i удалить из дерева.
3. Если число вершин дерева больше 2, перейти к шагу 1.

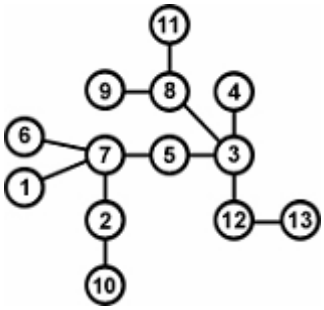


Рис. 12.

Дерево, изображенное на рисунке 12, имеет такой код Прюфера:

(7, 3, 7, 8, 8, 3, 2, 7, 5, 3, 12).

Распаковка кода Прюфера осуществляется по следующему алгоритму:

1. Пусть T состоит из вершин $\{v_1, v_2, \ldots, v_n\}$, таких, что номер вершины v_i равен i, где n – длина кода A плюс 2;
2. $B = [1: n]$;
3. для i от 1 до $n + 1$ цикл;
4. $b = \min \{ k \in B: k \neq A[j]$ для любого $j \geq i\}$;
5. В T добавить ребро, соединяющее вершины с номерами b и $A[i]$;
6. $B = B \setminus \{b\}$;
7. В T добавить ребро, соединяющее оставшиеся в B вершины.

Рассмотрим восстановление дерева по коду Прюфера (2, 2, 2, 3, 1, 5).

Длина вектора равна 6, следовательно, число вершин в дереве $n = 8$.

j	B	$A[j],\ldots,A[n-2]$	Ребро
1	1, 2, 3, 4, 5, 6, 7, 8	2, 2, 2, 3, 1, 5	2, 4
2	1, 2, 3, 5, 6, 7, 8	2, 2, 3, 1, 5	2, 6
3	1, 2, 3, 5, 7, 8	2, 3, 1, 5	2, 7
4	1, 2, 3, 5, 8	3, 1, 5	2, 3
5	1, 3, 5, 8	1, 5	1, 3
6	1, 5, 8	5	1, 5
	5, 8		5, 8

На рисунке 13 показано результирующее дерево:

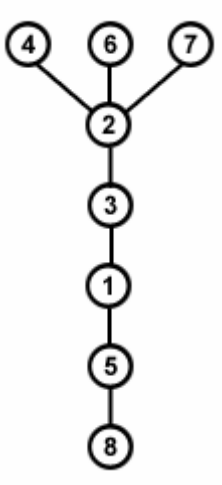


Рис. 13.

Уровневые коды

Пусть T – дерево с множеством вершин $\{v_1, v_2, \ldots, v_n\}$ и вершина z выбрана в качестве корня. Уровень вершины v в корневом дереве (T, z) – это расстояние от v до z плюс единица: $l(v) = d(z, v) + 1$. *Уровневый код* (обозначается $L(T, z) = (l_1, \ldots, l_n)$) – это последовательность целых чисел, полученная выписыванием уровней вершин дерева (T, z) в постфиксном порядке. Постфиксный порядок – это способ обхода корневого дерева, при котором сначала обходятся поддеревья слева направо, а потом посещается корень. Уровневый код называется *каноническим* (обозначается $L^*(T, z)$), если он является наибольшим в лексикографическом порядке среди всех уровневых кодов, описывающих дерево. Для дерева, изображенного на рисунке 14, имеем:

$L(T, z) = (3, 3, 2, 3, 4, 4, 3, 2, 3, 2, 1)$, $L^*(T, z) = (4, 4, 3, 3, 2, 3, 3, 2, 2, 3, 1)$.

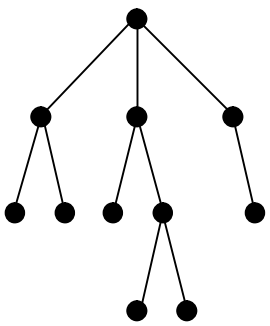


Рис. 14.

Вершина z^* называется *основным корнем* свободного дерева T, если либо z – единственный центр T, либо z – один из центров z_1 и z_2 бицентрального дерева T, определяемый по следующему правилу. Пусть T_1 и T_2 – поддеревья, получаемые из T удалением ребра, соединяющего z_1 и z_2. В качестве z берется z_1, если T_1 имеет меньше вершин, чем T_2, либо если T_1 и T_2 состоят из равного числа вершин, но $L^*(T_1, z_1)$ лексикографически предшествует $L^*(T_1, z_1)$; в остальных случаях в качестве z берется z_2. Уровневый код $L(T, z^*)$ называется *главным уровневым кодом* дерева T, а $L^*(T, z^*)$ – *главным каноническим уровневым кодом* дерева T.

Дерево из предыдущего примера является бицентральным с корнями z_1 и z_2. Удаление ребра $\{z_1, z_2\}$ приводит к деревьям T_1 и T_2 (см. рисунок 15). В дереве T_1 6 вершин, а в T_2 5 вершин, поэтому в качестве основного корня берется вершина z_2. Соответствующее корневое дерево (T, z_2) изображено на рисунке 11 справа. Главный канонический уровневый код дерева T будет иметь вид:

$$L^*(T, z_2) = (4, 4, 3, 4, 3, 2, 3, 3, 2, 2, 1).$$

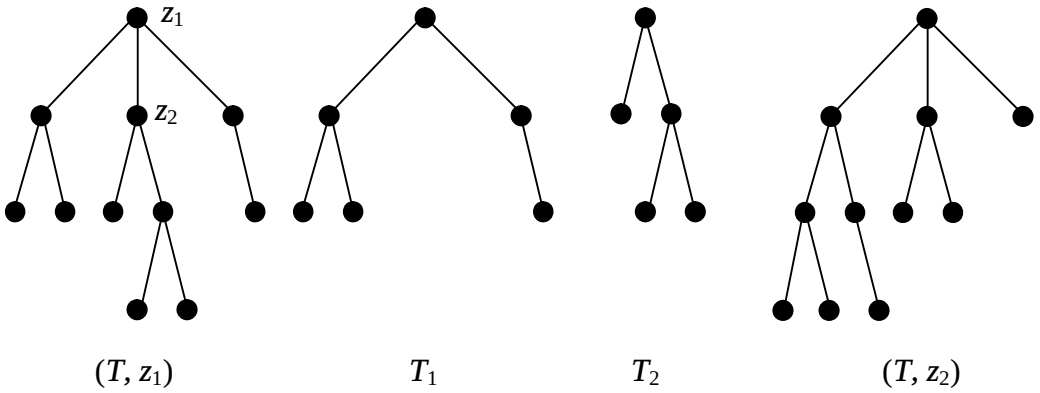


Рис. 15.

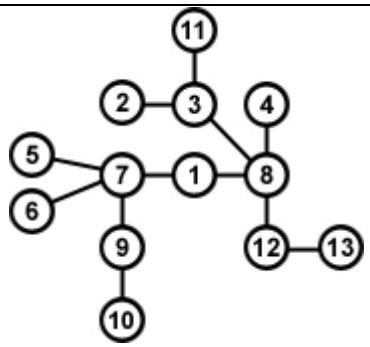
Задание 5. Свойства деревьев

Для заданного дерева определите:

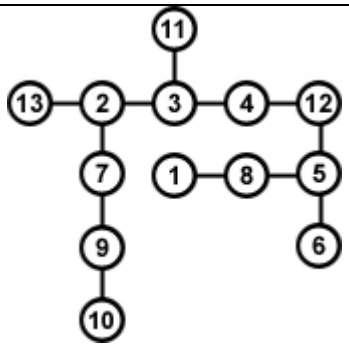
а) эксцентриситеты вершин, центр дерева, радиус и диаметр;

б) центроид дерева;

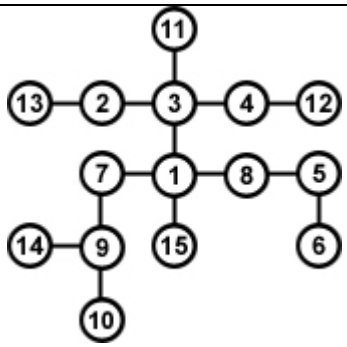
в) код Прюфера.



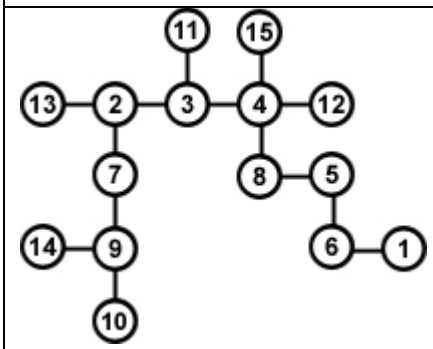
5.1



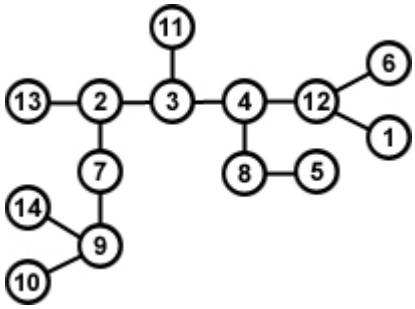
5.2



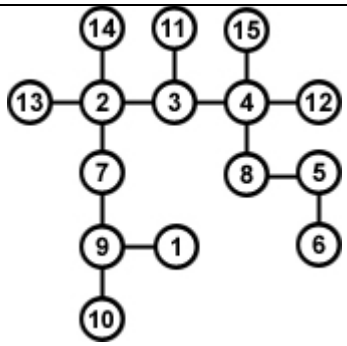
5.3



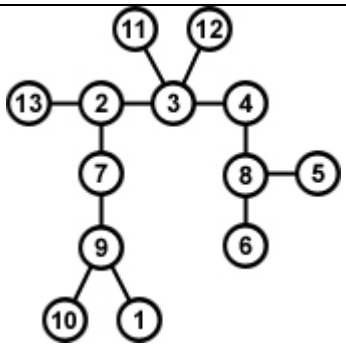
5.4



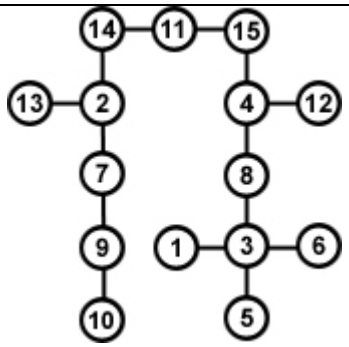
5.5



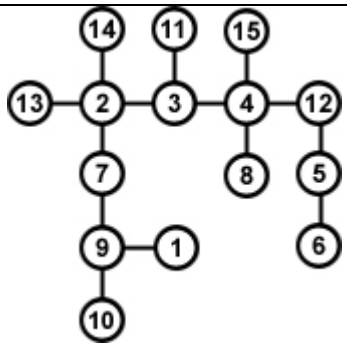
5.6



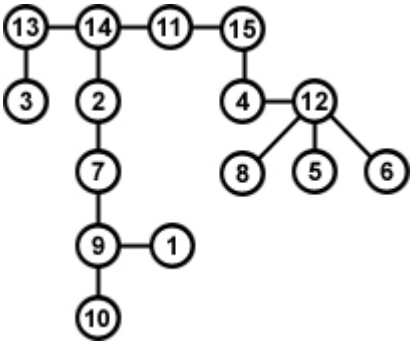
5.7



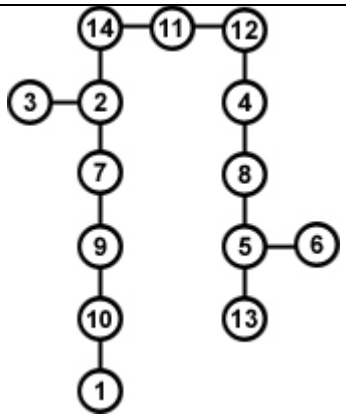
5.8



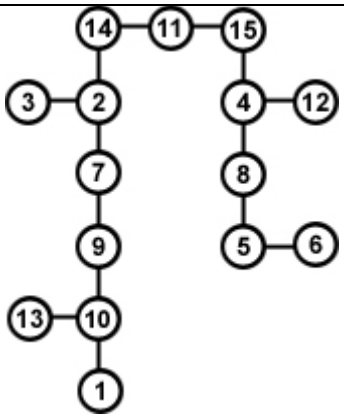
5.9



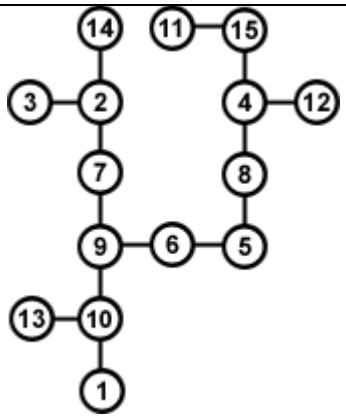
5.10



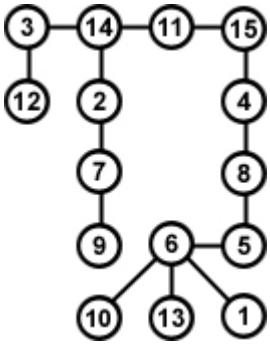
5.11



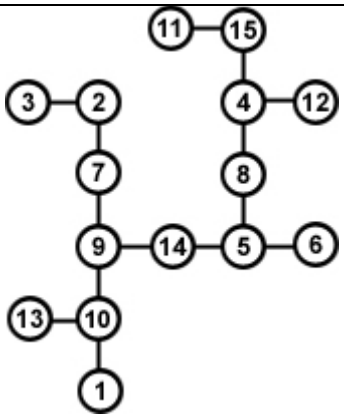
5.12



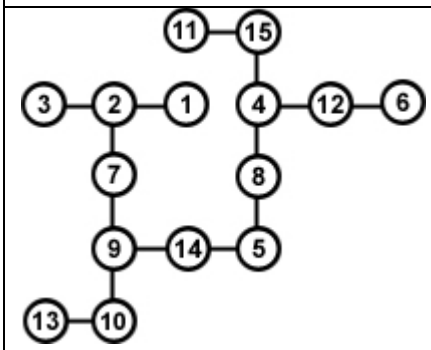
5.13



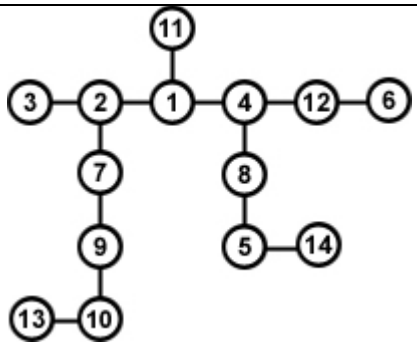
5.14



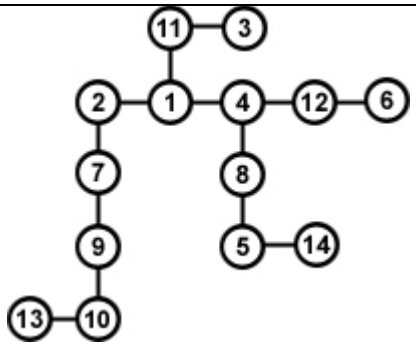
5.15



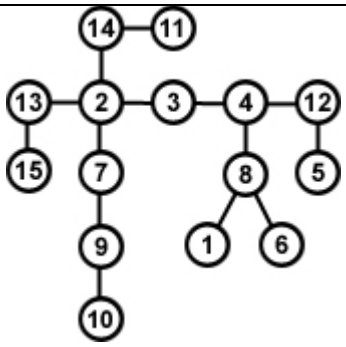
5.16



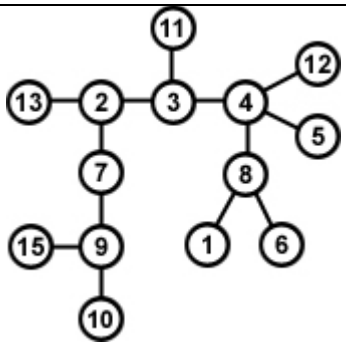
5.17



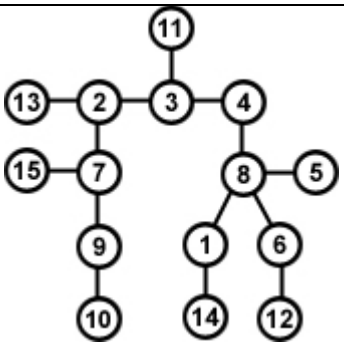
5.18



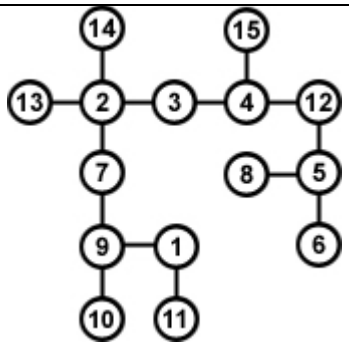
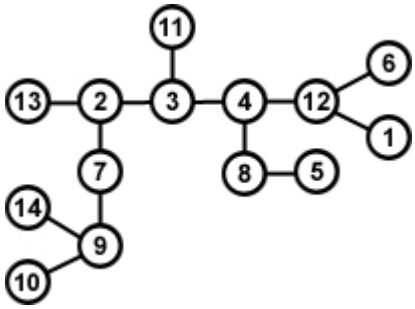
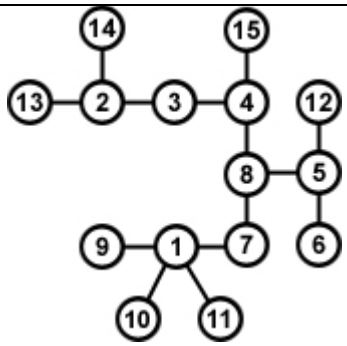
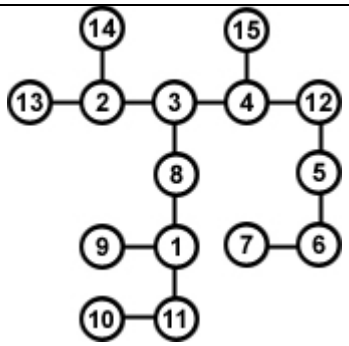
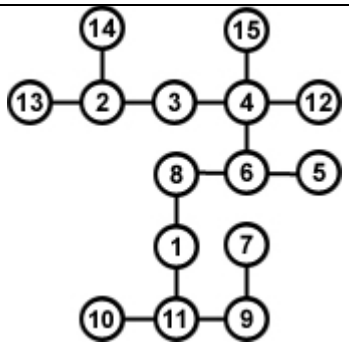
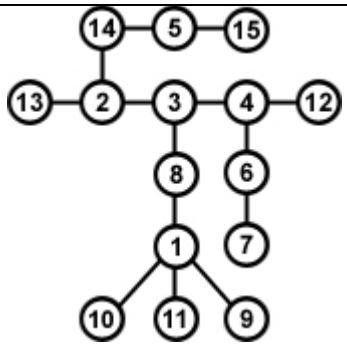
5.19



5.20



5.21

 5.22	 5.23	 5.24
 5.25	 5.26	 5.27

Задание 6. Кодирование деревьев

По заданному коду Прюфера восстановите дерево T. Изобразите построенное дерево как корневое (T, 1) с корнем в вершине 1. Для дерева (T, 1) определите уровневый и канонический уровневый коды. Для дерева T рассчитайте главный канонический уровневый код.

6.1. (2, 2, 7, 2, 11, 11, 7, 7, 6, 9, 4, 5).
6.2. (1, 1, 7, 6, 13, 1, 7, 12, 6, 9, 4, 5, 3).
6.3. (1, 2, 8, 3, 1, 10, 1, 1, 6, 5, 3, 2, 9).
6.4. (2, 5, 7, 12, 10, 11, 7, 7, 6, 9, 4, 5).
6.5. (12, 2, 1, 1, 1, 1, 3, 3, 4, 1, 2, 3, 8, 9).
6.6. (2, 2, 2, 2, 10, 10, 10, 2, 3, 1, 2, 2).
6.7. (7, 5, 4, 2, 1, 1, 11, 11, 2, 3, 4, 5, 8).
6.8. (3, 3, 3, 3, 1, 12, 2, 1, 7, 7, 9, 3, 8).
6.9. (2, 1, 3, 2, 11, 1, 1, 1, 6, 6, 6, 6, 7).
6.10. (5, 3, 2, 2, 2, 4, 4, 4, 5, 5, 4, 5, 9, 9).
6.11. (1, 2, 3, 4, 8, 7, 7, 7, 6, 3, 3, 3, 8, 9).
6.12. (6, 2, 1, 2, 5, 5, 4, 3, 7, 7, 7, 7, 9, 9).
6.13. (2, 1, 1, 6, 6, 6, 1, 1, 2, 3, 4, 5, 9, 9).
6.14. (2, 6, 2, 2, 3, 3, 13, 13, 3, 1, 2, 2, 13).
6.15. (12, 12, 12, 2, 2, 3, 1, 1, 3, 3, 4, 5, 9, 10).
6.16. (1, 2, 4, 1, 2, 4, 5, 6, 7, 8, 9, 10, 13, 13).
6.17. (2, 2, 1, 1, 2, 2, 7, 7, 6, 9, 3, 3, 7, 5, 2).
6.18. (5, 5, 5, 4, 2, 4, 1, 3, 1, 4, 5, 6, 9, 9).
6.19. (2, 12, 3, 12, 1, 1, 7, 7, 3, 4, 1, 10, 10).
6.20. (5, 1, 6, 7, 1, 8, 9, 5, 7, 6, 7, 3, 5, 6, 2).
6.21. (11, 12, 13, 1, 1, 2, 2, 1, 4, 5, 6, 7, 9, 8).
6.22. (1, 4, 1, 4, 1, 5, 1, 6, 1, 7, 8, 9, 6, 5).
6.23. (6, 6, 4, 7, 3, 2, 8, 4, 7, 9, 5, 3, 7, 8).
6.24. (9, 9, 7, 3, 6, 12, 4, 6, 2, 5, 6, 6, 7, 3).
6.25. (2, 2, 1, 2, 3, 4, 5, 7, 8, 5, 9, 4, 5).
6.26. (5, 2, 3, 1, 1, 8, 4, 2, 4, 4, 5, 6, 8).
6.27. (1, 1, 2, 5, 1, 7, 8, 9, 8, 8, 7, 4, 9).

Реконструируемость

Говорят, что граф H является *реконструкцией* графа G, если их колоды совпадают. Граф называется *реконструируемым*, если он *изоморфен* каждой своей реконструкции. Популярность задачи реконструируемости графов связана с известной гипотезой –

Гипотеза (Келли[1], Улам[2], 1945). *Каждый неориентированный граф с числом вершин, большим двух, является реконструируемым.*

Очевидно, что по колоде графа можно восстановить количество его вершин и ребер. Нетрудно доказать, что по колоде неориентированного графа можно восстановить и его вектор степеней. Исключений из гипотезы для неориентированных графов неизвестно, поэтому считается, что она для них выполняется. Для ориентированных графов известны бесконечные семейства пар нереконструируемых орграфов.

Пример. По заданной колоде реконструируйте граф.

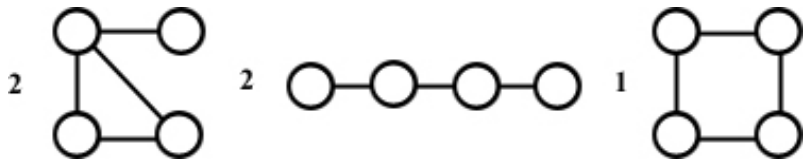


Рис. 16.

Так как в колоде 5 графов с 4 вершинами, значит число вершин исходного графа – 5.

Сумма ребер графов колоды $S = 2*4 + 2*3 + 4 = 18$. Значит число ребер исходного графа $m = 18 / (5 – 2) = 6$. Приведем два возможных рассуждения, приводящих к решению задания.

Решение 1. Посмотрим на последний граф колоды: он отличается от исходного на два ребра. Добавим к нему вершину и соединим ее с парой смежных и несмежных вершин. Далее из двух получившихся графов выбираем подходящий.

Решение 2. Рассчитаем степени вершин исходного графа:

$m_1 = m_2 = 6 – 4 = 2$;

$m_3 = m_4 = 6 – 3 = 3$;

[1] Пол Келли (Paul Joseph Kelly, 1915 – 1995) – американский математик.

[2] Станислав Мартин Улам (Stanislaw Marcin Ulam, 1909 – 1984) – польский математик.

$m_5 = 6 - 4 = 2$.

Таким образом, вектор степеней исходного графа имеет вид (3, 3, 2, 2, 2).

Реконструирование удобнее всего начать с подграфа, у которого есть вершина с максимальной степенью (на рисунке он расположен слева). Ни одна вершина с максимальной степенью не может быть смежна с добавленной вершиной. Кроме того, у выбранного подграфа есть вершина степени 1, а так как исходный граф не содержит вершин с такой степенью, значит, добавленная вершина обязательно должна быть связана с вершиной степени 1. Приняв во внимание то, что всего в исходном графе 6 ребер, получаем два случая.

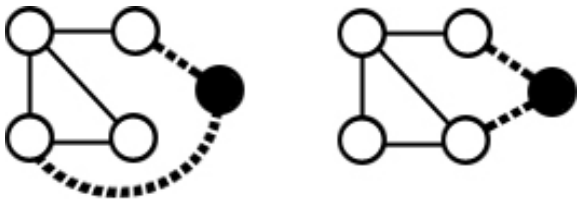
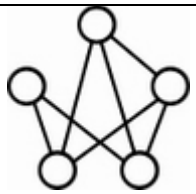
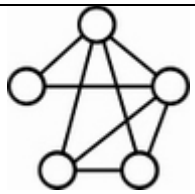
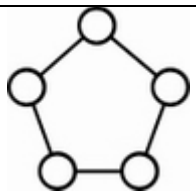
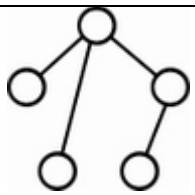
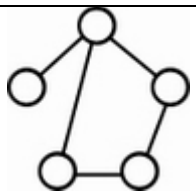
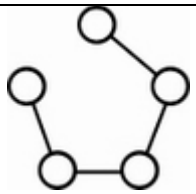
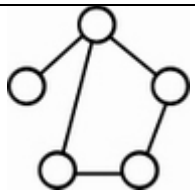
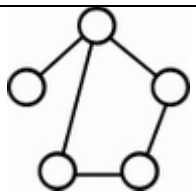
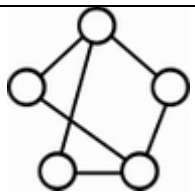
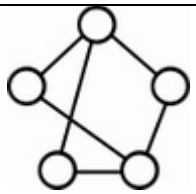
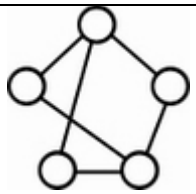
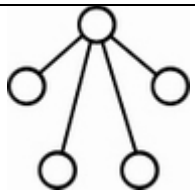
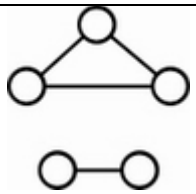
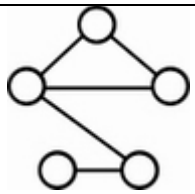
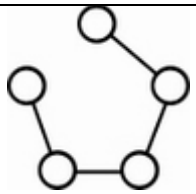
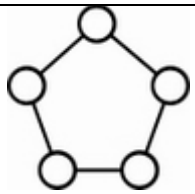
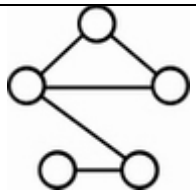
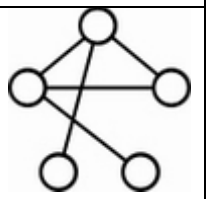
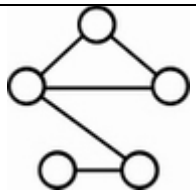
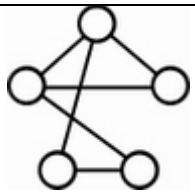


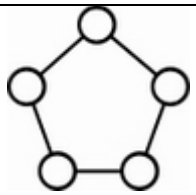
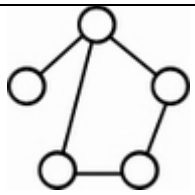
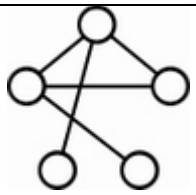
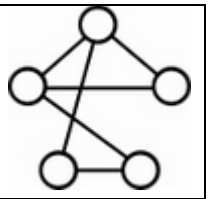
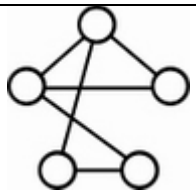
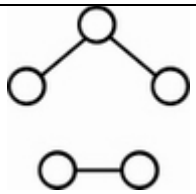
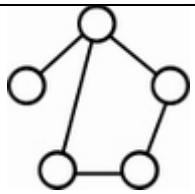
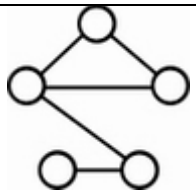
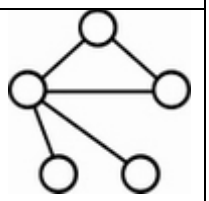
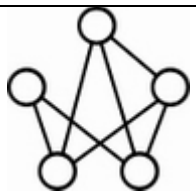
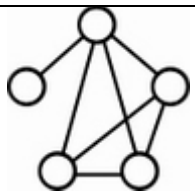
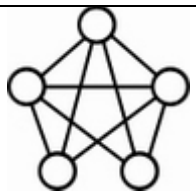
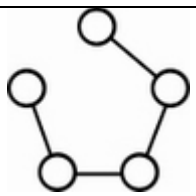
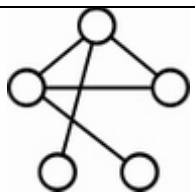
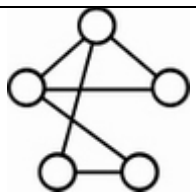
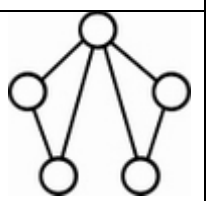
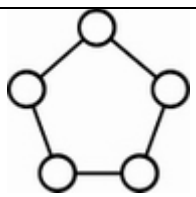
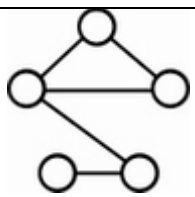
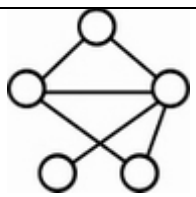
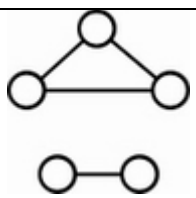
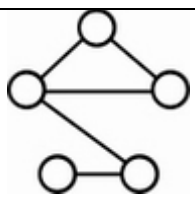
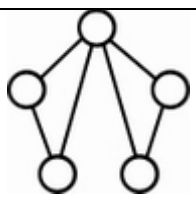
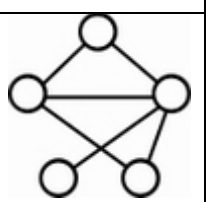
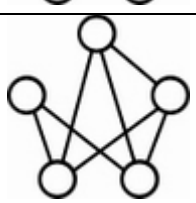
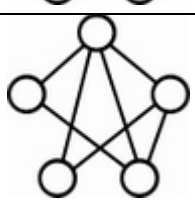
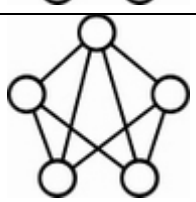
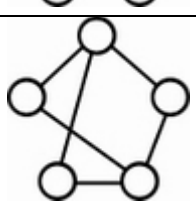
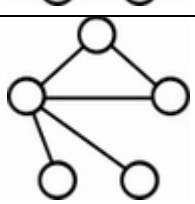
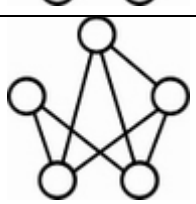
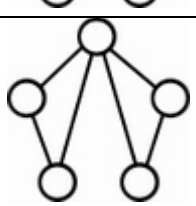
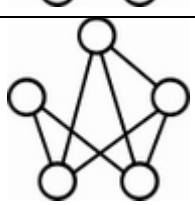
Рис. 17.

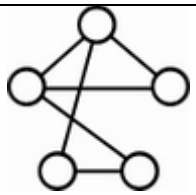
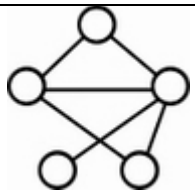
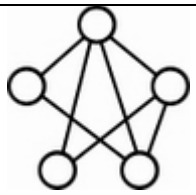
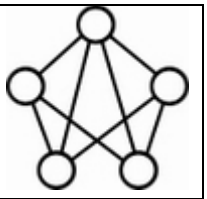
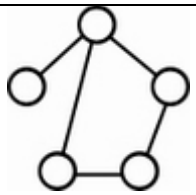
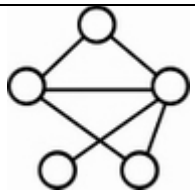
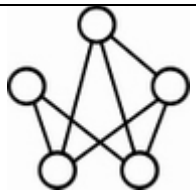
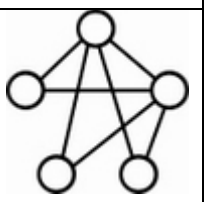
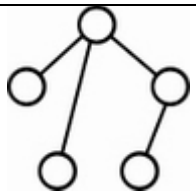
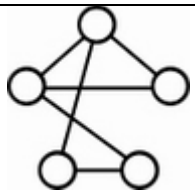
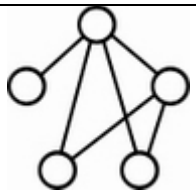
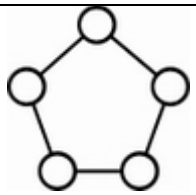
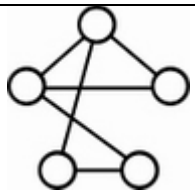
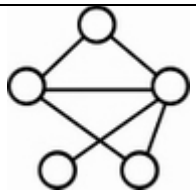
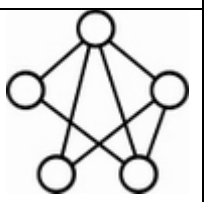
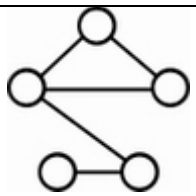
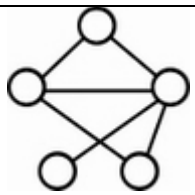
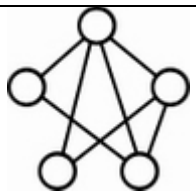
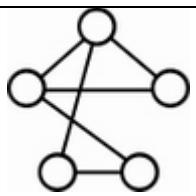
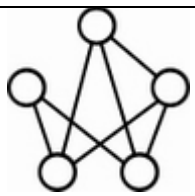
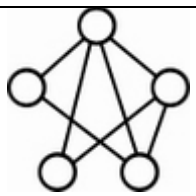
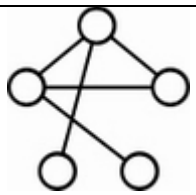
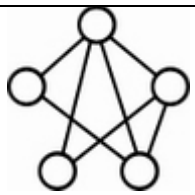
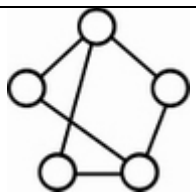
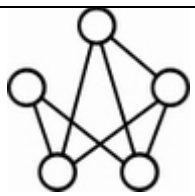
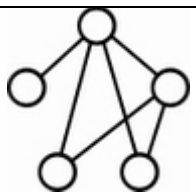
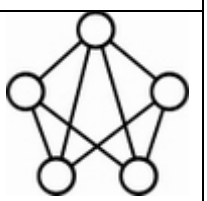
Очевидно, что графы изоморфны.

Задание 7

По заданной колоде реконструируйте граф.

7.1	4		2					
7.2	2		2		2			
7.3	2		4					
7.4	4		2					
7.5	6							
7.6	4		2					
7.7	2		4					
7.8	2		1		2		1	
7.9	4		2					

7.10	1		2		1		2	
7.11	6							
7.12	1		2		2		1	
7.13	3		2		1			
7.14	1		2		2		1	
7.15	2		2		2			
7.16	1		2		1		2	
7.17	2		2		2			
7.18	2		2		2			
7.19	2		4					

7.20	1		2		2		1	
7.21	2		2		1		1	
7.22	2		2		2			
7.23	1		2		2		1	
7.24	2		2		2			
7.25	2		2		2			
7.26	3		3					
7.27	1		2		2		1	

Факторграфы

Ориентированным графом или *орграфом* называется пара $G = (V, a)$, где a – отношение на множестве вершин V, называемое *отношением смежности*. Элементы множества a называются *дугами*. Если $(u,v) \in a$, то говорят, что u – *начало дуги*, а v – *конец дуги*. Дуга вида $(u,u) \in a$ называется *петлей*. Вершина не являющая началом никакой дуги, кроме быть может петли, называется *стоком*, а вершина не являющая концом никакой дуги, кроме быть может петли, называется *источником*.

Орграф $G = (V, a)$ называется *направленным графом* или *диграфом*, если отношение a антисимметрично. Полный направленный граф называется *турниром*. На рисунке изображены 3-вершинные турниры, которые называются *циклической тройкой* и *транзитивной тройкой*.

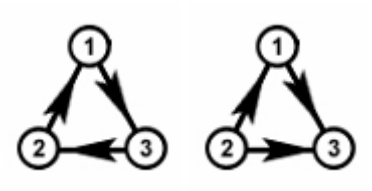


Рис. 18.

В таблице приведено количество n-вершинных орграфов[1], диграфов[2] и турниров[3] (без петель):

n	1	2	3	4	5	6	7	8	9	10
Орграфы	1	3	16	218	9608	1540944	~$9*10^8$	~$2*10^{12}$	~$1*10^{16}$	~$3*10^{20}$
Диграфы	1	2	7	42	582	21480	2142288	575016219	$4*10^{11}$	$8*10^{14}$
Турниры	1	1	2	4	12	56	456	6880	191536	9733056

Маршрутом в графе $G = (V, a)$ называется последовательность дуг $(v_0,v_1),(v_1,v_2),...,(v_{n-1},v_n)$. При этом говорят, что v_0 – *начальная вершина* маршрута, а v_n – *конечная*. Говорят также, что вершина v_n *достижима* из v_0. Маршрут, в котором никакая дуга не встречается более одного раза, называется *путем*. Если начальная и конечная вершины совпадают, то путь называется *циклическим*. Путь, каждая вершина которого принадлежит не более чем двум его дугам, считается *простым*. Простой циклический путь называется *контуром*, а

[1] См. последовательность A000055: http://www.research.att.com/~njas/sequences/A000055
[2] См. последовательность A001174: http://www.research.att.com/~njas/sequences/A001174
[3] См. последовательность A000568: http://www.research.att.com/~njas/sequences/A000568

простой путь, не являющийся контуром, называется *ориентированной цепью*. Петля называется *тривиальным контуром*. Орграф, не имеющий нетривиальных контуров, называется *бесконтурным*.

Будем считать, что каждая вершина достижима из самой себя. Тогда отношение достижимости является отношением эквивалентности на множестве вершин графа. Классы этого отношения называются компонентами связности графа.

Пусть даны орграф $G = (V, a)$, и отношение эквивалентности на множестве его вершин $\Theta \subseteq VxV$.

Факторграфом орграфа G по эквивалентности Θ называется орграф G/Θ, вершинами которого являются классы эквивалентности Θ. При этом из вершины $\Theta(u)$ проводится дуга в $\Theta(v)$, если существует вершина u' из класса $\Theta(u)$ и v' из класса $\Theta(v)$, такие, что $(u', v') \in a$.

Через ε обозначим отношение взаимной достижимости вершин орграфа, считая, что вершина достижима из самой себя. Очевидно, что ε является отношением эквивалентности. Классы отношения ε называются *сильными компонентами* орграфа.

Конденсацией орграфа называется его факторграф по отношению взаимной достижимости ε, то есть вершинами конденсации являются сильные компоненты орграфа. Очевидно, что конденсация любого орграфа является бесконтурным графом.

Пример. На рисунке 19 сверху изображен граф, а снизу его конденсация.

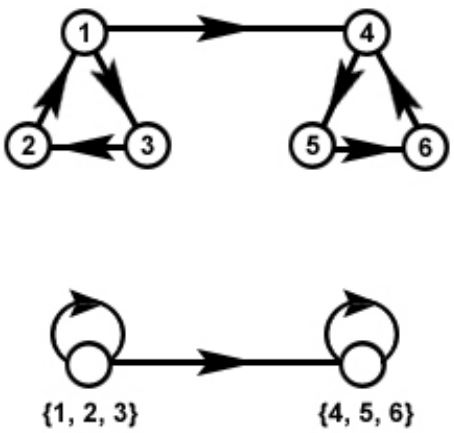


Рис. 19.

Подмножество $V^* \subseteq V$ называется *базой* орграфа $G = (V, \alpha)$, если выполняется два условия:

1) любая вершина орграфа достижима из подходящей вершины базы;
2) никакая вершина базы не достижима из других ее вершин.

Пример: граф из предыдущего примера имеет три базы: {1}, {2}, {3}.

Теорема. *Подмножество $V^* Í V$ тогда и только тогда является базой орграфа $G = (V, a)$, когда оно образовано вершинами, взятыми по одной из каждого источника конденсации.*

Пусть орграф G является моделью некоторой системы, допускающий одиночный отказ. Для обнаружения отказа проводится проверка элементов системы: если элемент исправен, то результатом проверки будет 0, а в противном случае – 1. Предположим, система обладает таким свойством, что ошибка наследуется всеми элементами, достижимыми из неисправного (в смысле орграфа G). *Проверяющим тестом* называется некоторая совокупность проверок элементов системы, позволяющая установить, имеется ли в системе отказ (без его локализации). Проверяющий тест называется минимальным, если он содержит минимально возможное количество проверок элементов.

Теорема. *Проверяющий тест системы минимален тогда и только тогда, когда он состоит из проверок элементов, которые соответствуют вершинам орграфа системы взятым по одной из каждого стока конденсации.*

Если орграф содержит нетривиальные контуры, то проверяющий тест не может быть локализующим. Интерес представляют бесконтурные графы, для которых можно строить локализующие проверяющие тесты.

Пусть $G = (V, a)$ – бесконтурный орграф. Для вершины v через $S(v)$ обозначим множество всех стоков, достижимых из v. Рассмотрим отношение σ на множестве вершин орграфа G: $\sigma \subseteq V \times V$: две вершины принадлежат отношению σ тогда и только тогда, когда из них достижимы одни и те же стоки: $(u, v) \in \sigma \Leftrightarrow S(u) = S(v)$. Очевидно, что отношение σ является эквивалентностью.

Теорема. *Если бесконтурный граф имеет тождественное отношение σ, то минимальный проверяющий тест для системы, представленной этим орграфов является также и локализующим тестом, то есть указывает неисправную вершину.*

Задание 8. Достижимость, конденсация и базы

Для заданного графа вычислите матрицу достижимости, найдите максимальные сильно связные подграфы, источники и стоки, постройте факторграф по заданной эквивалентности, конденсацию и выпишите все возможные базы.

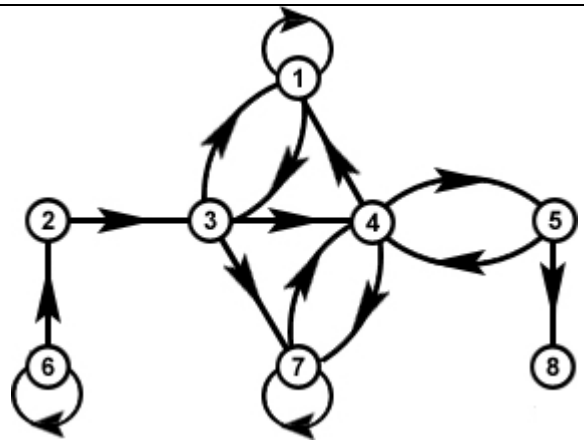
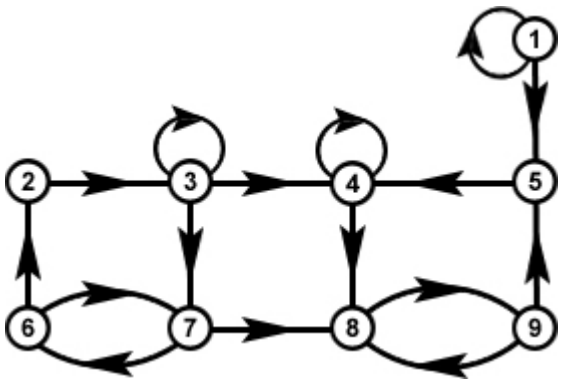
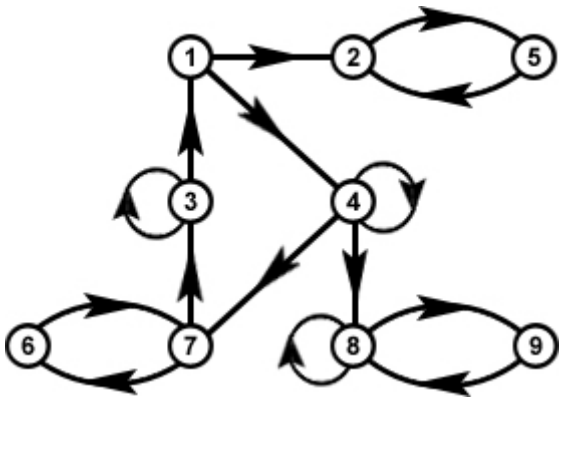
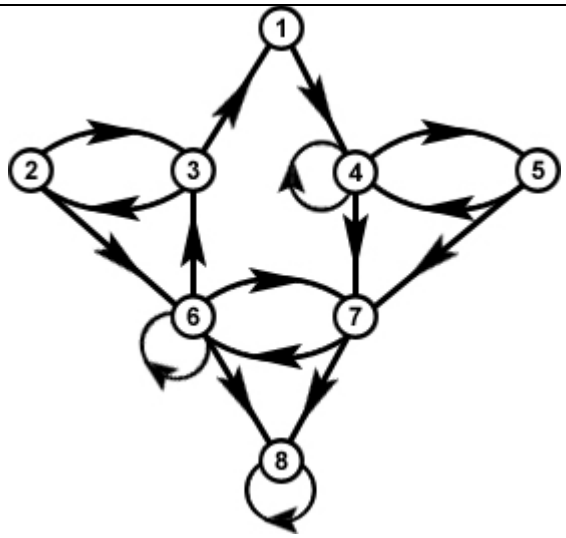
Множества эквивалентных вершин для групп задач:

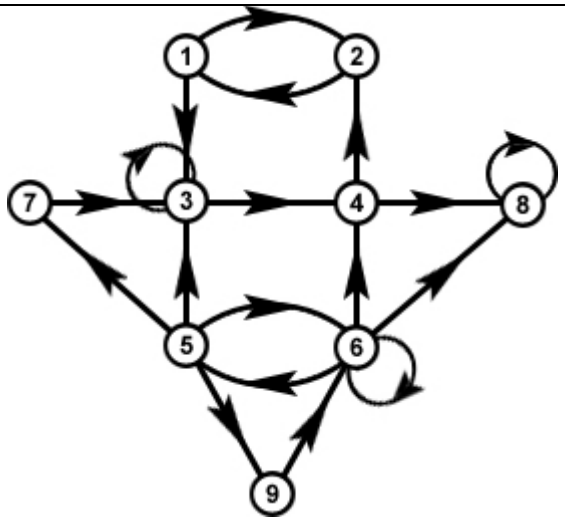
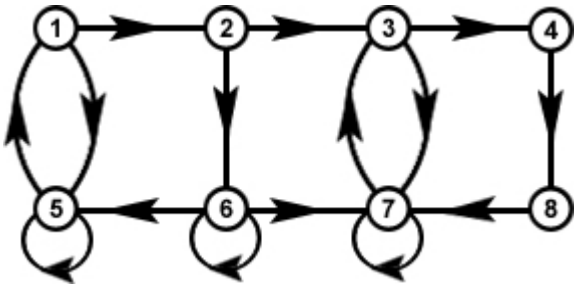
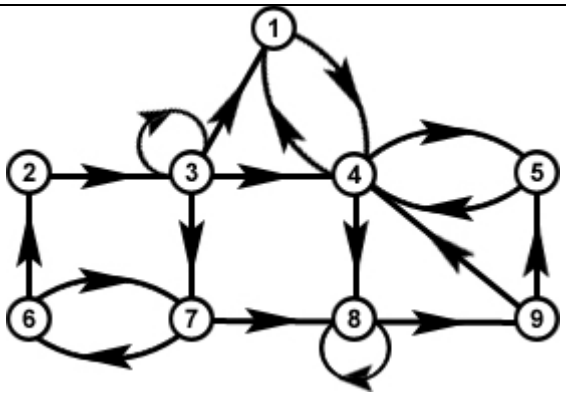
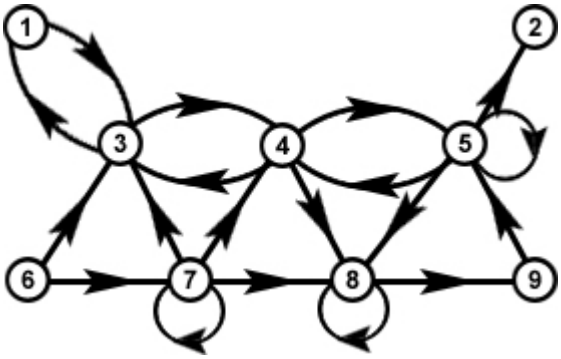
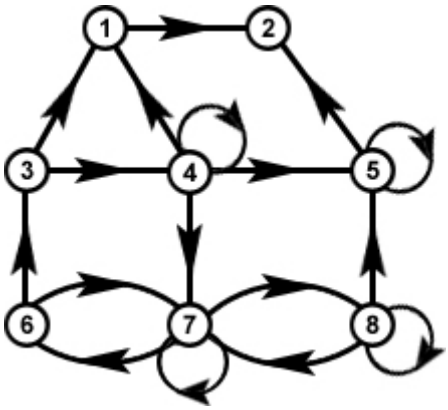
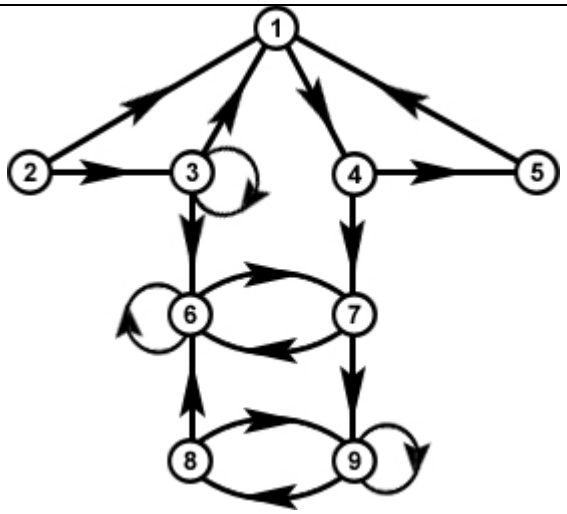
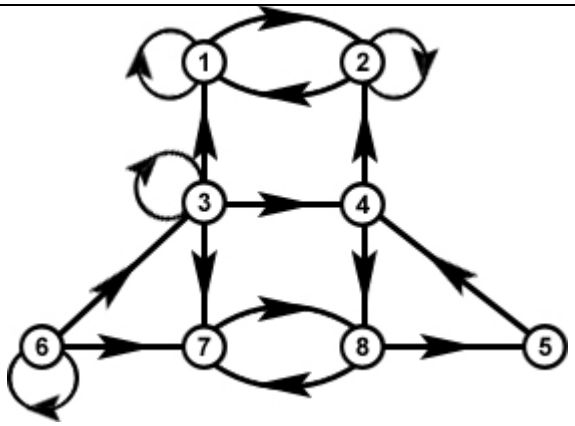
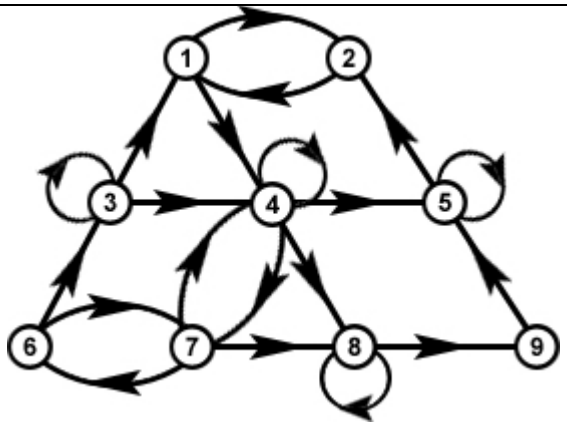
A) {1, 3, 4}, {2}, {5, 6}, {7, 8};

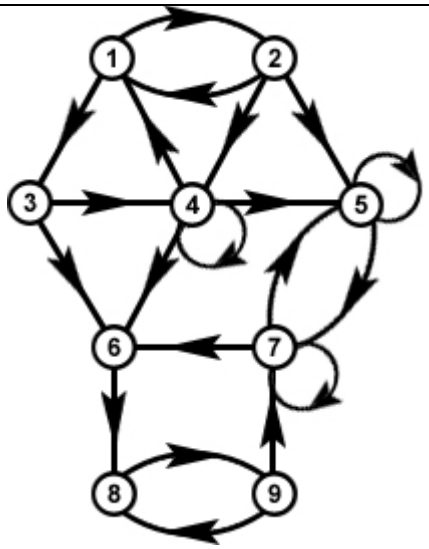
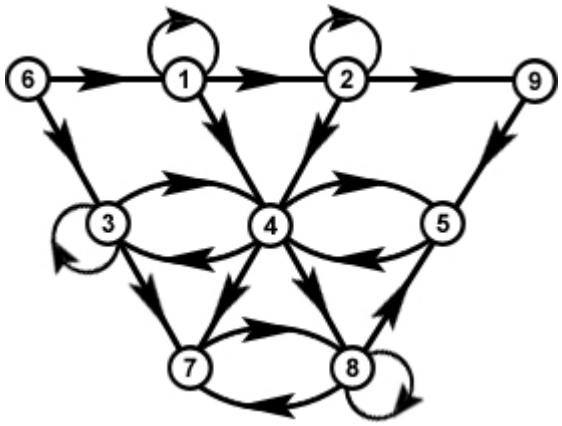
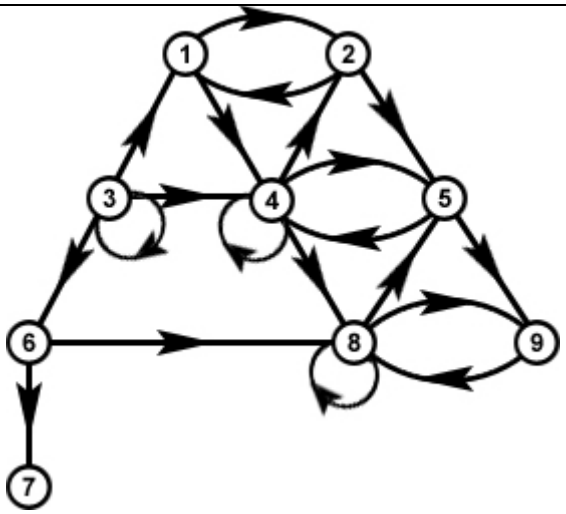
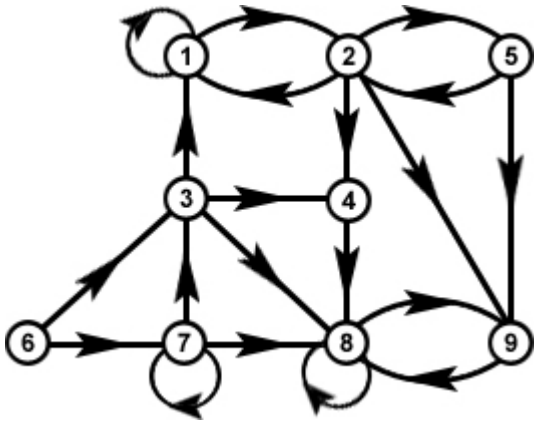
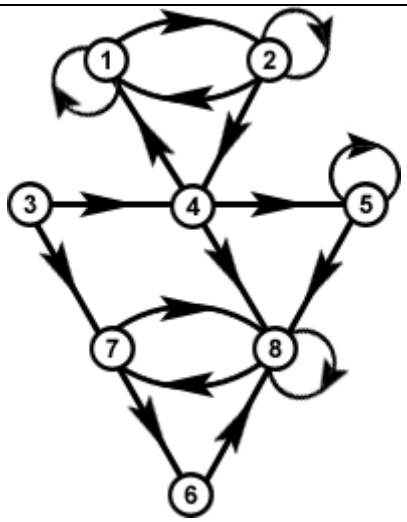
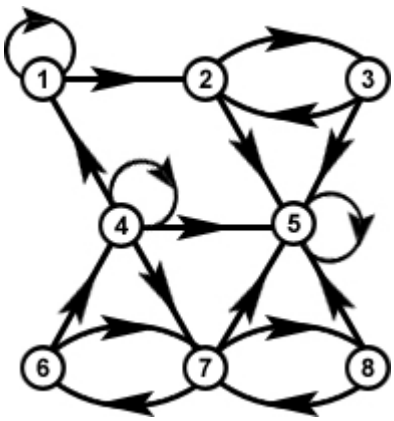
B) {1, 2, 9}, {3, 4}, {5, 6, 7}, {8};

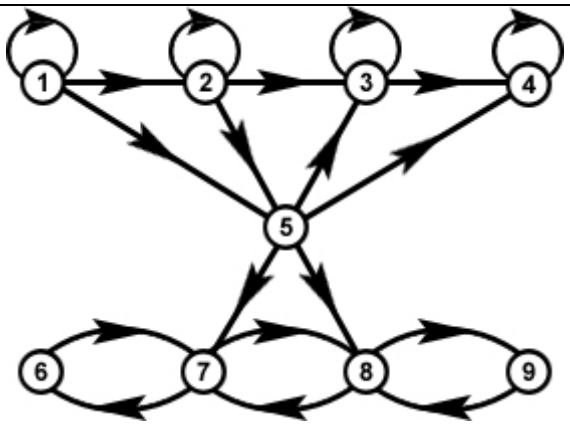
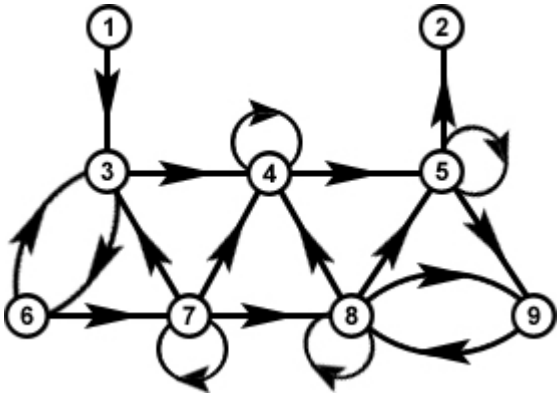
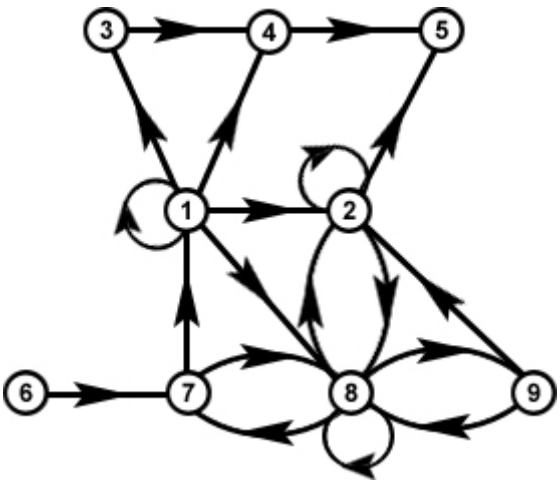
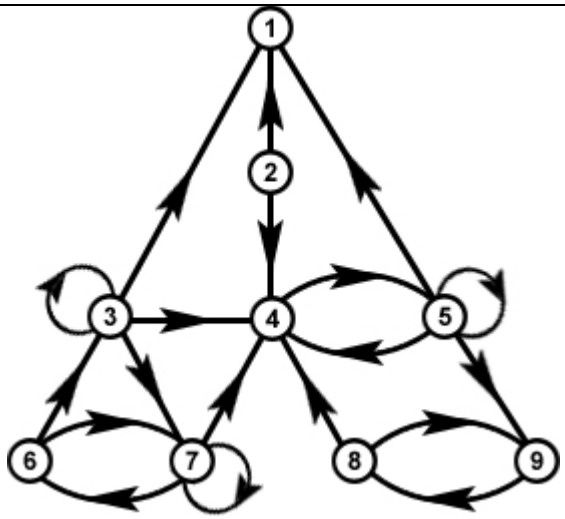
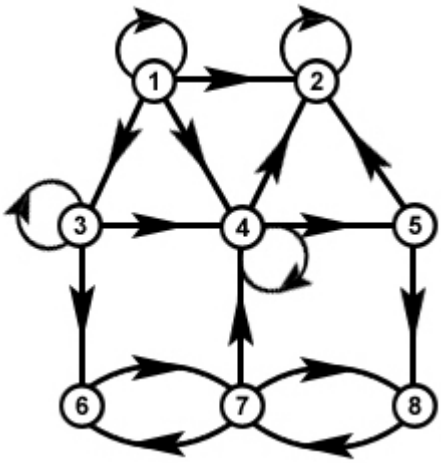
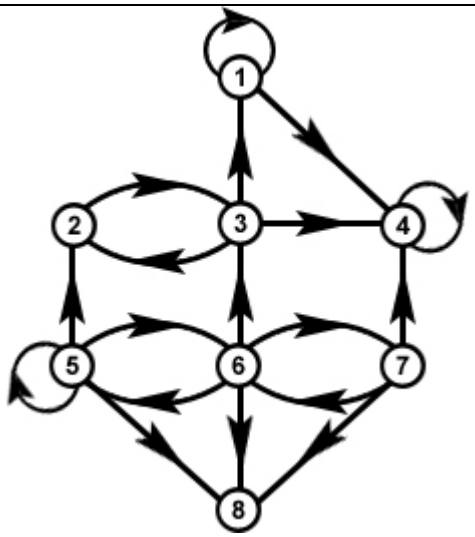
C) {1, 5, 7}, {2, 3}, {4, 6}, {8, 9};

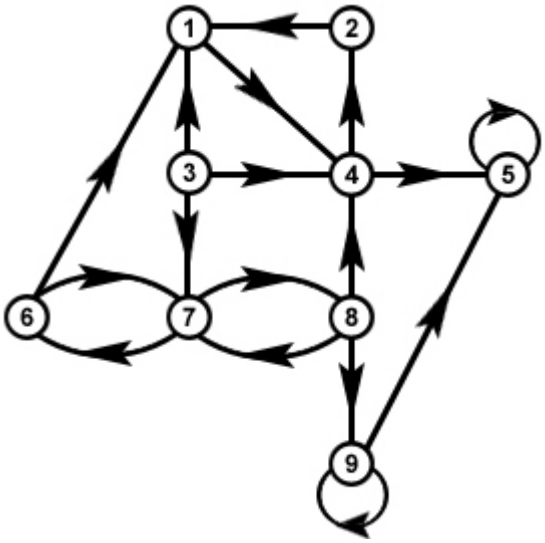
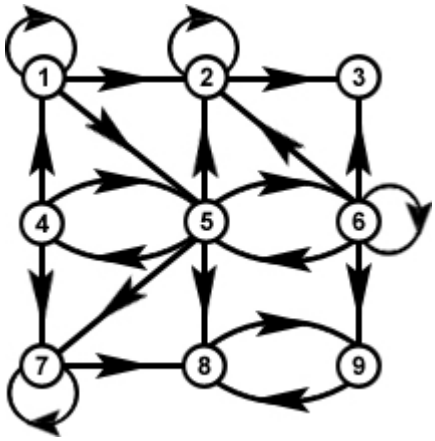
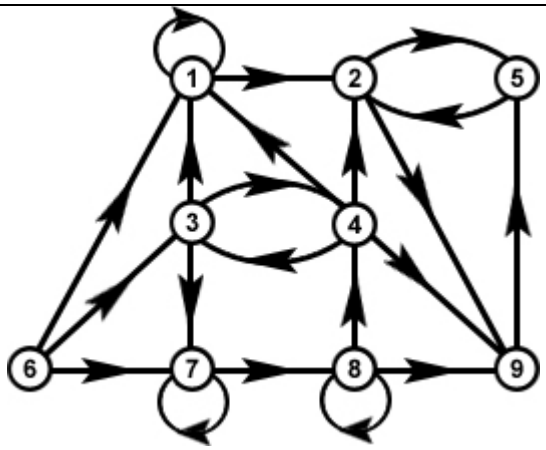
D) {1, 9}, {2, 3, 4}, {5, 7}, {6, 8}.

1 A		2 B	
3 B		4 A	

5 C		6 A	
B		8 D	
9 A		10 C	
11 A		12 D	

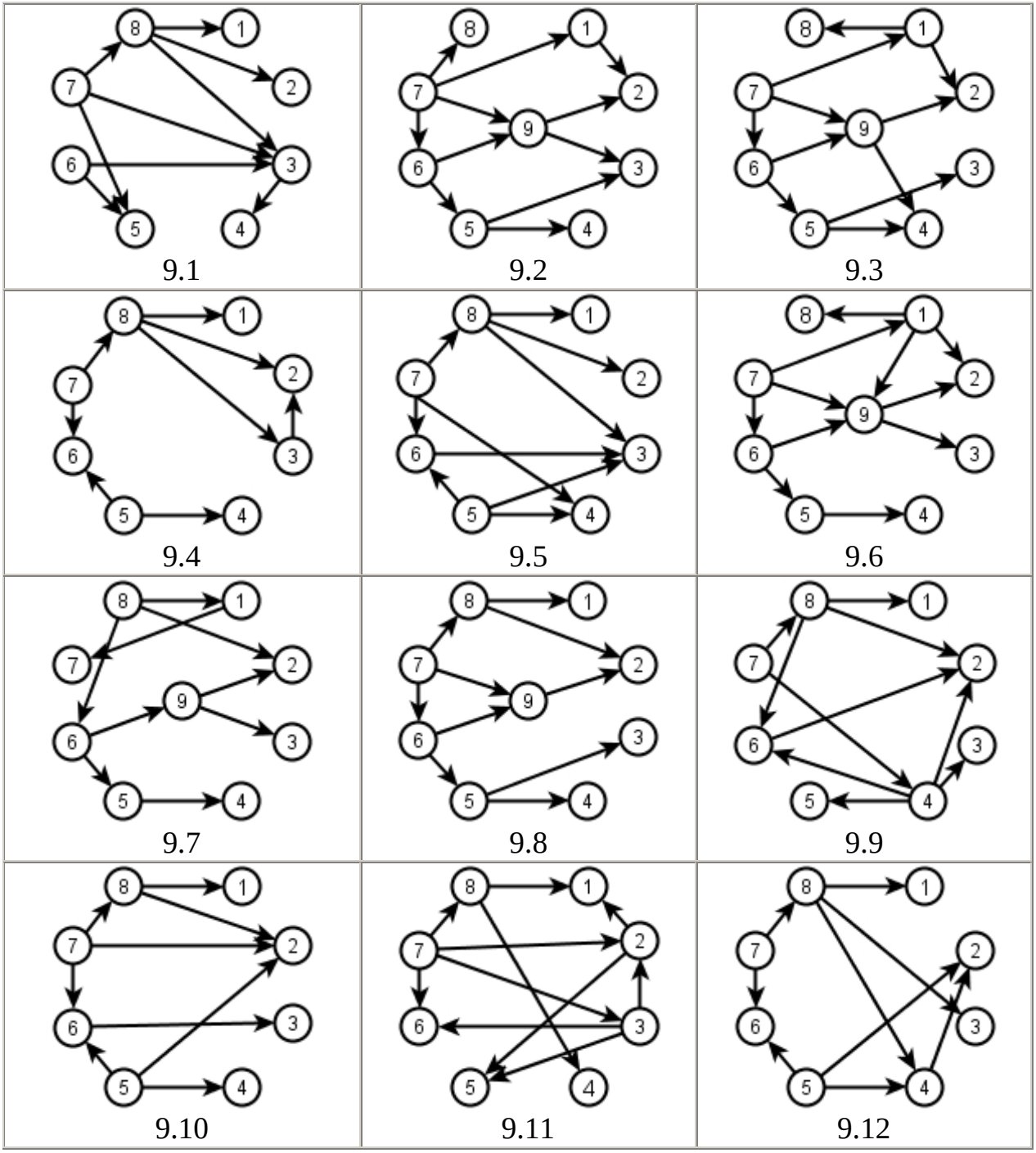
13 B		14 C	
15 D		16 B	
17 A		18 A	

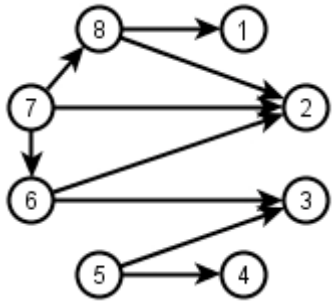
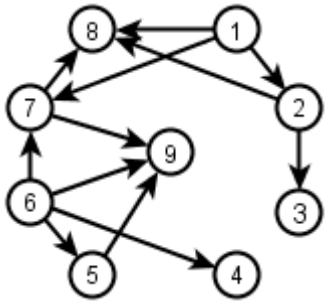
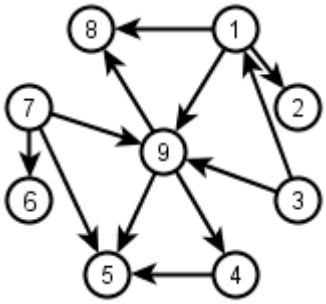
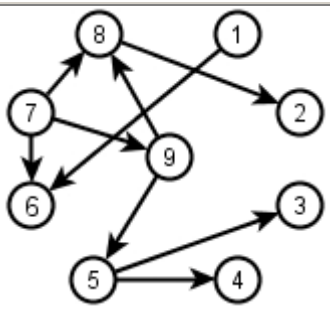
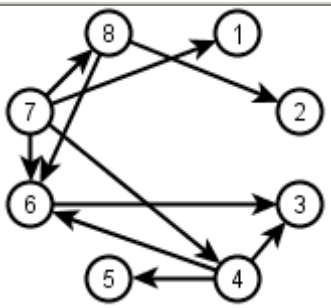
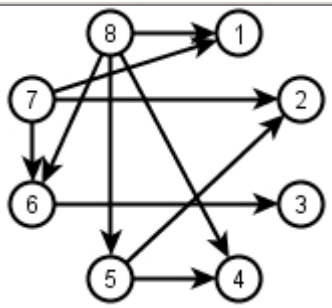
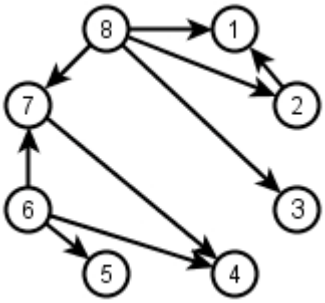
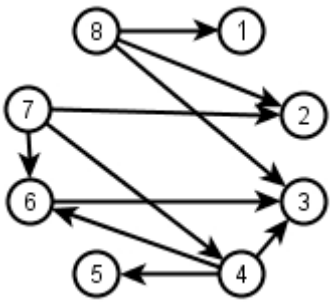
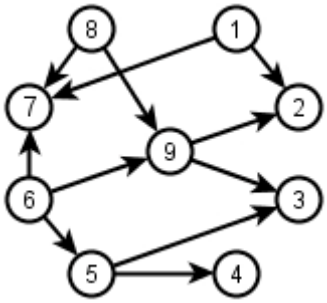
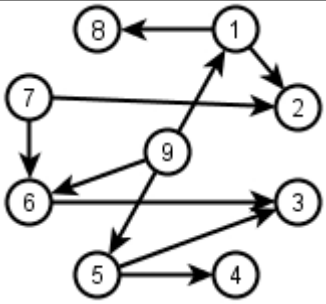
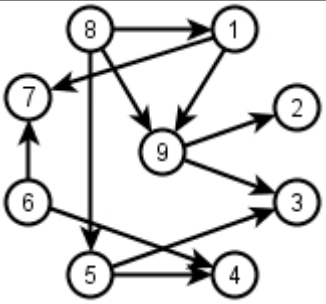
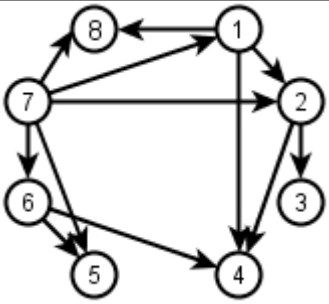
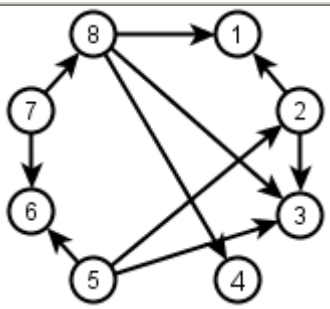
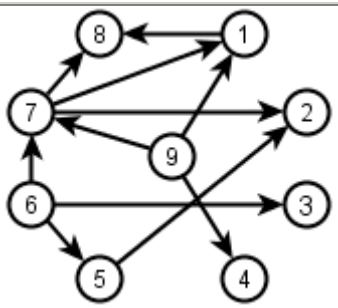
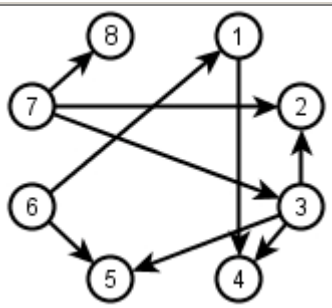
19 D		20 D	
21 B		22 C	
23 A		24 A	

25 C		26 D	
27 B			

Задание 9. Проверяющие тесты

Для заданного графа постройте минимальный проверяющий тест и проверьте, является ли он локализующим.



 9.13	 9.14	 9.15
 9.16	 9.17	 9.18
 9.19	 9.20	 9.21
 9.22	 9.23	 9.24
 9.25	 9.26	 9.27

Алгоритмы на графах

Графовые модели широко применяются в различных прикладных областях. Многие задачи сводятся к последовательному просмотру вершин или ребер графа и определению его характеристик. Далее мы рассмотрим несколько способов представления графов, а также некоторые алгоритмы для решения задач построения остовного дерева и нахождения кратчайших путей между вершинами.

Существует ряд способов задания графов. Для решения конкретной задачи можно выбрать тот или иной способ, в зависимости от удобства его применения. Перечислим некоторые, наиболее известные способы.

Матрица инцидентности: матрица размером $n \times m$ (n – число вершин, m – число ребер), элемент a_{ij} которой равен 1, если i-я вершина инцидентна j-му ребру, и 0 в противном случае. Матрица инцидентности неудобна для ввода и обработки на ЭВМ, кроме того, она не несет прямой информации о ребрах.

Матрица смежности: матрица размером $n \times n$, элемент a_{ij} которой равен 1, если i-я вершина смежна с j-ой, и 0 в противном случае. Матрица смежности неориентированного графа является симметричной. Для взвешенного графа вместо 1 используется значение весовой функции и такая матрица называется *матрицей весов*. Так как размер матрицы смежности есть $O(n^2)$, то алгоритмы, использующие такой способ представления графов, имеют не меньшую сложность.

Списки смежности: каждый i-й список содержит номера вершин, смежных i-ой вершине. Списки смежности удобны для ввода в ЭВМ, экономят память, но не могут использоваться в случае взвешенного графа.

Приведем пример описания класса графа и реализацию ввода-вывода графа средствами языка C++.

```
#include <iostream>
#include <fstream>

using namespace std;

#define MAX_VERTEX 20

// пример класса для работы с графами
class CGraph{
public:
	CGraph(){
		memset(m_a, 0, sizeof(m_a));
		m_nVertexNum = 0;
	}
	// чтение графа из потока
	void ReadGraph(istream &InStream){
		InStream >> m_nVertexNum;
		for (int i = 0; i < m_nVertexNum; i++){
			for (int j = 0; j < m_nVertexNum; j++){
				InStream >> m_a[i][j];
			}
		}
	}
	// печать графа в поток
	void Print(ostream &OutStream){
		for (int i = 0; i < m_nVertexNum; i++){
			for (int j = 0; j < m_nVertexNum; j++){
				OutStream << m_a[i][j] << " ";
			}
			OutStream << endl;
		}
	}

protected:
	int m_nVertexNum; // количество вершин графа
	int m_a[MAX_VERTEX][ MAX_VERTEX];  // матрица смежности графа
};

void main(){
	CGraph G1, G2;

	// чтение графа из стандартного потока ввода
	G1.ReadGraph(cin);
	// печать графа в стандартный поток вывода
	G1.Print(cout);

	ifstream In("input.txt");
	ofstream Out("output.txt");

	// чтение графа из файла "input.txt"
	G2.ReadGraph(In);
	// печать графа в файл "output.txt"
	G2.Print(Out);

	In.close();
	Out.close();
}
```

Обход графа

Основные алгоритмы просмотра вершин графа делятся на две группы – обход в глубину или в ширину. Обход в глубину предполагает последовательный просмотр вершин графа построением цепи от текущей вершины. При обходе в ширину в первую очередь просматриваются вершины, смежные с текущей.

Поиск в глубину

Идея метода. Поиск начинается с некоторой вершины *v*. Рассматривается новая (из не рассмотренных ранее) вершина *u*, смежная с *v*, и помечается как просмотренная. Процесс повторяется с вершиной *u*. Если на очередном шаге была просмотрена вершина *w* и нет вершин, смежных с *w* и не рассмотренных ранее, то возвращаемся из вершины *w* к вершине, которая была просмотрена до нее. Если были просмотрены все вершины или мы вернулись к начальной вершине *v*, но новых вершин смежных с ней не осталось, то процесс обхода закончен. Поиск в глубину удобен для рекурсивной реализации или реализации с помощью стека.

Пример. Дан граф (см. рис. 20). Поиск начинается с вершины 1. На левом рисунке приведен исходный граф, а на правом рисунке у вершин в скобках указана та очередность, в которой вершины графа просматривались в процессе поиска в глубину.

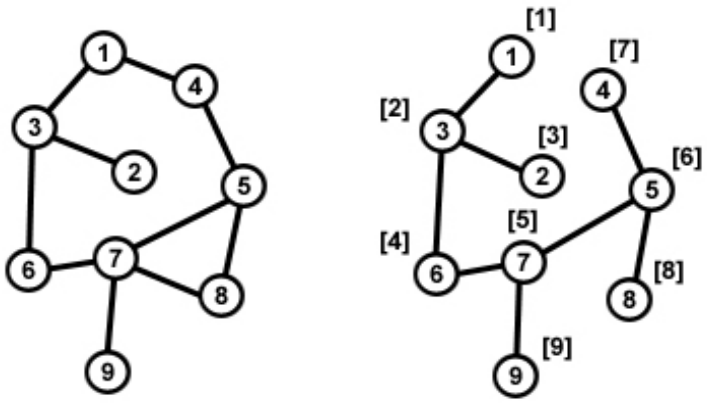


Рис. 20.

Поиск в ширину

Идея метода. Пусть дан граф $G=(V, \alpha)$ и некоторая начальная вершина v. Алгоритм поиска в ширину перечисляет все достижимые из v вершины в порядке увеличения расстояния от v. В процессе поиска из графа выделяется часть, называемая «деревом поиска в ширину» с корнем в v. Это дерево содержит все достижимые из v вершины и только их. Для реализации данного принципа обычно используется структура данных «очередь».

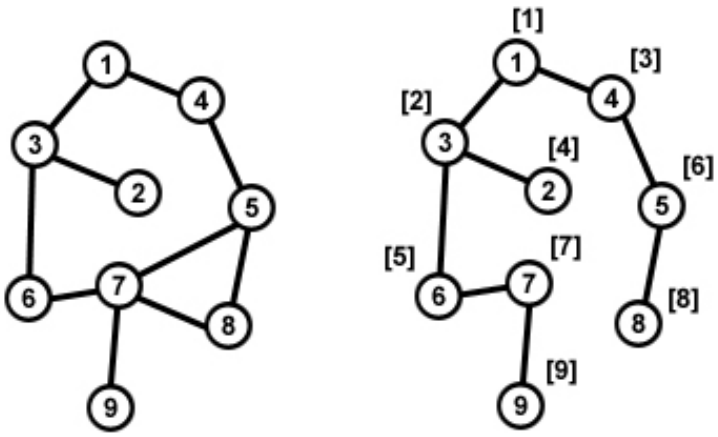


Рис. 21.

Пример. Дан граф (см. рис. 21). Поиск начинается с вершины 1. На рисунке 21 слева приведен исходный граф, а справа у вершин в скобках указана та очередность, в которой вершины графа просматривались в процессе поиска в ширину.

Минимальное остовное дерево

Для произвольного связного неориентированного графа $G=(V, \alpha)$ дерево $T = (V, \alpha^*)$, где $\alpha^* \subseteq \alpha$ называют *остовным деревом* (*стягивающим деревом*, *каркасом*, *остовом*). Ребра такого дерева называют *ветвями*, а остальные ребра графа – *хордами*. Граф, каждому ребру которого приписано некоторое число, называется *взвешенным графом* или сетью. Весом остова называется сумма весов его ребер. Требуется найти остов минимального веса. Большинство алгоритмов решения задачи (в том числе и рассматриваемы далее алгоритмы Прима и Краскала) основываются на принципе «жадности», в основе которого лежит правило выбора локально оптимального решения.

Алгоритм Краскала[1] (1956)

Дано: связный взвешенный n-вершинный граф $G=(V, \alpha)$.

Результат: минимальное остовное дерево $T = (V, \alpha^*)$, где $\alpha^* \subseteq \alpha$.

Шаг 1. Начать с вполне несвязного графа G, содержащего n вершин.

Шаг 2. Упорядочить ребра графа G в порядке неубывания их весов.

Шаг 3. Начав с первого ребра, очередное ребро добавить к дереву T, если его добавление не приводит к появлению цикла в T.

Шаг 4. Повторять шаг 3 до тех пор, пока число ребер в T не станет равным $n - 1$. Получившееся дерево является минимальным остовом.

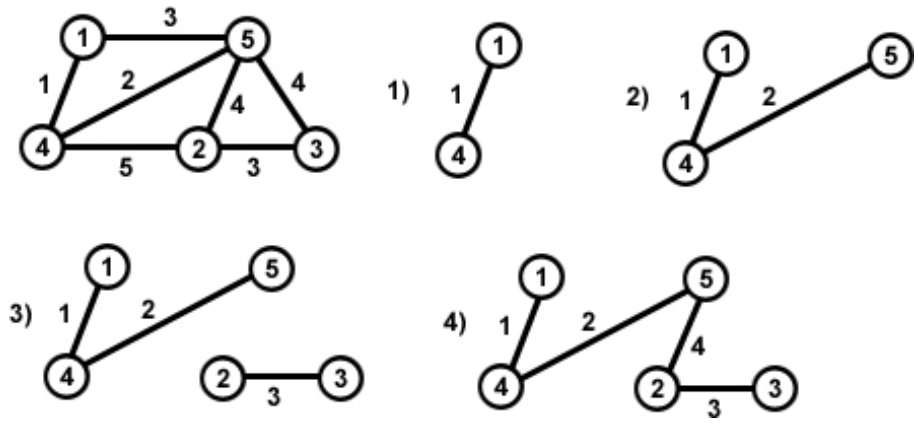


Рис. 22.

Пример. Граф и процесс построения минимального остова по алгоритму Краскала (см. рис. 22).

Введем массив меток вершин графа (Mark[n]). Начальные значения элементов массива равны номерам соответствующих вершин. Ребро включается в остов в том случае, если вершины, соединяемые им, имеют разные значения меток. Для примера, приведенного выше, пошаговое изменение значений меток показано в таблице.

[1] Джозеф Краскал (Joseph Bernard Kruskal, Jr., род. 1928) – американский математик.

Номер итерации	Ребро	Значения элементов Mark
0	–	[1,2,3,4,5]
1	{1,4}	[1,2,3,1,5]
2	{4,5}	[1,2,3,1,1]
3	{2,3}	[1,2,2,1,1]
4	{2,5}	[1,1,1,1,1]

Алгоритм Прима[1] (1957)

Дано: связный взвешенный n-вершинный граф $G=(V, \alpha)$.

Результат: минимальное остовное дерево $T = (V, \alpha^*)$, где $\alpha^* \subseteq \alpha$.

Отличие от метода Краскала заключается в том, что на каждом шаге достраивается дерево. Как показано в примере, дерево начинается с произвольной корневой вершины r и растет до тех пор, пока не охватит все вершины в V. На каждом шаге к дереву добавляется ребро наименьшего веса среди ребер, соединяющих вершины дерева с вершинами из оставшейся части графа. Данная стратегия является жадной, поскольку на каждом шаге к дереву добавляется ребро, которое вносит минимально возможный вклад в общий вес.

Пример. Граф и процесс построения минимального остова по алгоритму Прима (см. рис. 23). Через *SM* обозначено множество вершин графа, а через *SP* – множество вершин строящегося дерева.

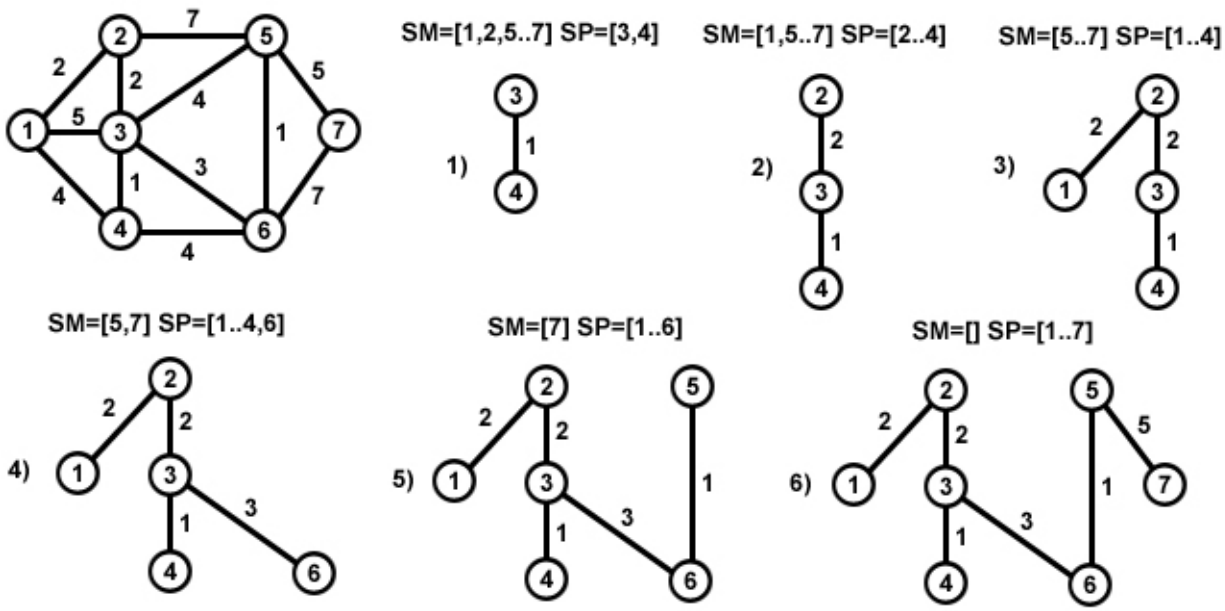


Рис. 23.

[1] Роберт Прим (Robert Clay Prim, род. 1921) – американский математик.

Кратчайшие пути

В задаче о кратчайшем пути дается взвешенный ориентированный или неориентированный граф. Каждой дуге или ребру приписан вещественный вес $w(u, v)$. *Весом* пути называется сумма весов дуг, входящих в этот путь. *Весом кратчайшего пути* из вершины u в v называется минимальный из весов путей u в v, а если таких путей нет, то вес кратчайшего пути считается равным ∞. *Кратчайшим путем* из u в v называется всякий путь из u в v, вес которого равен весу кратчайшего пути из u в v. Если в графе существует контур отрицательной длины, достижимый из вершины u, то понятие кратчайшего пути утрачивает смысл для всех вершин v, достижимых из вершин этого контура.

В задаче о кратчайшем пути требуется найти кратчайший путь от заданной вершины (источника) до всех остальных графа. Очевидно, что любая часть кратчайшего пути сама является кратчайшим путем. Чтобы восстановить кратчайший путь, будем для каждой вершины v графа запоминать ее предшественника $\pi(v)$ в этом пути. Выписывая цепочку предшественников в обратном порядке, можно получить кратчайший путь.

Алгоритм Дейкстры[1] (1959)

Алгоритм Дейкстры решает задачу о кратчайших путях из вершины s_0 до всех остальных вершин для графа $G=(V, \alpha)$ с неотрицательными весами. Обозначим через $\lambda(v)$ оценку кратчайшего пути из s_0 в v. Вначале все вершины считаются не окрашенными. Для запоминания окрашенных вершин будем использовать множество U. Последнюю окрашенную вершину будем хранить в s. Множество неокрашенных вершин будет $V\backslash U$.

Шаг 1. Положим $\lambda(s_0) := 0$, $\pi(s_0) :=$ NIL а для всех остальных вершин $\lambda(v) = \infty$, $\pi(v) :=$ NIL. Окрасим вершину s_0 и положим $U := \{s_0\}$, $s := s_0$.

Шаг 2. Для каждой неокрашенной вершины v смежной с последней окрашенной вершиной s осуществляется *релаксация* по дуге (s, v), то есть осуществляется пересчет оценки кратчайшего пути:

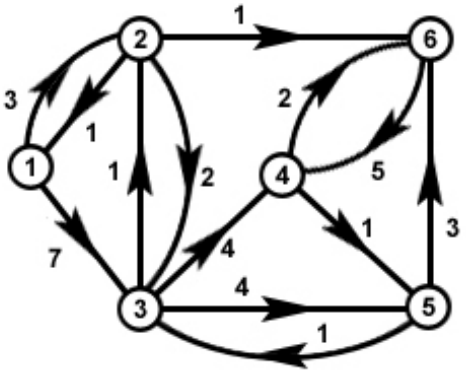
если $\lambda(v) > \lambda(s) + w(s, \text{i})$, *то* $\lambda(v) := \lambda(s) + w(s, \text{i})$; $\pi(v) :=$ s.

Из неокрашенных вершин выбирается вершина v с минимальным значением $\lambda(v)$ и окрашивается: $U := U \cup \{v\}$; $s := v$.

Шаг 3. Если все вершины окрашены, то есть $U = V$, то алгоритм завершен, иначе перейти к шагу 2.

[1] Эдсгер Дейкстра (Edsger Wybe Dijkstra; 1930 – 2002) – нидерландский математик.

Пример:



$$A = \begin{pmatrix} 0 & 3 & 7 & \infty & \infty & \infty \\ 1 & 0 & 2 & \infty & \infty & 1 \\ \infty & 1 & 0 & 4 & 4 & \infty \\ \infty & \infty & \infty & 0 & 1 & 2 \\ \infty & \infty & 1 & \infty & 0 & 3 \\ \infty & \infty & \infty & 5 & \infty & 0 \end{pmatrix}$$

Рис. 24.

№ итерации	λ(1)/ π(1)	λ(2) / π(2)	λ(3) / π(3)	λ(4) / π(4)	λ(5) / π(5)	λ(6) / π(6)	T
1	0/NIL	**3/1**	7/1	∞/NIL	∞/NIL	∞/NIL	[2,3,4,5,6]
2	0/NIL	3/1	5/2	∞/NIL	∞/NIL	**4/2**	[3,4,5,6]
3	0/NIL	3/1	**5/2**	9/6	∞/NIL	4/**2**	[3,4,5]
4	0/NIL	3/1	5	**9/6**	9/3	4/**2**	[4,5]
5	0/NIL	3/1	5/2	9/6	**9/3**	4/**2**	[5]

Восстановим кратчайший путь до вершины 4. Выписываем метки предшественников, начиная с вершины 4: 6, 2, 1. Переписав список вершин в обратном порядке, получаем кратчайший путь: 1, 2, 6, 4. Вес построенного кратчайшего пути равен 6.

Алгоритм Беллмана[1]-Форда[2] (1958, 1962)

Алгоритм Беллмана-Форда решает задачу о кратчайших путях из вершины s_0 до всех остальных вершин для графа $G=(V, \alpha)$, в котором допускаются отрицательные веса. Если из источника достижим контур отрицательной длины, то алгоритм позволяет это обнаружить. По-прежнему обозначим через $\lambda(v)$ оценку кратчайшего пути из s_0 в v.

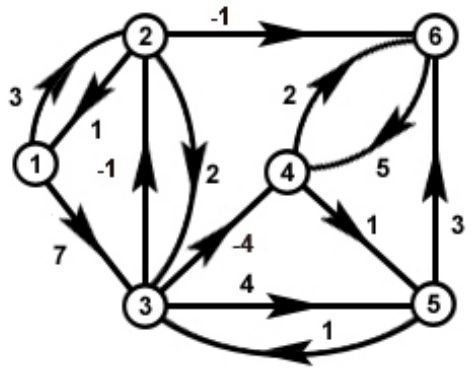
Вначале положим $\lambda(s_0) := 0$, $\pi(s_0) :=$ NIL, а для всех остальных вершин $\lambda(v) = \infty$, $\pi(v) :=$ NIL.

[1] Ричард Беллман (Richard Ernest Bellman, 1920 – 1984) – американский математик.
[2] Лестер Форд (Lester Randolph Ford, Jr., род. 1927) – американский математик.

Далее $|V|-1$ раз осуществляется релаксация по всем дугам графа. Далее проверяется, нет ли контура отрицательной длины, достижимого из источника s_0. Если такого контура нет, то оценки $\lambda(v)$ равны значению веса кратчайшего пути из s_0 в v, а перечисление предшественников, начиная с $\pi(v)$, в обратном порядке позволяет восстановить кратчайший путь.

Для определения контура отрицательной длины, достижимого из источника s_0, осуществляется релаксация по всем ребрам. Если хотя бы одна оценка $\lambda(v)$ изменит свое значение, то такой контур существует.

Пример:



$$A = \begin{pmatrix} 0 & 3 & 7 & \infty & \infty & \infty \\ 1 & 0 & 2 & \infty & \infty & -1 \\ \infty & -1 & 0 & -4 & 4 & \infty \\ \infty & \infty & \infty & 0 & 1 & 2 \\ \infty & \infty & 1 & \infty & 0 & 3 \\ \infty & \infty & \infty & 5 & \infty & 0 \end{pmatrix}$$

Рис. 25.

№ итерации	λ(1)/ π(1)	λ(2) / π(2)	λ(3) / π(3)	λ(4) / π(4)	λ(5) / π(5)	λ(6) / π(6)
1	0/NIL	3/1	7/1	∞/NIL	∞/NIL	∞/NIL
2	0/NIL	3/1	5/2	1/3	5/3	2/2
3	0/NIL	3/1	5/2	1/3	2/4	2/2
4	0/NIL	3/1	3/5	-1/3	0/4	1/4
5	0/NIL	2/3	1/5	-3/3	-2/4	0/4
6	0/NIL	0/3	-1/5	-5/3	-4/4	-3/4

Очевидно, что эффективность алгоритма Беллмана-Форда составляет $O(nm)$, где n и m число вершин и дуг графа соответственно.

Алгоритм Флойда[1]-Уоршала[2] (1962)

Алгоритм Флойда-Уоршала использует технику динамического программирования и позволяет находить кратчайшие пути между всеми парами вершин

[1] Роберт Флойд (Robert Floyd, 1936 – 2001) – американский математик.
[2] Стивен Уоршал (Stephen Warshall, 1935 – 2006) – американский математик.

графа $G = (V, \alpha)$. Как и в алгоритме Беллмана-Форда допускаются отрицательные веса, однако запрещаются контуры отрицательной длины.

Обозначим через A матрицу весов дуг, а через $D^m[i,j]$ – оценку кратчайшего пути из вершины i в j с промежуточными вершинами из множества $\{1, \ldots, m\}$. Вначале имеем: $D^0[i,j] := A[i,j]$.

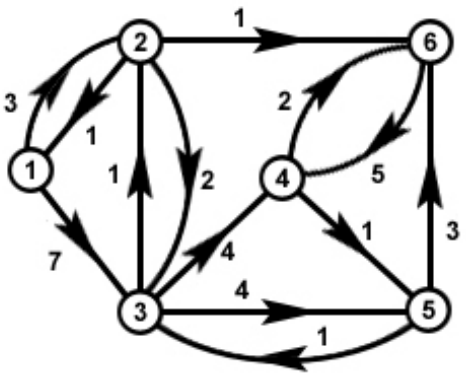
Для динамики используется соотношение:

$D^{(m+1)}[i, j] = \min\{D^m[i, j], D^m[i, m+1] + D^m[m+1, j]\}$.

В самом деле, кратчайший путь из i в j с промежуточными вершинами из множества $\{1, \ldots, (m + 1)\}$ может содержать вершину с номером $m + 1$ или нет. Если путь не содержит вершину $(m + 1)$, то $D^{(m+1)}[i, j] = D^m[i, j]$, а если содержит эту вершину, то путь можно разделить на две части: от i до $(m + 1)$ и от $(m + 1)$ до j, каждая из которых будет содержать промежуточные вершины из множества $\{1, \ldots, m\}$.

D^n содержит оценку кратчайшего пути с промежуточными вершинами из множества $\{1, \ldots, n\}$, то есть с любыми вершинами графа G, и следовательно, совпадает с искомым весом кратчайшего пути. Время работы алгоритма составляет $O(n^3)$.

Пример:



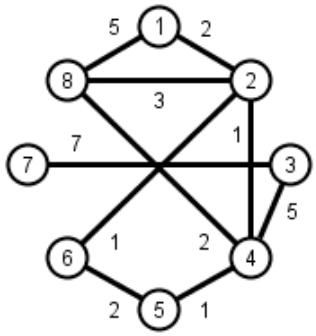
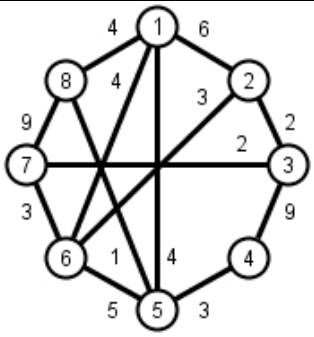
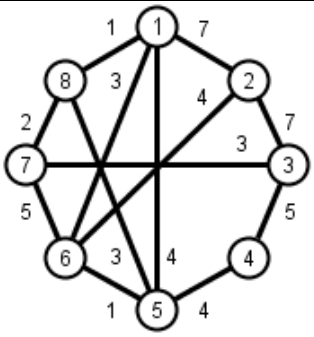
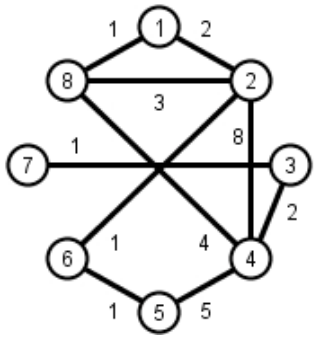
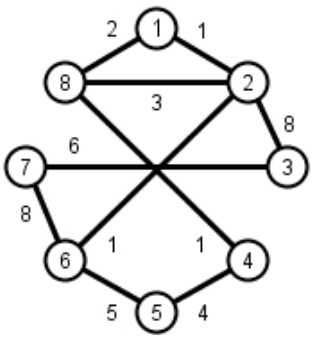
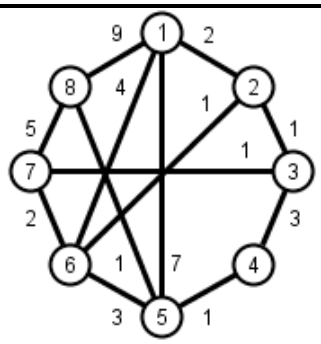
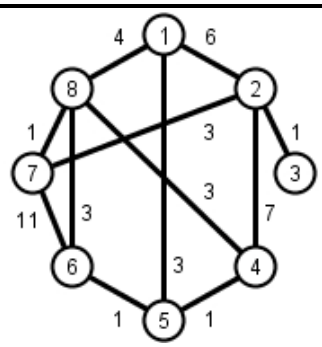
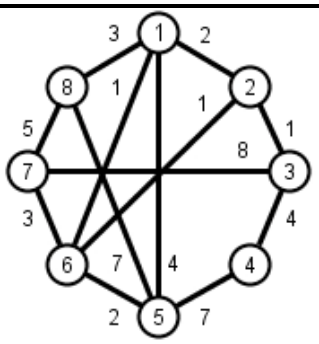
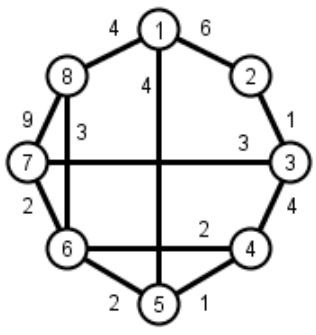
$$A = \begin{pmatrix} 0 & 3 & 7 & \infty & \infty & \infty \\ 1 & 0 & 2 & \infty & \infty & 1 \\ \infty & 1 & 0 & 4 & 4 & \infty \\ \infty & \infty & \infty & 0 & 1 & 2 \\ \infty & \infty & 1 & \infty & 0 & 3 \\ \infty & \infty & \infty & 5 & \infty & 0 \end{pmatrix}$$

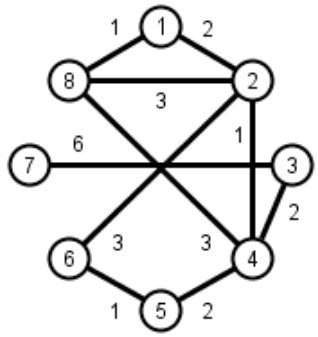
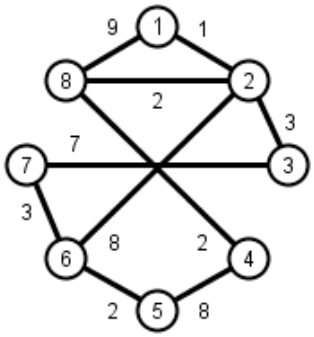
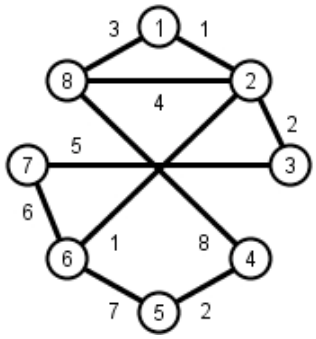
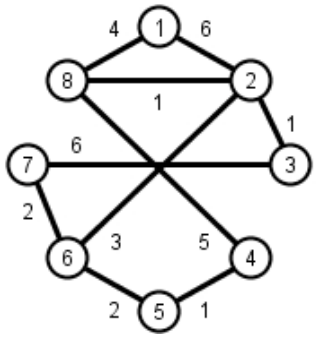
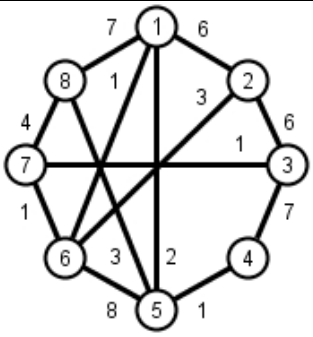
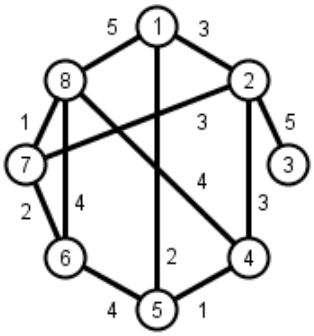
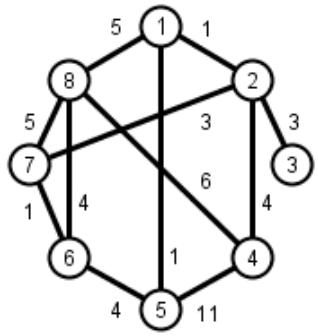
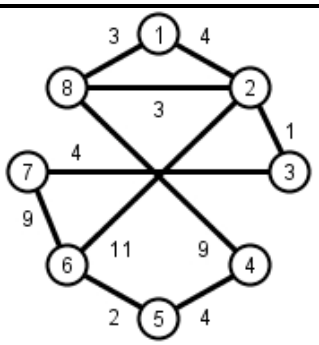
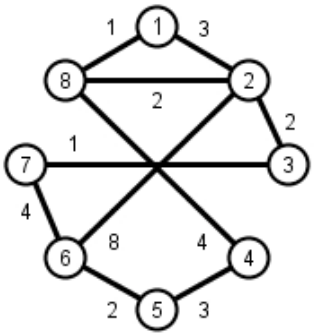
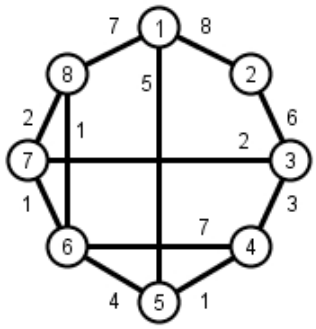
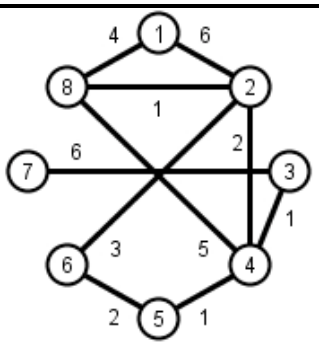
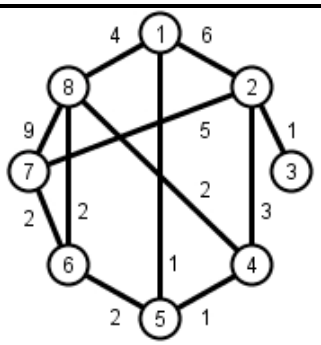
Рис. 26.

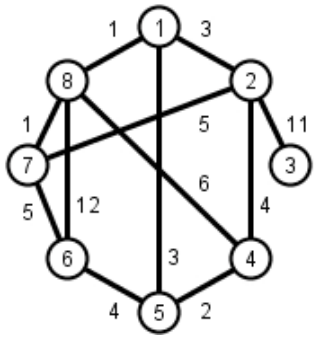
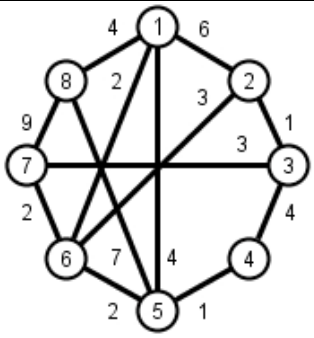
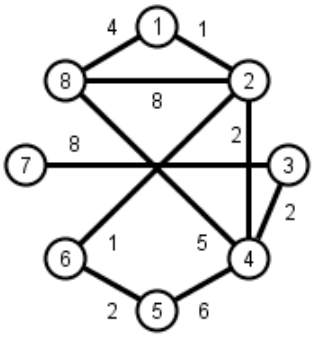
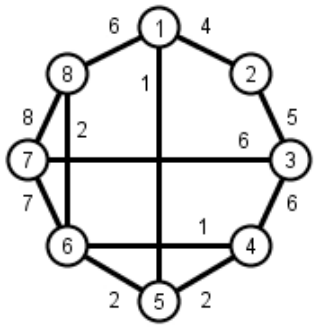
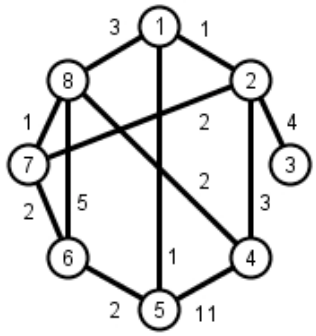
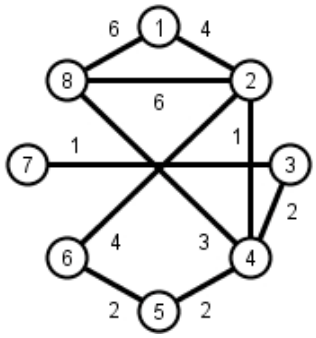
$D^{(1)}=\begin{matrix} 0 & 3 & 7 & \infty & \infty & \infty \\ 1 & 0 & 2 & \infty & \infty & 1 \\ \infty & 1 & 0 & 4 & 4 & \infty \\ \infty & \infty & \infty & 0 & 1 & 2 \\ \infty & \infty & 1 & \infty & 0 & 3 \\ \infty & \infty & \infty & 5 & \infty & 0 \end{matrix}$	$D^{(2)}=\begin{matrix} 0 & 3 & 5 & \infty & \infty & 4 \\ 1 & 0 & 2 & \infty & \infty & 1 \\ 2 & 1 & 0 & 4 & 4 & 2 \\ \infty & \infty & \infty & 0 & 1 & 2 \\ \infty & \infty & 1 & \infty & 0 & 3 \\ \infty & \infty & \infty & 5 & \infty & 0 \end{matrix}$	$D^{(3)}=\begin{matrix} 0 & 3 & 5 & 9 & 9 & 4 \\ 1 & 0 & 2 & 6 & 6 & 1 \\ 2 & 1 & 0 & 4 & 4 & 2 \\ \infty & \infty & \infty & 0 & 1 & 2 \\ 3 & 2 & 1 & 5 & 0 & 3 \\ \infty & \infty & \infty & 5 & \infty & 0 \end{matrix}$
$D^{(4)}=\begin{matrix} 0 & 3 & 5 & 9 & 9 & 4 \\ 1 & 0 & 2 & 6 & 6 & 1 \\ 2 & 1 & 0 & 4 & 4 & 2 \\ \infty & \infty & \infty & 0 & 1 & 2 \\ 3 & 2 & 1 & 5 & 0 & 3 \\ \infty & \infty & \infty & 5 & 6 & 0 \end{matrix}$	$D^{(5)}=\begin{matrix} 0 & 3 & 5 & 9 & 9 & 4 \\ 1 & 0 & 2 & 6 & 6 & 1 \\ 2 & 1 & 0 & 4 & 4 & 2 \\ 4 & 3 & 2 & 0 & 1 & 2 \\ 3 & 2 & 1 & 5 & 0 & 3 \\ 9 & 8 & 7 & 5 & 6 & 0 \end{matrix}$	$D^{(6)}=\begin{matrix} 0 & 3 & 5 & 9 & 9 & 4 \\ 1 & 0 & 2 & 6 & 6 & 1 \\ 2 & 1 & 0 & 4 & 4 & 2 \\ 4 & 3 & 2 & 0 & 1 & 2 \\ 3 & 2 & 1 & 5 & 0 & 3 \\ 9 & 8 & 7 & 5 & 6 & 0 \end{matrix}$

Задание 10. Минимальные остовы

Для заданного графа найдите минимальные остовы с помощью алгоритмов Прима и Краскала.

 10.1	 10.2	 10.3
 10.4	 10.5	 10.6
 10.7	 10.8	 10.9

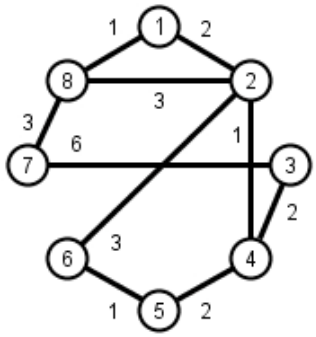
 10.10	 10.11	 10.12
 10.13	 10.14	 10.15
 10.16	 10.17	 10.18
 10.19	 10.20	 10.21

 10.22	 10.23	 10.24
 10.25	 10.26	 10.27

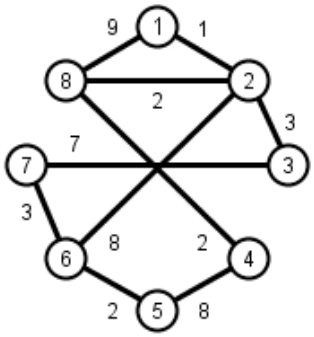
Задание 11. Кратчайшие пути

Для заданного графа найдите

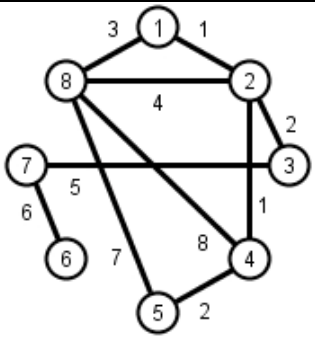
1) кратчайшие пути от вершины 1 до всех остальных вершин с помощью алгоритмов Дейкстри (если применим) и Форда-Беллмана.
2) кратчайшие пути между всеми парами вершин с помощью алгоритма Флойда-Уоршала.



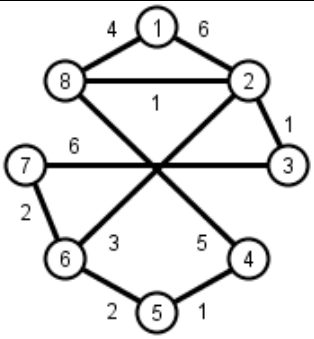
11.1



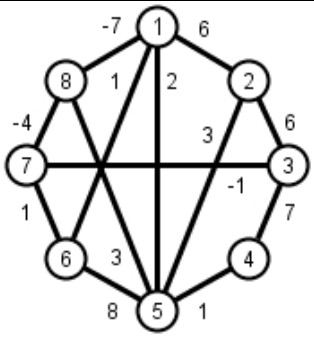
11.2



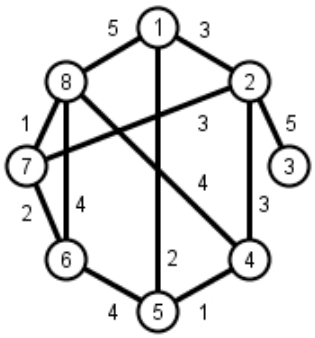
11.3



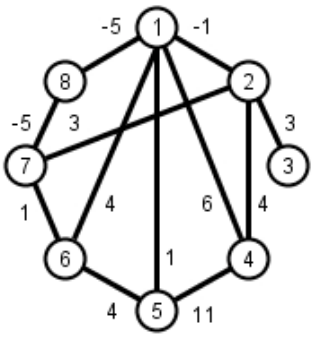
11.4



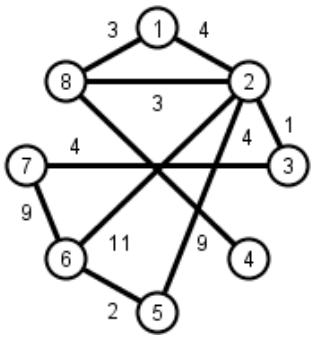
11.5



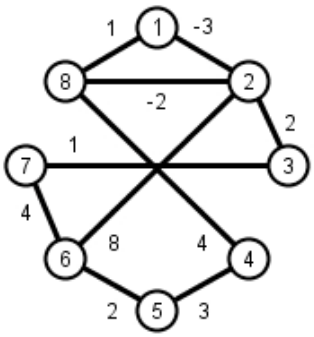
11.6



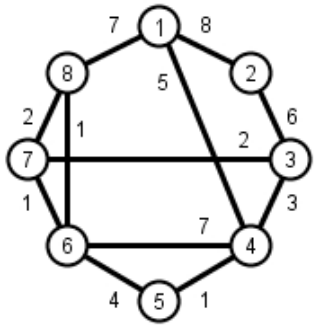
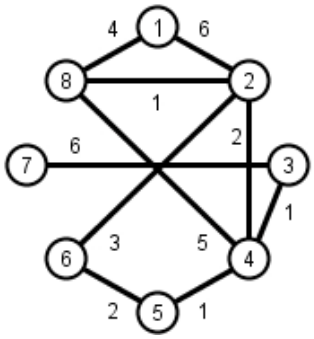
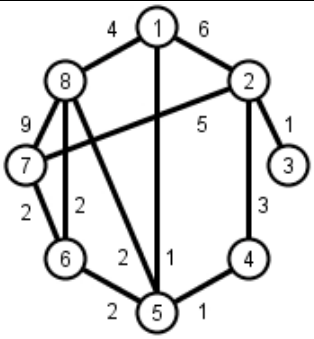
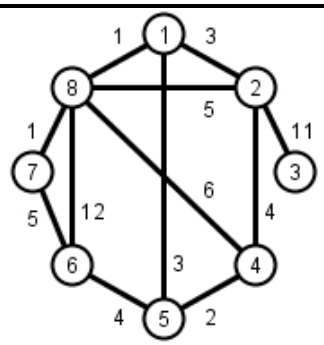
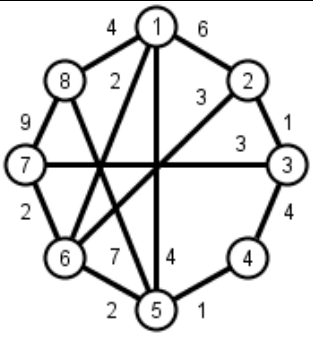
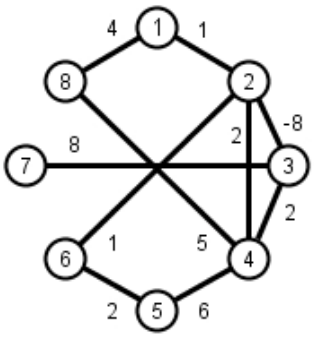
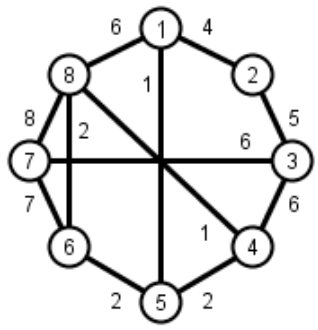
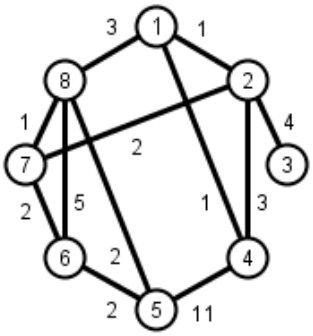
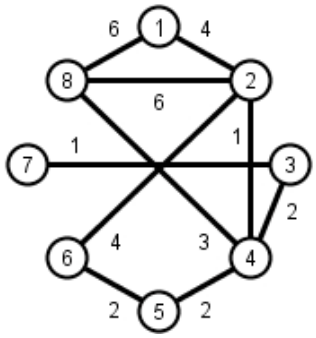
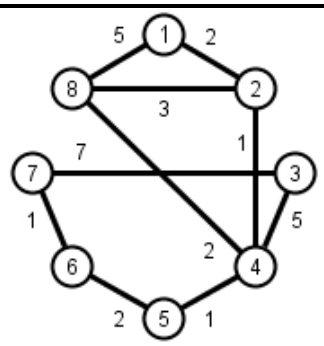
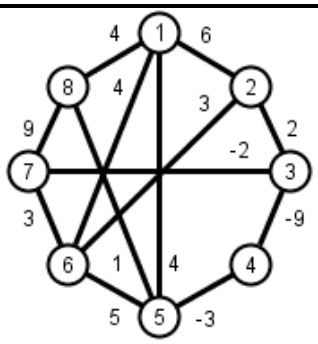
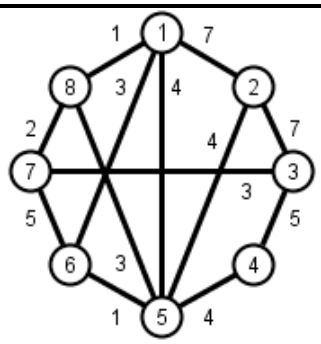
11.7

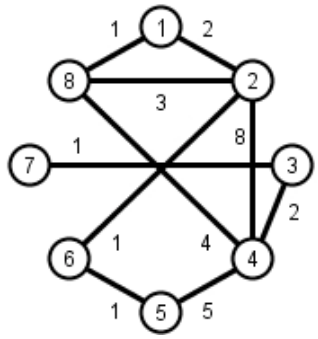
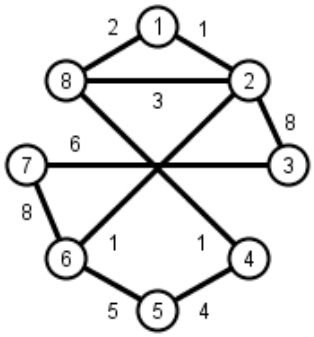
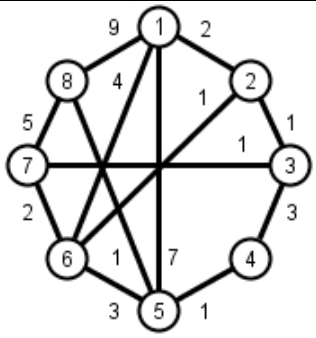
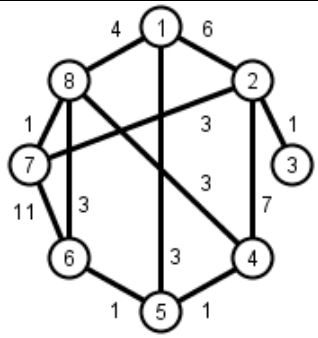
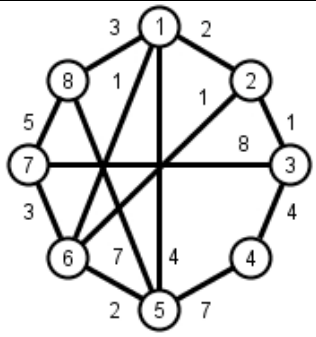
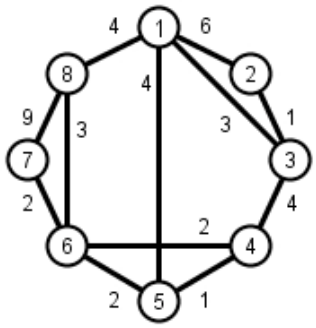


11.8



11.9

 11.10	 11.11	 11.12
 11.13	 11.14	 11.15
 11.16	 11.17	 11.18
 11.19	 11.20	 11.21

 11.22	 11.23	 11.24
 11.25	 11.26	 11.27

Дополнительные задания

Задание 12

Составьте программу на любом языке программирования с использованием только стандартных библиотек и модулей. Исходные данные должны вводиться из текстового файла, результаты должны сохраняться в текстовый файл. Для ввода, вывода и представления графов или сетей можно использовать любые способы.

12.1. Для заданного графа определите, является ли он связным.
12.2. Найдите все точки сочленения заданного графа.
12.3. Найдите все мосты заданного графа.
12.4. Для заданного графа определите, является ли он эйлеровым, и если да, то вывести эйлеров цикл.
12.5. Для заданного графа определите, имеет ли он эйлерову цепь, и если да, то вывести ее.
12.6. Для заданного орграфа определите, является ли он эйлеровым, и если да, то вывести эйлеров цикл.
12.7. Для заданного орграфа определите, имеет ли он эйлерову цепь, и если да, то вывести ее.
12.8. Для заданного орграфа определите, является ли он гамильтоновым и, если да, найдите гамильтонов цикл.
12.9. Для заданного орграфа определите, является ли он сильно связным.
12.10. Для заданного орграфа определите, является ли он бесконтурным.
12.11. Для заданного вектора определите, является ли он графическим по критерию Гавела-Хакими.
12.12. Для заданного вектора определите, является ли он графическим по критерию Эрдёша-Галлаи.
12.13. Для заданного числа вершин n постройте все вектора степеней n-вершинных графов.
12.14. Для заданного числа вершин n постройте все вектора степеней n-вершинных деревьев.
12.15. Постройте хотя бы одну реализацию заданного вектора степеней.
12.16. Постройте связную реализацию заданного вектора степеней.

12.17. Постройте все неизоморфные реализации заданного вектора степеней.
12.18. Проверить изоморфизм двух заданных графов.
12.19. Для заданного графа построить максимальный матричный код.
12.20. Для заданного графа построить минимальный матричный код.
12.21. Для двух заданных графов проверить вложение одного из них в другой.
12.22. Для заданной сети построить остовное дерево по алгоритму Прима.
12.23. Для заданной сети построить остовное дерево по алгоритму Краскала.
12.24. Для заданной сети найти кратчайшие пути от заданной вершины до всех остальных по алгоритму Дейкстры.
12.25. Для заданной сети найти кратчайшие пути от заданной вершины до всех остальных по алгоритму Беллмана-Форда.
12.26. Для заданной сети найти кратчайшие пути между всеми парами вершин по алгоритму Флойда-Уоршалла.
12.27. Для заданного дерева рассчитайте код Прюфера.
12.28. Для заданного дерева рассчитайте эксцентриситеты его вершин.
12.29. Для заданного дерева найдите его центр.
12.30. Для заданного дерева найдите его центроид.

Задание 13

13.1. Докажите, что самодополнительный граф не может содержать изолированных и полных вершин. Постройте все самодополнительные графы с числом вершин $n < 6$.

13.2. Докажите, что граф двудольный тогда и только тогда, когда он не содержит циклов нечетной длины.

13.3. Докажите, что если граф G несвязный, то его дополнение будет связным графом.

13.4. Докажите, что в любом 6-вершинном графе найдется либо 3 попарно смежных, либо 3 попарно несмежных вершины.

13.5. Докажите, что точки сочленения графа не являются точками сочленения в дополнении.

13.6. Каково наибольшее число точек сочленения в n-вершинном графе?

13.7. Каково наибольшее число мостов в n-вершинном графе?

13.8. Докажите, что всякое дерево является двудольным графом.

13.9. Докажите, что число ребер графа реконструируемо.

13.10. Докажите, что вектор степеней графа реконструируем.

13.11. Докажите, что всякий несвязный граф реконструируем.

13.12. Докажите, что всякое дерево реконструируемо.

13.13. Докажите, что любой турнир с числом вершин больше 3 содержит транзитивную тройку.

13.14. Докажите, что граф является вполне несвязным тогда и только тогда, когда все его подграфы тоже вполне несвязные.

13.15. Докажите, что граф является полным тогда и только тогда, когда все его подграфы тоже полные.

13.16. Докажите, что центр дерева не изменяется при удалении всех листьев.

13.17. Докажите, что хроматическое число нетривиального дерева равно 2.

13.18. Докажите, что хроматическое число двудольного графа равно 2.

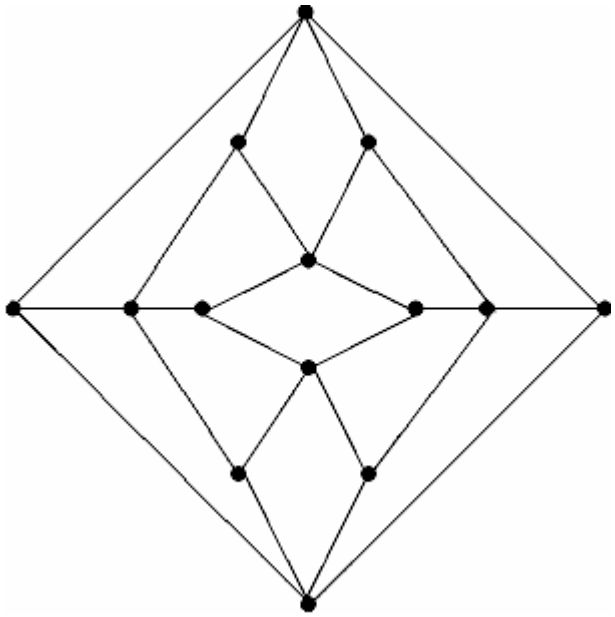
13.19. Найдите хроматическое число графа Петерсена.

13.20. Докажите или опровергните, что граф Петерсена гамильтонов.

13.21. Докажите или опровергните, что граф Петерсена гипогамильтонов.

13.22. Докажите или опровергните, что граф Петерсена является минимальным 1-расширением цикла.

13.23. Докажите или опровергните, что ромбический додекаэдр, изображенный на рисунке, является гамильтоновым.



13.24. Найдите хроматическое число ромбического додекаэдра.

13.25. Опишите минимальные вершинные 1-расширения звезд.

13.26. Опишите минимальные реберные 1-расширения звезд.

13.27. Опишите минимальные вершинные 1-расширения графов, являющихся объединением двух цепей.

13.28. Опишите минимальные реберные 1-расширения графов, являющихся объединением двух цепей.

13.29. Опишите графы, которые имеют минимальное вершинное 1-расширение, отличающееся на одно дополнительное ребро.

13.30. Опишите графы, которые имеют минимальное вершинное 1-расширение, отличающееся на два дополнительных ребра.

Литература

1. *Асанов М.О., Баранский В.А., Расин В.В.* Дискретная математика: графы, матроиды, алгоритмы. – Ижевск, 2001.
2. *Богомолов А.М., Салий В.Н.* Алгебраические основы теории дискретных систем. – М.: Наука, 1997.
3. *Зыков А.А.* Основы теории графов. – М.: Вузовская книга, 2004.
4. *Касьянов В.Н., Евстигнеев В.А.* Графы в программировании: обработка, визуализация и применение. – СПб.: БХВ, 2003.
5. *Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.* Алгоритмы: построение и анализ, 2-е издание. – М.: Издательский дом «Вильямс», 2005.
6. *Кристофидес Н.* Теория графов. Алгоритмический подход. – М.: Мир, 1978.
7. Лекции по теории графов, 2-е издание / *Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И.* – М.: Книжный дом «ЛИБРОКОМ», 2009.
8. *Липский В.* Комбинаторика для программистов. – М.: Мир, 1988.
9. *Окулов С.М.* Дискретная математика. Теория и практика решения задач по информатике. – М.: БИНОМ, 2008.
10. *Оре О.* Графы и их применение. – М.: Едиториал УРСС, 2002.
11. *Оре О.* Теория графов. – М.: Наука, 1980.
12. *Татт У.* Теория графов. – М.: Мир, 1988.
13. *Харари Ф.* Теория графов, 2-е издание. – М.: Едиториал УРСС, 2003.