Министерство науки и высшего образования Российской Федерации
Ярославский государственный университет им. П. Г. Демидова
Кафедра теоретической информатики

В. С. Рублев

Элементы теории графов

(индивидуальные работы № 8–10 по дисциплине «Дискретная математика»)

Учебно-методическое пособие

Ярославль
ЯрГУ
2020

УДК 519.1(075.8)
ББК В174.2я73
Р82

Рекомендовано
Редакционно-издательским советом университета
в качестве учебного издания. План 2020 года

Рецензент
кафедра теоретической информатики Ярославского государственного университета им. П. Г. Демидова

Рублев, Вадим Сергеевич.
Р82 Элементы теории графов : (индивидуальные работы № 8-10 по дисциплине «Дискретная математика») : учебно-методическое пособие / В. С. Рублев ; Яросл. гос. ун-т им. П. Г. Демидова. – Ярославль : ЯрГУ, 2020. – 172 с.

Пособие содержит варианты индивидуальных заданий по теме “Элементы теории графов”, а также необходимый материал для ее самостоятельного изучения и выполнения индивидуальных заданий. Для качественного усвоения курса в издании даны подробные определения, примеры, иллюстрации, обоснования, а также методические рекомендации для выполнения индивидуальных заданий.

Пособие предназначено для студентов, обучающихся по дисциплине «Дискретная математика».

УДК 519.1(075.8)
ББК В174.2я73

© ЯрГУ, 2020

Оглавление

1. Определения графов

Функциональное соответствие и его частный случай – взаимно-однозначное соответствие – наиболее простые бинарные отношения множеств. В общем случае, когда соответствие не является функциональным и элемент может иметь много образов, наглядное его представление – граф. Представление отношений в виде графов использовал еще Леонард Эйлер, но как наука теория графов возникла в 30-е годы прошлого века с выходом монографии Кенига "Теория графов". В России первой книгой, посвященной этой теории, является книга Клода Бержа "Теория графов и ее приложения" в переводе профессора А. А. Зыкова (патриарха графистов на пространстве бывшего СССР), изданная в начале 1960-х годов. Она проста для чтения, достаточное количество экземпляров этой книги имеется в библиотеке ЯрГУ. С тех пор было издано очень много учебников и монографий по теории графов как зарубежных, так и отечественных авторов.

Конечный граф (в последующем термин "конечный" всегда будет подразумеваться) G определяется как пара 2 конечных множеств, одно из которых называется множеством вершин графа (каждый элемент – вершина графа), а другое определяется как множество пар вершин (первого множества) и называется множеством ребер (каждый элемент – ребро, соединяющее 2 вершины графа). Таким образом, в обозначении $G(X, U)$ X – множество вершин, а U – множество ребер (пар вершин). Например,

$$G(\{x_1, x_2, x_3, x_4\}, \{\{x_1, x_2\}, \{x_1, x_3\}, \{x_3, x_4\}\})$$

– граф, содержащий 4 вершины x_1, x_2, x_3, x_4 и 3 ребра.

Геометрически вершины графа изображаются точками, а ребра — линиями, соединяющими эти вершины-точки. При этом неважно, где располагаются вершины и пересекаются или не пересекаются ребра. На рис. 1 приведены 3 изображения вышеприведенного графа G.

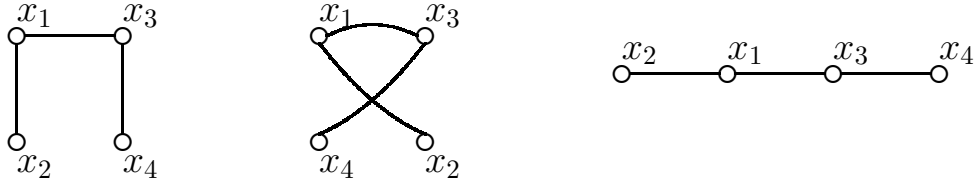

Рис. 1

Если в определении множества ребер U допустить пары одинаковых элементов (вершин), то каждая такая пара изображается линией, идущей из вершины в ту же самую вершину, а потому называется *петлей*, а соответствующий граф – *графом с петлями*. Примером служит следующий граф

$$G(\{x_1, x_2, x_3\}, \{\{x_1, x_2\}, \{x_2, x_2\}\}),$$

содержащий 3 вершины и 2 ребра, одно из которых петля (см. рис. 2).

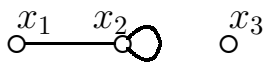

Рис. 2

Если в определении множества U ребер допустить повторяющиеся элементы (ребра), то получим *мультиграф*. Например, граф

$$G(\{y_1, y_2, y_3\}, \{\{y_1, y_2\}, \{y_2, y_3\}, \{y_3, y_2\}, \{y_2, y_3\}\})$$

содержит 3 ребра, соединяющих вершины y_2 и y_3 (см. рис. 3).

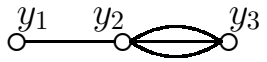

Рис. 3

Чаще всего мы будем рассматривать графы без петель и повторяющихся ребер, которые называют *обыкновенными графами*.

Если элементы множества U являются упорядоченными парами вершин, то граф называется *ориентированным графом*, или *орграфом*, а элементы множества U называются *дугами*. Например, орграф

$$G(\{y_1, y_2, y_3\}, \{(y_1, y_2), (y_2, y_1), (y_2, y_3)\})$$

содержит 3 вершины и 3 дуги, 2 из которых соединяют вершины y_1, y_2 и идут в разных направлениях. В наглядном изображении орграфа дуга – это стрелка, указывающая направление от вершины начала дуги к вершине конца дуги (см. рис. 4).

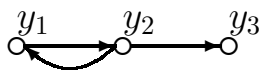

Рис. 4

Вершины x и y ребра $\{x, y\}$ называются *концами ребра*. Если вершина x является концом ребра u, то говорят, что x и u *инцидентны* (вершина x инцидентна ребру u, а ребро u инцидентно вершине x). Вершины x и y определяются как *смежные*, если существует ребро $\{x, y\}$, связывающее эти вершины (инцидентное этим вершинам). Два ребра u и v являются *смежными*, если они инцидентны одной и той же вершине.

Степень вершины x определяется как число $d(x)$ ребер, инцидентных x. Обозначим $n(G) \equiv |X|$ – число вершин графа $G(X, U)$, а $m(G) \equiv |U|$ – число ребер этого графа.

Занумеруем все вершины графа $X = \{x_1, x_2, \ldots, x_n\}$. Число ребер и степени вершин связаны следующим соотношением:

$$m = \frac{1}{2} \sum_{i=1}^{n} d(x_i),$$

которое обосновывается таким образом: если мы просуммируем степени всех вершин, то мы пересчитаем дважды каждое ребро (один раз, когда мы учитываем ребро в степени одной из вершин, которой оно инцидентно, а второй раз, когда мы учитываем ребро в степени другой вершины, которой оно инцидентно).

В орграфе дуга имеет *начало* (вершина, из которой она выходит) и *конец* (вершина, в которую она входит). В соответствии с этим могут быть дуги, *выходящие* из вершины, и дуги, *входящие* в вершину. Поэтому вместо степени вершины в орграфе для вершины определяют *полустепень исхода* $d^-(x)$ как число дуг, исходящих из вершины, и *полустепень захода* $d^+(x)$ как число дуг, входящих в вершину. Подсчитать все дуги можно, если просуммировать все полустепени исхода или же просуммировать все полустепени захода. Из этого следует соотношение:

$$m = \sum_{i=1}^{n} d^-(x_i) = \sum_{i=1}^{n} d^+(x_i).$$

Вершина графа, имеющая степень 0, называется *изолированной*. Вершина графа, имеющая степень 1, называется *висячей*.

Граф называется *полным*, если любые 2 его вершины соединены ребром, и пустым, если все его вершины изолированы (множество ребер пусто). Полный граф, имеющий n вершин, обозначается как K_n. На

рис. 5 приведено изображение графа K_5.

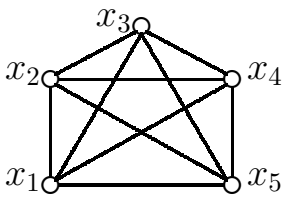

Рис. 5

Граф $G(X, U)$ называется *двудольным*, если множество X его вершин может быть разбито на 2 таких непересекающихся подмножества ($X = X_1 \cup X_2, X_1 \cap X_2 = \emptyset$ – доли графа), для которых смежными могут быть только вершины разных долей: $\{x_i, x_j\} \in U \wedge x_i \in X_1 \ \rightarrow x_j \in X_2$. На рис. 6 приведен пример такого графа с долями: $X_1 = \{x_1, x_2, x_3\}$, $X_2 = \{x_4, x_5\}$.

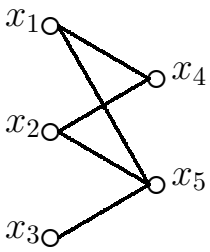

Рис. 6

Полный двудольный граф – это двудольный граф, у которого любые 2 вершины, принадлежащие разным долям, смежны. Полный двудольный граф, имеющий n вершин в первой доле и m вершин во второй доле (и потому $n \cdot m$ ребер), обозначается как $K_{n,m}$. Например, если в граф, изображенный на рис. 6, добавить ребро $\{x_3, x_4\}$, то мы получим полный двудольный граф $K_{3,2}$.

Обобщением двудольного графа является n-дольный граф. Он определяется разбиением множества вершин на n подмножеств-долей:

$$X = \bigcup_{i=1}^{n} X_i, \quad X_k \cap X_j = \emptyset \ (k \neq j),$$

а также смежностью только вершин, принадлежащих соседним долям:

$$\{x, y\} \in U, \ x \in X_i \ (1 < i < n) \ \rightarrow \ y \in X_{i+1} \ \vee \ y \in X_{i-1}.$$

На рис. 7 изображен 3-дольный граф.

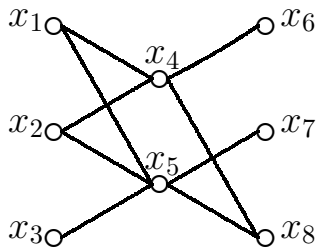

Рис. 7

2. Изоморфизм графов и алгоритмические способы задания графа

Структура графа не изменится, если переименовать его вершины и ребра: характеристики графа, такие как количество вершин, количество ребер, набор степеней вершин, количество элементарных циклов с заданным числом вершин и другие не зависят от наименования вершин и ребер. Поэтому графы, отличающиеся только наименованием вершин и ребер, будем называть *изоморфными*. В теории графы изучаются с точностью до изоморфизма. Дадим точное определение изоморфизма графов.

Графы $G_1(X, U)$ и $G_2(Y, V)$ называются изоморфными, если между множествами их вершин X и Y можно установить взаимнооднозначное соответствие φ, сохраняющее смежность, т. е. такое, что

$$\forall x_i, x_j \in X \quad \{x_i, x_j\} \in U \;\leftrightarrow\; \{\varphi(x_i), \varphi(x_j)\} \in V.$$

Таким образом, при изоморфизме смежным вершинам графа G_1 соответствуют смежные вершины графа G_2, а несмежным вершинам из G_1 – несмежные вершины из G_2.

Пусть дан граф $G(X, U)$. В силу изоморфизма мы можем считать, что множество вершин $X = \{1, 2, \ldots, n\}$. Таким образом, для задания множества вершин достаточно указать их число n. Множество ребер можно задать *списком ребер* как списком m пар номеров вершин. При этом безразлично, какой из номеров в паре будет первым, а какой – вторым. Но в некоторых случаях удобно, когда номера вершин в

паре (ребра) упорядочены и когда сами ребра упорядочены по номеру первой в паре вершины. Для определения степени $d(k)$ вершины k необходимо подсчитать количество пар, в которых k находится на первом или втором месте. Трудоемкость такого алгоритма – $O(m)$. Например, список

$$\{1,2\},\{1,3\},\{1,4\},\{2,4\},\{3,4\}$$

определяет граф с 5 ребрами, где степень вершины 3 равна 2. Если же каждое ребро определить в списке дважды (один раз, когда одна из вершин стоит на первом месте, а второй раз, когда другая вершина стоит на первом месте) и этот список упорядочить по первой вершине, то тогда для определения степени вершины необходимо подсчитать количество пар, где эта вершина стоит на первом месте. Трудоемкость такого алгоритма $O(\log m + \max_i d(i))$. Для примера, написанного выше, этот список выглядит следующим образом:

$$\{1,2\},\{1,3\},\{1,4\},\{2,1\},\{2,4\},\{3,1\},\{3,4\},\{4,1\},\{4,2\},\{4,3\}.$$

Для орграфа порядок вершин в дуге ориентирует ее и потому определяет порядок в паре. Например, *список дуг*

$$(1,2),(1,4),(2,4),(3,1),(4,3)$$

определяет орграф с пятью дугами, которые получены ориентацией ребер предыдущего графа. $d_4^+ = 2,\ d_4^- = 1$.

В случае неориентированного графа иногда удобно задавать каждое ребро дважды. Например, это упрощает и ускоряет алгоритм определения степени вершины.

Другой удобный способ задания графа $G(X,U)$ – *матрица смежности вершин*. Она представляет собой матрицу A размерности $n \times n$, элемент a_{ij} которой равен 1, если вершины i и j смежны (есть ребро $\{i,j\}$), или 0, если несмежны (нет такого ребра).

$$a_{ij} = \begin{cases} 1, & \{i,j\} \in U \\ 0, & \{i,j\} \notin U. \end{cases}$$

Матрица смежности обыкновенного графа без петель имеет нулевую главную диагональ и является симметрической: $a_{ij} = a_{ji}$. В качестве

примера приведем матрицу смежности для обыкновенного графа, заданного выше списком ребер:

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

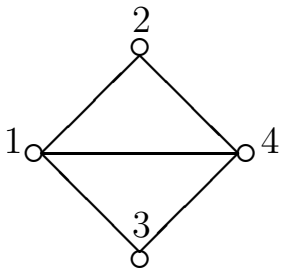

Рис. 8

Изображение этого графа приведено на рис. 8.

Граф с петлями имеет на диагонали единицы.

Элементы матрицы смежности орграфа определяются следующим образом:

$$a_{ij} = \begin{cases} 1, & (i,j) \in U \\ -1, & (j,i) \in U \\ 0, & (i,j) \notin U, (j,i) \notin U. \end{cases}$$

Матрица смежности орграфа является кососимметрической:

$$a_{ji} = -a_{ij}.$$

В качестве примера приведем матрицу смежности орграфа, заданного выше списком дуг:

$$\begin{bmatrix} 0 & 1 & -1 & 1 \\ -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ -1 & -1 & 1 & 0 \end{bmatrix}$$

На рис. 9 приведено изображение этого графа.

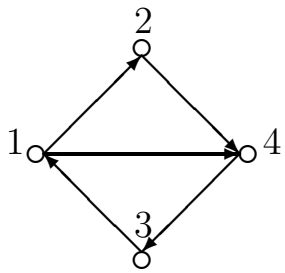

Рис. 9

В случае мультиграфа элемент матрицы смежности вершин обыкновенного графа может иметь значение k, если k дуг связывают вершины, соответствующие индексам элемента матрицы.

Еще один способ задания графа – *матрица инцидентности*. Элемент b_{ij} ($i \in \overline{1,n},\ j \in \overline{1,m}$) матрицы инцидентности B определяется следующим образом:

$$b_{ij} = \begin{cases} 1, & \text{вершина } i \text{ инцидентна ребру } j, \\ 0, & \text{вершина } i \text{ не инцидентна ребру } j. \end{cases}$$

В каждом столбце матрицы инцидентности имеются ровно 2 единицы, соответствующие вершинам, которые являются концами ребра. Число единиц в каждой строке определяет степень соответствующей вершины. В качестве примера приведем матрицу инцидентности для графа, заданного выше списком ребер:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Матрица инцидентности орграфа определяется подобным образом.

$$b_{ij} = \begin{cases} -1, & \text{из вершины } i \text{ исходит дуга } j, \\ 1, & \text{в вершину } i \text{ входит дуга } j, \\ 0, & \text{вершина } i \text{ не инцидентна дуге } j. \end{cases}$$

В каждом столбце матрицы инцидентности орграфа имеются ровно 1 единица и 1 минус единица, соответствующие вершинам, которые являются выходящим и входящим концами дуги. Число единиц в каждой строке определяет полустепень захода соответствующей вершины,

а число минус единиц в каждой строке – полустепень исхода соответствующей вершины. В качестве примера приведем матрицу инцидентности для графа, заданного выше списком дуг:

$$\begin{bmatrix} -1 & -1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 1 & 1 & 0 & -1 \end{bmatrix}$$

Выбор алгоритмического способа задания графа зависит от удобства конкретных алгоритмов работы с графом. Матрица смежности вершин может являться неэкономным способом, если каждый элемент задается в памяти компьютера как целое число. Но если такую матрицу определить как массив строк из нулей и единиц (строка бит в памяти компьютера), то можно не только сэкономить на памяти, но и получить эффективный способ для некоторых операций работы с графом. Так, если нужно найти все вершины, каждая из которых смежна с каждой из вершин заданного множества, то достаточно взять логическое произведение строк соответствующих вершинам множества. В вышеописанном примере матрицы смежности обыкновенного графа для получения вершин, смежных с вершинами 1 и 4, после логического умножения первой и четвертой строк матрицы получим строку

$$0\ 1\ 1\ 0,$$

которая определяет вершины 2 и 3.

3. Операции над графами

Рассмотрим операции, выделяющие части графа или образующие новые графы по уже заданным.

Граф $G_1(X_1, U_1)$ называется подграфом графа $G(X, U)$, если

$$X_1 \subseteq X, \quad U_1 \subseteq U.$$

Если $G_1 \neq G$, то подграф *собственный*. Если подграф G_1 графа G является полным (любые 2 его вершины смежны) и содержащим n вершин, то он называется *кликой* из n вершин.

Подграф G_1 графа G называется *остовным*, если $X_1 = X$, т. е. он содержит все вершины графа. Таким образом, остовный подграф образуется из графа удалением некоторых ребер, а любой подграф – из графа, возможно, удалением некоторых вершин (не всех) и инцидентных им ребер, а также иногда удалением еще некоторых ребер. Например, для графа $G(\{1,2,3,4\},\{\{1,3\},\{1,4\},\{2,3\}\})$ следующий подграф $G_1(\{1,2,3,4\},\{\{1,3\},\{1,4\}\})$ является остовным подграфом, а подграф $G_2(\{2,3,4\},\{\{2,3\}\})$ не является остовным подграфом.

Граф $\overline{G}(X,\overline{U})$ называется дополнением графа $G(X,U)$ (дополнительным графом), если $u \in \overline{U} \leftrightarrow u \notin U$. Таким образом, в дополнении графа смежные вершины не смежны, а несмежные вершины смежны. Например, для графа пятиугольника

$$G(\{1,2,3,4,5\},\{\{1,2\},\{2,3\},\{3,4\},\{4,5\},\{5,1\}\})$$

дополнением является звезда

$$\overline{G}(\{1,2,3,4,5\},\{\{1,4\},\{4,2\},\{2,5\},\{5,3\},\{3,1\}\}),$$

которая изоморфна графу G, а для графа

$$P(\{1,2,3,4\},\{\{1,2\},\{2,3\},\{2,4\},\{3,4\}\})$$

дополнением является граф

$$\overline{P}(\{1,2,3,4\},\{\{1,3\},\{1,4\}\}),$$

не изоморфный графу P.

Матрица смежности $\overline{A}$ дополнительного графа получается *инвертированием* не принадлежащих главной диагонали элементов матрицы A основного графа, т. е. заменой нулей на единицы и единиц на нули для всех элементов матрицы, кроме диагонали. Так, для графа пятиугольника матрица смежности вершин:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix},$$

а матрица смежности вершин дополнительного графа – звезды:

$$\overline{A} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

В тех случаях, когда уже определена матрица смежности графа, для задания подграфа, содержащего все ребра графа, которые инцидентны вершинам подграфа, не обязательно задавать его матрицу смежности – достаточно задать булев вектор из нулей и единиц, где единицы определяют вершины подграфа. Логическое произведение этой строки на строки матрицы, соответствующие вершинам подграфа, определяют все ребра, инцидентные вершинам подграфа.

В некоторых случаях полезными операциями над графами являются *объединение графов* и *прямое произведение графов*. Под объединением графов $G(X,U)$ и $H(Y,V)$ понимают граф $E(X \cup Y, U \cup V)$. Прямое произведение графов $G(X,U) \times H(Y,V) = E(X \times Y, W)$, где
$W = \{\ \{(x_i^g, y_k^h), (x_j^g, y_l^h)\} \mid$
$(\{x_i^g, x_j^g\} \in U \wedge\ y_k^h = y_l^h)\ \vee\ (\{y_k^h, y_l^h\} \in V \wedge\ x_i^g = x_j^g)\}.$

4. Методика установления изоморфизма графов

Следующая лемма обосновывается просто.

Лемма. *Графы $G_1(X,U)$ и $G_2(Y,V)$ изоморфны тогда и только тогда, когда изоморфны их дополнения $\overline{G_1}(X,\overline{U})$ и $\overline{G_2}(Y,\overline{V})$.*

Этой леммой имеет смысл пользоваться в тех случаях, когда количество ребер дополнительного графа меньше количества ребер основного графа.

Из определения изоморфизма следует, что для изоморфных графов должно совпадать количество вершин, количество ребер, количество вершин с одинаковой степенью, количество элементарных циклов с заданным числом вершин и другие характеристики графов, не зависящие от наименования (или нумерации) вершин. Поэтому если при попытке установить изоморфность графов оказывается, что одна из таких *характеристик различна*, то *графы неизоморфны*. Если же одна или

несколько таких характеристик совпадают, то из этого вовсе не следует изоморфность графов: для обоснования изоморфности необходимо установить 1-1с, сохраняющее смежность.

В общем случае нет хорошего алгоритма для установления изоморфности двух графов, существенно отличного (по сложности) от перебора $n!$ вариантов проверки 1-1с, где n – число вершин каждого из графов. Однако для графов с небольшим количеством вершин можно указать некоторую последовательность действий, либо приводящую к выводу о неизоморфности графов, либо ведущую к построению функции изоморфизма. Мы будем считать, что графы, для которых исследуется вопрос их изоморфности, связны, так как в противном случае можно исследовать вопрос для каждой пары компонент связности графов.

1. Найти количество вершин $n(G_i)$ $(i = 1, 2)$ и количество ребер $m(G_i)$ $(i = 1, 2)$ каждого из графов. Если они не совпадают: $n(G_1) \neq n(G_2)$ или $m(G_1) \neq m(G_2)$, то графы неизоморфны.

2. Найти наборы степеней вершин каждого из графов $(n_1^1, n_2^1, \dots)$ и $(n_1^2, n_2^2, \dots)$, где n_k^i означает количество вершин степени k i-го графа. Если эти наборы не совпадают, то графы неизоморфны.

3. Если наборы степеней вершин графов совпадают и есть вершины разных степеней, то делаем попытку построения изоморфизма φ следующим образом:

   1^o. Находим степень i_0, для которой число вершин в наборе минимально: $n_{i_0} = \min_i\{n_i\}$.

   2^o. Делаем попытку установления соответствия φ между вершинами множества $X_i = \{x \in X |\ d(x) = i_0\}$ и вершинами $Y_i = \{y \in Y |\ d(y) = i_0\}$. Для этого для каждой вершины $x \in X_i$ находим набор степеней вершин, смежных с ней. То же делаем для каждой вершины $y \in Y_i$. Если эти наборы можно сопоставить с несколькими вариантами, то выбрать один из них (и положить соответствие между сопоставленными вершинами), а остальные запомнить для последующих выборов, если принятый выбор окажется неприемлемым. Если наборы нельзя сопоставить, то значит выбор предыдущего варианта сопоставления (если он есть) является неудачным и потому

следует вернуться к нему (ликвидировав все варианты выборов после него) и взять следующий вариант сопоставления. Если такого нет, то снова вернуться к предыдущему варианту выбора сопоставления. Если такого нет, то графы неизоморфны.

3^o. Для каждого из наборов уже сопоставленных вершин найти вершины, смежные с ними, но не входящие в сопоставление, установить их относительную смежность и выбрать для сопоставления в каждом наборе те вершины, которые имеют наименьшую относительную смежность. Если таких вариантов несколько, то выбрать один из них, запомнив остальные для последующих выборов. Этот пункт выполнять до тех пор, пока все вершины обоих графов не окажутся сопоставленными (и это сопоставление дает изоморфизм графов) либо пока не будет установлено, что нет вариантов сопоставления. В последнем случае надо вернуться к предыдущему варианту сопоставления (ликвидировав все варианты выборов после него) и взять следующий вариант сопоставления. Если такого нет, то снова вернуться к предыдущему варианту выбора сопоставления. Если такого нет, то графы неизоморфны.

4. Если все вершины имеют одинаковую степень, то можно применить прием попытки установления соответствия между вершинами, входящими в элементарные циклы (цикл, не содержащий повторяющихся вершин и ребер) одинаковой длины. Если количество циклов какой-либо длины (например, наименьшей из имеющихся) будет различным, то графы не являются изоморфными. Однако если такие количества одинаковы, то нельзя еще сделать вывод об изоморфизме графов. В этом случае надо попытаться сопоставлять вершины, которые не входят ни в один из элементарных циклов некоторой длины (например, минимальной длины) или сопоставлять вершины, которые являются общими для некоторых циклов одной и той же длины либо являются концами ребер связывающих вершины разных циклов одинаковой длины и т. д. После такого сопоставления надо перейти к дальнейшему сопоставлению, используя предыдущий пункт этого алгоритма.

5. Предыдущий прием хорошо работает, когда графы не являются изоморфными. Например, в одном графе есть цикл определенной длины, которого нет в другом графе. Если же это не так, то требуется очень большое внимание при подсчете циклов. Неаккуратность приводит к ошибочному заключению.

   В случае предполагаемой изоморфности при одинаковости степеней всех вершин каждого графа можно применить следующий прием сопоставления вершин по набору элементарных циклов, в которые входит эта вершина. Определим для каждого ребра набор степеней вершин, смежных с обоими вершинами ребра, и будем сопоставлять те ребра обоих графов, для которых такие наборы одинаковы. При невозможности такого сопоставления для каких-либо ребер графы неизоморфны. При неоднозначности сопоставления выбираем один из вариантов, запоминая остальные для последующих выборов. В сопоставляемых ребрах также выбирается один из двух вариантов сопоставления вершин, инцидентных ребрам (второй вариант запоминается). Прием повторяется, но при этом определяются наборы степеней вершин, смежных с обоими концами ребер, не вошедших уже в сопоставление. Либо такое повторение приводит к установлению изоморфизма, либо при неудаче в сопоставлении следует вернуться к предыдущему выбору (ликвидировав все варианты выборов после него) и взять следующий вариант сопоставления. Если такого нет, то снова вернуться к предыдущему варианту выбора сопоставления. Если такого нет, то графы неизоморфны.

5. Примеры задач на изоморфизм графов с их решениями

Пример 1. Граф $G_1(X, U)$ задан матрицей смежности вершин

$$\begin{bmatrix} 010 & 001 & 10 \\ 101 & 001 & 00 \\ 010 & 110 & 00 \\ 001 & 001 & 01 \\ 001 & 000 & 11 \\ 110 & 100 & 00 \\ 100 & 010 & 01 \\ 000 & 110 & 10 \end{bmatrix},$$

а граф $G_2(Y, V)$ задан списком ребер: ({1,2}, {1,4}, {1,5}, {2,3}, {2,6}, {3,4}, {3,7}, {4,8}, {5,6}, {5,8}, {6,7}, {7,8}).

Реализации (изображения) графов приведены на рис. 10 и 11.

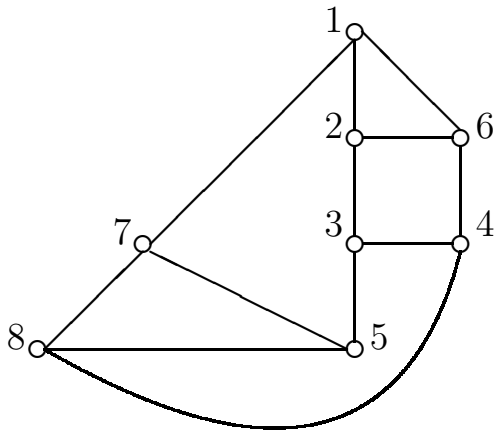

Рис. 10 (граф G_1)

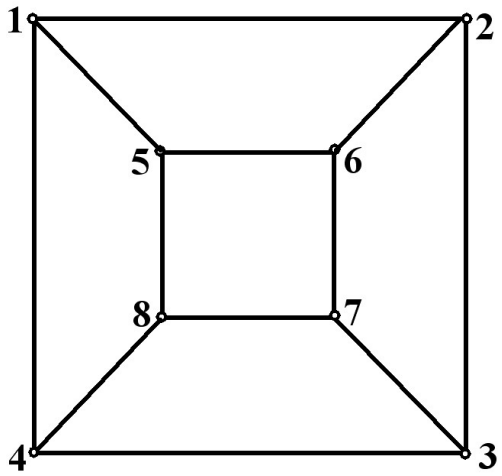

Рис. 11 (граф G_2)

Оба графа имеют по 8 вершин и все вершины степени 3. Поэтому число ребер у них одинаковое, и алгоритм сопоставления по степеням может иметь полный перебор сопоставления вершин, если графы не изоморфны. Попробуем использовать прием сопоставления по длине элементарных циклов этих графов. Наглядно видно на рисунках, что G_1 имеет 2 элементарных цикла длины 3, а G_2 не имеет таких циклов. Поэтому графы G_1 и G_2 не изоморфны.

Пример 2. Граф $G_1(X, U)$ задан следующей матрицей смежности вершин:

$$\begin{bmatrix} 010 & 001 & 10 \\ 101 & 001 & 00 \\ 010 & 110 & 00 \\ 001 & 001 & 01 \\ 001 & 001 & 11 \\ 110 & 110 & 00 \\ 100 & 010 & 01 \\ 000 & 110 & 10 \end{bmatrix},$$

а граф $G_2(Y, V)$ задан списком ребер:
({1,4}, {1,5}, {1,7} {1,8}, {2,4}, {2,5}, {2,8}, {3,5}, {3,6}, {3,8}, {4,7}, {6,7}, {6,8}).

Оба графа имеют по 8 вершин, 2 из которых имеют степень 4, а остальные 6 – степень 3. При попытке установить изоморфность графов есть 2 варианта отождествления вершин степени 4 графов G_1 и G_2:

{5–1, 6–8} и {5–8, 6–1}. Выберем первый вариант φ ={5–1, 6–8}, а второй – {5–8, 6–1} оставим в памяти для последующего выбора, если первый вариант будет отвергнут.

Вершина 5 графа G_1, кроме уже выбранной в соответствие вершины 6, смежна с вершинами 3, 7, 8, из которых 7 и 8 смежны между собой, а соответствующая ей вершина 1 графа G_2, кроме уже выбранной в соответствие вершины 8, смежна с вершинами 4, 5, 7, из которых 4 и 7 смежны между собой. Поэтому продолжением является включение в строящийся изоморфизм φ соответствия 3–5, как единственных вершин соответствующих графов, которые смежны с уже включенной в φ парой вершин 5–1 и несмежны с другими вершинами, смежными с этой парой: φ={5–1, 6–8, 3–5}.

Вершина 6 графа G_1, кроме уже выбранной в соответствие вершины 5, смежна с вершинами 1, 2, 4, из которых 1 и 2 смежны между собой, а соответствующая ей вершина 8 графа G_2, кроме уже выбранной в соответствие вершины 1 смежна с вершинами 2, 3, 6, из которых 3 и 6 смежны между собой. Поэтому продолжением является включение в строящийся изоморфизм φ соответствия 4–2, как единственных вершин соответствующих графов, которые смежны с уже включенной в φ парой вершин 6–8 и не смежны с другими вершинами, смежными с этой парой: φ={5–1, 6–8, 3–5, 4–2}.

Среди еще не рассмотренных вершин, смежных с вершиной 5 графа G_1, вершины 7 и 8, а среди еще не рассмотренных вершин графа G_2, смежных с соответствующей вершиной 1, вершины 4 и 7, которые также смежны между собой. Поэтому для дальнейшего есть два варианта выбора соответствия вершин: {7–4, 8–7} и {7–7, 8–4}. Выберем первый вариант φ={5–1, 6–8, 3–5, 4–2, 7–4, 8–7}, а второй – {5–1, 6–8, 3–5, 4–2, 7–7, 8–4} оставим в памяти для последующего выбора, если первый вариант этого выбора будет отвергнут.

Среди еще не рассмотренных вершин графа G_1 вершины 1 и 2, смежные с вершиной 6 и смежные между собой. Им соответствуют в графе G_2 вершины 3 и 6, смежные с соответствующей вершиной 8 и смежные между собой. Есть два варианта выбора соответствия вершин графов: {1–3, 2–6} и {1–6, 2–3}. Но выбор первого варианта ведет к противоречию: вершина 1 графа G_1 смежна с вершиной 7 этого графа, а соответствующая ей вершина 3 графа G_2 несмежна с вершиной 4 этого

графа, которая выбрана как соответствующая вершине 7 графа G_1. Поэтому этот вариант выбора не подходит. Перейдем к выбору следующего второго варианта {1–6, 2–3}. Но и в этом варианте выбора есть противоречие: вершина 1 графа G_1 смежна с вершиной 7 этого графа, а соответствующая ей вершина 6 графа G_2 несмежна с вершиной 4 этого графа, которая выбрана как соответствующая вершине 7 графа G_1. Так как вариантов выбора в данной точке рассмотрения больше нет, то следует отступить к предыдущей точке рассмотрения и вместо варианта {7–4, 8–7} выбрать следующий вариант {7–7, 8–4}. При этом φ={5–1, 6–8, 3–5, 4–2, 7–7, 8–4}.

Теперь снова нужно рассмотреть 2 варианта соответствия оставшихся вершин графов: {1–3, 2–6} и {1–6, 2–3}. При выборе первого варианта опять возникает противоречие: вершина 1 графа G_1 смежна с вершиной 7 этого графа, а соответствующая ей вершина 3 графа G_2 несмежна с вершиной 7 этого графа, которая выбрана как соответствующая вершине 7 графа G_1. Поэтому перейдем к следующему варианту выбора соответствия вершин: {1–6, 2–3}.

На этот раз противоречия не возникает: вершина 1 графа G_1 смежна с вершиной 7 этого графа, а соответствующая ей вершина 6 графа G_2 также смежна с вершиной 7 этого графа, которая выбрана как соответствующая вершине 7 графа G_1. Аналогично вершина 2 графа G_1 смежна с вершинами 1, 3, 6 этого графа, а соответствующая ей вершина 3 графа G_2 также смежна с вершинами 5, 6, 8 этого графа, которые выбраны как соответствующие вершинам 3, 1, 6 графа G_1.

Таким образом, мы получаем окончательный вариант соответствия: φ={5–1, 6–8, 3–5, 4–2, 7–7, 8–4, 1–6, 2–3}. Взаимнооднозначное соответствие $\varphi = y(x)$, где x – вершина графа G_1, а y – соответствующая ей вершина графа G_2, определяется следующей таблицей:

x	1	2	3	4	5	6	7	8
y	6	3	5	2	1	8	7	4

Проверка, что φ является изоморфизмом, дается следующей таблицей соответствия ребер графов G_1 и G_2:

x_i, x_j	$1,2$	$1,6$	$1,7$	$2,3$	$2,6$	$3,4$	$3,5$	$4,6$	$4,8$	$5,6$	$5,7$	$5,8$	$7,8$
y_k, y_l	$6,3$	$6,8$	$6,7$	$3,5$	$3,8$	$5,2$	$5,1$	$2,8$	$2,4$	$1,8$	$1,7$	$1,4$	$7,4$

Отметим, что мы не использовали при построении изоморфизма второй вариант соответствия в первой точке рассмотрения. Можно проверить, что в этом случае получается другой вариант изоморфизма графов.

Пример 3. Граф $G_1(X, U)$ возьмем из примера 1, а граф $G_2(Y, V)$ зададим списком ребер:
({1,3}, {1,4}, {1,6}, {2,3}, {2,5}, {2,7}, {3,8}, {4,6}, {4,7}, {5.7}, {5,8}, {6,8}).

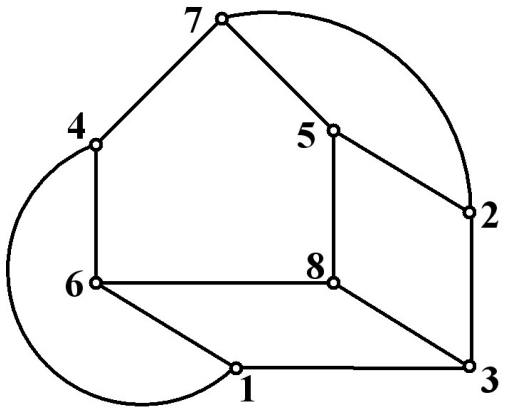

Рис. 12 (граф G_2)

Оба графа имеют по 8 вершин, и все вершины имеют степень 3. Поэтому число ребер у них одинаковое, и алгоритм сопоставления по степеням будет иметь полный перебор сопоставления вершин. Попробуем сопоставить элементарные циклы минимальной длины. В графе G_1 есть 2 таких цикла (1, 2, 6, 1) и (5, 7, 8, 5); а в графе G_2 также 2 цикла длины 3: (1, 4, 6, 1) и (2, 5, 7, 2). Только вершины 3 и 4 графа G_1 не входят в циклы длины 3 и при этом они смежны между собой. Соответствующие вершины графа G_2 3 и 8 также не входят в циклы длины 3 и смежны между собой. Поэтому есть 2 возможности сопоставления: (3-3, 4-8) и (3-8, 4-3).

Сначала попробуем первую возможность (3-3, 4-8). Вершины 3 и 4 графа G_1 смежны соответственно с вершинами 2 и 6 первого цикла, которые смежны между собой и смежны с вершиной 1 этого цикла. Для графа G_2 вершины 3 и 8 смежны соответственно с вершинами 1 и 6 первого цикла, которые смежны между собой и смежны с вершиной 4 этого цикла. Поэтому продолжим построение соответствия (3-3, 4-8, 2-1, 6-6, 1-4). Далее, вершины 3 и 4 графа G_1 смежны соответственно с

вершинами 5 и 8 второго цикла, которые также смежны между собой и смежны с вершиной 7 этого цикла. А для графа G_2 вершины 3 и 8 смежны соответственно с вершинами 2 и 5 второго цикла, которые смежны между собой и смежны с вершиной 7 этого цикла. Поэтому продолжим построение соответствия вершин (3-3, 4-8, 2-1, 6-6, 1-4, 5-2, 8-5, 7-7). В результате мы получаем следующее 1-1с вершин

x	1	2	3	4	5	6	7	8
y	4	1	3	8	2	6	7	5

,

которое сохраняет 1-1с между ребрами

$\{x_i, x_j\}$	$1,2$	$1,6$	$1,7$	$2,3$	$2,6$	$3,4$	$3,5$	$4,6$	$4,8$	$5,7$	$5,8$	$7,8$
$\{y_k, y_l\}$	$4,1$	$4,6$	$4,7$	$1,3$	$1,6$	$3,8$	$3,2$	$8,6$	$8,5$	$2,7$	$2,5$	$7,5$

и, следовательно, является изоморфизмом. Вторая неиспользованная возможность также должна привести к построению другого изоморфизма.

Продемонстрируем на этом примере алгоритм сопоставления вершин по набору длин циклов, в которые входят сопоставляемые вершины. Граф G_1 содержит элементарные циклы

2 цикла длины 3: (1, 2, 6, 1) и (5, 7, 8, 5);

2 цикла длины 4: (2, 3, 4, 6, 2) и (3, 4, 8, 5, 3);

4 цикла длины 5: (1, 2, 3, 5, 7, 1), (1, 2, 3, 4, 6, 1), (3, 4, 8, 7, 5, 3) и (1, 6, 4, 8, 7, 1);

а также циклы длины большей 5.

Граф G_2 содержит элементарные циклы

2 цикла длины 3: (1, 4, 6, 1) и (2, 5, 7, 2);

2 цикла длины 4: (1, 3, 8, 6, 1) и (2, 3, 8, 5, 2);

4 цикла длины 5: (4, 6, 8, 5, 7, 4), (1, 3, 8, 6, 4, 1), (2, 3, 8, 5, 7, 2) и (1, 3, 2, 7, 4, 1);

а также циклы длины, большей 5.

Так как число элементарных циклов с одной длиной, похоже, одинаково, попробуем установить изоморфизм графов. Для этого определим для каждой вершины длину элементарных циклов (до 5), в которые входит эта вершина.

Для графа G_1

1 – 3, 5, 5, 5 (вершина 1 входит в 1 цикл длины 3 и 3 цикла длины 5);

2 – 3, 4, 5, 5;

3 – 4, 4, 5, 5, 5;

4 – 4, 4, 5, 5, 5;

5 – 3, 4, 5, 5;

6 – 3, 4, 5, 5, 5;

7 – 3, 5, 5, 5;

8 – 3, 4, 5, 5.

Для графа G_2

1 – 3, 4, 5, 5, 5;

2 – 3, 4, 5, 5;

3 – 4, 4, 5, 5, 5;

4 – 3, 5, 5, 5;

5 – 3, 4, 5, 5;

6 – 3, 4, 5, 5;

7 – 3, 5, 5, 5;

8 – 4, 4, 5, 5, 5.

Набор длин элементарных циклов (3, 4, 5, 5, 5) имеет только вершина 6 графа G_1 и вершина 1 графа G_2. Поэтому устанавливаем единственно возможное соответствие этих вершин графа: φ=$\{$(6–1)$\}$.

Смежными для вершины 6 в G_1 являются вершины (с набором циклов, в которые входят): 1 (3, 5, 5, 5), 2 (3, 4, 5, 5) и 4 (4, 4, 5, 5, 5). Смежными для соответствующей вершины 1 в G_2 – вершины: 3 (4, 4, 5, 5, 5), 4 (3, 5, 5, 5) и 6 (3, 4, 5, 5). Одинаковые наборы циклов имеют пары вершин: 1 из G_1 и 4 из G_2, 2 из G_1 и 6 из G_2, а также 4 из G_1 и 3 из G_2,. Добавляем эти пары в строящееся соответствие: φ=$\{$(6–1), (1–4), (2–6), (4–3)$\}$.

Смежными для вошедших в соответствие вершин 2 и 4 из G_1 является вершина 3, еще не вошедшая в соответствие, а смежными для соответствующим им вершинам 6 и 3 из G_2 является вершина 8. Поэтому добавляем пару (3–8) в строящееся соответствие: φ=$\{$(6–1), (1–4), (2–6), (4–3), (3–8)$\}$.

Далее, вершина 1 из G_1, уже вошедшая в соответствие, среди вершин, не вошедших в соответствие, смежна только с вершиной 7, а соответствующая ей вершина 4 из G_2 среди вершин, не вошедших в соот-

ветствие, смежна только с вершиной 7. Поэтому добавляем пару (7–7): φ={(6–1), (1–4), (2–6), (4–3), (3–8), (7–7)}.

Затем смежной вершине 3 из G_1, уже вошедшей в соответствие, среди вершин, не вошедших в соответствие, является только вершина 5, а для соответствующей ей вершины 8 из G_2 среди вершин, не вошедших в соответствие, смежной является только вершина 5. Поэтому добавляем пару (5–5): φ={(6–1), (1–4), (2–6), (4–3), (3–8), (7–7), (5–5)}.

Остается еще не вошедшая в соответствие вершина 8 из G_1, смежная с вершинами 4, 5 и 7, а в G_2 вершина 2, которая смежна с соответствующими вершинами 3, 5 и 7. Добавляем эту пару (8–2) в соответствие: φ={(6–1), (1–4), (2–6), (4–3), (3–8), (7–7), (5–5), (8–2)}.

Взаимнооднозначное соответствие $\varphi = y(x)$, где x – вершина графа G_1, а y – соответствующая ей вершина графа G_2, определяется следующей таблицей:

x	1	2	3	4	5	6	7	8
y	4	6	8	3	5	1	7	2

.

Проверка, что φ является изоморфизмом, дается следующей таблицей соответствия ребер графов G_1 и G_2:

$\{x_i, x_j\}$	$1,2$	$1,6$	$1,7$	$2,3$	$2,6$	$3,4$	$3,5$	$4,6$	$4,8$	$5,7$	$5,8$	$7,8$
$\{y_k, y_l\}$	$4,6$	$4,1$	$4,7$	$6,8$	$6,1$	$8,3$	$8,5$	$3,1$	$3,2$	$5,7$	$5,2$	$7,2$

.

6. Планарность графа

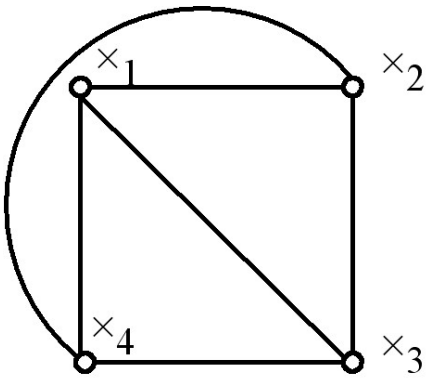

Рис. 13

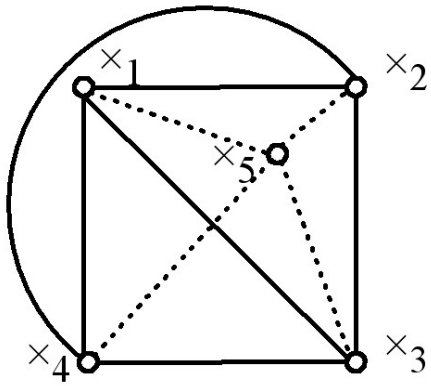

Рис. 14

Граф называется планарным (плоским), если он допускает реализацию на плоскости без пересечения ребер. Например, *полные* (содержащие все возможные ребра) графы K_3 из 3 вершин и K_4 из 4 вершин являются плоскими (см. рис. 13), а K_5 плоским не является. Последнее обосновывается тем, что его клика 4 вершин x_1, x_2, x_3, x_4 (полный подграф, содержащий эти вершины) при реализации на плоскости без пересечения ребер разбивает плоскость на 4 области (см. рис. 14):

1) ограниченную ребрами цикла (x_1, x_2, x_3, x_1) (вне ее лежит вершина x_4);

2) ограниченную ребрами цикла (x_1, x_3, x_4, x_1) (вне ее лежит вершина x_2);

3) ограниченную ребрами цикла (x_1, x_2, x_4, x_1) (вне ее лежит вершина x_3);

4) лежащую вне цикла (x_2, x_3, x_4, x_2) (внутри цикла лежит вершина x_1).

Теперь, в какую бы из областей мы ни поместили вершину x_5, какая-то из первых 4 вершин x_i ($i \in \overline{1,4}$) будет лежать вне этой области, а потому ребро $\{x_5, x_i\}$ пересечет какое-либо из ребер, лежащих на границе области. Таким образом граф K_5 не является планарным.

Другим важным примером непланарного подграфа является полный двудольный граф $K_{3,3}$ – каждая доля графа имеет по 3 вершины и между любыми вершинами разных долей есть ребро (см. рис. 15). Обоснование этого факта проводится подобным образом – его подграф, содержащий все 3 вершины x_1, x_2, x_3 первой доли и 2 вершины x_4, x_5 второй доли, а также все ребра, соединяющие любые вершины разных долей, разбивает плоскость на 3 области (см. рис. 16):

1) ограниченную ребрами цикла $(x_1, x_5, x_2, x_4, x_1)$ (вне ее лежит вершина x_3);

2) ограниченную ребрами цикла $(x_2, x_5, x_3, x_4, x_2)$ (вне ее лежит вершина x_1);

3) лежащую вне цикла $(x_1, x_5, x_3, x_4, x_1)$ (внутри цикла лежит вершина x_2).

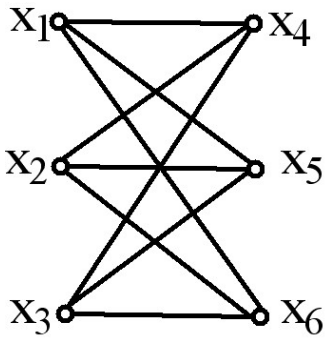

Рис. 15

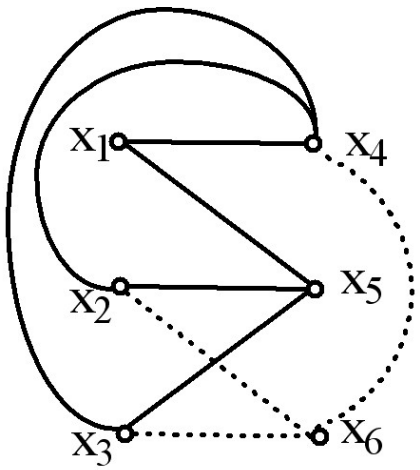

Рис. 16

Теперь, в какую бы из областей мы ни поместили вершину x_6, какая-то из первых 3 вершин x_i ($i \in \overline{1,3}$) будет лежать вне этой области, а потому ребро $\{x_6, x_i\}$ пересечет какое-либо из ребер, лежащих на границе области. Таким образом граф $K_{3,3}$ не является планарным.

Если граф содержит в качестве своей части непланарный подграф, то он также является непланарным.

Введем операцию *подразбиения ребра* $\{x_i, x_j\} \in U$ произвольного графа $G\{X, U\}$: к $n = |X|$ вершинам графа добавляется $(n+1)$-я вершина x_{n+1}, и это ребро заменяется на 2 ребра: $\{x_i, x_{n+1}\}$ и $\{x_{n+1}, x_j\}$. Совершенно ясно, что операция подразбиения ребра не меняет свойство планарности (непланарности) графа.

Два графа называются *гомеоморфными*, если при помощи операций подразбиения ребра их можно сделать изоморфными. Поэтому если граф содержит в качестве своей части подграф, гомеоморфный K_5 или $K_{3,3}$, то он не является планарным. Оказывается, это условие является не только необходимым, но и достаточным, что устанавливает теорема Понтрягина–Куратовского.

Теорема Понтрягина–Куратовского. *Для того чтобы граф был планарен, необходимо и достаточно, чтобы он в качестве своих частей не содержал бы подграфов, гомеоморфных K_5 и $K_{3,3}$.*

На этой теореме основан алгоритм проверки планарности графа. Ввиду его сложности мы приведем методику проверки планарности для небольших графов.

1. Попытаться реализовать граф на плоскости без пересечения ребер. Если это сразу не удается, то после некоторого числа попыток перенесения расположения вершин с целью уменьшить число пересечения ребер это, возможно, удастся (если граф планарный).

2. Найти число вершин, степень которых не меньше 4. Если таких вершин меньше 5, то подграфа, гомеоморфного K_5, нет.

3. Если число вершин, степень которых не меньше 4, больше 4, то попытаться найти подграф, гомеоморфный K_5:

   (a) Если среди вершин со степенью не меньше 4 есть 5 вершин, попарно связных между собой, то подграф, изоморфный K_5, найден и граф непланарен.

   (b) Если этого нет, то следует удалить висячие вершины (степени 1), а также вершины степени 2 операцией, обратной операции подразбиения ребра (замена двух ребер, инцидентных такой вершине, на одно ребро, связывающее смежные с ней вершины).

   (c) Если после этого шага нет 5 вершин степени 4, попарно связных между собой, то следует рассмотреть вершины степени 3. Удалением одного из ребер каждой такой вершины (здесь возможен перебор удаляемых ребер, при котором порядок перебора зависит от графа) следует добиться получения подграфа, гомеоморфного K_5, что устанавливает непланарность графа.

   (d) Если выполнение предыдущего пункта не дало результата, то следует перейти к следующему пункту – поиску подграфа, гомеоморфного $K_{3,3}$.

4. Найти число вершин со степенью не меньше 3. Если таких вершин меньше 6, то подграфа, гомеоморфного $K_{3,3}$, нет. Если подграф, гомеоморфный K_5, также не найден, то необходимо снова попытаться реализовать граф на плоскости без пересечения ребер.

5. Если число вершин, степень которых не меньше 3, больше 5, то следует попытаться найти подграф, гомеоморфный $K_{3,3}$:

(a) Если среди вершин со степенью не меньше 3 есть 2 набора по 3 вершины таких, что каждая вершина из первого набора смежна с 3 вершинам из второго набора, то подграф, изоморфный $K_{3,3}$, найден и граф непланарен.

(b) Если этого нет, то следует удалить висячие вершины (степени 1), а также вершины степени 2 операцией, обратной операции подразбиения ребра (замена двух ребер, инцидентных такой вершине, на одно ребро, связывающее смежные с ней вершины).

(c) Если после этого подграф, изоморфный $K_{3,3}$, не находится, то следует по очереди удалять (здесь возможен перебор удаляемых ребер, при котором порядок перебора зависит от графа) только одно из ребер вершин степени 3 до тех пор, пока подграф, изоморфный $K_{3,3}$, не будет найден.

(d) Если последние действия не дали результата, то следует снова попытаться реализовать граф на плоскости без пересечения ребер.

7. Примеры задач на планарность графа с их решениями

В примерах мы будем вершины отмечать только их номером (без буквы), а ребра парой номеров в фигурных скобках.

Пример 4.

$$\begin{bmatrix} 010 & 001 & 10 \\ 101 & 001 & 00 \\ 010 & 110 & 00 \\ 001 & 001 & 01 \\ 001 & 000 & 11 \\ 110 & 100 & 00 \\ 100 & 010 & 01 \\ 000 & 110 & 10 \end{bmatrix}$$

Анализ матрицы смежности вершин устанавливает, что все вершины имеют степень 3. Попытка реализации графа на плоскости без пересечения ребер (см. рис. 17) удачна – граф планарен.

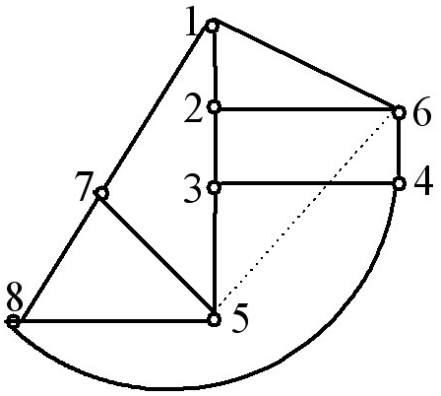

Рис. 17

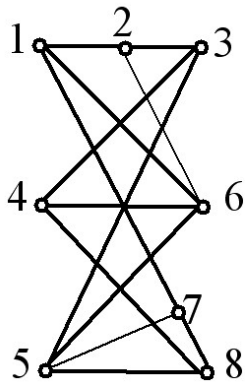

Рис. 18

Пример 5.

$$\begin{bmatrix} 010 & 001 & 10 \\ 101 & 001 & 00 \\ 010 & 110 & 00 \\ 001 & 001 & 01 \\ 001 & 001 & 11 \\ 110 & 110 & 00 \\ 100 & 010 & 01 \\ 000 & 110 & 10 \end{bmatrix}$$

Этот граф отличается от предыдущего добавлением ребра {5, 6} (на рис. 17 обозначено пунктиром). Однако теперь получается пересечение ребер {3, 4} и {5, 6}. От пересечения ребер не удается избавиться перемещением вершин 6 и 4 в область цикла (1, 2, 3, 7, 5, 1), так как тогда ребро {8, 4} начинает пересекаться с ребром {1, 7} или {5, 7}. Попытаемся установить непланарность графа.

Имеется 2 вершины степени 4 (5 и 6) и 6 вершин степени 3. Так как после добавления ребра $(5, 6)$ не удается реализовать граф на плоскости, то следует искать подграф, гомеоморфный $K_{3,3}$, и добавленное ребро должно быть между вершинами разных долей.

Вершина 5 смежна с вершинами 3, 6, 7, 8, причем из этих 4 вершин только вершины 7 и 8 смежны между собой. Поэтому, по-видимому,

2 из вершин 3, 7, 8 принадлежат одной доле вместе с вершиной 6, и так как 7 и 8 смежны между собой, то это либо 3 и 7, либо 3 и 8.

Исследуем теперь связь остальных вершин 1, 2, 4, 5 с вершинами 3, 6, 7, 8. Вершина 4 смежна с вершинами 3, 6, 8. Вершина 1 смежна с вершинами 2, 6, 7. Вершина 2 смежна с вершинами 1, 3, 6. Вершина 5 смежна с вершинами 3, 6, 7, 8.

Анализ этих связей показывает, что вершинам 3, 6, 8 смежны вершины 4 и 5. Вершина 1, смежная с вершиной 6, связана с вершиной 8 через вершину 7 и с вершиной 3 через вершину 2. Поэтому если удалить ребра (5, 7) и (2, 6), то мы получим подграф, гомеоморфный $K_{3,3}$ (см. рис. 18), что и доказывает непланарность графа.

8. Индивидуальное задание 8

8.1. Общее задание

1. Определить, является ли граф, заданный матрицей смежности вершин, *планарным*. В случае планарности построить реализацию графа на плоскости без пересечения ребер. В случае непланарности найти часть графа (построив реализацию с удаляемыми при этом вершинами или ребрами, обозначив это пунктиром), которая является гомеоморфной K_5 или $K_{3,3}$.

2. Определить, являются ли *изомофными* 2 графа, один из которых взят из задачи 1, а второй задан списком ребер. В случае изоморфности графов построить *изоморфизм* – взаимнооднозначное соответствие вершин первого и второго графов, сохраняющее смежность вершин. В случае неизоморфности графов обосновать это.

8.2. Методические рекомендации для задания 8

Задача 1

1. Начинать исследование графа нужно с попытки изобразить граф на плоскости без пересечения ребер. Рекомендуется в каком-либо

порядке добавлять сначала вершины и связывающие их ребра в изображение графа.

Например, в качестве начальной вершины выбрать одну из вершин с наибольшей степенью; затем каждый раз добавлять вершину, смежную с уже выбранными и имеющую наибольшую степень среди добавляемых. Если возникнет пересечение ребер, то поменять порядок ранее добавленных вершин, чтобы пересечение исчезло.

Альтернативной к этой идее построения графа без пересечния ребер может служить способ, когда каждый раз выбирается вершина, смежная с уже выбранными, с наименьшей степенью из добавляемых.

Если пересечение не удается ликвидировать, то скорее всего граф непланарный и следует найти обоснование непланарности в следующих пунктах рекомендаций.

2. Обоснование непланарности графа следует начать с попытки найти подграф, гомеоморфный K_5. В этом случае следует рассмотреть все вершины степени не ниже 4. Если их меньше 5, то перейти к следующему пункту рекомендаций.

   Если вершин не меньше 5, рассмотреть по очереди все подграфы из 5 вершин и для каждого такого подграфа проверить, смежны ли все его вершины (в случае смежности любой пары Вы нашли подграф, изоморфный K_5).

   Иначе для каждой пары несмежных вершин установить, есть ли путь, их связывающий через другие вершины. Если такой путь есть, то на этом пути для всех промежуточных вершин удалить лишние ребра. Если это возможно сделать для каждой пары из пятерки вершин (они смежны либо связаны через другие вершины), то Вы нашли подграф, гомеоморфный K_5. **Типичной ошибкой** является указание **общей вершины на пути связи 2 различных пар пятерки**.

   Если ни для одной пятерки вершин это не получается, то перейти к следующему пункту рекомендаций.

3. Попытаться обосновать непланарность графа попыткой найти подграф, гомеоморфный $K_{3,3}$. В этом случае следует рассмотреть все вершины степени не ниже 3. Если их меньше 6, то такого подграфа нет и следует снова перейти к 1-му пункту рекомендаций.

   Если таких вершин не меньше 6, то следует для каждой вершины определить список остальных, смежных с данной. Если для какой-либо тройки этих вершин есть 3 общие вершины в этих списках, то эти общие вершины и тройка вершин, имеющие их смежными, образуют подграф, изоморфный $K_{3,3}$.

   Если для тройки вершин в списках их смежных вершин только 2 общие вершины, то следует посмотреть, не является ли другая необщая вершина, смежная с одной из тройки, также смежной или имеющей связи через еще какие-то вершины с другими 2 вершинами этой тройки. В этом случае, удалив лишние ребра, можно найти подграф, гомеоморфный $K_{3,3}$. **Типичной ошибкой** является указание **общей вершины на пути 2 различных таких связей**.

   Если ни для одной шестерки вершин это не получается, то перейти снова к 1-му пункту рекомендаций: возможно, граф планарен и следует поискать его реализацию без пересечения ребер.

4. Если граф планарен, то указать его реализацию без пересечения ребер. В противном случае следует реализовать граф с выделенным подграфом K_5 или $K_{3,3}$ и всеми **удаленными ребрами**, нарисованными **пунктирными линиями**.

Задача 2

1. Определить основные характеристики каждого из графов: число вершин, число ребер, наборы степеней вершин. Если какая-то из этих характеристик не совпадает у обоих графов, то графы неизоморфны.

2. Если степени вершин различны, то найти ту степень, которую имеет наименьшее число вершин и сопоставить эти вершины друг с другом в разных графах.

3. Если сопоставление единственное, то выбрать его и перейти к сопоставлению их смежных вершин по степеням и смежности между собой.

4. Если сопоставление не единственное, то выбрать одно из них (а другие запомнить в этой *точке ветвления*) и продолжать процесс сопоставления, определяя предварительно характеристики степени и смежности с только что добавленными вершинами и выбирая для сопоставления вершины с одинаковыми характеристиками степени и смежности среди только что добавленных вершин. При этом могут получиться новые точки ветвления.

5. Если процесс сопоставления каких-либо двух вершин различных графов закончился неудачно (этот вариант сопоставления невозможен), то вернуться к предыдущей точке ветвления, вычекнуть неудачный вариант и выбрать следующий вариант сопоставления. Если в точке ветвления вычеркнуты уже все варианты, то точку ветвления надо вычеркнуть и возвратиться к предыдущей точке ветвления.

6. Если все варианты сопоставления закончились неудачно и точек ветвления уже нет, то графы неизомрфны.

7. Если же удалось сопоставить все вершины, то значит получено 1-1с вершин, дающее один из возможных изоморфизмов графов. Следует оформить таблицу сопоставления вершин графов и затем по ней получить таблицу сопоставления ребер графов.

8. Если в пункте 2 этих рекомендаций все степени вершин одинаковы, то для каждой вершины следует рассмотреть ее смежные вершины и определить их взаимную смежность. Тогда нужно для сопоставления рассматривать лишь те варианты, где характеристики взаимной смежности одинаковы. Если таких вариантов несколько, то надо организовать точку ветвления, выбрать один из вариантов и далее действовать, как в пункте 4 этих рекомендаций.

9. Описанный прием исследования изоморфизма графов путем сопоставления вершин по количеству элементарных циклов, в которые они входят, *требует очень большой аккуратности.* **Типичной**

ошибкой его применения является **пропуск некоторых циклов**. Поэтому при недостаточной внимательности студента этот алгоритм не рекомендуется.

8.3. Варианты индивидуального задания 8

Задача 1

1.
$$\begin{bmatrix} 001 & 111 & 00 \\ 001 & 001 & 01 \\ 110 & 000 & 10 \\ 100 & 000 & 11 \\ 100 & 000 & 11 \\ 110 & 000 & 10 \\ 001 & 111 & 00 \\ 010 & 110 & 00 \end{bmatrix}$$

2.
$$\begin{bmatrix} 000 & 011 & 01 \\ 001 & 111 & 00 \\ 010 & 001 & 10 \\ 010 & 000 & 11 \\ 110 & 000 & 11 \\ 111 & 000 & 01 \\ 001 & 110 & 00 \\ 100 & 111 & 00 \end{bmatrix}$$

3.
$$\begin{bmatrix} 001 & 100 & 100 \\ 000 & 110 & 011 \\ 100 & 001 & 011 \\ 110 & 001 & 100 \\ 010 & 001 & 010 \\ 001 & 110 & 001 \\ 100 & 100 & 001 \\ 011 & 010 & 000 \\ 011 & 001 & 100 \end{bmatrix}$$

4.

$$\begin{bmatrix} 000 & 101 & 101 \\ 001 & 000 & 011 \\ 010 & 110 & 010 \\ 101 & 000 & 110 \\ 001 & 001 & 110 \\ 100 & 010 & 011 \\ 100 & 110 & 000 \\ 011 & 111 & 001 \\ 110 & 001 & 010 \end{bmatrix}$$

5.

$$\begin{bmatrix} 000 & 111 & 100 \\ 000 & 000 & 111 \\ 000 & 110 & 000 \\ 101 & 001 & 001 \\ 101 & 000 & 010 \\ 100 & 100 & 011 \\ 110 & 000 & 001 \\ 010 & 011 & 000 \\ 010 & 101 & 100 \end{bmatrix}$$

6.

$$\begin{bmatrix} 010 & 100 & 1100 \\ 100 & 110 & 0000 \\ 000 & 011 & 0011 \\ 110 & 000 & 0100 \\ 011 & 000 & 0100 \\ 001 & 000 & 0011 \\ 100 & 000 & 0011 \\ 100 & 110 & 0000 \\ 001 & 001 & 1000 \\ 001 & 001 & 1000 \end{bmatrix}$$

7.

$$\begin{bmatrix} 011 & 001 & 010 \\ 101 & 100 & 001 \\ 110 & 011 & 101 \\ 010 & 010 & 011 \\ 001 & 101 & 010 \\ 101 & 010 & 100 \\ 001 & 001 & 001 \\ 100 & 110 & 000 \\ 011 & 100 & 100 \end{bmatrix}$$

8.

$$\begin{bmatrix} 010 & 011 & 000 \\ 100 & 000 & 111 \\ 000 & 101 & 001 \\ 001 & 000 & 111 \\ 100 & 000 & 100 \\ 101 & 000 & 010 \\ 010 & 110 & 010 \\ 010 & 101 & 100 \\ 011 & 100 & 000 \end{bmatrix}$$

9.

$$\begin{bmatrix} 000 & 001 & 1100 \\ 000 & 101 & 1100 \\ 000 & 101 & 0011 \\ 011 & 000 & 0011 \\ 000 & 000 & 0111 \\ 111 & 000 & 0000 \\ 110 & 000 & 0100 \\ 110 & 010 & 1000 \\ 001 & 110 & 0000 \\ 001 & 110 & 0000 \end{bmatrix}$$

10.

$$\begin{bmatrix} 010 & 011 & 01 \\ 101 & 001 & 11 \\ 010 & 100 & 11 \\ 001 & 011 & 10 \\ 100 & 101 & 11 \\ 110 & 110 & 00 \\ 011 & 110 & 00 \\ 111 & 010 & 00 \end{bmatrix}$$

11.

$$\begin{bmatrix} 011 & 110 & 00 \\ 101 & 000 & 11 \\ 110 & 101 & 00 \\ 101 & 010 & 10 \\ 100 & 101 & 01 \\ 001 & 010 & 11 \\ 010 & 101 & 01 \\ 010 & 011 & 10 \end{bmatrix}$$

12.

$$\begin{bmatrix} 000 & 011 & 10 \\ 000 & 001 & 11 \\ 000 & 010 & 11 \\ 000 & 001 & 11 \\ 101 & 001 & 00 \\ 110 & 110 & 00 \\ 111 & 100 & 00 \\ 011 & 100 & 00 \end{bmatrix}$$

13.

$$\begin{bmatrix} 010 & 001 & 11 \\ 100 & 101 & 01 \\ 000 & 011 & 10 \\ 010 & 010 & 10 \\ 001 & 100 & 01 \\ 111 & 000 & 01 \\ 101 & 100 & 00 \\ 110 & 011 & 00 \end{bmatrix}$$

14.

$$\begin{bmatrix} 001 & 110 & 01 \\ 000 & 110 & 11 \\ 100 & 001 & 11 \\ 110 & 010 & 10 \\ 110 & 101 & 00 \\ 001 & 010 & 11 \\ 011 & 101 & 00 \\ 111 & 001 & 00 \end{bmatrix}$$

15.

$$\begin{bmatrix} 000 & 111 & 10 \\ 001 & 001 & 11 \\ 010 & 110 & 11 \\ 101 & 001 & 01 \\ 101 & 001 & 10 \\ 110 & 110 & 01 \\ 111 & 010 & 00 \\ 011 & 101 & 00 \end{bmatrix}$$

16.

$$\begin{bmatrix} 000 & 001 & 11 \\ 000 & 001 & 11 \\ 000 & 011 & 01 \\ 000 & 011 & 01 \\ 001 & 100 & 10 \\ 111 & 100 & 00 \\ 110 & 010 & 00 \\ 111 & 100 & 00 \end{bmatrix}$$

17.

$$\begin{bmatrix} 000 & 111 & 01 \\ 000 & 100 & 11 \\ 000 & 101 & 01 \\ 111 & 000 & 10 \\ 100 & 001 & 10 \\ 101 & 010 & 10 \\ 010 & 111 & 00 \\ 111 & 000 & 00 \end{bmatrix}$$

18.

$$\begin{bmatrix} 011 & 011 & 000 \\ 100 & 101 & 010 \\ 100 & 000 & 110 \\ 010 & 010 & 001 \\ 100 & 100 & 011 \\ 110 & 000 & 101 \\ 001 & 001 & 010 \\ 011 & 010 & 100 \\ 000 & 111 & 000 \end{bmatrix}$$

19.

$$\begin{bmatrix} 010 & 000 & 111 \\ 101 & 000 & 001 \\ 010 & 110 & 001 \\ 001 & 010 & 101 \\ 001 & 101 & 010 \\ 000 & 010 & 110 \\ 100 & 101 & 001 \\ 100 & 011 & 001 \\ 111 & 100 & 110 \end{bmatrix}$$

20.

$$\begin{bmatrix} 011 & 101 & 000 \\ 100 & 010 & 001 \\ 100 & 101 & 100 \\ 101 & 000 & 011 \\ 010 & 001 & 100 \\ 101 & 010 & 010 \\ 001 & 010 & 000 \\ 000 & 101 & 001 \\ 010 & 100 & 010 \end{bmatrix}$$

21.

$$\begin{bmatrix} 000 & 100 & 1011 \\ 001 & 011 & 0100 \\ 010 & 001 & 0001 \\ 100 & 010 & 0010 \\ 010 & 100 & 1000 \\ 011 & 000 & 1100 \\ 100 & 011 & 0010 \\ 010 & 001 & 0001 \\ 100 & 100 & 1000 \\ 101 & 000 & 0100 \end{bmatrix}$$

22.

$$\begin{bmatrix} 001 & 111 & 01 \\ 000 & 110 & 11 \\ 100 & 011 & 10 \\ 110 & 001 & 10 \\ 111 & 000 & 01 \\ 101 & 100 & 10 \\ 011 & 101 & 01 \\ 110 & 010 & 10 \end{bmatrix}$$

23.

$$\begin{bmatrix} 001 & 011 & 10 \\ 001 & 110 & 01 \\ 110 & 001 & 01 \\ 010 & 010 & 11 \\ 110 & 100 & 10 \\ 101 & 000 & 11 \\ 100 & 111 & 00 \\ 011 & 101 & 00 \end{bmatrix}$$

24.

$$\begin{bmatrix} 011 & 001 & 10 \\ 100 & 010 & 01 \\ 100 & 010 & 01 \\ 000 & 001 & 11 \\ 011 & 000 & 10 \\ 100 & 100 & 01 \\ 100 & 110 & 00 \\ 011 & 101 & 00 \end{bmatrix}$$

25.

$$\begin{bmatrix} 001 & 100 & 01 \\ 001 & 101 & 10 \\ 110 & 010 & 00 \\ 110 & 001 & 10 \\ 001 & 000 & 11 \\ 010 & 100 & 11 \\ 010 & 111 & 00 \\ 100 & 011 & 00 \end{bmatrix}$$

26.

$$\begin{bmatrix} 000 & 111 & 10 \\ 001 & 011 & 01 \\ 010 & 111 & 00 \\ 101 & 000 & 11 \\ 111 & 000 & 10 \\ 111 & 000 & 01 \\ 100 & 110 & 01 \\ 010 & 101 & 10 \end{bmatrix}$$

27.

$$\begin{bmatrix} 001 & 101 & 11 \\ 000 & 110 & 11 \\ 100 & 011 & 01 \\ 110 & 010 & 01 \\ 011 & 101 & 10 \\ 101 & 010 & 10 \\ 110 & 011 & 00 \\ 111 & 100 & 00 \end{bmatrix}$$

28.

$$\begin{bmatrix} 001 & 100 & 01 \\ 000 & 100 & 11 \\ 100 & 010 & 10 \\ 110 & 011 & 00 \\ 001 & 100 & 01 \\ 000 & 100 & 11 \\ 011 & 001 & 00 \\ 110 & 011 & 00 \end{bmatrix}$$

29.

$$\begin{bmatrix} 010 & 111 & 00 \\ 101 & 001 & 10 \\ 010 & 000 & 11 \\ 100 & 001 & 11 \\ 100 & 000 & 11 \\ 110 & 100 & 00 \\ 011 & 110 & 00 \\ 001 & 110 & 00 \end{bmatrix}$$

30.

$$\begin{bmatrix} 000 & 011 & 100 \\ 001 & 101 & 100 \\ 010 & 000 & 011 \\ 010 & 010 & 110 \\ 100 & 101 & 000 \\ 110 & 010 & 001 \\ 110 & 100 & 001 \\ 001 & 100 & 001 \\ 001 & 001 & 110 \end{bmatrix}$$

31.

$$\begin{bmatrix} 001 & 001 & 110 \\ 000 & 101 & 100 \\ 100 & 110 & 100 \\ 011 & 000 & 101 \\ 001 & 000 & 011 \\ 110 & 000 & 110 \\ 111 & 101 & 001 \\ 100 & 011 & 001 \\ 000 & 110 & 110 \end{bmatrix}$$

32.

$$\begin{bmatrix} 010 & 000 & 111 \\ 100 & 010 & 011 \\ 000 & 011 & 010 \\ 000 & 010 & 101 \\ 011 & 100 & 000 \\ 001 & 000 & 011 \\ 100 & 100 & 000 \\ 111 & 001 & 000 \\ 110 & 101 & 000 \end{bmatrix}$$

33.

$$\begin{bmatrix} 001 & 010 & 0010 \\ 000 & 100 & 0111 \\ 100 & 011 & 0010 \\ 010 & 001 & 0001 \\ 101 & 000 & 1000 \\ 001 & 100 & 0100 \\ 000 & 010 & 0011 \\ 010 & 001 & 0001 \\ 111 & 000 & 1000 \\ 010 & 100 & 1100 \end{bmatrix}$$

34.

$$\begin{bmatrix} 001 & 111 & 00 \\ 000 & 011 & 11 \\ 100 & 110 & 10 \\ 101 & 001 & 01 \\ 111 & 000 & 11 \\ 110 & 100 & 11 \\ 011 & 011 & 00 \\ 010 & 111 & 00 \end{bmatrix}$$

35.

$$\begin{bmatrix} 000 & 111 & 10 \\ 001 & 101 & 01 \\ 010 & 010 & 11 \\ 110 & 011 & 00 \\ 101 & 100 & 10 \\ 110 & 100 & 01 \\ 101 & 010 & 01 \\ 011 & 001 & 10 \end{bmatrix}$$

36.

$$\begin{bmatrix} 001 & 011 & 10 \\ 000 & 110 & 01 \\ 100 & 110 & 00 \\ 011 & 001 & 10 \\ 111 & 000 & 00 \\ 100 & 100 & 01 \\ 100 & 100 & 01 \\ 010 & 001 & 10 \end{bmatrix}$$

37.

$$\begin{bmatrix} 001 & 001 & 11 \\ 000 & 110 & 10 \\ 100 & 011 & 10 \\ 010 & 001 & 01 \\ 011 & 000 & 01 \\ 101 & 100 & 10 \\ 111 & 001 & 00 \\ 100 & 110 & 00 \end{bmatrix}$$

38.

$$\begin{bmatrix} 001 & 110 & 01 \\ 000 & 111 & 10 \\ 100 & 001 & 11 \\ 110 & 000 & 11 \\ 110 & 001 & 10 \\ 011 & 010 & 01 \\ 011 & 110 & 00 \\ 101 & 101 & 00 \end{bmatrix}$$

39.

$$\begin{bmatrix} 001 & 100 & 11 \\ 000 & 111 & 01 \\ 100 & 001 & 11 \\ 110 & 011 & 10 \\ 010 & 100 & 11 \\ 011 & 100 & 01 \\ 101 & 110 & 00 \\ 111 & 011 & 00 \end{bmatrix}$$

40.

$$\begin{bmatrix} 001 & 000 & 11 \\ 000 & 101 & 10 \\ 100 & 111 & 00 \\ 011 & 000 & 01 \\ 001 & 000 & 11 \\ 011 & 000 & 01 \\ 110 & 010 & 00 \\ 100 & 111 & 00 \end{bmatrix}$$

41.

$$\begin{bmatrix} 010 & 010 & 11 \\ 101 & 100 & 01 \\ 010 & 011 & 00 \\ 010 & 011 & 10 \\ 101 & 100 & 10 \\ 001 & 100 & 01 \\ 100 & 110 & 00 \\ 110 & 001 & 00 \end{bmatrix}$$

42.

$$\begin{bmatrix} 001 & 011 & 010 \\ 000 & 001 & 011 \\ 100 & 000 & 111 \\ 000 & 011 & 110 \\ 100 & 100 & 100 \\ 110 & 100 & 001 \\ 001 & 110 & 000 \\ 111 & 100 & 000 \\ 011 & 001 & 000 \end{bmatrix}$$

43.

$$\begin{bmatrix} 001 & 100 & 100 \\ 000 & 100 & 111 \\ 100 & 001 & 101 \\ 110 & 010 & 100 \\ 000 & 101 & 110 \\ 001 & 010 & 011 \\ 111 & 110 & 001 \\ 010 & 011 & 000 \\ 011 & 001 & 100 \end{bmatrix}$$

44.

$$\begin{bmatrix} 001 & 100 & 010 \\ 001 & 001 & 101 \\ 110 & 001 & 010 \\ 100 & 011 & 000 \\ 000 & 100 & 101 \\ 011 & 100 & 001 \\ 010 & 010 & 000 \\ 101 & 000 & 001 \\ 010 & 011 & 010 \end{bmatrix}$$

45.

$$\begin{bmatrix} 010 & 001 & 0010 \\ 101 & 000 & 0101 \\ 010 & 010 & 0100 \\ 000 & 001 & 1011 \\ 001 & 000 & 1001 \\ 100 & 100 & 1000 \\ 000 & 111 & 0010 \\ 011 & 000 & 0001 \\ 100 & 100 & 1000 \\ 010 & 110 & 0100 \end{bmatrix}$$

46.

$$\begin{bmatrix} 001 & 111 & 10 \\ 000 & 111 & 11 \\ 100 & 010 & 11 \\ 110 & 001 & 10 \\ 111 & 000 & 01 \\ 110 & 100 & 01 \\ 111 & 100 & 00 \\ 011 & 011 & 00 \end{bmatrix}$$

47.

$$\begin{bmatrix} 001 & 011 & 01 \\ 000 & 110 & 11 \\ 100 & 011 & 10 \\ 010 & 001 & 11 \\ 111 & 000 & 10 \\ 101 & 100 & 01 \\ 011 & 110 & 00 \\ 110 & 101 & 00 \end{bmatrix}$$

48.

$$\begin{bmatrix} 001 & 000 & 11 \\ 000 & 101 & 10 \\ 100 & 111 & 00 \\ 011 & 000 & 10 \\ 001 & 000 & 11 \\ 011 & 000 & 01 \\ 110 & 110 & 00 \\ 100 & 011 & 00 \end{bmatrix}$$

49.

$$\begin{bmatrix} 001 & 001 & 01 \\ 000 & 110 & 11 \\ 100 & 010 & 10 \\ 010 & 011 & 01 \\ 011 & 100 & 01 \\ 100 & 100 & 10 \\ 011 & 001 & 00 \\ 110 & 110 & 00 \end{bmatrix}$$

50.

$$\begin{bmatrix} 001 & 010 & 11 \\ 000 & 101 & 11 \\ 100 & 011 & 10 \\ 010 & 011 & 10 \\ 101 & 100 & 01 \\ 011 & 100 & 01 \\ 111 & 100 & 00 \\ 110 & 011 & 00 \end{bmatrix}$$

51.

$$\begin{bmatrix} 001 & 111 & 00 \\ 000 & 011 & 11 \\ 100 & 011 & 10 \\ 100 & 011 & 01 \\ 111 & 100 & 10 \\ 111 & 100 & 01 \\ 011 & 010 & 01 \\ 010 & 101 & 10 \end{bmatrix}$$

52.

$$\begin{bmatrix} 000 & 011 & 10 \\ 000 & 001 & 11 \\ 000 & 010 & 11 \\ 000 & 001 & 11 \\ 101 & 001 & 00 \\ 110 & 110 & 00 \\ 111 & 100 & 00 \\ 011 & 100 & 00 \end{bmatrix}$$

53.

$$\begin{bmatrix} 010 & 001 & 11 \\ 100 & 101 & 01 \\ 000 & 011 & 10 \\ 010 & 010 & 10 \\ 001 & 100 & 01 \\ 111 & 000 & 01 \\ 101 & 100 & 00 \\ 110 & 011 & 00 \end{bmatrix}$$

54.

$$\begin{bmatrix} 001 & 110 & 01 \\ 000 & 110 & 11 \\ 100 & 001 & 11 \\ 110 & 010 & 10 \\ 110 & 101 & 00 \\ 001 & 010 & 11 \\ 011 & 101 & 00 \\ 111 & 001 & 00 \end{bmatrix}$$

55.

$$\begin{bmatrix} 000 & 111 & 10 \\ 001 & 001 & 11 \\ 010 & 110 & 11 \\ 101 & 001 & 01 \\ 101 & 001 & 10 \\ 110 & 110 & 01 \\ 111 & 010 & 00 \\ 011 & 101 & 00 \end{bmatrix}$$

56.

$$\begin{bmatrix} 000 & 001 & 11 \\ 000 & 001 & 11 \\ 000 & 011 & 01 \\ 000 & 011 & 01 \\ 001 & 100 & 10 \\ 111 & 100 & 00 \\ 110 & 010 & 00 \\ 111 & 100 & 00 \end{bmatrix}$$

57.

$$\begin{bmatrix} 000 & 111 & 01 \\ 000 & 100 & 11 \\ 000 & 101 & 01 \\ 111 & 000 & 10 \\ 100 & 001 & 10 \\ 101 & 010 & 10 \\ 010 & 111 & 00 \\ 111 & 000 & 00 \end{bmatrix}$$

58.

$$\begin{bmatrix} 011 & 011 & 000 \\ 100 & 101 & 010 \\ 100 & 000 & 110 \\ 010 & 010 & 001 \\ 100 & 100 & 011 \\ 110 & 000 & 101 \\ 001 & 001 & 010 \\ 011 & 010 & 100 \\ 000 & 111 & 000 \end{bmatrix}$$

59.

$$\begin{bmatrix} 010 & 000 & 111 \\ 101 & 000 & 001 \\ 010 & 110 & 001 \\ 001 & 010 & 101 \\ 001 & 101 & 010 \\ 000 & 010 & 110 \\ 100 & 101 & 001 \\ 100 & 011 & 001 \\ 111 & 100 & 110 \end{bmatrix}$$

60.

$$\begin{bmatrix} 011 & 101 & 000 \\ 100 & 010 & 001 \\ 100 & 101 & 100 \\ 101 & 000 & 011 \\ 010 & 001 & 100 \\ 101 & 010 & 010 \\ 001 & 010 & 000 \\ 000 & 101 & 001 \\ 010 & 100 & 010 \end{bmatrix}$$

61.

$$\begin{bmatrix} 000 & 100 & 1011 \\ 001 & 011 & 0100 \\ 010 & 001 & 0001 \\ 100 & 010 & 0010 \\ 010 & 100 & 1000 \\ 011 & 000 & 1100 \\ 100 & 011 & 0010 \\ 010 & 001 & 0001 \\ 100 & 100 & 1000 \\ 101 & 000 & 0100 \end{bmatrix}$$

62.

$$\begin{bmatrix} 001 & 111 & 01 \\ 000 & 110 & 11 \\ 100 & 011 & 10 \\ 110 & 001 & 10 \\ 111 & 000 & 01 \\ 101 & 100 & 10 \\ 011 & 101 & 01 \\ 110 & 010 & 10 \end{bmatrix}$$

63.

$$\begin{bmatrix} 001 & 011 & 10 \\ 001 & 110 & 01 \\ 110 & 001 & 01 \\ 010 & 010 & 11 \\ 110 & 100 & 10 \\ 101 & 000 & 11 \\ 100 & 111 & 00 \\ 011 & 101 & 00 \end{bmatrix}$$

64.

$$\begin{bmatrix} 011 & 001 & 10 \\ 100 & 010 & 01 \\ 100 & 010 & 01 \\ 000 & 001 & 11 \\ 011 & 000 & 10 \\ 100 & 100 & 01 \\ 100 & 110 & 00 \\ 011 & 101 & 00 \end{bmatrix}$$

Задача 2

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

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

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

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

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

6. {1,2}, {1,6}, {1,9}, {2,3}, {2,8}, {2,10}, {3,5}, {3,8}, {4,6}, {4,7}, {4,9}, {5,7}, {5,10}, {6,7}, {7,9}, {8,10}

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

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

9. {1,4}, {1,7}, {1,9}, {1,10}, {2,3}, {2,5}, {2,6}, {2,8}, {3,6}, {3,10}, {4,5}, {4,9}, {5,7}, {6,7}, {6,8}, {7,9}, {8,10}

10. {1,3}, {1,4}, {1,5}, {1,6}, {1,8}, {2,4}, {2,5}, {2,7}, {2,8}, {3,5}, {3,6}, {3,7}, {4,6}, {4,7}, {5,8}, {6,7}, {7,8}

11. {1,4}, {1,5}, {1,6}, {1,7}, {2,3}, {2,5}, {2,6}, {2,8}, {3,4}, {3,5}, {3,6}, {4,7}, {4,8}, {5,7}, {6,8}, {7,8}

12. {1,3}, {1,4}, {1,8}, {2,4}, {2,7}, {2,8}, {3,5}, {3,7}, {4,5}, {4,6}, {5,8}, {6,7}, {6,8}

13. {1,3}, {1,4}, {1,8}, {2,3}, {2,4}, {2,6}, {2,7}, {3,5}, {4,6}, {4,7}, {5,7}, {5,8}, {6,7}, {6,8}

14. {1,3}, {1,4}, {1,5}, {1,8}, {2,4}, {2,5}, {2,6}, {2,7}, {3,6}, {3,7}, {3,8}, {4,7}, {4,8}, {5,6}, {5,7}, {6,8}

15. {1,3}, {1,4}, {1,6}, {1,7}, {1,8}, {2,4}, {2,5}, {2,7}, {2,8}, {3,5}, {3,6}, {3,8}, {4,5}, {4,8}, {5,6}, {5,7}, {6,7}

16. {1,3}, {1,7}, {1,8}, {2,4}, {2,6}, {2,7}, {3,4}, {3,5}, {3,6}, {4,8}, {5,7}, {5,8}, {6,8}

17. {1,2}, {1,4}, {1,5}, {1,6}, {2,3}, {2,6}, {2,7}, {3,7}, {3,8}, {4,6}, {4,7}, {4,8}, {5,7}, {5,8}

18. {1,5}, {1,6}, {1,7}, {2,3}, {2,4}, {2,6}, {2,7}, {3,8}, {3,9}, {4,5}, {4,7}, {4,8}, {5,6}, {6,9}, {7,9}, {8,9}

19. {1,3}, {1,6}, {1,7}, {1,8}, {2,4}, {2,6}, {2,7}, {3,4}, {3,5}, {3,7}, {4,7}, {4,9}, {5,8}, {5,9}, {6,7}, {6,8}, {7,9} {8,9}

20. {1,2}, {1,7}, {1,8}, {1,9}, {2,5}, {2,8}, {2,9}, {3,5}, {3,6}, {3,8}, {4,5}, {4,7}, {4,9}, {6,8}, {6,9}

21. {1,3}, {1,5}, {1,9}, {2,4}, {2,8}, {2,9}, {2,10}, {3,5}, {3,6}, {3,9}, {4,6}, {4,10}, {5,7}, {6,8}, {7,9}, {7,10}, {8,10}

22. {1,3}, {1,4}, {1,5}, {1,6}, {2,5}, {2,6}, {2,7}, {2,8}, {3,4}, {3,5}, {3,7}, {4,6}, {4,8}, {5,7}, {5,8}, {6,7}, {6,8}

23. {1,4}, {1,5}, {1,6}, {1,7}, {2,3}, {2,4}, {2,6}, {2,8}, {3,5}, {3,7}, {3,8}, {4,5}, {4,6}, {5,7}, {6,8}, {7,8}

24. {1,3}, {1,5}, {1,6}, {1,7}, {2,4}, {2,5}, {2,8}, {3,4}, {3,5}, {4,6}, {4,7}, {6,8}, {7,8}

25. {1,3}, {1,6}, {1,7}, {1,8}, {2,4}, {2,5}, {2,7}, {3,5}, {3,6}, {3,7}, {4,6}, {4,8}, {5,8}, {6,7}

26. {1,1,3}, {1,5}, {1,7}, {1,8}, {2,4}, {2,6}, {2,7}, {2,8}, {3,5}, {3,6}, {3,7}, {4,5}, {4,6}, {4,7}, {5,8}, {6,8}

27. {1,3}, {1,4}, {1,7}, {1,8}, {2,4}, {2,5}, {2,6}, {2,8}, {3,6}, {3,7}, {3,8}, {4,5}, {4,6}, {4,7}, {5,7}, {5,8}, {6,8}

28. {1,3}, {1,4}, {1,5}, {1,6}, {2,3}, {2,6}, {2,8}, {3,7}, {4,7}, {4,8}, {5,7}, {5,8}, {6,7}

29. {1,2}, {1,5}, {1,7}, {1,8}, {2,3}, {2,4}, {2,8}, {3,5}, {3,6}, {4,5}, {4,6}, {4,7}, {5,7}, {6,8}

30. {1,3}, {1,5}, {1,6}, {1,8}, {2,6}, {2,8}, {2,9}, {3,7}, {3,8}, {3,9}, {4,5}, {4,6}, {4,7}, {4,8}, {5,7}, {6,9}

31. {1,3}, {1,4}, {1,7}, {2,4}, {2,7}, {2,8}, {2,9}, {3,6}, {3,7}, {3,9}, {4,5}, {4,7}, {5,6}, {5,7} {5,8}, {6,8}, {6,9}, {7,9}

32. {1,3}, {1,4}, {1,8}, {2,3}, {2,6}, {2,7}, {2,9}, {3,6}, {3,8}, {4,5}, {4,6}, {5,7}, {5,9}, {6,9}, {8,9}

33. {1,2}, {1,6}, {1,9}, {2,3}, {2,8}, {2,10}, {3,5}, {3,8}, {4,6}, {4,7}, {4,9}, {4,10}, {5,7}, {5,10}, {6,7}, {7,9}, {8,10}

34. {1,3}, {1,4}, {1,5}, {1,6}, {1,7}, {2,4}, {2,5}, {2,6}, {2,7}, {2,8}, {3,5}, {3,7}, {3,8}, {4,6}, {4,7}, {5,8}, {6,8}

35. {1,3}, {1,5}, {1,6}, {1,8}, {2,4}, {2,5}, {2,7}, {2,8}, {3,5}, {3,6}, {3,7}, {4,6}, {4,7}, {4,8}, {5,7}, {6,8}

36. {1,3}, {1,7}, {1,8}, {2,4}, {2,6}, {2,7}, {3,4}, {3,5}, {3,6}, {4,7}, {5,7}, {5,8}, {6,8}

37. {1,3}, {1,6}, {1,8}, {2,4}, {2,5}, {2,7}, {2,8}, {3,5}, {3,7}, {4,5}, {4,6}, {4,8}, {5,8}, {6,7}

38. {1,3}, {1,4}, {1,5}, {1,8}, {2,4}, {2,5}, {2,7}, {2,8}, {3,6}, {3,7}, {3,8}, {4,5}, {4,7}, {5,6}, {6,7}, {6,8}, {7,8}

39. {1,3}, {1,4}, {1,5}, {1,6}, {2,5}, {2,6}, {2,7}, {2,8}, {3,5}, {3,6}, {3,7}, {4,5}, {4,6}, {4,8}, {5,7}, {6,8}

40. {1,3}, {1,4}, {1,5}, {1,6}, {2,3}, {2,6}, {2,8}, {3,7}, {4,7}, {4,8}, {5,7}, {5,8}, {6,7}

41. {1,5}, {1,6}, {1,8}, {2,3}, {2,4}, {2,5}, {2,6}, {3,6}, {3,7}, {4,7}, {4,8}, {5,7}, {5,8}, {6,8}

42. {1,3}, {1,4}, {1,7}, {2,4}, {2,5}, {2,8}, {2,9}, {3,6}, {3,8}, {3,9}, {4,6}, {4,7}, {5,6}, {5,8}, {6,9}, {7,9}

43. {1,4}, {1,6}, {1,7}, {1,9}, {2,3}, {2,8}, {2,9}, {3,4}, {3,5}, {3,8}, {4,7}, {4,8} {5,6}, {5,7}, {5,8}, {6,8}, {6,9}, {8,9}

44. {1,4}, {1,5}, {1,6}, {1,7}, {2,7}, {2,8}, {2,9}, {3,4}, {3,5}, {4,6}, {4,9}, {5,8}, {6,8}, {6,9}, {7,9}

45. {1,6}, {1,7}, {1,8}, {2,4}, {2,6}, {2,7}, {2,8}, {3,4}, {3,6}, {3,9}, {3,10}, {4,9}, {4,10}, {5,8}, {5,9}, {5,10}, {7,8}

46. {1,2}, {1,5}, {1,6}, {1,8}, {2,3}, {2,6}, {2,7}, {2,8}, {3,4}, {3,7}, {3,8}, {4,5}, {4,6}, {4,7}, {5,6}, {5,7}, {5,8}

47. {1,2}, {1,5}, {1,6}, {1,8}, {2,3}, {2,7}, {2,8}, {3,4}, {3,7}, {3,8}, {4,5}, {4,6}, {4,7}, {5,6}, {5,7}, {6,8}

48. {1,5}, {1,6}, {1,7}, {2,6}, {2,7}, {2,8}, {3,5}, {3,7}, {3,8}, {4,6}, {4,7}, {4,8}, {5,6}

49. {1,2}, {1,6}, {1,7}, {1,8}, {2,4}, {2,6}, {2,8}, {3,5}, {3,6}, {3,7}, {4,5}, {4,7}, {5,8}, {6,8}

50. {1,3}, {1,4}, {1,5}, {1,8}, {2,4}, {2,5}, {2,7}, {2,8}, {3,6}, {3,7}, {3,8}, {4,5}, {4,7}, {5,6}, {6,7}, {6,8}

51. {1,4}, {1,5}, {1,6}, {1,7}, {2,3}, {2,6}, {2,7}, {2,8}, {3,4}, {3,5}, {3,7}, {3,8}, {4,6}, {4,8}, {5,6}, {5,7}, {6,8}

52. {1,3}, {1,4}, {1,8}, {2,4}, {2,7}, {2,8}, {3,5}, {3,7}, {4,5}, {4,6}, {5,8}, {6,7}, {6,8}

53. {1,3}, {1,4}, {1,8}, {2,3}, {2,4}, {2,6}, {2,7}, {3,5}, {4,6}, {4,7}, {5,7}, {5,8}, {6,7}, {6,8}

54. {1,3}, {1,4}, {1,5}, {1,8}, {2,4}, {2,5}, {2,6}, {2,7}, {3,6}, {3,7}, {3,8}, {4,7}, {4,8}, {5,6}, {5,7}, {6,8}

55. {1,3}, {1,4}, {1,6}, {1,7}, {1,8}, {2,4}, {2,5}, {2,7}, {2,8}, {3,5}, {3,6}, {3,8}, {4,5}, {4,8}, {5,6}, {5,7}, {6,7}

56. {1,3}, {1,7}, {1,8}, {2,4}, {2,6}, {2,7}, {3,4}, {3,5}, {3,6}, {4,8}, {5,7}, {5,8}, {6,8}

57. {1,2}, {1,4}, {1,5}, {1,6}, {2,3}, {2,6}, {2,7}, {3,7}, {3,8}, {4,6}, {4,7}, {4,8}, {5,7}, {5,8}

58. {1,5}, {1,6}, {1,7}, {2,3}, {2,4}, {2,6}, {2,7}, {3,8}, {3,9}, {4,5}, {4,7}, {4,8}, {5,6}, {6,9}, {7,9}, {8,9}

59. {1,3}, {1,6}, {1,7}, {1,8}, {2,4}, {2,6}, {2,7}, {3,4}, {3,5}, {3,7}, {4,7}, {4,9}, {5,8}, {5,9}, {6,7}, {6,8}, {7,9}, {8,9}

60. {1,2}, {1,7}, {1,8}, {1,9}, {2,5}, {2,8}, {2,9}, {3,5}, {3,6}, {3,8}, {4,5}, {4,7}, {4,9}, {6,8}, {6,9}

61. {1,3}, {1,5}, {1,9}, {2,4}, {2,8}, {2,9}, {2,10}, {3,5}, {3,6}, {3,9}, {4,6}, {4,10}, {5,7}, {6,8}, {7,9}, {7,10}, {8,10}

62. {1,3}, {1,4}, {1,5}, {1,6}, {2,5}, {2,6}, {2,7}, {2,8}, {3,4}, {3,5}, {3,7}, {4,6}, {4,8}, {5,7}, {5,8}, {6,7}, {6,8}

63. {1,4}, {1,5}, {1,6}, {1,7}, {2,3}, {2,4}, {2,6}, {2,8}, {3,5}, {3,7}, {3,8}, {4,5}, {4,6}, {5,7}, {6,8}, {7,8}

64. {1,3}, {1,5}, {1,6}, {1,7}, {2,4}, {2,5}, {2,8}, {3,4}, {3,5}, {4,6}, {4,7}, {6,8}, {7,8}

9. Маршруты в графах

Последовательность

$$\mu[x_1, u_1, x_2, u_2, x_3, \ldots, u_{n-1}, x_n] \quad (x_i \in X, u_i \in U),$$

в которой чередуются вершины и ребра и при этом

$$\forall i \in \overline{1, n-1} \quad u_i = \{x_i, x_{i+1}\},$$

называется *маршрутом*.
Чаще маршрут изображается последовательностью вершин

$$\mu[x_1, x_2, \ldots, x_n],$$

для которой любые две соседние вершины смежны.

Длина маршрута определяется либо как число ребер маршрута, либо как сумма длин ребер (при введении весов ребер, называемых их длиной):

$$l(\mu) = \sum_{u \in \mu} l(u).$$

Первый случай может быть включен во второй, если длину каждого ребра по умолчанию считать равной 1.

Цепь – маршрут, в котором все ребра попарно различны. Так, маршрут $\mu[x_1, x_2, x_3, x_4, x_5, x_3, x_2, x_6]$ не является цепью – ребро $\{x_2, x_3\}$ повторяется дважды.

Простая цепь – это цепь, в которой нет повторяющихся вершин. Так, цепь $\mu[x_1, x_2, x_3, x_4, x_2, x_5]$ не является простой.

Маршрут *замкнутый*, если первая вершина маршрута совпадает с последней.

Цикл – замкнутая цепь (нет повторяющихся ребер).

Простой цикл – цикл, у которого все вершины различны, кроме совпадающих первой и последней вершин.

Граф *связен*, если для любых двух его вершин существует цепь, их соединяющая. Если граф G не связен, то его можно разделить на *компоненты связности* – связные подграфы, каждый из которых не является собственным подграфом никакого другого связного подграфа из G. Так, например, граф

$$G(\{1,2,3,4,5,6,7\}, \{\{1,2\},\{1,3\},\{2,3\},\{2,4\},\{5,6\}\})$$

состоит из трех компонент связности $G_1(\{5,6\}\{\{5,6\}\})$, $G_2(\{7\},\emptyset)$, $G_3(\{1,2,3,4\},\{\{1,2\},\{1,3\},\{2,3\},\{2,4\}\})$.

Расстояние между вершинами – длина кратчайшей цепи, их связывающей:

$$\rho(x_i, x_j) = \min_{\mu=\mu[x_i,\ldots,x_j]} l(\mu).$$

Диаметр связного графа – расстояние между двумя наиболее удаленными вершинами в графе:

$$d(G) = \max_{x_i,x_j \in G} \rho(x_i, x_j).$$

Радиусом графа называется минимум по всем вершинам расстояний от каждой из них до: наиболее удаленной от нее вершины:

$$r(G) = \min_{x \in X} \max_{y \in X} \rho(x, y).$$

Центр графа – множество вершин графа, расстояние от каждой из которых до наиболее удаленной вершины графа равно радиусу графа:

$$C = \{x |\ x \in X, \max_{x_i \in X} \rho(x, x_i) = r(G)\}.$$

Так, для графа, изображенного на рис. 10, $r(G) = 2$, $C(G) = \{1, 3, 4, 7\}$.

В орграфе цепь $\nu[x_1, \ldots, x_n]$, проходимая в направлении ориентации дуг $((x_i, x_{i+1}) \in \overline{U}\ \ (i \in \overline{1, n-1}))$, называется *путем*. *Простой путь*

есть простая ориентированная цепь. *Контур* есть ориентированный цикл, а *простой контур* есть простой ориентированный цикл.

Орграф $G(X, \overline{U})$ называется *связным*, если из любой его вершины $x \in X$ в любую его вершину $y \in X$ есть путь. При связности орграфа аналогичным для обыкновенного графа образом вводятся диаметр, радиус и центр орграфа. Если орграф не связен, но обыкновенный граф, полученный из него потерей ориентации дуг (заменой дуг на ребра) является связным, то орграф называется *слабо связным*. Так, орграф, изображенный на рис. 9, связный с диаметром 3, радиусом 2 и центром $\{3\}$. А орграф $G(\{1,2,3,4\}, \{(1,2),(1,3),(1,4),(2,4),(4,3)\})$, отличающийся от предыдущего только ориентацией дуги между вершинами 1 и 3, не является связным, он слабо связный.

10. Задача о кратчайшем маршруте

Задача о кратчайшем маршруте для произвольных вершин a и b связного графа G формулируется следующим образом:

$$\textit{найти кратчайший маршрут } \mu[a,\ldots,b]: \quad l(\mu) = l(a,b).$$

Один из эффективных алгоритмов нахождения кратчайшего маршрута предложен нидерландским информатиком Эдсгер Вибэ Дейкстра (лауреат премии Тьюринга, один из создателей языка АЛГОЛ, концепции структурного программирования, методов верификации и распределенных вычислений) и назван в его честь *алгоритмом Дейкстры*. В этом алгоритме для каждой вершины x вводятся 2 характеристики:
λ_x – расстояние от вершины a до x и
k_x – номер предыдущей для x вершины на кратчайшем маршруте от 1 до x.

Пошаговое описание алгоритма:

1^o. Составляем множество S всех вершин графа. Полагаем $\lambda_a = 0$, $k_a = 0$ и для каждой другой вершины x графа $\lambda_x = \infty, k_x = 0$.

2^o Среди всех вершин множества S ищем вершину[1] y с минимальным

$$\lambda_y = \min_{x \in S} \lambda_x.$$

[1] При неоднозначности выбора вершины в этом или любом другом алгоритме на графах всегда выбираем вершину с наименьшим номером.

3^o. Если $y = b$, то переходим к п. 6^o.

4^o. Исключаем вершину y из множества S и пересчитываем характеристики остальных из множества S вершин, смежных y:

$$\text{если } \lambda_x > \lambda_y + l(y,x), \text{то } \lambda_x = \lambda_y + l(y,x) \quad (x \in S,\ \{y,x\} \in U),$$

где $l(y,x)$ – длина ребра $\{y,x\}$.

5^o. Переходим к п. 2^o.

6^o. Конец: λ_b – длина кратчайшего маршрута, а сам маршрут находится по пометкам k_x предыдущих вершин: $\mu[a,\ldots,k_{k_b},k_b,b]$.

11. Пример задачи на нахождение кратчайшего маршрута в графе с ее решением

Пример 6. В списке ребер с их длинами каждый элемент есть упорядоченная пара: ребро как множество двух смежных вершин и длина ребра:
(({1,2}, 6), ({1,3}, 3), ({1,4}, 9), ({1,5}, 8), ({2,3}, 1), ({2,5}, 2),
({2,6}, 2), ({3,7}, 4), ({4,5}, 2), ({4,8}, 1), ({5,9}, 7), ({6,7}, 2),
({7,8}, 3), ({8,9}, 2)).

А. В алгоритме Дейкстры для каждой вершины x вводятся 2 характеристики:
λ_x – расстояние от начальной вершины (в условиях задачи – 1) до x,
k_x – номер предыдущей вершины на кратчайшем маршруте от 1 до x.
В таблице выполнения алгоритма для каждого номера вершины заводим 2 колонки для λ и k. Дадим пошаговое описание алгоритма:

1^o. Составляем множество S всех вершин графа. Полагаем $\lambda_1 = 0, k_1 = 0$ и для каждой другой вершины x графа $\lambda_x = \infty, k_x = 0$.

2^o. Среди всех вершин множества S ищем вершину j с минимальным

$$\lambda_j = \min_{i \in S} \lambda_i.$$

3^o. Если j – номер конечной (последней в условиях задачи) вершины, то переходим к п. 6^o.

4^o. Исключаем вершину j из множества S и пересчитываем характеристики вершин из множества S, смежных j:

$$\text{если } \lambda_i > \lambda_j + l_{ji}, \text{то } \lambda_i = \lambda_j + l_{ji} \quad (i \in S,\ \{j,i\} \in U),$$

где l_{ji} – длина ребра $\{j,i\}$.

5^o Переходим к п. 2^o.

6^o Конец: λ_j – длина кратчайшего маршрута, а сам маршрут находится по пометкам k предыдущих вершин: $j, k_j, k_{k_j}, \ldots$.

В. Сведем выполнение алгоритма в следующую таблицу:

N	$\overline{1}$		$\overline{2}$		$\overline{3}$		$\overline{4}$		$\overline{5}$		$\overline{6}$		$\overline{7}$		$\overline{8}$		$\overline{9}$	
	λ	k	λ	k	λ	k	λ	k	λ	k	λ	k	λ	k	λ	k	λ	k
1	0	0	∞	0	∞	0	∞	0	∞	0	∞	0	∞	0	∞	0	∞	0
2			6	1	3	1	9	1	8	1								
3			4	3									7	3				
4									6	2	6	2						
5							8	5			6	2					13	5
6													7	3				
7							8	5							10	7		
8															9	4		
9																	11	8

В этой таблице каждый столбец соответствует вершине графа и двум его характеристикам λ и k, каждая строка таблицы соответствует определению новых значений характеристик при выполнении шагов 1^o и 4^o алгоритма, выбор вершины с минимальной характеристикой λ обозначается заключением в рамку характеристик этой вершины (для наглядности при необходимости эти характеристики сносятся в строку определения минимума λ), а исключение вершины из множества S помечается чертой над ее номером.

Мы получили длину 11 кратчайшего маршрута из вершины 1 в вершину 9. Начиная с вершины 9, определяем последовательность номеров предшествующих вершин в найденном маршруте:

$$9, 8, 4, 5, 2, 3, 1.$$

Таким образом, кратчайший маршрут, соединяющий вершины 1 и 9:

$$\mu[1, 3, 2, 5, 4, 8, 9],$$

а его длина

$$l(\mu) = l(1,3) + l(3,2) + l(2,5) + l(5,4) + l(4,8) + l(8,9) = 11.$$

12. Эйлеровы маршруты

Эйлеровым маршрутом в графе называется маршрут (цепь, цикл), содержащий все ребра графа. *Эйлеровым ориентированным маршрутом* в орграфе называется ориентированный маршрут (путь, контур), содержащий все дуги орграфа. Критерий существования эйлеровой цепи или цикла дает следующая теорема.

Теорема об эйлеровом маршруте. *Для того чтобы граф содержал эйлерову цепь, необходимо и достаточно, чтобы он был связен и имел ровно 2 вершины нечетной степени. Для того чтобы граф содержал эйлеров цикл, необходимо и достаточно, чтобы он был связен и не содержал вершин нечетной степени.*

Доказательство. Необходимость. Если эйлерова цепь есть, то граф связен (любые его вершины связаны этой цепью или ее частью). Нечетной степенью обладают только начальная и конечная вершины цепи. Поэтому число вершин с нечетной степенью равно 2. Если мультиграф содержит эйлеров цикл, то он связен и все его вершины имеют четную степень (при прохождении цикла сколько раз мы войдем в некоторую его вершину, столько раз мы и выйдем из нее).

Достаточность. Доказательство проведем по индукции по числу ребер графа.

В случае когда граф не имеет вершин нечетной степени, при числе ребер $m(G) = 2$ имеется ровно 2 вершины a и b, каждая из которых имеет степень 2, и граф имеет цикл $(a,\ b,\ a)$, который является эйлеровым.

В случае когда граф G связен и имеет ровно 2 вершины нечетной степени a и b, при числе ребер $m(G) = 1$ утверждение справедливо, так как это ребро – $\{a, b\}$ и эйлеров путь – $\mu[a, b]$, а при числе ребер $m(G) = 2$ утверждение также справедливо, так как есть эйлеров путь $\mu[a, c, b]$, где c – вершина, имеющая степень 2.

Таким образом, основание индукции имеет место.

Пусть теперь утверждение теоремы справедливо для любого связного графа G, имеющего не более двух вершин нечетной степени, и $m(G) \leq k$, где $k > 1$, и докажем, что оно справедливо при $m(G) = k+1$. Пусть сначала граф G имеет ровно две вершины нечетной степени.

Отправим путешественника из вершины a с условием не проходить по одному и тому же ребру дважды. Если путешественник пришел в вершину $x \neq b$, то при $x = a$ количество пройденных ребер, инцидентных a, четно (число выходов из вершины a равно числу входов в эту вершину) и нечетная степень вершины a позволяет путешественнику выйти из a по непройденному еще ребру, а если $x \neq a$, то количество пройденных ребер, инцидентных x, нечетно (число входов в вершину x на 1 больше числа выходов из этой вершины) и четная степень вершины x позволяет также выйти из этой вершины по не пройденному еще ребру. Если невозможно идти дальше (все ребра, инцидентные вершине, уже пройдены), то это вершина b. Цепь, пройденную путешественником, обозначим через $\mu_0[a, \ldots, b]$.

Однако при этом могут быть еще непройденные ребра графа. В подграфе G', образованном удалением пройденных ребер, все вершины имеют уже четную степень и число его ребер $m(G') < k$. Пусть $C_1, \ldots, C_p$ – компоненты связности G', содержащие хотя бы 1 ребро. Так как $m(C_i) < k$ ($i \in \overline{1,k}$, то по предположению индукции для них есть эйлеровы циклы $\mu_1, \ldots, \mu_p$. Так как G связен, то цепь μ_0 встречает при ее прохождении все подграфы $C_1, \ldots, C_p$. Допустим, что это происходит в вершинах $x_1 \in C_1, \ldots, x_p \in C_p$. Обозначим через $\mu_0[x, \ldots, y]$ отрезок цепи μ_0 от вершины x до вершины y. Рассмотрим цепь

$$\mu_0[a, \ldots, x_1] + \mu_1 + \mu_0[x_1, \ldots, x_2] + \mu_2 + \cdots + \mu_p + \mu_0[x_p, \ldots, b].$$

Это и есть эйлерова цепь графа G, которую необходимо было построить.

Пусть теперь граф G не имеет вершин нечетной степени. Тогда, удалив любое его ребро $\{a, b\}$, мы получим связный граф, содержащий ровно 2 вершины нечетной степени a и b. Как мы только что доказали, он имеет эйлерову цепь, а потому, добавив к этой цепи удаленное ребро $\{a, b\}$, мы получим эйлеров цикл графа, что и требовалось. Теорема доказана.

Замечание. Теорема очевидным образом переносится на мультиграф (граф, между вершинами которого может быть более одного ребра).

Теорема об эйлеровом ориентированном маршруте.

1. Для того чтобы орграф содержал эйлеров путь, необходимо и достаточно, чтобы он был слабо связен и для всех вершин, кроме двух, полустепень исхода была равна полустепени захода, для одной вершины полустепень исхода была на 1 больше полустепени захода и для другой вершины полустепень исхода была на 1 меньше полустепени захода.

2. Для того чтобы орграф содержал эйлеров контур, необходимо и достаточно, чтобы он был слабо связен и для всех вершин полустепень исхода была равна полустепени захода.

Доказательство этой теоремы аналогично доказательству предыдущей для неориентированного графа. Равенство полустепеней исхода и захода для всех вершин, кроме двух, гарантирует, что, выйдя из вершины с полустепенью исхода большей полустепени захода, мы можем остановиться только в вершине с полустепенью захода большей полустепени исхода, так как полустепени исхода и захода всех остальных вершин равны.

Алгоритм отыскания эйлерова маршрута. Конструкцию доказательства теорем нельзя использовать для построения эффективного алгоритма отыскания эйлерова маршрута, так как оно содержит неконструктивное предположение индукции.

Введем понятие *перешейка* графа, которое необходимо для описания эффективного алгоритма.

Перешейком в графе называется ребро, удаление которого приводит к несвязности графа: нет маршрута из одного конца удаленного ребра в другой (в одну компоненту связности попадает один конец ребра, а в другую – другой конец ребра).

Перешейком в орграфе называется дуга, удаление которой приводит к несвязности графа: из вершины конца дуги нет ориентированного маршрута в ее начало.

Опишем алгоритм построения эйлерова маршрута.

1^0. Выходим из начальной вершины в зависимости от вида маршрута: при цикле или контуре – из произвольной вершины, при цепи – из любой вершины нечетной степени, при ориентированном пути

– из вершины, полустепень исхода которой больше полустепени захода. Все ребра (дуги) считаем непомеченными.

2^0. Двигаемся из вершины при неориентированном маршруте по любому непомеченному ребру, не являющемуся перешейком в подграфе непомеченных ребер, и помечаем пройденное ребро, а при ориентированном маршруте – по любой исходящей из вершины непомеченной дуге, не являющейся перешейком в подграфе из непомеченных дуг, и помечаем ее.

3^0. Повторяем предыдущий пункт до тех пор, пока не придем в вершину, все инцидентные ребра (исходящие дуги) которой уже помечены. Полученный маршрут и будет эйлеровым.

Для обоснования алгоритма нужно показать, что при выполнении пункта 2^0 из достигнутой вершины всегда есть непомеченное исходящее ребро (дуга), не являющееся перешейком, если эта вершина не удовлетворяет условиям в пункте 3^0 алгоритма. Действительно, граф из непомеченных ребер (дуг) удовлетворяет условиям теорем об эйлеровом маршруте и эйлеров маршрут этого графа как раз выходит из достигнутой вершины. Поэтому существует выходящее из этой вершины ребро (дуга) этого маршрута, которое не является перешейком.

Приведем теперь простой алгоритм, позволяющий определить, является ли выбранное ребро перешейком. Для этого в графе из непомеченных ребер будем определять компоненту связности другой вершины ребра (дуги), включая в нее эту вершину, затем смежные ей вершины по непомеченным ребрам (исходящим дугам), затем смежные им вершины по непомеченным ребрам (исходящим дугам) и т. д. до тех пор, пока либо не будет включена другая вершина выбранного ребра (дуги), либо процесс формирования компоненты связности не остановится с невключенной в нее другой вершиной выбранного ребра (дуги). В первом случае выбранное ребро не является перешейком, а во втором случае – является.

13. Примеры задач на нахождение эйлерова маршрута в графе с их решениями

Пример 7.

$$\begin{bmatrix} 001 & 101 \\ 001 & 010 \\ 110 & 010 \\ 100 & 001 \\ 011 & 000 \\ 100 & 100 \end{bmatrix}$$

Матрица смежности вершин графа симметрична, поэтому граф неориентированный. Граф содержит 6 вершин и 7 ребер. Для анализа связности графа образуем компоненту связности C_1, содержащую вершину 1, и включим в нее эту вершину: $C_1 = \{1\}$. Вершина 1 смежна с вершинами 3, 4, 6. Добавим эти вершины: $C_1 = \{1, 3, 4, 6\}$. Вершина 3 смежна с вершинами 1, 2 и 5. Добавим их: $C_1 = \{1, 2, 3, 4, 5, 6\}$. Так как C_1 содержит все вершины графа, то граф связен.

Определяем степени вершин графа по матрице (число единиц в каждой строке): $d_1 = 3,\ d_2 = 2,\ d_3 = 3,\ d_4 = 2,\ d_5 = 2,\ d_6 = 2$. Граф содержит 4 вершины четной степени и 2 – нечетной. Поэтому граф не может содержать эйлеров цикл (все вершины для этого должны иметь четную степень), но содержит эйлерову цепь, так как выполнены условия существования эйлеровой цепи:
Для того чтобы граф содержал эйлерову цепь, необходимо и достаточно, чтобы он был связен и содержал ровно две вершины нечетной степени.

Опишем алгоритм нахождения эйлеровой цепи. Для этого все ребра в начале алгоритма считаем непомеченными, а по мере их включения в маршрут будем делать пометки соответствующих ребер.

1^o. Выбираем в качестве начальной любую вершину нечетной степени (для определенности берем вершину с меньшим номером) и делаем ее текущей.

2^o. Если из текущей вершины нет непройденных (непомеченных) ребер, то алгоритм закончен: все ребра пройдены и цепь составляем в порядке их прохождения.

3^o. Из текущей вершины выбираем любое ребро (для определенности ребро, ведущее в вершину с наименьшим номером).

4^o. Если оно не является *перешейком* (из текущей вершины есть еще непройденные ребра, а при временной пометке данного ребра граф непомеченных ребер не содержит маршрута из текущей вершины в другую вершину данного ребра), то выбираем его, помечаем и делаем текущей вершину смежную по этому ребру с предыдущей смежной вершиной. Переходим к п. 2^o.

5^o. Если выбранное на предыдущем шаге ребро является *перешейком*, то выбираем следующее ребро, ведущее из текущей вершины, и переходим к п. 4^o.

Выполнение алгоритма сводим в следующую таблицу:

N	$\overline{1,3}$	$\overline{1,4}$	$\overline{1,6}$	$\overline{2,3}$	$\overline{2,5}$	$\overline{3,5}$	$\overline{4,6}$	I	не перешеек?
0								1	
1	$-$							3	$3,2,5,3-$
		$+$						4	$4,6,1+$
2							$+$	6	$6,1,3,2,5,3=$
3			$+$					1	$1,3,2,5,3=$
4	$+$							3	$3,2,5,3=$
5				$+$				2	$2,5,3=$
6					$+$			5	$5,3=$
7						$+$		3	$3=$

При неоднозначности выбора начинаем с вершины с наименьшим номером. В данном примере 2 вершины 1 и 3 имеют нечетную степень, а потому мы начинаем выполнение алгоритма с вершины с наименьшим из этих двух номеров, т. е. с 1. Каждый столбец (кроме первого и 2 последних) отвечает ребру графа, и все ребра идут в лексикографическом порядке. Для пометки ребра ставим над ним черту. В первом столбце отмечаем номер шага выполнения алгоритма, а в предпоследнем – номер текущей вершины. В последнем столбце проверяем, не является ли выбранное ребро перешейком, выписывая в список последовательно все вершины, смежные с концом ребра, затем с выписанными вершинами и т. д. до тех пор, пока не встретится вершина-начало проверяемого

ребра (тогда ребро не перешеек, и мы ставим +) или список далее не растет (если все вершины уже включены и список зканчивается вершиной конца эйлерового пути, то мы ставим знак =, а в противном случае ставим знак –, так как ребро является перешейком). В этом случае мы выбираем следующее ребро, имеющее то же начало, что и перешеек. А при знаках + или = переходим к выбору следующего по порядку ребра с началом в конце предыдущего ребра.

Если рассматриваемое ребро является перешейком, то на пересечении текущей строки выполнения алгоритма и столбца ребра ставим *минус*. В противном случае ставим *плюс*, помечаем ребро и заносим новую текущую вершину.

По окончании алгоритма столбец текущей вершины определит эйлерову цепь. На шаге 1 выполнения алгоритма ребро {1,3} является перешейком, так как после его временного исключения не существует маршрута, связывающего вершины 1 и 3: компонента связности в графе непомеченных ребер, содержащая вершину 1, содержит также смежные с ней вершины 4 и 6, но никаких других вершин (в частности, вершины 3) не содержит.

В результате выполнения алгоритма получена следующая эйлерова цепь:

$$(1, 4, 6, 1, 3, 2, 5, 3).$$

Пример 8.

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 1 & 0 & 0 & 0 & 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ -1 & 0 & -1 & 0 & 0 & 0 & 1 & 1 \\ 0 & -1 & 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & -1 & -1 & 0 & 0 \end{bmatrix}$$

Матрица смежности вершин кососимметрична, поэтому это орграф. Определяем связность графа как неориентированного. Включаем в компоненту связности C_1 вершину 1: $C_1 = \{1\}$. Вершина 1 смежна с вершинами 3 и 5, добавляем их: $C_1 = \{1, 3, 5\}$. Вершины 3 и 5 смежны с вершинами 6, 7, 8. Добавляем их в компоненту связности: $C_1 =$

$\{1, 3, 5, 6, 7, 8\}$. Вершины 6 и 8 смежны с вершинами 2 и 4. Добавляем их в компоненту связности: $C_1 = \{1, 2, 3, 4, 5, 6, 7, 8\}$. Поскольку компонента связности содержит все вершины графа, он связен как неориентированный.

Находим полустепени исхода и захода для каждой вершины и сравниваем их:

1. $d_1^- = 1, d_1^+ = 1 \to d_1^- = d_1^+$
2. $d_2^- = 1, d_2^+ = 1 \to d_2^- = d_2^+$
3. $d_3^- = 2, d_3^+ = 2 \to d_3^- = d_3^+$
4. $d_4^- = 1, d_4^+ = 1 \to d_4^- = d_4^+$
5. $d_5^- = 2, d_5^+ = 2 \to d_5^- = d_5^+$
6. $d_6^- = 2, d_6^+ = 2 \to d_6^- = d_6^+$
7. $d_7^- = 1, d_7^+ = 1 \to d_7^- = d_7^+$
8. $d_8^- = 2, d_8^+ = 2 \to d_8^- = d_8^+$

Из того, что орграф слабо связен (связен как неориентированный) и для каждой вершины полустепень исхода и полустепень захода совпадают, следует, что орграф содержит эйлеров контур, так как выполнены условия существования эйлерова контура.
Для того чтобы орграф содержал эйлеров контур, необходимо и достаточно, чтобы он был слабо связен и для каждой вершины полустепени исхода и захода были бы равны.

Опишем алгоритм нахождения эйлерова контура. Для этого все дуги в начале алгоритма считаем непомеченными, а по мере их включения в маршрут будем делать пометки соответствующих дуг.

1^o. Выбираем в качестве начальной любую вершину (для определенности берем вершину с нименьшим номером, т. е. 1) и делаем ее текущей.

2^o. Если из текущей вершины нет непройденных (непомеченных) выходящих дуг, то алгоритм закончен: все дуги пройдены и контур составляем в порядке их прохождения.

3^o. Из текущей вершины выбираем любую непомеченную выходящую дугу (для определенности дугу, ведущую в вершину с наименьшим номером).

4^o. Если она не является *перешейком* (из текущей вершины есть еще непройденные выходящие дуги, а при временной пометке данной дуги из ее конечной вершины не существует пути до текущей вершины в орграфе непомеченных дуг), то выбираем ее, помечаем и делаем текущей конечную вершину этой дуги. Переходим к п. 2^o.

5^o. Если выбранная на предыдущем шаге дуга является *перешейком*, то выбираем следующую непомеченную выходящую дугу из текущей вершины, помечаем ее и переходим к п. 4^o.

Выполнение алгоритма сводим в следующую таблицу:

N	$\overline{1,5}$	$\overline{2,6}$	$\overline{3,1}$	$\overline{3,5}$	$\overline{4,6}$	$\overline{5,7}$	$\overline{5,8}$	$\overline{6,3}$	$\overline{6,8}$	$\overline{7,3}$	$\overline{8,2}$	$\overline{8,4}$	I	перш
0													1	
1	+												5	
2						+							7	
3										+			3	
4			−	+									5	1−
5							+						8	
6											+		2	
7		+											6	
8								−	+				8	3, 1−
9												+	4	
10					+								6	
11								+					3	
12			+										1	

Каждый столбец (кроме первого и последнего) отвечает дуге орграфа. Для пометки дуги ставим над ней черту. В первом столбце отмечаем номер шага выполнения алгоритма, а в последнем – номер текущей вершины. Если рассматриваемая дуга является перешейком, то на пересечении текущей строки выполнения алгоритма и столбца дуги ставим *минус*. В противном случае ставим *плюс*, помечаем дугу и заносим

новую текущую вершину. По окончании алгоритма столбец текущей вершины определит эйлеров контур. Отметим, что на шаге 4 выполнения алгоритма дуга (3,1) является перешейком, так как из вершины 1 нет уже выходящих непомеченных дуг, а потому нет пути до вершины 3 в графе непомеченных дуг. На шаге 8 выполнения алгоритма дуга (6,3) также является перешейком, так как из вершины 3 можно достигнуть только вершины 1 в графе непомеченных дуг, но никак не вершины 6.

В результате выполнения алгоритма получен следующий эйлеров контур:

$$(1, 5, 7, 3, 5, 8, 2, 6, 8, 4, 6, 3, 1).$$

14. Деревья

14.1. Определение деревьев и их свойства

Деревом *называется связный граф, не содержащий циклов.*

Примеры деревьев с числом вершин n от 1 до 4 изображены на рис. 1.

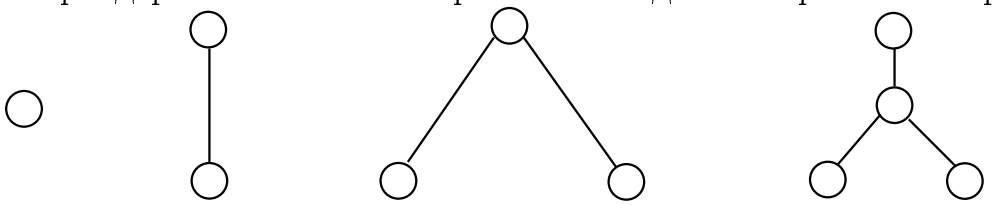

Рис. 19

Изобразим дерево как иерархическую схему следующим образом:

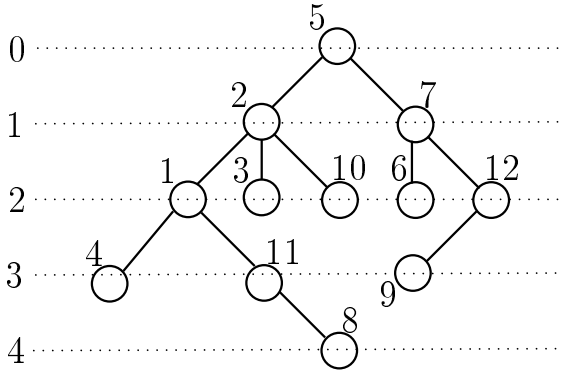

Рис. 20

1) все вершины располагаются на нескольких уровнях с номерами 0, 1,. . . ;

2) на уровне 0 (самом высоком) находится только 1 вершина – она называется *корнем* дерева;

3) на уровне 1 находятся все вершины, смежные с вершиной уровня 0;

4) на каждом следующем уровне с номером $k = 2, 3, \ldots$ находятся все вершины, смежные с вершинами уровня $k - 1$, которые не находятся уже на уровне $k - 2$;

5) каждое ребро соединяет вершины двух соседних уровней.

На рис. 20 приведен пример иерархической схемы для дерева D с 12 вершинами, заданного следующим списком ребер:
$\{\{1,2\},\{1,4\},\{1,11\},\{2,3\},\{2,5\},\{2,10\},\{5,7\},\{6,7\},\{7,12\},\{8,11\},$
$\{9,12\}\}$.

Отметим, что в качестве корня можно выбрать любую вершину дерева. Такое дерево с выделенной в качестве корня вершиной называется *корневым деревом*. В корневом дереве каждая вершина, кроме корня, смежна лишь с одной вершиной, находящейся на более высоком уровне (с номером на 1 меньше), которая называется *отцом* этой вершины. Для каждой вершины определяется единственный маршрут, связывающий ее с корнем дерева. Поэтому вершины корневого дерева, находящиеся на одном уровне, несмежны между собой (предположение противного приводит к циклу, который состоит из ребра между ними и маршрутов, связывающих эти вершины с корнем). Все вершины, имеющие одного и того же *отца*, называются его *сыновьями*. Вершины, имеющие одного и того же *отца*, называются *братьями*[1]. Вершины, не имеющие *сыновей*, называются *листьями*, или *висячими вершинами*. Для каждой вершины дерева x, не являющейся *листом*, часть графа, образованная этой вершиной и всеми вершинами, которые связаны с корнем через эту вершину, является *поддеревом*. В этом *поддереве* вершина x играет роль корня, а *поддеревья* вершины x, образованные *сыновьями* вершины x, называются *ветвями*, выходящими из вершины x. В целом термины *корень, ветви, листья* идут от схожести с деревом-растением, если перевернуть иерархическую схему.

Свойства деревьев:

[1] Иногда применяется терминология *мать, дочь, сестра*.

1. Маршрут, связывающий любые 2 вершины дерева, является простой цепью, так как в предположении, что есть еще один маршрут, мы приходим к циклу, которого не может существовать в дереве.

2. Количество n вершин и количество m ребер дерева связаны соотношением:

$$m = n - 1.$$

   Это равенство следует из того факта, что каждое ребро дерева связывает некоторую вершину дерева с ее *отцом*, и так как для каждой вершины дерева, кроме корня, отец единственный, то между множеством вершин без корня и множеством ребер устанавливается 1-1с.

3. Связный граф, имеющий $n-1$ ребро, является деревом. Действительно, в предположении, что этот связный граф не является деревом, он имеет цикл, и из этого цикла можно исключить, по крайней мере, 1 ребро так, что он останется связным. Будем исключать такие ребра до тех пор, пока все циклы не исчезнут, но граф останется связным. Тогда это будет дерево, имеющее число ребер меньше, чем $n-1$, что противоречит предыдущему свойству дерева.

4. Добавление любого ребра к дереву приводит к циклу с этим ребром, что также следует из свойства 2 (а также следует из свойства 1, так как появляется второй маршрут, связывающий концы добавленного ребра).

Для ориентированных корневых деревьев применяют 2 способа ориентации:
1) от *отцов* к *сыновьям* и
2) от *сыновей* к *отцам*.

14.2. Способы задания деревьев

Кроме описанных ранее способов задания графа (список ребер, матрица смежности вершин, матрица инцидентности вершин и ребер), применяют еще 2 специфических для деревьев способа:

1. *Список отцов* – для каждой вершины с номером $i = 1, \ldots, n$ на i-м месте в списке находится номер *отца* этой вершины, если она не является корнем дерева, или 0 для корня дерева.
   Так, в вышеописанном примере дерева, заданного списком ребер, список отцов будет выглядеть следующим образом:

$$(2, 5, 2, 1, 0, 7, 5, 11, 12, 2, 1, 7).$$

2. *Список сыновей и братьев.* Для этого все сыновья одного и того же отца упорядочиваются в список братьев (например, по номерам этих вершин), и самый первый в этом списке *сын* называется *старшим сыном* (например, с наименьшим номером), а для каждой вершины *списка братьев* следующая вершина называется *следующим братом.* Для каждой вершины *списка сыновей и братьев* с номером $i = 1, \ldots, n$ на i-м месте в списке находится пара – *старший сын* (вершина с наименьшим номером из сыновей) и *следующий брат* (вершина со следующим по порядку возрастания номером брата) для этой вершины. Если *сыновей* вершины нет, то записывается номер 0, а если нет *следующего брата*, то также записывается номер 0.
   Так, для вышеописанного примера дерева *список сыновей и братьев* будет выглядеть следующим образом:
   $((4, 3), (1, 7), (0, 10), (0, 11), (2, 0), (0, 12), (6, 0), (0, 0), (0, 0), (0, 0),$
   $(8, 0), (9, 0)).$

Список отцов удобен для алгоритмов, в которых идет движение по дереву снизу вверх, а *список сыновей и братьев* удобен в тех алгоритмах, в которых движение по дереву идет вниз и слева направо. В тех же случаях, когда движение может происходить в различных направлениях, удобно оба списка совместить в один *список отцов, сыновей и братьев.* Так, для вышеописанного примера такой *список отцов, сыновей и братьев* будет выглядеть следующим образом:
$((2, 4, 3), (5, 1, 7), (2, 0, 10), (1, 0, 11), (0, 2, 0), (7, 0, 12), (5, 6, 0), (11, 0, 0),$
$(12, 0, 0), (2, 0, 0), (1, 8, 0), (7, 9, 0)).$

Еще 1 способ относится к заданию специальных деревьев – бинарных деревьев, каждая вершина которых имеет не более 2 сыновей. В этом случае для каждой вершины задается список из номеров левого

и правого сына; если какого-то сына нет, то в качестве номера задается 0. Список сыновей корня в списке бинарного дерева находится на первом месте. Так, для бинарного дерева, заданного списком сыновей и братьев

$$((2,0),\ (4,3),\ (6,0),\ (8,5),\ (0,0),\ (0,7),\ (9,0),\ (0,0),\ (0,0))$$

список бинарного дерева будет выглядеть следующим образом:

$$((2,3),\ (4,5),\ (6,7),\ (8,0),\ (0,0),\ (0,0),\ (0,9),\ (0,0),\ (0,0)).$$

14.3. Приложения деревьев

Деревья очень часто используются в различных дискретных задачах. Но одни из самых важных в информатике приложений – это компиляция кода при трансляции выражений языков программирования, где используется *дерево выражений*, и информационный поиск, в котором важную роль играют *поисковые деревья*.

14.3.1. Дерево выражений

Рассмотрим в качестве примера арифметическое выражение

$$(a+b)*c+a*(b-d).$$

Его можно представить в виде дерева, где корнем дерева является последняя выполняемая операция, а 2 ветви, идущие из корня, – это 2 *поддерева* выражений операндов операции корня. Продолжая этот процесс, мы построим дерево выражения, в котором *листьями* будут терминальные символы a, b, c, d, а любая операция выражения – вершиной, не являющейся *листом* дерева. На рис. 21 изображено дерево выражения для вышеуказанного примера.

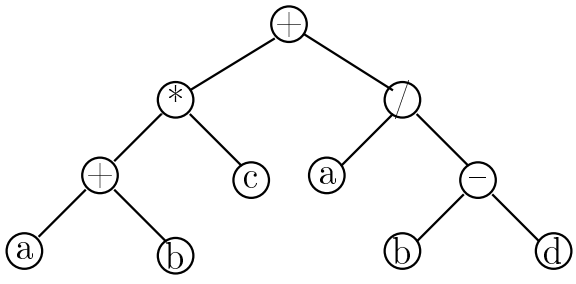

Рис. 21

Дерево выражения определяет все возможные способы вычисления выражения. При этом вычисления идут по дереву снизу вверх. Для определения значения каждой вершины сначала вычисляются значения ее сыновей-операндов, а затем берется от них функция, соответствующая операции в вершине $(+, -, *, /)$. Таким образом, одна из задач – обойти все вершины дерева, побывав в каждой вершине раньше, чем в ее отце.

Общепринятый способ записи выражений с бинарными операциями (например, арифметических выражений) называется *инфиксной* записью: операция записывается между операндами. Другой способ записи выражений, при котором операция идет после своих операндов, называется *постфиксной* записью. Она получается при обходе вершин дерева снизу вверх и слева направо. Так, для вышеописанного примера выражения *постфиксная* запись выражается следующим образом:

$$a\ b\ +\ c\ *\ a\ b\ d\ -\ /\ +\ .$$

Заметим, что постфиксная запись выражения является бесскобочной. Такой вид выражения непривычен и ненагляден. Однако вычисление значения выражения можно легко организовать с помощью стека при сканировании выражения слева направо:

1) если элемент выражения – не знак операции, то записываем его в стек;

2) если элемент выражения – знак операции, то из стека извлекаем 2 последних записанных операнда, выполняем операцию с этими операндами и результат записываем в стек.

Таким образом, для выражения сначала в стек будут записаны операнды a и b, затем при выполнении операции сложения эти операнды будут извлечены из стека, а в стек запишется $a + b$.

Затем в стек запишется c, и при выполнении следующей операции умножения из стека будут извлечены операнды c и $a + b$, а в стек запишется $(a + b) * c$.

Затем в стек будут записаны a, b, d, и при выполнении последующей операции вычитания из стека будут извлечены операнды d и b, а в стек запишется $b - d$.

Затем при выполнении следующей операции деления из стека будут извлечены операнды $b - d$ и a, а в стек запишется $a/(b - d)$.

Наконец, при выполнении последней операции сложения из стека будут извлечены операнды $a/(b-d)$ и $(a+b)*c$, а в стек запишется окончательный результат выражения $(a+b)*c+a*(b-d)$.

При компиляции программного кода выполняется подобный алгоритм, но вместо записи операндов в стек записываются их адреса, а вместо вычисления выражения производится генерация команды компьютера с соответствующими адресами операндов.

Для получения постфиксной записи выражения необходим специальный обход дерева выражения. Приведем алгоритм, использующий способ записи дерева в виде списка отцов, сыновей и братьев.

1^o. В качестве начального значения *текущей вершины* выбираем корень.

2^o. Если у *текущей вершины* нет сыновей, то выписываем ее и переходим к пункту 3^o, иначе – к пункту 5^o.

3^o. Если у *текущей вершины* есть следующий брат, то выбираем его в качестве *текущей вершины* и переходим к пункту 2^o; иначе переходим к пункту 4^o.

4^o. Если у *текущей вершины* нет отца, то выполнение алгоритма закончено; иначе в качестве *текущей вершины* выбираем отца, выписываем ее и переходим к пункту 3^o.

5^o. В качестве *текущей вершины* выбираем старшего сына и переходим к пункту 2^o.

14.3.2. Поисковое дерево

Информация для быстрого поиска часто организуется в специальное поисковое бинарное дерево, в котором каждая вершина указывает на информацию и снабжается специальной записью (число или строка символов), называемой *ключом вершины*. При поиске информации задается *поисковый ключ* и ищется вершина, значение ключа которой совпадает со значением поискового ключа. Для этого выполняется следующий алгоритм:

1. В качестве текущей вершины выбирается корень дерева.

2. Значение поискового ключа сравнивается с ключом текущей вершины, и если они совпадают, то поиск успешно завершен.

3. Если значение поискового ключа меньше, то в качестве текущей вершины выбирается левый сын, если он есть, и после этого переход на пункт 2; если его нет – то поиск завершен неуспешно.

4. Если значение поискового ключа больше, то в качестве текущей вершины выбирается правый сын, если он есть, и после этого переход на пункт 2; если его нет – то поиск завершен неуспешно.

Для выполнения такого алгоритма в списке бинарного дерева каждый элемент помимо списка сыновей имеет ключ, находящийся на первом месте.
Пусть, например, имеется массив ключей $\{1, 3, 4, 7, 12, 13, 15, 19, 20\}$. Зададим бинарное поисковое дерево следующим списком (рис. 22):
($(12, 2, 3)$, $(4, 4, 5)$, $(15, 6, 7)$, $(3, 8, 0)$, $(7, 0, 0)$, $(1, 0, 0)$, $(19, 0, 9)$, $(1, 0, 0)$, $(20, 0, 0)$).

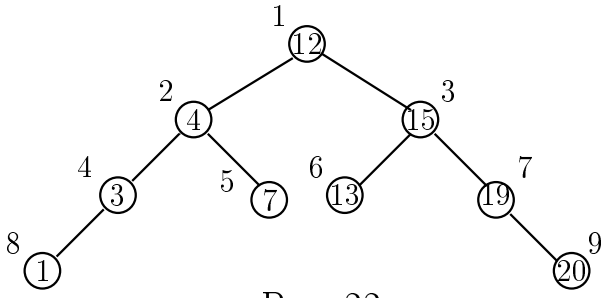

Рис. 22

При поиске ключа 7 он будет последовательно сравниваться с ключами 12, 4 и 7, а поиск закончится успешно. При поиске ключа 14 он будет последовательно сравниваться с ключами 12, 15, 13, а поиск закончится неуспешно. Но в этом случае может быть добавлена новая вершина в поисковое дерево как правый сын вершины с последним при сравнении ключом 13.

В качестве поисковых используются не обязательно бинарные деревья. В этом случае с вершиной, имеющей m сыновей, связывается упорядоченная по возрастанию последовательность из $m - 1$ ключа и происходит последовательное сравнение поискового ключа с этими ключами вершины до тех пор, пока поисковый ключ не попадет в один

из m интервалов, на которые этими ключами разбиваются все возможные значения. После этого происходит переход к сыну вершины по порядку, соответствующему найденному интервалу ключей.

14.4. Задача о кратчайшем остовном дереве

Рассмотрим следующий пример. Необходимо спроектировать кратчайшую телефонную сеть, связывающую n абонентов. В этом случае необходимо из полного графа K_n с заданными длинами ребер выбрать связный остовной подграф, имеющий наименьшую суммарную длину своих ребер. Этот подграф обязательно является деревом, так как иначе есть цикл и, следовательно, можно удалить любое ребро этого цикла, и при этом граф останется связным и остовным, но суммарная длина его ребер станет меньше. Таким образом, кратчайший остовной подграф есть дерево, но дерево, имеющее наименьшую длину.

Постановка задачи. *Дан связный граф G(X,U). Необходимо выделить его кратчайший остовной подграф D(X,V), т. е. такой, что*

$$l(V) = \sum_{i \in V} l_i$$

минимальна для всех связных остовных подграфов.

Для решения этой задачи есть несколько алгоритмов. Но наиболее простым и широко используемым является алгоритм Краскала[2].

1. В качестве начального подграфа D берем пустой подграф $D(X, \emptyset)$ и упорядочиваем множество ребер U графа $G(X, U)$ по возрастанию длин.

2. Добавляем к D кратчайшее ребро из U, не приводящее к циклу (при добавлении ребра 2 компоненты связности, которым принадлежат концы ребра, сливаются в одну; если же оба конца ребра принадлежат одной компоненте связности, то добавление такого ребра приводит к циклу). Добавленное ребро или пропущенные при этом ребра помечаем как вычеркнутые.

3. Повторяем пункт 2 до тех пор, пока число добавленных ребер V не станет равным $n - 1$. Полученный граф $D(X, V)$ и есть кратчайшее остовное дерево.

[2] Алгоритм Прима является более эффективным по трудоемкости, но более сложным.

Обоснование. Покажем сначала, что полученый при выполнении алгоритма граф $D(X, V)$ является деревом и что это дерево кратчайшее.

1. Предположим, что граф $D(X, V)$ не является связным. Тогда он состоит из k $(k > 1)$ компонент связности, каждая из которых не имеет циклов, а потому является деревом. Пусть k_1 $(k_1 \leq k)$ компонент связности имеют ребра. Без ограничения общности можно считать, что это первые k_1 компонент связности. $k_1 > 1$, так как в противном случае получаем противоречие: если $k_1 = 1$, то эта компонента содержит $n-1$ ребро по построению и это меньше, чем n вершин, а, значит, не может быть деревом. Число m ребер графа D равно

$$n - 1 = m = \sum_{i=1}^{k} m_i = \sum_{i=1}^{k_1} (n_i - 1) \leq n - k_1 < n - 1.$$

Полученное противоречие показывает, что D является связным графом. D не имеет циклов и, следовательно, является деревом.

2. Допустим, что D не является кратчайшим деревом. Пусть сначала все ребра имеют различную длину. Пусть $D_1(X, V_1)$ является кратчайшим деревом.

Упорядочим ребра множества V_1 по возрастанию длин и перенумеруем их. Также поступим с ребрами множества V. Пусть v_i^1 – первое ребро дерева D_1, не совпадающее с соответствующим ребром v_i. Заметим, что $l(v_i) < l(v_i^1)$. Добавим ребро v_i к дереву D_1. В этом графе, назовем его G_1, появится единственный цикл. Он содержит добавленное ребро и, по крайней мере, одно ребро v_j^1 с номером $j \geq i$, так как предыдущие ребра принадлежали и дереву D, не содержащему циклов.

Удалим теперь из графа G_1 ребро v_j^1. Граф, назовем его D_2, при этом останется связным и не будет содержать циклов, т. е. D_2 является деревом. При этом $l(D_2) < l(D_1)$, что противоречит тому, что D_1 является кратчайшим деревом. Это противоречие доказывает, что дерево D, полученное алгоритмом, является кратчайшим.

Пусть теперь ребра из множества U могут иметь одинаковую длину. Обозначим $\delta_i = l(u_{i+1}) - l(u_i)$ $(i \in \overline{1, m-1})$, $\delta_0 = \min\limits_{\delta_i > 0} \delta_i$. Положим $\varepsilon = \min\{0.5, \delta_0/2\}$. Изменим теперь длины ребер этого множества, увеличив в упорядоченной последовательности длину i-го ребра на величину ε^{m-i}. При этом порядок ребер в упорядоченной по их длинам

последовательности не изменится, но все ребра будут иметь различную длину, а граф G будет иметь единственное кратчайшее дерево D. Устремим $\varepsilon \to 0$. Порядок ребер не будет меняться, и дерево D будет оставаться кратчайшим. Поэтому в пределе мы получим, что D является кратчайшим деревом. Заметим, что в случае одинаковости длин ребер кратчайшее остовное дерево может не быть единственным.

15. Пример задачи на построение кратчайшего остовного дерева с ее решением

Построить кратчайшее остовное дерево для графа, заданного списком ребер с их длинами:
(({1,2}, 6), ({1,3}, 3), ({1,4}, 9), ({1,5}, 8), ({2,3}, 1), ({2,5}, 2),
({2,6}, 2), ({3,7}, 4), ({4,5}, 2), ({4,8}, 1), ({5,9}, 7), ({6,7}, 2),
({7,8}, 3), ({8,9}, 2)).

А. Дадим пошаговое описание алгоритма Краскала с описанием таблицы выполнения:

1^o. Сортируем множество ребер графа по возрастанию длин ребер; в таблице выполнения выделяем первую колонку, в которую выписываем отсортированные ребра вместе с их длиной.

2^o. В порядке возрастания длин ребер выбираем ребра, не приводящие к циклу с уже выбранными ребрами (в колонке выбора N ставим по порядку номер выбора от 1 до $n-1$, если рассматриваемое ребро не приводит к циклу с уже выбранными, и знак "–" в противном случае). Для этого заводим несколько колонок (не большее чем $k = n/2$) для компонент связности графа выбранных ребер $C_1, C_2, \ldots$ и проверяем, принадлежат ли вершины выбранного ребра уже введенным компонентам связности:

1) если не принадлежат ни одной введенной компоненте связности, то выбираем это ребро (в колонке выбора ставим очередной номер выбора), заводим новую компоненту связности в следующей свободной колонке и включаем в нее вершины этого ребра;

2) если одна из вершин принадлежит одной компоненте связности (находится в колонке компоненты связности), а другая принадлежит другой компоненте связности, то выбираем это ребро (в колонке выбора ставим очередной номер выбора) и объединяем компоненты связности (переносим вершины второй компоненты связности в первую, вычеркивая их аккуратно из второй компоненты);

3) если обе вершины ребра принадлежат одной компоненте связности, то включение этого ребра приводит к циклу; поэтому это ребро не включаем (в колонку выбора ставим знак "–").

3^o. Как только $n-1$ ребро будет выбрано, прекращаем выполнение алгоритма: ребра, помеченные номерами в колонке выбора N, составляют кратчайшее остовное дерево, а сумма их длин – длину кратчайшего остовного дерева.

В. Выполнение алгоритма:

$\{a,b\},l$	N	C_1	C_2	C_3	C_4
$\{2,3\},1$	1	$2,3$			
$\{4,8\},1$	2		$\not{A}$, $\not{8}$		
$\{2,5\},2$	3	5			
$\{2,6\},2$	4	6			
$\{4,5\},2$	5	$4,8$			
$\{6,7\},2$	6	7			
$\{8,9\},2$	7	9			
$\{1,3\},3$	8	1			
$\{7,8\},3$	–				
$\{3,7\},4$					
$\{1,2\},6$					
$\{5,9\},7$					
$\{1,5\},8$					
$\{1,4\},9$					

Длина полученного дерева равна 15.

C. Реализация графа с выделением кратчайшего остовного дерева:

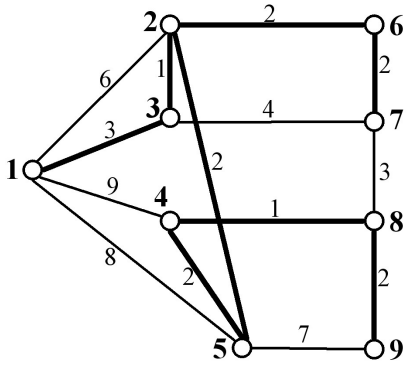

Рис. 23

D. Таблица списка отцов, сыновей и братьев

N	F	S	B
1	0	3	0
2	3	5	0
3	1	2	0
4	5	8	0
5	2	4	6
6	2	7	0
7	6	0	0
8	4	9	0
9	8	0	0

Построенная реализация дерева по этому списку отцов, сыновей и братьев приведена на рис. 10.

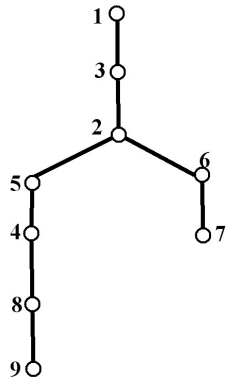

Рис. 24

16. Индивидуальное задание 9

16.1. Общее задание

1. Для графа, заданного списком ребер с их длинами, найти кратчайший маршрут из первой вершины графа (с номером 1) в последнюю вершину (с наибольшим номером):

a) описать алгоритм Дейкстры нахождения кратчайшего маршрута в графе с пояснением всех вводимых в алгоритме обозначений;

b) свести выполнение алгоритма для заданного графа в таблицу;

c) построить реализацию графа с указанием длин ребер и выделением жирными ребрами найденного кратчайшего маршрута.

2. Для графа (или орграфа), заданного матрицей смежности вершин, найти эйлеров маршрут (цикл, контур, цепь, путь):

a) анализировать граф на существование эйлерова маршрута (цикла, контура, цепи, пути), сформулировав соответствующий критерий существования;

b) описать алгоритм нахождения эйлерова маршрута (цикла или контура, цепи, пути);

c) свести выполнение алгоритма для заданного графа в таблицу;

d) выписать полученный маршрут в качестве результата.

3. Для графа задачи 1 найти кратчайшее остовное дерево:

a) описать алгоритм Краскала нахождения кратчайшего остовного дерева с пояснением всех вводимых в алгоритме обозначений;

b) свести выполнение алгоритма для заданного графа в таблицу;

c) построить реализацию графа с указанием длин ребер и выделением жирными ребрами найденного кратчайшего остовного дерева;

d) построить таблицу списка отцов, сыновей и братьев для описания полученного дерева и проверить полученное описание построением по нему дерева.

Отметим, что во всех задачах при неоднозначности выбора вершины (например, для корня, для старшего сына, для следующего брата) всегда следует выбирать вершину с наименьшим номером. Также при неоднозначности выбора ребра или дуги всегда следует выбирать первое в лексикографическом порядке упорядочения соответственно ребро или дугу.

16.2. Методические рекомендации для задания 9

Задача 1

1. В таблице прокрутки алгоритма порядок ребер лексикографический (сначала ребра упорядочиваются по меньшему концу ребра, а при равных номерах вершин таких концов по номеру другого конца ребра).

2. При неоднозначности выбора очередного ребра с наименьшим значением λ берется первое ребро в этом порядке. **Типичной ошибкой** является **нарушение этого порядка**.

3. При пересчете характеристики λ для ребер эта характеристика изменяется только при уменьшении ее значения. Игнорирование этого также является **типичной ошибкой**.

Задача 2

1. Прежде всего необходимо проверить связность графа или слабую связность для орграфа. Для этого используем волновой алгоритм: в пустой список включаем вершину 1; затем к ней добавляем смежные с ней вершины, не вошедшие еще в список; затем добавляем вершины, смежные к только что добавленным и не вошедшие еще в список; и т. д., пока вершины не перестанут добавляться. Если в списке все вершины графа, то он связен (слабо связен), а в противном случае связности нет и эйлерова маршрута также.

2. Подсчитываем степени вершин (полустепени исхода и захода для орграфа). Если все степени четны (полустепени исхода равны полустепеням захода), то есть эйлеров цикл (эйлеров контур для орграфа). Если есть только одна пара вершин, для которых степени

нечетны (полустепени исхода и захода отличаются на 1), то есть эйлерова цепь (путь для орграфа).

3. При организации таблицы прокрутки алгоритма эйлерова маршрута ребра неориентированного графа выписываются в лексикографическом порядке, а для орграфа дуги упорядочиваются по началу дуги, и при одиниковых вершинах начала дуг по их концам.

4. В качестве начальной вершины замкнутого эйлерова маршрута выбирается вершина 1, а при незамкнутом маршруте выбирается вершина с меньшим номером для эйлеровой цепи или вершина с полустепенью исхода большей, чем полустепень захода для эйлерова пути.

5. При каждом выборе ребра (дуги) из текущей вершины проверяется, не является ли оно *перешейком*. Для этого в строке шага и в колонке ребра (дуги), подозреваемого на перешеек, делается пометка знаком "–", а в колонке "Перш" (перешеек) выписывются поочередно в список: вершина другого конца ребра (конца дуги); другие вершины непройденных ребер (дуг), идущие из этой вершины; вершины непройденных ребер (дуг), идущие из других концов ребер (дуг) только что добавленных вершин в список; и т. д., пока в список не войдет начальная вершина ребра (дуги), подозреваемого на перешеек, или процесс добавления вершин не закончится.

   В 1-м случае подозреваемое ребро (дуга) не является перешейком, и тогда знак "–" в этой строке заменяется на знак "+", в колонке I записывается вершина другого конца ребра (дуги) и определяется следующая строка с выбором ребра (дуги) с началом в этой вершине.

   Во втором случае, если процесс добавления вершин закончился в конечной вершине эйлерова маршрута, также подозреваемое ребро (дуга) не является перешейком и делается то же, что и в предыдущем абзаце.

   Иначе подозреваемое ребро (дуга) является перешейком, и в этом

случае надо перейти к выбору следующего по порядку ребра (дуги) с проверкой на перешеек.

6. **Типичными ошибками** являются **отсутствие исследования на перешеек** или **неаккуратность** при таком исследовании.

Задача 3

1. В таблице прокрутки алгоритма ребра упорядочиваются по длине, а при одной длине – в лексикографическом порядке. Несоблюдение этого является **типичной ошибкой**.

2. Ребра рассматриваются по порядку с попыткой добавления их (их вершин) в основную компоненту C^1 связности графа. Если компонента еще пустая или только одна вершина содержится в этой компоненте, то добавление компоненты (обеих вершин или одной, еще не присутствовавшей) является успешным.

3. Если компонента не пустая и ни одна вершина ребра не входит в эту компоненту, то делается попытка вставки вершин ребра в следующую компоненту.

4. Если компонента содержит обе вершины ребра (вставка приведет к циклу), то вставка ребра отменяется.

5. Если одна из вершин ребра принадлежит одной компоненте, а другая – другой, то обе компоненты сливаются в компоненту с меньшим номером.

6. После $n-1$ шагов добавления ребер в копоненте C^1 иолучим все вершины и ребра дерева.

7. **Типичной ошибкой** является **несоблюдение алгоритма**.

8. При построении списка отцов, сыновей и братьев необходимо при неоднозначности выбора всегда выбирать вершину с наименьшим номером: в качестве корня – 1, в качестве старшего сына вершины – наименьший номер из ее сыновей, в качестве следующего брата – наименьший следующий номер из сыновей отца этой вершины. Отсутствие отца, старшего сына или следующего брата обозаначется номером 0.

16.3. Варианты индивидуального задания 9

Задачи 1, 3

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

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

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

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

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

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

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

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

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

10. (({1,4}, 4), ({1,6}, 4), ({1,7}, 2) ({2,3}, 1), ({2,5}, 1), ({2,6}, 2), ({2,8}, 4), ({3,4}, 3), ({3,7}, 5), ({3,8}, 2), ({4,5}, 2), ({5,6}, 3), ({5,8}, 3), ({6,7}, 1)).

11. (({1,2}, 4), ({1,3}, 1), ({1,5}, 3), ({2,3}, 2), ({2,4}, 1), ({2,7}, 4), ({3,4}, 4), ({4,7}, 2), ({4,8}, 6), ({5,6}, 5), ({5,7}, 4), ({6,7}, 1), ({6,8}, 2), ({7,8}, 4)).

12. (({1,5}, 4), ({1,6}, 2), ({1,7}, 4) ({2,3}, 1), ({2,5}, 3), ({2,6}, 5), ({2,8}, 2), ({3,4}, 1), ({3,7}, 2), ({3,8}, 4), ({4,5}, 2), ({4,7}, 3), ({4,8}, 3), ({6,7}, 1)).

13. (({1,3}, 4), ({1,4}, 3), ({1,6}, 2), ({2,3}, 2), ({2,4}, 3), ({2,6}, 4), ({2,7}, 3), ({2,8}, 6), ({3,4}, 1), ({3,6}, 1), ({4,7}, 6), ({5,2}, 2), ({5,6}, 2), ({5,8}, 7), ({7,8}, 2)).

14. (({1,5}, 2), ({1,6}, 6), ({1,7}, 6), ({1,8}, 5), ({2,6}, 2), ({2,9}, 1), ({3,6}, 1), ({3,7}, 2), ({3,9}, 2), ({4,7}, 1), ({4,9}, 2), ({5,8}, 2), ({5,9}, 7), ({6,7}, 1), ({7,8}, 1)).

15. (({1,4}, 2), ({1,6}, 5) ({1,7}, 6), ({1,8}, 6), ({2,7}, 2), ({2,8}, 1), ({2,9}, 2), ({3,8}, 2), ({3,9}, 1), ({4,6}, 2), ({4,9}, 7), ({5,7}, 1), ({5,9}, 2), ({6,7}, 1), ({7,8}, 1)).

16. (({1,2}, 4), ({1,3}, 2), ({1,4}, 3), ({1,5}, 3), ({2,3}, 1), ({2,6}, 2), ({2,7}, 3), ({3,7}, 4), ({4,6}, 3), ({5,8}, 4), ({6,8}, 1), ({6,9}, 4) ({7,9}, 3), ({8,9}, 2)).

17. (({1,6}, 4), ({1,7}, 2) ({1,8}, 6), ({2,7}, 4), ({2,9}, 3), ({3,6}, 3), ({3,8}, 1), ({3,9}, 2), ({4,7}, 3), ({4,9}, 4), ({5,7}, 1), ({5,8}, 2), ({5,9}, 6)).

18. (({1,2}, 4), ({1,3}, 2), ({1,5}, 3), ({2,3}, 1), ({2,5}, 1), ({2,7}, 2), ({3,4}, 2), ({3,7}, 4), ({4,7}, 2), ({4,8}, 7), ({5,6}, 6), ({5,7}, 3), ({6,7}, 3), ({6,8}, 2), ({7,8}, 6)).

19. (({1,2}, 1), ({1,3}, 4), ({1,4}, 6), ({2,5}, 1), ({2,6}, 4), ({3,4}, 2), ({3,5}, 1), ({4,7}, 2), ({4,8}, 4), ({5,8}, 7), ({6,8}, 4), ({7,8}, 1)).

20. (({1,2}, 7), ({1,3}, 6), ({1,4}, 2), ({2,3}, 2), ({2,5}, 2), ({3,4}, 3), ({3,5}, 4), ({3,6}, 2), ({3,7}, 3), ({4,7}, 6), ({5,6}, 1), ({5,8}, 2), ({6,7}, 1), ({6,8}, 4), ({7,8}, 3)).

21. (({1,2}, 1), ({1,3}, 4), ({1,4}, 3), ({2,3}, 2), ({2,5}, 4), ({3,5}, 1), ({3,6}, 4), ({4,6}, 4), ({4,7}, 5), ({5,6}, 2), ({5,8}, 6), ({6,7}, 1), ({6,8}, 4), ({7,8}, 2)).

22. (({1,2}, 3), ({1,3}, 2) ({1,4}, 4), ({2,4}, 1), ({2,5}, 5), ({3,4}, 1), ({3,6}, 4), ({4,5}, 4), ({4,6}, 2), ({4,7}, 4), ({5,7}, 1), ({5,8}, 2), ({6,7}, 1), ({6,8}, 4), ({7,8}, 2)).

23. (({1,2}, 4), ({1,3}, 2), ({1,5}, 3), ({2,3}, 1), ({2,5}, 1), ({2,6}, 2), ({3,4}, 3), ({3,7}, 5), ({4,5}, 2), ({4,8}, 4), ({5,6}, 3), ({6,7}, 1), ({6,8}, 4), ({7,8}, 2)).

24. (({1,2}, 2), ({1,3}, 4), ({1,4}, 3), ({1,5}, 3), ({2,3}, 1), ({2,6}, 4), ({3,6}, 3), ({3,7}, 2), ({4,8}, 4), ({5,7}, 3), ({6,9}, 3), ({7,8}, 1), ({7,9}, 4) ({8,9}, 2)).

25. (({1,5}, 6), ({1,6}, 2), ({1,7}, 5), ({2,3}, 4), ({2,4}, 1), ({2,5}, 2), ({2,8}, 2), ({3,5}, 2), ({3,6}, 1), ({3,8}, 6), ({4,5}, 1), ({4,7}, 1), ({4,8}, 2), ({5,6}, 4), ({5,7}, 1)).

26. (({1,2}, 3), ({1,5}, 2) ({1,7}, 4) ({2,4}, 3), ({2,6}, 4), ({3,5}, 4), ({3,9}, 3), ({4,6}, 1), ({4,7}, 2), ({4,9}, 4), ({5,7}, 1), ({6,9}, 2), ({7,8}, 3), ({8,9}, 3)).

27. (({1,2}, 4), ({1,3}, 3), ({1,5}, 2), ({2,3}, 1), ({2,4}, 2), ({2,5}, 1), ({2,6}, 2), ({2,7}, 4), ({3,4}, 3), ({4,7}, 1), ({4,8}, 4), ({5,6}, 3), ({6,7}, 2), ({6,8}, 4), ({7,8}, 2)).

28. (({1,2}, 4), ({1,3}, 2), ({1,4}, 6), ({2,3}, 1), ({2,4}, 2), ({2,5}, 4), ({2,7}, 4), ({3,5}, 5), ({4,6}, 4), ({4,7}, 1), ({5,8}, 3), ({6,7}, 2), ({6,8}, 1), ({7,8}, 4)).

29. (({1,2}, 2), ({1,3}, 3), ({1,4}, 6), ({1,5}, 4), ({2,7}, 3), ({2,8}, 1), ({3,7}, 4), ({4,6}, 1), ({4,8}, 2), ({5,6}, 3), ({6,9}, 2) ({7,9}, 4), ({8,9}, 6)).

30. (({1,5}, 3), ({1,6}, 1), ({1,7}, 4), ({2,3}, 1), ({2,4}, 2), ({2,5}, 4), ({2,7}, 4), ({2,8}, 4), ({3,5}, 5), ({3,8}, 2), ({4,6}, 4), ({4,7}, 1), ({4,8}, 6), ({6,7}, 2)).

31. (({1,2}, 2), ({1,4}, 3), ({1,6}, 4), ({2,5}, 3), ({2,6}, 1), ({3,5}, 2), ({3,6}, 4), ({3,7}, 1), ({3,8}, 2), ({4,6}, 1), ({4,7}, 3), ({5,6}, 2), ({5,8}, 4), ({6,7}, 2), ({7,8}, 4)).

32. (({1,2}, 2), ({1,3}, 1), ({1,4}, 7), ({1,5}, 2), ({2,7}, 2), ({2,8}, 1), ({3,8}, 2), ({4,6}, 2), ({4,9}, 2), ({5,7}, 1), ({6,7}, 1), ({6,9}, 5) ({7,8}, 1), ({7,9}, 6), ({8,9}, 6)).

33. (({1,6}, 3), ({1,7}, 4) ({1,8}, 2), ({2,3}, 1), ({2,6}, 4), ({2,9}, 2), ({3,6}, 3), ({3,7}, 2), ({3,9}, 4), ({4,8}, 4), ({4,9}, 3), ({5,7}, 3), ({5,9}, 3), ({7,8}, 1)).

34. (({1,2}, 3), ({1,3}, 2), ({1,4}, 4), ({1,5}, 6), ({2,7}, 4), ({3,6}, 3), ({3,8}, 1), ({4,7}, 3), ({5,7}, 1), ({5,8}, 2), ({6,9}, 4), ({7,9}, 2) ({8,9}, 6)).

35. (({1,5}, 6), ({1,6}, 2), ({1,7}, 2), ({2,3}, 4), ({2,5}, 1), ({2,8}, 2), ({3,4}, 1), ({3,5}, 2), ({3,6}, 2), ({3,7}, 1), ({3,8}, 6), ({4,7}, 1), ({4,8}, 5), ({5,6}, 4), ({6,7}, 1)).

36. (({1,2}, 2), ({1,3}, 4), ({1,4}, 3), ({2,3}, 1), ({2,5}, 2), ({2,6}, 4), ({3,4}, 1), ({3,6}, 2), ({4,6}, 3), ({4,7}, 6), ({5,6}, 2), ({5,8}, 7), ({6,7}, 3), ({6,8}, 6), ({7,8}, 2)).

37. (({1,5}, 3), ({1,6}, 4), ({1,7}, 2), ({2,3}, 1), ({2,5}, 5), ({2,6}, 4), ({2,8}, 2), ({3,4}, 1), ({3,6}, 4), ({3,8}, 2), ({4,6}, 2), ({4,7}, 4), ({4,8}, 4), ({5,6}, 1), ({6,7}, 1)).

38. (({1,2}, 4), ({1,3}, 2), ({1,5}, 3), ({2,3}, 1), ({2,5}, 1), ({2,6}, 2), ({3,4}, 3), ({3,7}, 5), ({4,5}, 2), ({4,8}, 4), ({5,6}, 3), ({6,7}, 1), ({6,8}, 4), ({7,8}, 2)).

39. (({1,2}, 2), ({1,3}, 2), ({1,4}, 4), ({2,3}, 1), ({2,5}, 5), ({2,6}, 4), ({3,4}, 1), ({3,6}, 4), ({4,6}, 2), ({4,7}, 4), ({5,6}, 1), ({5,8}, 3), ({6,7}, 1), ({6,8}, 4), ({7,8}, 2)).

40. (({1,3}, 3), ({1,4}, 4), ({1,6}, 2), ({1,8}, 3), ({2,6}, 4), ({2,4}, 3), ({2,9}, 3), ({3,5}, 4), ({4,6}, 1), ({4,7}, 2), ({5,7}, 1), ({5,9}, 2) ({7,8}, 3), ({7,9}, 4)).

41. (({1,2}, 1), ({1,3}, 2), ({1,4}, 2), ({1,5}, 7), ({2,6}, 2), ({3,6}, 1), ({3,7}, 2), ({4,7}, 1), ({5,8}, 2), ({5,9}, 2), ({6,7}, 1), ({6,9}, 6), ({7,8}, 1) ({7,9}, 6), ({8,9}, 5)).

42. (({1,2}, 1), ({1,3}, 4), ({1,4}, 7), ({1,5}, 4), ({2,5}, 2), ({3,7}, 4), ({4,6}, 1), ({4,7}, 1), ({5,6}, 2), ({5,8}, 6), ({6,8}, 4), ({7,8}, 1)).

43. (({1,5}, 2), ({1,6}, 4), ({1,7}, 3), ({2,3}, 2), ({2,5}, 2), ({2,8}, 7), ({3,4}, 3), ({3,5}, 4), ({3,6}, 2), ({3,7}, 3), ({3,8}, 6), ({4,7}, 6), ({4,8}, 2), ({5,6}, 1), ({6,7}, 1)).

44. (({1,5}, 6), ({1,6}, 4), ({1,7}, 1), ({2,5}, 2), ({2,8}, 1), ({3,7}, 4), ({3,8}, 4), ({4,6}, 1), ({4,7}, 1), ({4,8}, 7), ({5,6}, 2), ({5,8}, 4)).

45. (({1,2}, 3), ({1,3}, 4), ({1,4}, 2), ({2,3}, 1), ({2,5}, 3), ({3,4}, 1), ({3,5}, 2), ({3,6}, 4), ({3,7}, 2), ({4,7}, 3), ({5,6}, 1), ({5,8}, 4), ({6,7}, 2), ({6,8}, 2), ({7,8}, 4)).

46. (({1,5}, 4), ({1,6}, 2), ({1,7}, 4), ({2,3}, 1), ({2,5}, 3), ({2,8}, 3), ({3,4}, 1), ({3,5}, 2), ({3,6}, 4), ({3,7}, 2), ({3,8}, 4), ({4,7}, 3), ({4,8}, 2), ({5,6}, 1), ({6,7}, 2)).

47. (({1,2}, 3), ({1,3}, 4), ({1,4}, 2), ({2,3}, 1), ({2,5}, 3), ({3,4}, 1), ({3,5}, 2), ({3,6}, 4), ({3,7}, 2), ({4,7}, 3), ({5,6}, 1), ({5,8}, 4), ({6,7}, 2), ({6,8}, 2), ({7,8}, 4)).

48. (({1,5}, 2), ({1,6}, 4), ({1,7}, 2), ({2,4}, 1), ({2,5}, 5), ({2,8}, 3), ({3,4}, 1), ({3,6}, 4), ({3,8}, 2) ({4,5}, 4), ({4,6}, 2), ({4,7}, 4), ({4,8}, 4), ({5,7}, 1), ({6,7}, 1)).

49. (({1,2}, 2), ({1,3}, 4), ({1,4}, 3), ({2,3}, 1), ({2,5}, 3), ({2,6}, 5), ({3,4}, 1), ({3,7}, 2), ({4,5}, 2), ({4,7}, 3), ({5,8}, 4), ({6,7}, 1), ({6,8}, 2), ({7,8}, 4)).

50. (({1,2}, 2), ({1,3}, 6), ({1,4}, 2), ({2,3}, 4), ({2,4}, 1), ({2,5}, 2), ({3,5}, 2), ({3,6}, 1), ({4,5}, 1), ({4,7}, 1), ({5,6}, 4), ({5,7}, 1), ({5,8}, 6), ({6,8}, 2), ({7,8}, 5)).

51. (({1,4}, 4), ({1,5}, 7), ({1,6}, 4), ({1,7}, 1), ({2,5}, 1), ({2,6}, 4), ({2,8}, 1), ({3,4}, 2), ({3,5}, 1), ({3,8}, 4), ({4,7}, 2), ({4,8}, 6)).

52. (({1,5}, 7), ({1,6}, 6), ({1,7}, 2), ({2,3}, 1), ({2,5}, 2), ({2,6}, 4), ({2,8}, 2), ({3,4}, 1), ({3,6}, 2), ({3,8}, 4), ({4,6}, 3), ({4,7}, 6), ({4,8}, 3), ({5,6}, 2), ({6,7}, 3)).

53. (({1,2}, 4), ({1,3}, 1), ({1,4}, 4), ({1,5}, 7), ({2,6}, 4), ({3,4}, 2), ({4,7}, 2), ({4,8}, 6), ({5,6}, 1), ({5,7}, 1), ({6,8}, 1), ({7,8}, 4)).

54. (({1,2}, 2), ({1,3}, 6), ({1,4}, 5), ({2,3}, 4), ({2,5}, 1), ({3,4}, 1), ({3,5}, 2), ({3,6}, 2), ({3,7}, 1), ({4,7}, 1), ({5,6}, 4), ({5,8}, 6), ({6,7}, 1), ({6,8}, 2), ({7,8}, 2)).

55. (({1,2}, 2), ({1,3}, 1), ({1,4}, 7), ({1,5}, 2), ({2,6}, 2), ({2,7}, 1), ({3,7}, 2), ({4,8}, 2), ({4,9}, 2), ({5,6}, 1), ({6,7}, 1), ({6,8}, 1) ({6,9}, 6), ({7,9}, 6), ({8,9}, 5)).

56. (({1,6}, 6), ({1,7}, 2) ({1,8}, 4), ({2,7}, 3), ({2,9}, 4), ({3,6}, 2), ({3,7}, 1), ({3,9}, 6), ({4,7}, 4), ({4,9}, 3), ({5,6}, 1), ({5,8}, 3), ({5,9}, 2)).

57. (({1,2}, 2), ({1,3}, 2), ({1,5}, 4), ({2,3}, 1), ({2,5}, 1), ({2,7}, 4), ({3,4}, 5), ({3,7}, 4), ({4,7}, 1), ({4,8}, 3), ({5,6}, 4), ({5,7}, 2), ({6,7}, 1), ({6,8}, 2), ({7,8}, 4)).

58. (({1,5}, 6), ({1,6}, 4), ({1,7}, 2), ({2,3}, 2), ({2,5}, 4), ({2,8}, 1), ({3,5}, 1), ({3,6}, 4), ({3,8}, 4), ({4,6}, 4), ({4,7}, 5), ({4,8}, 3), ({5,6}, 2), ({6,7}, 1)).

59. (({1,2}, 6), ({1,3}, 2), ({1,5}, 5), ({2,3}, 4), ({2,4}, 2), ({2,5}, 1), ({2,6}, 1), ({2,7}, 2), ({3,4}, 1), ({4,7}, 4), ({4,8}, 6), ({5,6}, 1), ({6,7}, 1), ({6,8}, 2), ({7,8}, 2)).

60. (({1,4}, 4), ({1,6}, 4), ({1,7}, 2) ({2,3}, 1), ({2,5}, 1), ({2,6}, 2), ({2,8}, 4), ({3,4}, 3), ({3,7}, 5), ({3,8}, 2), ({4,5}, 2), ({5,6}, 3), ({5,8}, 3), ({6,7}, 1)).

61. (({1,2}, 4), ({1,3}, 1), ({1,5}, 3), ({2,3}, 2), ({2,4}, 1), ({2,7}, 4), ({3,4}, 4), ({4,7}, 2), ({4,8}, 6), ({5,6}, 5), ({5,7}, 4), ({6,7}, 1), ({6,8}, 2), ({7,8}, 4)).

62. (({1,5}, 4), ({1,6}, 2), ({1,7}, 4) ({2,3}, 1), ({2,5}, 3), ({2,6}, 5), ({2,8}, 2), ({3,4}, 1), ({3,7}, 2), ({3,8}, 4), ({4,5}, 2), ({4,7}, 3), ({4,8}, 3), ({6,7}, 1)).

63. (({1,3}, 4), ({1,4}, 3), ({1,6}, 2), ({2,3}, 2), ({2,4}, 3), ({2,6}, 4), ({2,7}, 3), ({2,8}, 6), ({3,4}, 1), ({3,6}, 1), ({4,7}, 6), ({5,2}, 2), ({5,6}, 2), ({5,8}, 7), ({7,8}, 2)).

64. (({1,5}, 2), ({1,6}, 6), ({1,7}, 6), ({1,8}, 5), ({2,6}, 2), ({2,9}, 1), ({3,6}, 1), ({3,7}, 2), ({3,9}, 2), ({4,7}, 1), ({4,9}, 2), ({5,8}, 2), ({5,9}, 7), ({6,7}, 1), ({7,8}, 1)).

Задача 2

1.
$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

2.
$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

3.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 1 & -1 \\ 1 & 1 & -1 & 0 & 0 & 0 & -1 & 0 \\ 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4.

$$\begin{bmatrix} 0 & 1 & -1 & 0 & 1 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & -1 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 1 & 0 & -1 & 0 \end{bmatrix}$$

5.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \\ 0 & -1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 1 \\ 0 & -1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \end{bmatrix}$$

6.

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

7.

$$\begin{bmatrix} 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

8.

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

9.

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & -1 & -1 & 0 & 0 & 1 \\ 1 & -1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & -1 & -1 & 0 \end{bmatrix}$$

10.

$$\begin{bmatrix} 0 & 0 & -1 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 \\ 1 & 0 & 0 & -1 & 1 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 1 & 0 & -1 \\ 0 & 1 & -1 & 0 & 0 & 0 & 1 & -1 \\ 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 1 & 0 & -1 & 0 \end{bmatrix}$$

11.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

12.

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

13.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

14.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

15.

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

16.

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

17.

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 1 & -1 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 & 0 & -1 \\ 1 & -1 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 1 & -1 \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & 1 \\ -1 & 1 & 0 & -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & -1 & -1 & 0 & 0 \end{bmatrix}$$

18.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & -1 & -1 & 0 & 1 \\ 0 & 0 & -1 & -1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & -1 & 1 \\ -1 & 1 & 0 & 0 & 0 & 1 & 0 & -1 \\ 1 & -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 1 & -1 \\ 0 & -1 & 1 & 0 & 1 & -1 & 0 & 0 \\ -1 & 0 & -1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

19.

$$\begin{bmatrix} 0 & -1 & 0 & -1 & 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 0 & -1 & -1 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 1 & -1 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 1 & -1 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

20.

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

21.

$$\begin{bmatrix} 0 & 0 & -1 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & -1 & 1 & 1 & -1 \\ 1 & 0 & 0 & 0 & -1 & -1 & 1 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 1 & -1 & 1 & 0 & 0 & 0 & -1 \\ -1 & -1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

22.

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

23.

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 1 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

24.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

25.

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

26.

$$\begin{bmatrix} 0 & -1 & 1 & 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & -1 & -1 & 0 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & -1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix}$$

27.
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & -1 & 1 \\ 0 & -1 & 0 & -1 & 1 & 1 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

28.
$$\begin{bmatrix} 0 & -1 & 0 & 1 & 0 & 1 & -1 \\ 1 & 0 & 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ -1 & 1 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & -1 & 0 & 0 \end{bmatrix}$$

29.
$$\begin{bmatrix} 0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & -1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

30.

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

31.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 1 & -1 \\ -1 & 0 & 1 & 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

32.

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

33.

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

34.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

35.

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

36.

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & -1 & -1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 1 & -1 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 \end{bmatrix}$$

37.

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

38.

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & -1 & 1 & 1 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 & 0 & -1 & 0 \end{bmatrix}$$

39.

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 1 & -1 \\ 1 & 0 & 0 & 0 & -1 & 1 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 & 0 & -1 & 1 \\ -1 & 0 & 1 & 1 & 0 & -1 & 0 & 0 \\ 1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & -1 & 0 & 0 & -1 & 0 \end{bmatrix}$$

40.

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

41.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

42.

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

43.

$$\begin{bmatrix} 0 & -1 & -1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & -1 & 0 & 0 & -1 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & -1 & 0 \\ -1 & 0 & 0 & 0 & -1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix}$$

44.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

45.

$$\begin{bmatrix} 0 & 0 & 1 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 1 & 0 & 1 & 0 & -1 \\ -1 & 1 & 0 & 0 & -1 & 0 & 1 & 0 \\ -1 & -1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 & 1 & 1 \\ 1 & 0 & -1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 0 & 0 \end{bmatrix}$$

46.

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

47.

$$\begin{bmatrix} 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 \end{bmatrix}$$

48.

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

49.

$$\begin{bmatrix} 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \end{bmatrix}$$

50.

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

51.

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & -1 & 0 & 0 & -1 & 1 & 1 \\ -1 & 0 & -1 & 1 & 0 & 0 & 1 \\ -1 & 1 & 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & -1 & -1 & 0 & 0 \end{bmatrix}$$

52.

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

53.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

54.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

55.

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

56.

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

57.

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 1 & -1 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 & 0 & -1 \\ 1 & -1 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 1 & -1 \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & 1 \\ -1 & 1 & 0 & -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & -1 & -1 & 0 & 0 \end{bmatrix}$$

58.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & -1 & -1 & 0 & 1 \\ 0 & 0 & -1 & -1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & -1 & 1 \\ -1 & 1 & 0 & 0 & 0 & 1 & 0 & -1 \\ 1 & -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 1 & -1 \\ 0 & -1 & 1 & 0 & 1 & -1 & 0 & 0 \\ -1 & 0 & -1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

59.

$$\begin{bmatrix} 0 & -1 & 0 & -1 & 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 0 & -1 & -1 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 1 & -1 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 1 & -1 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

60.

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

61.

$$\begin{bmatrix} 0 & 0 & -1 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & -1 & 1 & 1 & -1 \\ 1 & 0 & 0 & 0 & -1 & -1 & 1 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 1 & -1 & 1 & 0 & 0 & 0 & -1 \\ -1 & -1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

62.

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

63.

$$\begin{bmatrix}
0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\
0 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\
1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 1 \\
0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 1 \\
-1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & -1 \\
1 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}$$

64.

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}$$

17. Транспортные сети и задача о наибольшем потоке

17.1. Определение сети

Сетью называется связный граф с выделенными вершинами, которые называются *полюсами* сети.

Так, корневое дерево это сеть, содержащая 1 полюс. Мы будем рассматривать такие сети, которые являются орграфами и у которых каж-

дый полюс – это вершина орграфа, обладающая одним из следующих свойств:

1. Множество входящих дуг в вершину – пусто; такой полюс называется *входом*.

2. Множество выходящих дуг из вершины – пусто; такой полюс называется *выходом*.

Пример сети приведен на рис. 25.

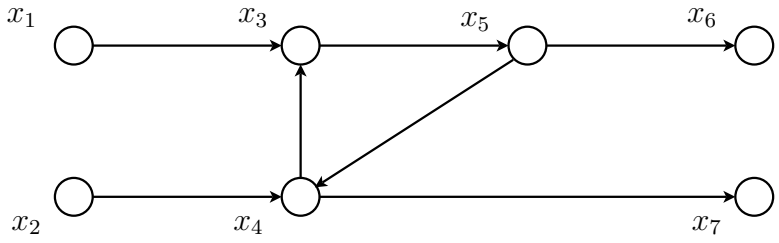

Рис. 25

Здесь $\{x_1, x_2\}$ – множество входов, а $\{x_6, x_7\}$ – множество выходов.

17.2. Транспортная сеть и задача о наибольшем потоке

Транспортной сетью называется связный орграф $G(X, U)$ с одним входом и одним выходом, каждой дуге u которого отнесено неотрицательное целое число $c(u)$, называемое *пропускной способностью* дуги.

Обычно вход транспортной сети называют *источником* и обозначают через x_0, а выход называют *стоком* и обозначают через z. На рис. 26 приведен пример транспортной сети с пропускными способностями, указанными около каждой дуги (не в круглых скобках).

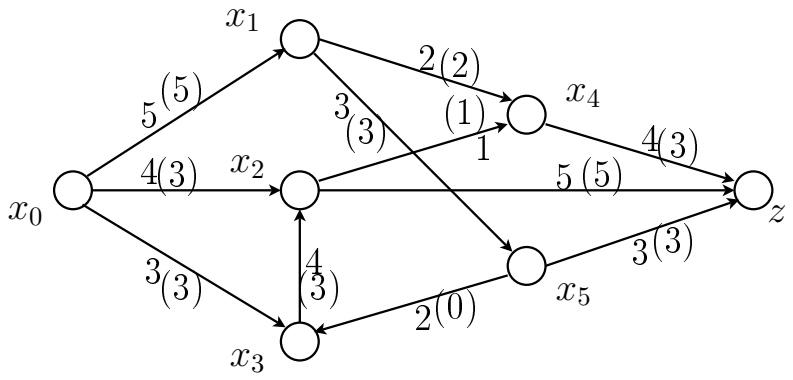

Рис. 26

Обозначим через U_x^+ множество дуг, входящих в вершину x, а через U_x^- – множество дуг, выходящих из x. *Потоком* φ транспортной сети $G(X,U)$ с пропускными способностями $c(u)$, входом x_0 и выходом z называется целочисленная функция, определенная на множестве дуг U и удовлетворяющая свойствам:

1) $\forall u \in U: \ \varphi(u) \geq 0$ – неотрицательность потока;

2) $\forall u \in U: \ \varphi(u) \leq c(u)$ – ограниченность пропускными способностями;

3) $\forall x \in X \setminus \{x_0, z\}: \ \sum_{u \in U_x^-} \varphi(u) = \sum_{u \in U_x^+} \varphi(u)$ – неразрывность потока.

Поток φ можно интерпретировать как поток вещества, идущий от входа к выходу по трубам-дугам сети. Из свойства 3 следует

$$\sum_{u \in U_z^+} \varphi(u) = \sum_{u \in U_{x_0}^-} \varphi(u) \equiv \varphi_z.$$

Эта величина φ_z называется *величиной потока*. На рис. 26 в круглых скобках около дуг указан поток по этим дугам, величина которого равна 11.

Задача о наибольшем потоке ставится следующим образом: *Дана транспортная сеть. Найти поток φ, имеющий наибольшую величину φ_z.*

17.3. Приложения задачи о наибольшем потоке

Задача о наибольшем потоке является дискретной моделью многих дискретных оптимизационных задач. Мы рассмотрим две такие задачи: *транспортную задачу* и *задачу о назначениях*.

17.3.1. Транспортная задача

Постановка задачи. *Предприятия $P_1, P_2, \ldots, P_n$ производят некоторый продукт, который требуется на рынках $R_1, R_2, \ldots, R_m$. Производительность продукта на предприятии P_i ($i \in \overline{1,n}$) равна p_i. Потребность в продукте на рынке R_j ($j \in \overline{1,m}$) равна r_j. Пропускная способность дороги, ведущей от предприятия P_i к рынку R_j, равна*

c_{ij} $(i \in \overline{1,n};\ j \in \overline{1,m})$. *Можно ли удовлетворить потребности всех рынков так, чтобы использовался весь производимый продукт? Как организовать перевозки* d_{ij} $(i \in \overline{1,n};\ j \in \overline{1,m})$ *в этом случае?*

Построим транспортную сеть, образовав для каждого предприятия P_i $(i \in \overline{1,n})$ вершину x_i и для каждого рынка R_j $(j \in \overline{1,m})$ вершину y_j. Соединим x_i с y_j дугой пропускной способности c_{ij}. Добавим источник x_0 и соединим его с каждой вершиной x_i $(i \in \overline{1,n})$ дугой пропускной способности p_i. Добавим сток z и соединим с ним каждую из вершин y_j $(j \in \overline{1,m})$ дугой пропускной способности r_j. Полученная транспортная сеть изображена на рис. 27.

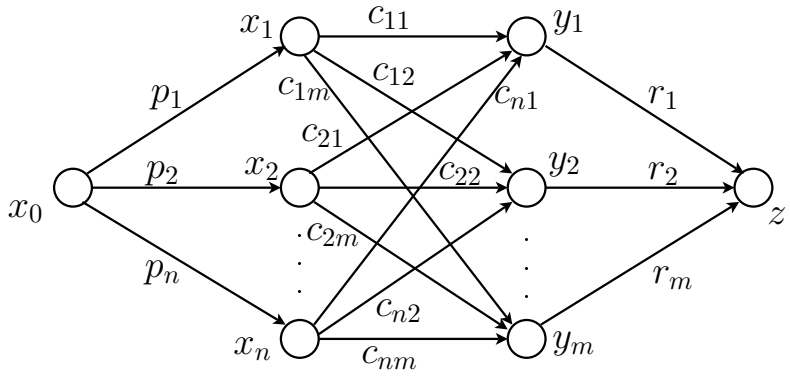

Рис. 27

Пусть транспортная задача имеет решение и d_{ij} $(i \in \overline{1,n};\ \in \overline{1,m})$ – количество продукта, которое перевозится от i-го производителя к j-му рынку. Тогда выполняются следующие условия:

1) сумма вывозимого от каждого производителя продукта равна произведенному им продукту $\sum_{j=1}^{m} d_{ij} = p_i$ $(i \in \overline{1,n})$;

2) сумма доставляемого каждому рынку продукта равна потребности рынка $\sum_{i=1}^{n} d_{ij} = r_j$ $(j \in \overline{1,m})$;

3) пропускная способность дороги, ведущей от P_i к R_j, позволяет перевезти это количество продукта $d_{ij} \le c_{ij}$.

Построим по этому решению поток φ для полученной транспортной сети. Для этого положим

$$\varphi(x_0, x_i) = p_i,\ \varphi(x_i, y_j) = d_{ij},\ \varphi(y_j, z) = r_j\ (i \in \overline{1,n},\ j \in \overline{1,m}).$$

Перечисленные выше условия (1)–(3) обеспечивают все свойства потока: неотрицательность, ограниченность пропускными способностями и неразрывность. Величина потока $\varphi_z = \sum_{j=1}^{m} r_j = \sum_{i=1}^{n} p_i$ является наиболее возможной. Таким образом, решение транспортной задачи индуцирует наибольший поток в транспортной сети.

Рассмотрим теперь необходимые условия решения транспортной задачи. Если задача имеет решение, то для использования всего производимого продукта необходимо, чтобы его количество равнялось суммарной потребности всех рынков, т. е. должно выполняться равенство:

$$\sum_{i=1}^{n} p_i = \sum_{j=1}^{m} r_j. \tag{1}$$

Для того чтобы дороги, ведущие к рынкам от каждого производителя, позволили вывести весь производимый продукт, необходимо выполнение условия на достаточную суммарную пропускную способность этих дорог:

$$\sum_{j=1}^{m} c_{ij} \geq p_i \qquad (i \in \overline{1,n}). \tag{2}$$

Для того чтобы дороги, ведущие к каждому рынку от производителей, позволили привезти весь потребный продукт, необходимо выполнение условия на достаточную суммарную пропускную способность этих дорог:

$$\sum_{i=1}^{n} c_{ij} \geq r_j \qquad (j \in \overline{1,m}). \tag{3}$$

Если эти условия выполнены, то оказывается, что транспортная задача всегда имеет решение. Это решение может быть найдено по наибольшему потоку для транспортной сети. Пусть φ – наибольший поток в транспортной сети. Тогда его величина $\varphi_z = \sum_{j=1}^{m} r_j$ и в силу неразрывности потока (а также выполнения условий (1)–(3)) $\varphi_z = \sum_{i=1}^{n} p_i = \sum_{i=1}^{n} \sum_{j=1}^{m} \varphi(x_i, y_j)$. Положим величину перевозки от производителя P_i к рынку R_j, равной $d_{ij} = \varphi(x_i, y_j)$ $(i \in \overline{1,n},\ j \in \overline{1,m})$. Выполнение условий (1)–(3) гарантирует, что такие перевозки возможны и дают решение транспортной задачи.

Если какое-либо из условий (1)–(3) не выполнено, то либо нельзя удовлетворить потребности рынка, либо не весь производимый про-

дукт может быть использован или перевезен. Но в этих случаях также решение задачи о наибольшем потоке дает возможность организовать перевозки так, чтобы наилучшим образом удовлетворить потребности рынков (по их сумме). Таким образом, транспортная задача может быть сведена к задаче о наибольшем потоке.

17.3.2. Задача о назначениях

Постановка задачи. *Даны* n *работников* X_i $(i \in \overline{1,n})$ *и* m *работ* Y_j $(j \in \overline{1,m})$. *Каждый работник* X_i *способен выполнять лишь некоторые работы* $Y_{i_1}, \ldots, Y_{i_{k_i}}$. *Как произвести назначения работников на работы, чтобы для каждой работы был назначен 1 работник и чтобы каждый работник был занят одной работой?*

Ясно, что необходимым условием положительного ответа на оба вопроса является равенство $m = n$, но это равенство не гарантирует решение: например, 3 работы могут выполнять лишь 2 работника (нехватка работников) или 3 работника могут выполнять лишь 2 какие-то определенные работы (нехватка работ). Поэтому возможны следующие постановки задачи:

1. При $n \geq m$ – можно ли обеспечить каждую работу работником и как это сделать?

2. При $n \leq m$ – можно ли назначить каждого работника на работу и как это сделать?

Решение задачи о назначениях будем выражать в виде булевой матрицы $D = \{d_{ij}\}$ $(i \in \overline{1,n};\ j \in \overline{1,m})$, где работник X_i $(i \in \overline{1,n})$ назначается на работу Y_j $(j \in \overline{1,m})$ в том и только в том случае, когда $d_{ij} = 1$. В каждой строке этой матрицы есть не более одной единицы, и в каждом столбце – также.

Построим транспортную сеть, образовав для каждого работника X_i $(i \in \overline{1,n})$ вершину x_i и для каждой работы Y_j $(j \in \overline{1,m})$ вершину y_j. Соединим вершину x_i с каждой вершиной y_j $(j \in \{i_1, \ldots, i_{k_i}\})$ дугой пропускной способности 1 только в том случае, если работник X_i имеет возможность выполнять работуY_j. Добавим источник x_0 и соединим его с каждой вершиной x_i $(i \in \overline{1,n})$ дугой пропускной способности 1. Добавим сток z и соединим с ним каждую из вершин y_j $(j \in \overline{1,m})$ дугой пропускной способности 1. Полученная транспортная сеть изоб-

ражена на рис. 28.

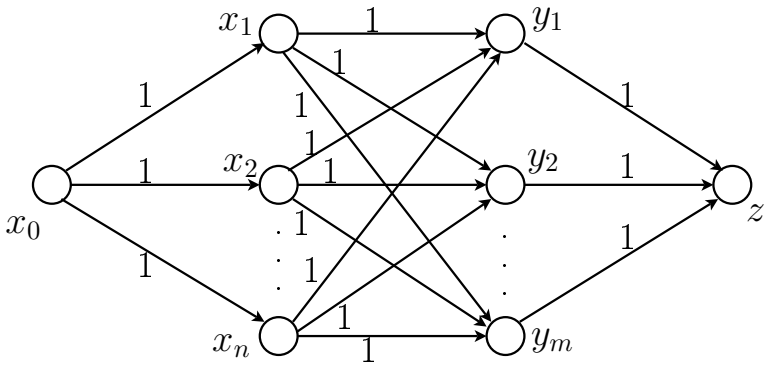

Рис. 28

Пусть сначала $n \geq m$ и задача о назначениях имеет решением матрицу D, для которой в каждом ее столбце имеется ровно 1 единица, а в каждой ее строке не больше 1 единицы. Образуем поток φ следующим образом:

$$\varphi(x_i, y_j) = d_{ij},\ \varphi(x_0, x_i) = \sum_{j=1}^{m} d_{ij},\ \varphi(y_j, z) = \sum_{i=1}^{n} d_{ij}\ (i \in \overline{1,n};\ j \in \overline{1,m}). \tag{4}$$

Нетрудно видеть, что для φ выполнены условия неотрицательности, ограниченности и неразрывности, что делает ее потоком по данной транспортной сети. Так как все назначения для работ определены, то $m = \sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij} = \sum_{j=1}^{m} \varphi(y_j, z) = \varphi_z$, т. е. поток имеет наибольшую величину (максимальный). В случае когда $n \leq m$ и задача о назначениях имеет решением матрицу D, для которой в каждой ее строке имеется ровно 1 единица, а в каждом ее столбце не больше 1 единицы, образуем поток φ равенствами (4). И в этом случае для φ выполнены условия неотрицательности, ограниченности и неразрывности, что делает ее потоком по данной транспортной сети. Так как все назначения для работников определены, то $n = \sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij} = \sum_{i=1}^{n} \varphi(x_0, x_i) = \varphi_z$, т. е. поток имеет наибольшую величину (максимальный). Поэтому в обоих случаях решение задачи о назначениях индуцирует наибольший поток в построенной транспортной сети.

Пусть теперь φ – наибольший поток в транспортной сети с величиной $\varphi_z = m$. Образуем матрицу $D = \{d_{ij}\}$, положив $d_{ij} = \varphi(x_i, y_j)$ $(i \in \overline{1,n};\ j \in \overline{1,m})$. Матрица D является решением задачи о назначениях

при $n \geq m$, поскольку в каждом столбце ее имеется лишь 1 единица, а в каждой ее строке – не более 1 единицы, что следует из величины потока и условия неразрывности потока. В случае когда φ – максимальный поток в транспортной сети с величиной $\varphi_z = n$, также образуем матрицу $D = \{d_{ij}\}$, положив $d_{ij} = \varphi(x_i, y_j)$ $(i \in \overline{1,n};\ j \in \overline{1,m})$. Матрица D является решением задачи о назначениях при $n \leq m$, поскольку в каждой ее строке имеется лишь 1 единица, а в каждом ее столбце не более 1 единицы, что следует из величины потока и условия неразрывности потока. Таким образом, в обоих случаях решение задачи о наибольшем потоке индуцирует решение задачи о назначениях.

Тем самым мы показали, что задача о назначениях сводится к задаче о наибольшем потоке. Позже мы покажем, как, в случае когда задача о назначениях не имеет решения, при помощи поиска решения задачи о наибольшем потоке можно найти так называемые *узкие места* – причины, по которым задача о назначениях не имеет решения.

17.4. Алгоритм Форда–Фалкерсона решения задачи о наибольшем потоке

Авторы теории потоков американские математики Л. Р. Форд и Д. Р. Фалкерсон дали следующий алгоритм решения задачи о наибольшем потоке, который состоит из двух частей:

I. Получение *полного потока*, т. е. такого потока, что для любого пути $\mu[x_0, z]$, ведущего из источника x_0 транспортной сети в сток z, найдется дуга, для которой поток по ней равен ее пропускной способности.

II. *Увеличение* потока *алгоритмом пометок.*

Для получения полного потока выполняем следующий алгоритм:

1. Все дуги транспортной сети не помечены, кроме дуг, которые имеют нулевые пропускные способности $c(x, y) = 0$, которые не рассматриваются. Назначается нулевой поток $\varphi = 0$ на каждой дуге (он допустим, так как для него выполняются условия неотрицательности, ограниченности и неразрывности потока).

2. Ищем любой путь μ из входа x_0 в выход z. Если его нет, то переходим к п. 4 алгоритма.

3. Находим минимальную пропускную способность $c_{min}(\mu)$ дуг пути μ и на дугах этого пути увеличиваем поток на эту величину:

$$\forall(x,y):(x,y)\in\mu \;\rightarrow\; \varphi(x,y)=\varphi(x,y)+c_{min}(\mu).$$

Удаляем (отмечаем) дуги пути, имеющие такую пропускную способность, и уменьшаем (временно, в пределах части I алгоритма) пропускную способность остальных дуг этого пути на это значение минимальной пропускной способности пути μ. Переходим к п. 2.

4. Восстанавливаем исходные пропускные способности дуг и переходим к п. 5 (часть II алгоритма).

Описание алгоритма пометок:

5. Стираем все предыдущие пометки вершин, если они имели место. Помечаем вершину x_0 пометкой $\{+0\}$ и выполняем с повторением п. 6 и п. 7, пока это возможно. Если помечена вершина z, переходим к п. 8, а иначе переходим к п. 9.

6. Если для дуги (x,y) вершина x помечена, а вершина y не помечена и $\varphi(x,y)<c(x,y)$ (поток по дуге меньше исходной пропускной способности), то вершину y помечаем пометкой $\{+x\}$.

7. Если для дуги (x,y) вершина x не помечена, а вершина y помечена и $\varphi(x,y)>0$ (поток по дуге ненулевой), то вершину x помечаем пометкой $\{-y\}$.

8. Если помечена вершина z, то находим маршрут прорыва χ (некоторые дуги маршрута могут следовать в обратном ориентации направлении), идущий из x_0 в z по обратным пометкам, начиная от вершины z ($\chi=(x_0,x_{i_k},\ldots,z)$, где каждая вершина, кроме x_0, имеет пометку предыдущей вершины этого пути), и изменяем поток по каждой дуге этого пути, увеличивая его на 1, если дуга ориентирована в направлении маршрута), или уменьшая его на 1, если дуга ориентирована в обратном направлении. Переходим к п. 5.

9. Конец алгоритма – полученный поток является наибольшим.

Для обоснования алгоритма введем понятие *разреза* транспортной сети.

Разрезом транспотной сети назавается множество V дуг сети, после удаления которых сеть распадается на 2 компоненты связности: G_0, содержащую источник x_0, и G_z, содержащую сток z.

Пропускной способностью разреза V называется суммарная пропускная способность $c(V)$ его дуг, идущих из вершин G_0 в вершины G_z.

В качестве примера рассмотрим транспортную сеть на рис. 26. Поток по ней (указан в круглых скобках около дуг сети) является полным (на любом пути из x_0 в z есть дуга, поток по которой равен ее пропускной способности). Его величина φ_z=11. Является ли он наибольшим? Разрез сети $V_1 = \{(x_0, x_1), (x_0, x_2), (x_0, x_3)\}$ имеет пропускную способность $c(V_1) = 12$, а пропускная способность разреза $V_2 = \{(x_1, x_4), (x_2, x_4), (x_2, z), (x_5, x_3)\}$ $c(V_2) = 11$. Совершенно ясно, что всякий поток проходит через любой разрез сети, а потому не может превысить пропускную способность разреза минимальной пропускной способности. Поэтому указанный на рис. 26 поток является наибольшим. Но наибольшая величина потока может не достичь минимальной пропускной способности разреза. Теорема Форда–Фалкерсона устанавливает, что это не так.

Теорема Форда–Фалкерсона. *Для заданной транспортной сети наибольшая величина потока равна наименьшей пропускной способности разреза.*

Доказательство. Пусть алгоритм пометок остановился, получен поток φ^0 и V – множество помеченных при этом вершин. При этом вершина z осталась непомеченной. Обозначим через V множество дуг, связывающих помеченные и непомеченные вершины. Поток по каждой дуге разреза, идущей из G_0 в G_z, равен ее пропускной способности – иначе была бы помечена одна из непомеченных вершин. Поток по каждой дуге разреза, идущей из G_z в G_0, равен 0 – иначе была бы помечена одна из непомеченных вершин. Поэтому величина потока φ_z равна пропускной способности разреза $c(V)$, и, следовательно, поток, полученный в алгоритме Форда–Фалкерсона, наибольший, а разрез, определяемый дугами, связывающими помеченные и не помеченные в алгоритме вершины, имеет минимальную величину.

Для задачи о назначениях возможности работников можно задать булевой матрицей $R = \{r_{ij}\}$ $(i \in \overline{1,n};\ j \in \overline{1,m})$. Если величина наибольшего потока $\varphi_z < m$, то при решении задачи о наибольшем потоке мы получаем минимальный разрез H, который дает возможность построить *узкое место* следующим образом:

1. Множество $L = \{i|\ r_{ij} \in R;\ (x_0, x_i) \notin H\}$ определяет группу работников (строки матрицы R), для которых не хватает работ (столбцов матрицы R, в которых стоят единицы для этих строк).

2. Множество $M = \{j|\ r_{ij} \in R;\ (y_j, z) \notin H\}$ определяет группу работ (столбцы матрицы R), для которых не хватает работников (строк матрицы R, в которых стоят единицы для этих столбцов).

18. Пример транспортной задачи и ее решения путем сведения к задаче о наибольшем потоке

Решить *транспортную задачу*, заданную следующей матрицей производительностей продукта в пунктах производства, спроса продукта в пунктах потребления и способностей транспортных перевозок (0-й столбец – производительности (s_1, s_2, s_3, s_4) продукта изготовителей (*производителей*) X_1, X_2, X_3, X_4, 0-я строка – потребности (d_1, d_2, d_3) в продукте торговых организаций (*потребителей*) Y_1, Y_2, Y_3, элементы на пересечении i-й строки $(i = 1, 2, 3, 4)$ и j-го столбца $(j = 1, 2, 3)$ – способности p_{ij} в перевозке транспортных магистралей от X_i к Y_j), сведением к *задаче о наибольшем потоке* и ее решению:

	15	12	11
10	2	5	4
9	5	2	3
11	4	6	2
8	5	1	3

А. Корректность данной транспортной задачи определяется следующими условиями:

1. $\sum_{i=1}^{4} s_i = \sum_{j=1}^{3} d_j$ – спрос в продукте всех потребителей равен производимому продукту (10+9+11+8=38=15+12+11);

2. $\sum_{j=1}^{3} p_{ij} \geq s_i$ $(i = 1, 2, 3, 4)$ – транспортные магистрали позволяют вывезти произведенный продукт от каждого производителя (2+5+4=11>10, 5+2+3=10>9, 4+6+2=12>11, 5+1+3=9>8);

3. $\sum_{i=1}^{4} p_{ij} \geq d_j$ $(j = 1, 2, 3)$ – транспортные магистрали позволяют доставить продукт каждому потребителю (2+5+4+5=16>15, 5+2+6+1=13>12, 4+3+2+3=12>11).

Ищется план перевозок b_{ij} $(i = 1, 2, 3, 4;\ j = 1, 2, 3)$, при котором выполняются условия:

1. $0 \leq b_{ij} \leq p_{ij}$ – перевозка от i-го производителя к j-му потребителю не превосходит способности соответствующей транспортной магистрали;

2. $\sum_{j=1}^{3} b_{ij} = s_i$ $(i = 1, 2, 3, 4)$ – вывозится весь продукт от каждого производителя;

3. $\sum_{i=1}^{4} b_{ij} = d_j$ $(j = 1, 2, 3)$ – удовлетворяется спрос каждого потребителя.

Построим транспортную сеть следующим образом (рис. 29):

1. Для каждого производителя X_i $(i = 1, 2, 3, 4)$ введем вершину x_i; для каждого потребителя Y_j $(j = 1, 2, 3)$ введем вершину x_{j+4}; введем вход сети x_0 и выход z.

2. Вход x_0 соединим с каждой вершиной x_i $(i = 1, 2, 3, 4)$ дугой пропускной способности $c(x_0, x_i) = c_{0i} = s_i$ (10, 9, 11, 8 соответственно).

3. Соединим каждую вершину x_{j+4} $(j = 1, 2, 3)$ с выходом z дугой пропускной способности $c(x_{j+4}, z) = c_{j+4,z} = d_j$ (15, 12, 11 соответственно).

4. Соединим каждую вершину x_i $(i = 1, 2, 3, 4)$ с каждой вершиной x_{j+4} $(j = 1, 2, 3)$ дугой пропускной способности $c(x_i, x_{j+4}) = c_{i,j+4} = p_{ij}$ (соответственно матрице).

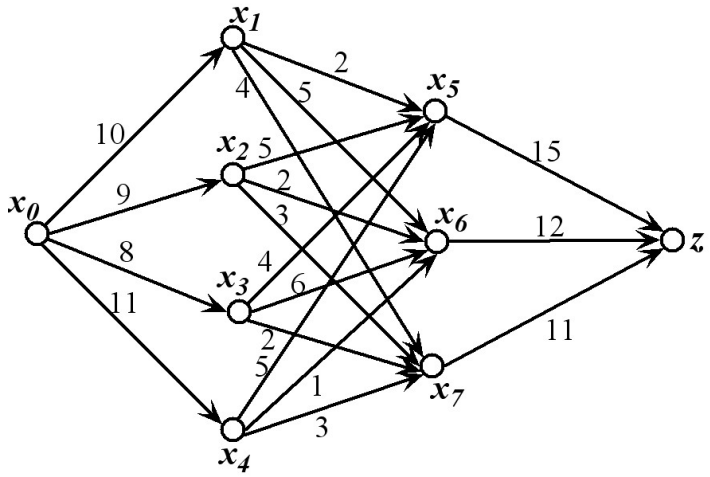

Рис. 29

В. Проведем сведение данной транспортной задачи к задаче о наибольшем потоке для построенной транспортной сети. Для этого нужно показать, что по решению транспортной задачи строится поток для этой транспортной сети, и он является наибольшим, и что по наибольшему потоку строится решение транспортной задачи.

Пусть матрица $b_{ij}(i = 1, 2, 3, 4;\ j = 1, 2, 3)$ является решением транспортной задачи. Определим функцию φ на дугах транспортной сети следующим образом:

1. На каждой дуге (x_i, x_{j+4}) $(i = 1, 2, 3, 4;\ j = 1, 2, 3)$ положим поток $\varphi_{i,j+4} = b_{ij}$.

2. На каждой дуге (x_0, x_i) $(i = 1, 2, 3, 4)$ положим поток $\varphi_{0i} = s_i$.

3. На каждой дуге (x_{j+4}, z) $(j = 1, 2, 3)$ положим поток $\varphi_{j+4,z} = d_j$.

Из неравенств $0 \le b_{ij} \le p_{ij}$ и равенств $\varphi_{i,j+4} = b_{ij}, c_{i,j+4} = p_{ij}$ $(i = 1, 2, 3, 4;\ j = 1, 2, 3)$ следуют неравенства $0 \le \varphi_{i,j+4} \le c_{i,j+4}$, т. е. функция φ на дугах (x_i, x_{j+4}) $(i = 1, 2, 3, 4;\ j = 1, 2, 3)$ неотрицательна и не превосходит пропускной способности соответствующих дуг.

Из равенств $\varphi_{0i} = s_i$ и $c_{0i} = s_i$ $(i = 1, 2, 3, 4)$ следует равенство $\varphi_{0i} = c_{0i}$, т. е. функция φ на дугах (x_0, x_i) $(i = 1, 2, 3, 4)$ неотрицательна и не превосходит (равна) пропускной способности соответствующих дуг.

Из равенств $\varphi_{j+4,z} = d_j$ и $c_{j+4,z} = d_j$ $(j = 1, 2, 3)$ следует равенство $\varphi_{j+4,z} = c_{j+4,z}$ $(j = 1, 2, 3)$, т. е. функция φ на дугах (x_{j+4}, z) $(j = 1, 2, 3)$

неотрицательна и не превосходит (равна) пропускной способности соответствующих дуг.

Таким образом, функция φ на каждой дуге транспортной сети неотрицательна и не превосходит пропускной способности этой дуги.

Из равенств $\varphi_{0i} = s_i$, $\varphi_{i,j+4} = b_{ij}$, $\sum_{j=1}^{3} b_{ij} = s_i$ $(i = 1, 2, 3, 4)$ следует равенство $\sum_{j=1}^{3} \varphi_{i,j+4} = \varphi_{0i}$ $(i = 1, 2, 3, 4)$, т. е. функция φ отвечает условию неразрывности потока в вершинах x_i $(i = 1, 2, 3, 4)$.

Из равенств $\varphi_{j+4,z} = d_j$ $(j = 1, 2, 3)$, $\varphi_{i,j+4} = b_{ij}$, $\sum_{i=1}^{4} b_{ij} = d_j$ $(j = 1, 2, 3)$ следует равенство $\sum_{i=1}^{4} \varphi_{i,j+4} = \varphi_{j+4,z}$ $(j = 1, 2, 3)$, т. е. функция φ отвечает условию неразрывности потока в вершинах x_{j+4} $(j = 1, 2, 3)$.

Таким образом, функция φ в каждой вершине транспортной сети (кроме входа и выхода) отвечает условию неразрывности потока и, значит, φ является потоком.

Из равенств $\varphi_{j+4,z} = d_j$ $(j = 1, 2, 3)$ и $c_{j+4,z} = d_j$ $(j = 1, 2, 3)$ следует равенство $\varphi_{j+4,z} = c_{j+4,z}$ $(j = 1, 2, 3)$, а потому поток на каждой дуге, входящей в выход z, максимален и, следовательно, имеет максимальную величину $\varphi_z = \sum_{j=1}^{3} d_j$. Мы показали, что решению транспортной задачи соответствует решение задачи о наибольшем потоке для построенной транспортной сети.

Покажем обратное: как по потоку φ, являющемуся решением задачи о наибольшем потоке для указанной сети построить решение транспортной задачи. Для этого положим $b_{ij} = \varphi_{i,j+4}$ $(i = 1, 2, 3, 4)$ $(j = 1, 2, 3)$ и покажем, что так образованная матрица b_{ij} $(i = 1, 2, 3, 4$ $j = 1, 2, 3)$ является решением транспортной задачи:

1. Из равенств $c_{i,j+4} = p_{ij}$ $(i = 1, 2, 3, 4;\ j = 1, 2, 3)$ и неравенств $0 \le \varphi_{i,j+4} \le c_{i,j+4}$ $(i = 1, 2, 3, 4;\ j = 1, 2, 3)$ следуют неравенства $0 \le b_{ij} \le p_{ij}$ $(i = 1, 2, 3, 4;\ j = 1, 2, 3)$, которые отвечают требованию 1 решения транспортной задачи.

2. Из равенств $\varphi_{0i} = s_i$ $(i = 1, 2, 3, 4)$, $\sum_{j=1}^{3} \varphi_{i,j+4} = \varphi_{0i}$ $(i = 1, 2, 3, 4)$ следуют равенства $\sum_{j=1}^{3} b_{ij} = s_i$ $(i = 1, 2, 3, 4)$, которые отвечают требованию 2 решения транспортной задачи.

3. Из равенств $\varphi_{j+4,z} = d_j$, $\sum_{i=1}^{4} \varphi_{i,j+4} = \varphi_{j+4,z}$ $(j = 1, 2, 3)$ следуют равенства $\sum_{i=1}^{4} b_{ij} = d_j$ $(j = 1, 2, 3)$, которые отвечают требова-

нию 3 решения транспортной задачи.

Таким образом, условия решения транспортной задачи для матрицы b_{ij} $(i = 1, 2, 3, 4;\ j = 1, 2, 3)$ показаны, что завершает сведение этой задачи к задаче о наибольшем потоке для построенной транспортной сети.

С. Алгоритм Форда-Фалкерсона решения задачи о наибольшем потоке состоит из двух частей:

I. Получение *полного потока*, т. е. такого потока, что для любого пути $\mu = (x_0, x_i, x_{j+4}, z)$ $(i \in \overline{1,4}, j \in \overline{1,3})$, ведущего из входа x_0 в выход z, найдется дуга, для которой поток по ней равен ее пропускной способности;

II. *Увеличение* потока *алгоритмом пометок.*

Для получения полного потока выполняем следующий алгоритм:

1. Все дуги транспортной сети не помечены, кроме дуг, которые имеют нулевые пропускные способности $c_{x,y} = 0$, которые не рассматриваются. Назначается нулевой поток $\varphi = 0$ на каждой дуге.

2. Ищем любой путь μ из входа x_0 в выход z. Если его нет, то переходим к п. 4 алгоритма.

3. Находим минимальную пропускную способность $c_{min}(\mu)$ дуг пути μ и на дугах этого пути увеличиваем поток на эту величину:

$$\forall (x, y) : (x, y) \in \mu \ \rightarrow \ \varphi_{x,y} = \varphi_{x,y} + c_{min}(\mu).$$

   Удаляем (отмечаем) дуги пути, имеющие такую пропускную способность, и уменьшаем (временно, в пределах части I алгоритма) пропускную способность остальных дуг этого пути на это значение минимальной пропускной способности пути μ. Переходим к п. 2.

4. Восстанавливаем исходные пропускные способности дуг и переходим к п. 5 (часть II алгоритма).

Реализацию этой части алгоритма ведем в таблице 1, в которой для каждой дуги с ненулевой пропускной способностью заводится столбец и

при выполнении алгоритма дуга отмечается чертой над ней. Исходные пропускные способности дуг подписываются под каждой дугой. При поиске очередного пути μ из x_0 в z для каждой дуги используется ее текущая пропускная способность, равная разности между исходной пропускной способностью и суммой уже имеющихся потоков в предыдущих строках с учетом их знаков). Каждая строка таблицы отмечает путь μ отметкой знаком "+" в столбцах дуг пути перед потоком по каждой дуге пути, равный минимуму текущих пропускных способностей дуг пути.

Описание алгоритма пометок:

5. Стираем все предыдущие пометки вершин, если они имели место. Помечаем вершину x_0 пометкой $\{+0\}$ и выполняем с повторением п. 6 и п. 7, пока это возможно. Если помечена вершина z, переходим к п. 8, а иначе переходим к п. 9.

6. Если для дуги (x, y) вершина x помечена, а вершина y не помечена и $\varphi_{x,y} < c_{x,y}$ (поток по дуге меньше исходной пропускной способности), то вершину y помечаем пометкой $\{+x\}$.

7. Если для дуги (x, y) вершина x не помечена, а вершина y помечена и $\varphi_{x,y} > 0$ (поток по дуге ненулевой), то вершину x помечаем пометкой $\{-y\}$.

8. Если помечена вершина z, то находим маршрут прорыва χ (некоторые дуги могут быть ориентированы в обратном направлении), идущий из x_0 в z по обратным пометкам, начиная от вершины z ($\chi = (x_0, x_{i_k}, \ldots, z)$, где каждая вершина, кроме x_0, имеет пометку предыдущей вершины этого пути), и изменяем поток по каждой дуге этого пути, увеличивая его на 1, если дуга ориентирована в направлении маршрута), или уменьшая его на 1, если дуга ориентирована в обратном направлении. Переходим к п. 5.

9. Конец алгоритма – полученный поток является наибольшим.

Реализацию этой части алгоритма ведем в таблице 2 и частично в таблице 1. В таблице 2 определяем столбец для каждой вершины. В каждой строке (нумерацию строк продолжаем, делая ее общей с таблицей 1) отмечаем вершины от помеченных в предыдущей строке. После

пометки вершины z изменяем в следующей строке таблицы 1 поток по дугам маршрута прорыва, записывая в столбец $+1$, если увеличивается поток по дуге, или записывая -1, если уменьшается поток по дуге. После завершения алгоритма в таблице 1 записываем результирующую строку, суммируя для каждой дуги потоки вместе с их знаками. По полученным значениям этой строки строим матрицу решения транспортной задачи.

D. Таблица 1 широкая, поэтому разделим ее на части 1A (начало) и 1B (продолжение).

Таблица 1A

N	$\overline{(0,1)}$	$\overline{(0,2)}$	$\overline{(0,3)}$	$\overline{(0,4)}$	$\overline{(1,5)}$	$\overline{(1,6)}$	$\overline{(1,7)}$	$\overline{(2,5)}$	$\overline{(2,6)}$	$\overline{(2,7)}$
	10	9	11	8	2	5	4	5	2	3
1	+2				+2					
2	+5					+5				
3	+3						+3			
4		+5						+5		
5		+2							+2	
6		+2								+2
7			+4							
8			+5							
9			+2							
10				+4						
11				+3						
17				+1	−1		+1			
	10	9	11	8	1	5	4	5	2	2

Таблица 1В

N	$\overline{(3,5)}$	$\overline{(3,6)}$	$\overline{(3,7)}$	$\overline{(4,5)}$	$(4,6)$	$\overline{(4,7)}$	$\overline{(5,z)}$	$\overline{(6,z)}$	$\overline{(7,z)}$
	4	6	2	5	1	3	15	12	11
1							$+2$		
2								$+5$	
3									$+3$
4							$+5$		
5								$+2$	
6									$+2$
7	$+4$						$+4$		
8		$+5$						$+5$	
9			$+2$						$+2$
10				$+4$			$+4$		
11					$-$	$+3$		$-$	$+3$
17				$+1$					$+1$
	4	5	2	5	0	3	15	12	11

Таблица 2

N	1	2	3	4	5	6	7	z
12				$+0$				
13					$+4$	$+4$		
14	-5	-5	-5					
15							$+1$	
16								$+7$

Маршрут прорыва, согласно таблице 2, $\chi = (x_0, x_4, x_5, x_1, x_7, z)$.

Е. Решение транспортной задачи выражается следующей матрицей:

	15	12	11
10	1	5	4
9	5	2	2
11	4	5	2
8	5	0	3

19. Примеры задачи о назначениях и ее решения сведением к задаче о наибольшем потоке

Пример 1. Решить *задачу о назначениях*, заданную следующей булевой матрицей R возможностей работников (строки) по выполнению работ (столбцы), сведением к *задаче о наибольшем потоке* и ее решением:

$$R = \begin{array}{|ccccc|}\hline 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ \hline\end{array}$$

А. Корректность задачи о назначениях определяется следующими условиями:

1. $n = m$ – число работников ($n = 5$) равно числу работ ($m = 5$).
2. Для любого подмножества работников число работ, которые они в совокупности умеют выполнять, не меньше числа работников; для любого подмножества работ число работников, которые их могут выполнять, не меньше числа работ. Эти условия непосредственно трудно проверяемы, поэтому если задача имеет решение, то условия выполнены, а если нет, то найдется узкое место – группа работников, для которых не хватает работ, которые они могут выполнять, а также группа работ, для выполнения которых работников не хватает.

Ищется булева матрица назначений N, для которой:

1) в каждой строке стоит ровно 1 единица (назначение на работу, определенную столбцом), которая соответствует такому же элементу матрицы R;

2) в каждом столбце стоит ровно 1 единица (назначение работника, определенного строкой), которая соответствует такому же элементу матрицы R.

Построим транспортную сеть следующим образом (рис. 30):

1. Для каждого работника W_i $(i = 1, \ldots, 5)$ введем вершину x_i; для каждой работы J_j $(j = 1, \ldots, 5)$ введем вершину x_{j+5}; введем вход сети x_0 и выход z.

2. Вход x_0 соединим с каждой вершиной x_i $(i = 1, \ldots, 5)$ дугой пропускной способности 1.

3. Соединим каждую вершину x_{j+5} $(j = 1, \ldots, 5)$ с выходом z дугой пропускной способности 1.

4. Соединим вершину x_i $(i = 1, \ldots, 5)$ с вершиной x_{j+5} $(j = 1, \ldots, 5)$ дугой пропускной способности 1, если $r_{ij} = 1$ (соответственно матрице возможностей R).

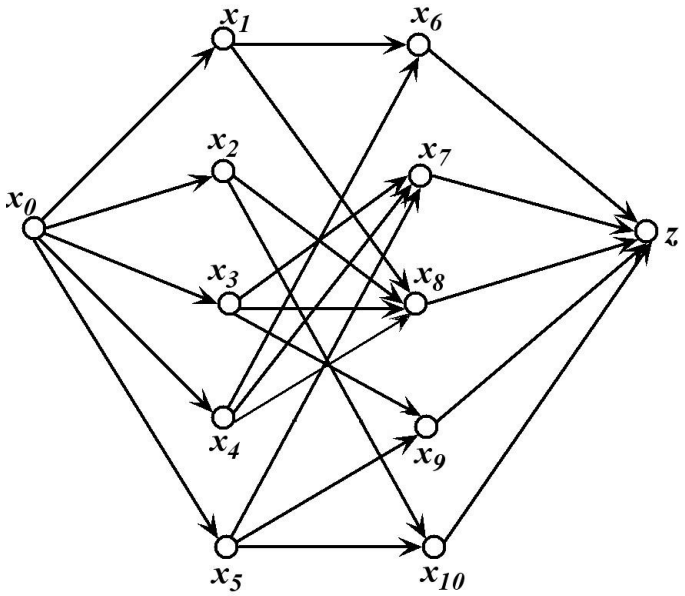

Рис. 30

В. Проведем сведение данной задачи о назначениях к задаче о наибольшем потоке для построенной транспортной сети. Для этого нужно показать, что по решению задачи о назначениях строится поток для этой транспортной сети и он является наибольшим, а также, что по наибольшему потоку, если его значение $\varphi_z = n = m$, строится решение задачи о назначениях.

Пусть матрица $N = \{n_{ij}\}(i \in \overline{1,5};\ j \in \overline{1,5})$ является решением задачи о назначениях (в каждой строке и в каждом столбце ровно 1 единица). Определим функцию φ на дугах транспортной сети следующим образом:

1. На каждой дуге (x_i, x_{j+5}) $(i \in \overline{1,5};\ j \in \overline{1,5})$, если она есть, положим поток $\varphi_{i,j+5} = n_{ij}$.

2. На каждой дуге (x_0, x_i) $(i \in \overline{1,5})$ положим поток $\varphi_{0i} = 1$.

3. На каждой дуге (x_{j+5}, z) $(j \in \overline{1,5})$ положим поток $\varphi_{j+5,z} = 1$.

Значение функции φ на каждой дуге сети неотрицательно и не превосходит 1 (пропускной способности).

Равенство $\varphi_{0,i} = 1 = \sum_{j=1}^{5} n_{ij}$ обеспечивает неразрывность потока в вершинах x_i $(i \in \overline{1,5})$.

Равенство $\sum_{i=1}^{5} \varphi_{i,j+5} = 1 = \varphi_{j+5,z}$ $(j \in \overline{1,5})$ отвечает условию неразрывности потока в вершинах x_{j+5} $(j \in \overline{1,5})$.

Таким образом, функция φ в каждой вершине транспортной сети (кроме входа и выхода) отвечает условию неразрывности потока и, значит, φ является потоком.

Из равенства $\sum_{j=1}^{5} \varphi_{j+5,z} = m$ следует, что поток имеет максимальную величину $\varphi_z = m$. Мы показали, что решению задачи о назначениях соответствует решение задачи о наибольшем потоке для построенной транспортной сети.

Покажем обратное: как по потоку φ, являющемуся решением задачи о наибольшем потоке для указанной сети при условии $\varphi_z = m$, построить решение задачи о назначениях. Для этого положим $n_{ij} = \varphi_{i,j+5}$ $(i \in \overline{1,5};\ j \in \overline{1,5})$ и покажем, что так образованная матрица $N = \{n_{ij}\}$ $(i \in \overline{1,5};\ j \in \overline{1,5})$ является решением задачи о назначениях:

1. Из неравенств $0 \le \varphi_{i,j+5} \le c_{i,j+5} = 1$ $(i \in \overline{1,5};\ j \in \overline{1,5})$ следуют неравенства $0 \le n_{ij} \le 1$ $(i \in \overline{1,5};\ j \in \overline{1,5})$, которые отвечают требованию булевости матрицы N.

2. Из равенств $\varphi_{0i} = 1 = \sum_{j=1}^{5} \varphi_{i,j+5} = \sum_{j=1}^{5} n_{ij}$ $(i \in \overline{1,5};\ j \in \overline{1,5})$ следует, что каждая строка матрицы N имеет ровно 1 единицу.

3. Из равенств $\varphi_{j+5,z} = 1 = \sum_{i=1}^{5} \varphi_{i,j+5} = n_{ij}$ $(j \in \overline{1,5})$ следует, что каждый столбец матрицы N содержит ровно 1 единицу, и, таким образом, матрица N является матрицей назначений.

С. Алгоритм Форда-Фалкерсона решения задачи о наибольшем потоке состоит из 2 частей:

I. Получение *полного потока*, т. е. такого потока, что для любого пути $\mu = (x_0, x_i, x_{j+5}, z)$ $(i \in \overline{1,5}, j \in \overline{1,5})$, ведущего из входа x_0 в выход z, найдется дуга, для которой поток по ней равен 1 (ее пропускной способности);

II. *Увеличение* потока *алгоритмом пометок*.

Для получения полного потока выполняем следующий алгоритм:

1. Все дуги транспортной сети не помечены, кроме дуг, которые имеют нулевые пропускные способности $c_{x,y} = 0$, которые не рассматриваются. Назначается нулевой поток $\varphi = 0$ на каждой дуге.

2. Ищем любой путь μ по непомеченным дугам из входа x_0 в выход z. Если его нет, то переходим к п. 4 алгоритма.

3. Пропускная способность всех дуг пути μ равна 1, и на дугах этого пути увеличиваем поток на 1:

$$\forall (x,y) : (x,y) \in \mu \ \rightarrow \ \varphi_{x,y} = \varphi_{x,y} + 1.$$

   Удаляем (отмечаем) все дуги пути, имеющие такую пропускную способность, и обнуляем (временно, в пределах части I алгоритма) пропускную способность всех дуг этого пути. Переходим к п. 2.

4. Восстанавливаем исходные пропускные способности дуг и переходим к п. 5 (часть II алгоритма).

Реализацию этой части алгоритма ведем в таблице 1, в которой для каждой дуги с ненулевой пропускной способностью заводится столбец, и при выполнении алгоритма дуга отмечается чертой над ней. Исходные пропускные способности дуг подписываются под каждой дугой. Каждая строка таблицы отмечает путь μ отметкой записью "+1" в столбцах дуг пути.

Описание алгоритма пометок:

5. Стираем все предыдущие пометки вершин, если они имели место. Помечаем вершину x_0 пометкой $\{+0\}$ и выполняем с повторением п. 6 и п. 7, пока это возможно. Если помечена вершина z, переходим к п. 8, а иначе переходим к п. 9.

6. Если для дуги (x, y) вершина x помечена, а вершина y не помечена и $\varphi_{x,y} = 0 < c_{x,y}$ (поток по дуге меньше исходной пропускной способности), то вершину y помечаем пометкой $\{+x\}$.

7. Если для дуги (x, y) вершина x не помечена, а вершина y помечена и $\varphi_{x,y} = 1 > 0$ (поток по дуге ненулевой), то вершину x помечаем пометкой $\{-y\}$.

8. Если помечена вершина z, то находим маршрут прорыва χ (некоторые дуги могут проходиться в обратном ориентации направлении), идущий из x_0 в z по обратным пометкам, начиная от вершины z ($\chi = (x_0, x_{i_k}, \ldots, z)$, где каждая вершина, кроме x_0, имеет пометку предыдущей вершины этого пути), и изменяем поток по каждой дуге этого пути, увеличивая его на 1, если дуга ориентирована в направлении маршрута), или уменьшая его на 1, если дуга ориентирована в обратном направлении. Переходим к п. 5.

9. Конец алгоритма – полученный поток является наибольшим.

Реализацию этой части алгоритма ведем в таблице 2 и частично в таблице 1. В таблице 2 заводим столбец для каждой вершины. В каждой строке (нумерацию строк продолжаем, делая ее общей с таблицей 1) отмечаем вершины от помеченных в предыдущей строке. После пометки вершины z изменяем в следующей строке таблицы 1 поток по дугам маршрута прорыва, записывая в столбец $+1$, если увеличивается поток по дуге, или записывая -1, если уменьшается поток по дуге. После завершения алгоритма в таблице 1 записываем результирующую строку, суммируя для каждой дуги потоки вместе с их знаками. По полученным значениям этой строки строим матрицу решения задачи о назначениях, если выполнено условие $\varphi_z = m = 5$.

Если значение потока $\varphi_z < m = 5$, то при решении задачи о наибольшем потоке мы получаем минимальный разрез H, который дает возможность построить *узкое место* следующим образом:

1. Множество $L = \{i|\ r_{ij} \in R;\ (x_0, x_j, z) \notin H\}$ определяет группу работников (строки матрицы R), для которых не хватает работ (столбцов матрицы R, в которых стоят единицы для этих строк).

2. Множество $M = \{j|\ r_{ij} \in R;\ (x_j + 5, z) \notin H\}$ определяет группу работ (столбцы матрицы R), для которых не хватает работников (строк матрицы R, в которых стоят единицы для этих столбцов).

D. Таблица 1 широкая, разделим ее на части 1A (начало) и 1B (продолжение).

Таблица 1A

N	$\overline{0,1}$	$\overline{0,2}$	$\overline{0,3}$	$\overline{0,4}$	$\overline{0,5}$	$\overline{1,6}$	$1,8$	$\overline{2,8}$	$\overline{2,10}$	$\overline{3,7}$	$3,8$	$3,9$
1	$+1$					$+1$						
2		$+1$						$+1$				
3			$+1$							$+1$		
4				$-$								
5					$+1$							
11				$+1$				-1	$+1$			
						1	0	0	1	1	0	0

Таблица 1B

N	$4,6$	$4,7$	$\overline{4,8}$	$5,7$	$\overline{5,9}$	$5,10$	$\overline{6,z}$	$\overline{7,z}$	$\overline{8,z}$	$\overline{9,z}$	$\overline{10,z}$
1							$+1$				
2									$+1$		
3								$+1$			
4	$-$	$-$	$-$								
5				$-$	$+1$					$+1$	
11			$+1$								$+1$
	0	0	1	0	1						

Таблица 2

N	1	2	3	4	5	6	7	8	9	10	z
6				$+0$							
7						$+4$	$+4$	$+4$			
8	-6	-8	-7								
9										$+2$	
10											$+10$

Маршрут прорыва, согласно таблице 2, $\chi = (x_0, x_4, x_8, x_2, x_{10}, z)$.

E. Получен максимальный поток, так как $\varphi_z = 5 = m$. Решение задачи о назначениях выражается следующей матрицей:

$$N = \begin{array}{|ccccc|}\hline 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline \end{array}$$

Пример 2. Решить *задачу о назначениях*, заданную следующей булевой матрицей R возможностей работников (строки) по выполнению работ (столбцы), путем сведения к *задаче о наибольшем потоке* и ее решению:

$$R = \begin{array}{|ccccc|}\hline 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ \hline \end{array}$$

А. Корректность задачи о назначениях определяется следующими условиями:

1. $n = m$ – число работников ($n = 5$) равно числу работ ($m = 5$).
2. Для любого подмножества работников число работ, которые они в совокупности умеют выполнять, не меньше числа работников; для любого подмножества работ число работников, которые их могут выполнять, не меньше числа работ. Эти условия непосредственно трудно проверяемы, поэтому если задача имеет решение, то условия выполнены, а если нет, то найдется узкое место – группа работников, для которых не хватает работ, которые они могут выполнять, а также группа работ, для выполнения которых работников не хватает.

Ищется булева матрица назначений N, для которой:

1) в каждой строке стоит ровно 1 единица (назначение на работу, определяемую столбцом), которая соответствует такому же элементу матрицы R;
2) в каждом столбце стоит ровно 1 единица (назначение работника, определяемого строкой), которая соответствует такому же элементу матрицы R.

Построим транспортную сеть следующим образом (рис. 31):

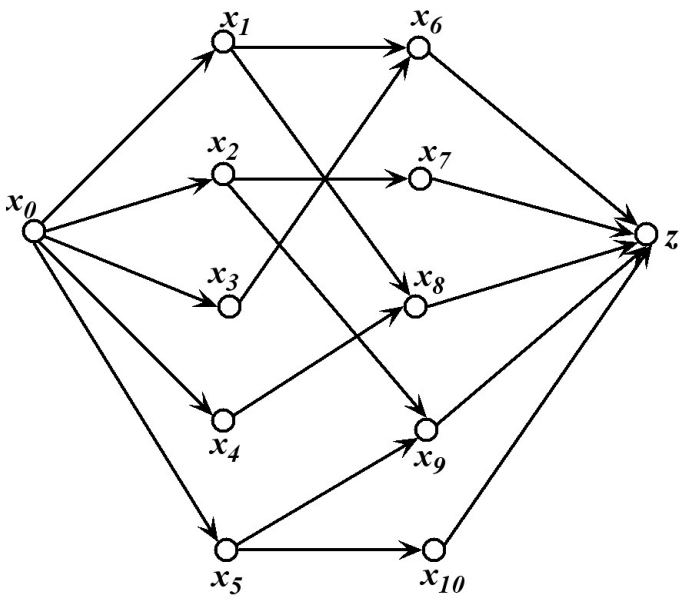

Рис. 31

1. Для каждого работника W_i $(i = 1, 2, 3, 4, 5)$ введем вершину x_i; для каждой работы J_j $(j = 1, 2, 3, 4, 5)$ введем вершину x_{j+5}; введем вход сети x_0 и выход z.

2. Вход x_0 соединим с каждой вершиной x_i $(i = 1, 2, 3, 4, 5)$ дугой пропускной способности 1.

3. Соединим каждую вершину x_{j+5} $(j = 1, 2, 3, 4, 5)$ с выходом z дугой пропускной способности 1.

4. Соединим вершину x_i $(i = 1, 2, 3, 4, 5)$ с вершиной x_{j+5} $(j = 1, 2, 3, 4, 5)$ дугой пропускной способности 1, если $r_{ij} = 1$ (соответственно матрице возможностей R).

B. Аналогично примеру 1.

C. Аналогично примеру 1.

D. Таблица 1 широкая, поэтому разделим ее на части 1A (начало) и 1B (продолжение).

Таблица 1A

N	$\overline{0,1}$	$\overline{0,2}$	$0,3$	$\overline{0,4}$	$\overline{0,5}$	$\overline{1,6}$	$1,8$	$\overline{2,7}$	$2,9$	$3,6$	$\overline{4,8}$	$\overline{5,9}$	$5,10$
1	$+1$					$+1$							
2		$+1$						$+1$					
3			$-$							$-$			
4				$+1$							$+1$		
5					$+1$							$+1$	

Таблица 1B

N	$\overline{6,z}$	$\overline{7,z}$	$\overline{8,z}$	$\overline{9,z}$	$10,z$
1	$+1$				
2		$+1$			
3					
4			$+1$		
5				$+1$	

Таблица 2

N	1	2	3	4	5	6	7	8	9	10	z
6			$+0$								
7						$+3$					
8	-6										
9								$+1$			
10				-8							

Процесс пометок остановился при непомеченной вершине z. Значение наибольшего потока $\varphi_z < 5 = m$. Поэтому задача о назначениях не имеет решения.

E. Для нахождения узкого места находим минимальный разрез как множество дуг, идущих из помеченных вершин в непомеченные вершины:

$$H = \{(0,2),\ (0,5),\ (6,z),\ (8,z)\}.$$

1. Множество $L = \{1,3,4\}$ определяет группу работников (строки матрицы R), для которых не хватает работ (столбцов матрицы R, в которых стоят единицы для этих строк), т. е. работники 1, 3, 4 могут выполнять только работы 1 и 3 (для этих работников не хватает работ).

2. Множество $M = \{2, 4, 5\}$ определяет группу работ (столбцы матрицы R), для которых не хватает работников (строк матрицы R, в которых стоят единицы для этих столбцов), т. е. работы 2, 4, 5 могут выполнять только работники 2 и 5 (для выполнения этих работ не хватает работников).

20. Индивидуальное задание 10

20.1. Общее задание

1. Решить транспортную задачу, заданную матрицей производительностей продукта в пунктах производства, спроса продукта в пунктах потребления и способностей транспортных перевозок:

a) указать условия, при которых весь производимый продукт может удовлетворить весь спрос продукта и перевозки могут быть осуществлены; построить транспортную сеть для указанной транспортной задачи;

b) описать сведение транспортной задачи к задаче о наибольшем потоке;

c) описать алгоритм решения задачи о наибольшем потоке;

d) свести выполнение алгоритма для заданной задачи в таблицы;

e) по решению задачи о наибольшем потоке получить матрицу перевозок – решение транспортной задачи.

2. Решить задачу о назначениях, заданную булевой матрицей размерности $n \times m$ возможностей n работников для m работ:

a) указать условия, при которых может существовать решение задачи о назначениях; построить транспортную сеть для указанной задачи о назначениях;

b) описать сведение задачи о назначениях к задаче о наибольшем потоке;

c) описать алгоритм решения задачи о наибольшем потоке для задачи о назначениях;

d) свести выполнение алгоритма для заданной задачи в таблицы;

e) если наибольший поток удовлетворяет условиям существования решения задачи о назначениях, то по решению задачи о наибольшем потоке *получить матрицу назначений*; в противном случае *найти узкое место*, которое не позволяет решить задачу о назначениях.

20.2. Методические рекомендации для задания 10

Задача 1

1. В таблице прокрутки 1-й части алгоритма все дуги должны быть расположены в строгом порядке (сначала по началу дуги, затем по концу дуги при одинаковом начале дуги).

2. При отыскании очередного пути от входа x_0 к выходу z последовательно определяются неотмеченные дуги, которые помечаются заком "—". Если помечена вершина z, то все пометки заменяются на знак "+", определяется остающаяся минимальная пропускная способность p дуги этого пути по разности пропускной способности этой дуги и суммы потоков, уже идущих по этой дуге, и к пометкам каждой дуги добавляется p. После этого дуга с остающейся нулевой пропускной способностью отмечается.

3. После окончания 1-й части алгоритма происходит переход ко 2-й части.

4. В таблице прокрутки 2-й части алгоритма все вершины располагаются от вершины с номером 1 до вершины с номером z (выход). Все прежние пометки (если они были) стираются. Вершины помечаются в соответствии с алгоритомом пометок до тех пор, пока не будет помечен выход z или процесс пометок не остановится в связи с невозможностью помечать далее вершины. **Типичной ошибкой** является **раннее завершение процесса пометок**.

5. Если помечен выход, то определяется путь прорыва и таблица 1-й части алгоритма корректируется. После этого снова повторяется

алгоритм пометок. **Типичной ошибкой** является **завершение алгоритма после коррекции** таблицы 1-й части его выполнения. В этом случае результат может оказаться неверным.

Задача 2

1. Выполнение 1-й и 2-й частей алгоритма проводится так же, как и для задачи 1, с той лишь разницей, что пропускные способности всех дуг равны 1.

2. Если в результате выполнения алгоритма все дуги, идущие в выход z, отмечены, то по таблице 1-й части находится матрица назначений, что завершает выполнение задачи.

3. Если процесс пометок остановился, но есть неотмеченные дуги, идущие в выход z, то задача о назначениях не имеет решения и требуется найти узкое место.

4. Для нахождения узкого места выписываются все дуги минимального разреза, идущие от помеченных вершин сети к непомеченным вершинам.

5. Дуги, идущие из входа x_0 к помеченным вершинам , определяют группу работников, для которых не хватает работ (работы определяются по исходной матрице R возможных назначений).

6. Дуги, идущие в выход z от помеченных вершин , определяют группу работ, для которых не хватает работников (работники определяются по исходной матрице R возможных назначений).

20.3. Варианты индивидуального задания 10

Задача 1

1.

	19	12	11
12	4	5	4
11	7	2	3
10	4	6	2
9	6	1	3

2.

	15	13	12	11
16	2	5	4	6
13	5	2	3	4
13	4	6	2	1
9	5	1	3	1

3.

	16	15	19
10	2	5	4
9	5	2	3
11	4	6	2
8	5	1	3
12	1	2	10

4.

	13	10	14
9	5	3	3
10	2	5	4
8	3	3	3
10	6	0	5

5.

	15	13	12	13
16	2	5	4	6
13	5	2	3	4
13	4	6	2	1
11	5	1	3	3

6.

	15	13	12	11	14
22	2	11	5	6	0
23	9	2	4	4	6
20	5	1	4	2	9

7.

	19	12	11
11	7	2	3
10	4	6	2
9	6	1	3
12	4	5	4

8.

	15	13	12	11
16	2	5	4	6
13	4	6	2	1
9	5	1	3	1
13	5	2	3	4

9.

	16	15	19
10	2	5	4
9	5	2	3
12	1	2	10
8	5	1	3
11	4	6	2

10.

	13	10	14
10	6	0	5
8	3	3	3
9	5	3	3
10	2	5	4

11.

	15	13	12	13
11	5	1	3	3
13	5	2	3	4
16	2	5	4	6
13	4	6	2	1

12.

	15	13	12	11	14
20	5	1	4	2	9
23	9	2	4	4	6
22	2	11	5	6	0

13.

	12	19	11
11	2	7	3
9	1	6	3
10	6	4	2
12	5	4	4

14.

	12	13	15	11
13	3	2	5	4
16	4	5	2	6
13	2	6	4	1
9	3	1	5	1

15.

	15	16	19
10	5	2	4
9	2	5	3
8	1	5	3
11	6	4	2
12	2	1	10

16.

	13	14	10
9	5	3	3
10	6	5	0
8	3	3	3
10	2	4	5

17.

	15	12	13	13
16	2	4	5	6
11	5	3	1	3
13	5	3	2	4
13	4	2	6	1

18.

	11	13	12	15	14
20	2	1	4	5	9
22	6	11	5	2	0
23	4	2	4	9	6

19.

	12	19	11
12	5	4	4
10	6	4	2
11	2	7	3
9	1	6	3

20.

	12	13	15	11
13	3	2	5	4
13	2	6	4	1
9	3	1	5	1
16	4	5	2	6

21.

	15	16	19
12	2	1	10
10	5	2	4
9	2	5	3
11	6	4	2
8	1	5	3

22.

	13	14	10
8	3	3	3
9	5	3	3
10	2	4	5
10	6	5	0

23.

	15	12	13	13
13	5	3	2	4
16	2	4	5	6
13	4	2	6	1
11	5	3	1	3

24.

	14	13	12	11	15
20	9	1	4	2	5
22	0	11	5	6	2
23	6	2	4	4	9

25.

	12	19	11
9	1	6	3
11	2	7	3
10	6	4	2
12	5	4	4

26.

	11	13	12	15
9	1	1	3	5
16	6	5	4	2
13	1	6	2	4
13	4	2	3	5

27.

	16	19	15
8	5	3	1
10	2	4	5
9	5	3	2
12	1	10	2
11	4	2	6

28.

	10	13	14
8	3	3	3
10	0	6	5
9	3	5	3
10	5	2	4

29.

	13	15	12	13
11	1	5	3	3
13	2	5	3	4
13	6	4	2	1
16	5	2	4	6

30.

	15	13	14	11	12
20	5	1	9	2	4
22	2	11	0	6	5
23	9	2	6	4	4

31.

	11	19	12
9	3	6	1
10	2	4	6
11	3	7	2
12	4	4	5

32.

	12	11	15	13
9	3	1	5	1
13	3	4	5	2
16	4	6	2	5
13	2	1	4	6

33.

	16	15	19
8	5	1	3
10	2	5	4
9	5	2	3
11	4	6	2
12	1	2	10

34.

	13	10	14
8	3	3	3
9	5	3	3
10	6	0	5
10	2	5	4

35.

	12	15	13	13
16	4	2	5	6
13	3	5	2	4
13	2	4	6	1
11	3	5	1	3

36.

	15	13	12	11	14
22	2	11	5	6	0
20	5	1	4	2	9
23	9	2	4	4	6

37.

	10	13	14
10	5	2	4
10	0	6	5
8	3	3	3
9	3	5	3

38.

	15	13	12	13
11	5	3	3	3
13	4	4	2	4
13	5	4	3	2
16	2	6	4	5

39.

	14	13	12	11	15
23	6	2	4	4	9
22	0	11	5	6	2
20	9	1	4	2	5

40.

	19	12	11
9	6	1	3
10	4	6	2
12	4	5	4
11	7	2	3

41.

	12	13	11	15
13	3	2	4	5
13	2	6	1	4
9	3	1	1	5
16	4	5	6	2

42.

	16	15	19
9	5	2	3
8	5	1	3
10	2	5	4
11	4	6	2
12	1	2	10

43.

	10	13	14
9	5	3	3
10	1	5	5
10	2	5	4
8	3	3	3

44.

	13	12	13	15
11	3	3	1	5
13	4	3	2	5
13	1	2	6	4
16	6	4	5	2

45.

	11	13	12	15	14
20	2	1	4	5	9
22	6	11	5	2	0
23	4	2	4	9	6

46.

	11	19	12
12	4	4	5
9	3	6	1
10	2	4	6
11	3	7	2

47.

	12	15	13	11
13	3	5	2	4
13	2	4	6	1
16	4	2	5	6
9	3	5	1	1

48.

	15	19	16
10	5	4	2
9	2	3	5
11	6	2	4
8	1	5	3
12	2	8	3

49.

	14	13	10
9	3	5	3
10	4	2	5
10	5	6	0
8	3	3	3

50.

	12	15	13	13
13	3	5	2	4
13	2	4	6	1
11	3	5	1	3
16	4	2	5	6

51.

	13	15	13	11
13	3	5	2	4
9	3	5	1	1
14	3	4	6	1
16	4	2	5	6

52.

	15	13	12	11
16	2	5	4	6
13	5	2	3	4
13	4	6	2	1
9	5	1	3	1

53.

	16	15	19
10	2	5	4
9	5	2	3
11	4	6	2
8	5	1	3
12	1	2	10

54.

	13	10	14
9	5	3	3
10	2	5	4
8	3	3	3
10	6	0	5

55.

	15	13	12	13
16	2	5	4	6
13	5	2	3	4
13	4	6	2	1
11	5	1	3	3

56.

	15	13	12	11	14
22	2	11	5	6	0
23	9	2	4	4	6
20	5	1	4	2	9

57.

	19	12	11
11	7	2	3
10	4	6	2
9	6	1	3
12	4	5	4

58.

	15	13	12	11
16	2	5	4	6
13	4	6	2	1
9	5	1	3	1
13	5	2	3	4

59.

	16	15	19
10	2	5	4
9	5	2	3
12	1	2	10
8	5	1	3
11	4	6	2

60.

	13	10	14
10	6	0	5
8	3	3	3
9	5	3	3
10	2	5	4

61.

	15	13	12	13
11	5	1	3	3
13	5	2	3	4
16	2	5	4	6
13	4	6	2	1

62.

	15	13	12	11	14
20	5	1	4	2	9
23	9	2	4	4	6
22	2	11	5	6	0

63.

	12	19	11
11	2	7	3
9	1	6	3
10	6	4	2
12	5	4	4

64.

	12	13	15	11
13	3	2	5	4
16	4	5	2	6
13	2	6	4	1
9	3	1	5	1

Задача 2

1.

1	0	1	0	0	0
0	1	0	1	0	1
0	0	1	0	1	0
0	1	0	0	0	1
0	0	0	1	0	1
1	1	0	1	0	0

2.

0	1	0	1	0
1	0	1	0	1
0	1	1	1	0
0	0	0	1	0
0	1	0	0	0

3.

1	1	1	0	0	0
0	0	0	0	1	1
1	0	1	0	0	0
0	0	0	1	0	1
0	1	1	0	0	0
0	0	0	1	1	0

4.

0	0	0	1	1
1	0	1	0	0
0	1	0	1	0
1	0	0	0	0
0	0	1	0	0

5.

0	0	0	0	1	1
1	1	0	0	0	0
1	1	1	0	0	0
0	0	0	1	0	1
1	0	1	0	0	0
0	1	1	0	0	0

6.

0	1	1	0	0
0	0	0	1	0
0	0	1	1	0
0	0	0	0	1
1	0	1	0	0

7.

1	0	1	0	0	0
0	0	0	1	0	1
0	1	0	1	0	1
1	1	0	1	0	0
0	0	1	0	1	0
0	1	0	0	0	1

8.

1	1	0	0	0
0	0	1	1	1
1	1	1	0	0
1	0	0	0	0
0	1	0	0	0

9.

1	0	1	0	1	0
0	1	0	0	0	1
1	0	1	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	1	0	1	0	0

10.

0	1	0	1	0
1	0	1	0	0
0	0	0	1	1
1	0	0	0	0
0	0	1	0	0

11.

0	1	0	0	0	1
1	0	0	0	1	0
1	0	1	0	1	0
0	0	0	1	0	1
1	0	1	0	0	0
0	0	1	0	1	0

12.

0	1	1	0	0
0	0	0	1	0
0	1	0	1	0
0	0	0	0	1
1	1	0	0	0

13.

0	1	0	1	0	1
0	0	1	0	1	0
0	1	0	0	0	1
1	0	1	0	0	0
0	0	0	1	0	1
1	1	0	1	0	0

14.

0	0	0	1	0
0	1	0	1	0
1	0	1	0	1
0	1	1	1	0
0	1	0	0	0

15.

0	0	0	1	1	0
1	1	1	0	0	0
0	1	1	0	0	0
0	0	0	0	1	1
1	0	1	0	0	0
0	0	0	1	0	1

16.

0	0	0	1	1
1	0	1	0	0
0	1	0	1	0
0	0	1	0	0
1	0	0	0	0

17.

1	1	0	0	0	0
1	1	1	0	0	0
0	0	0	1	0	1
1	0	1	0	0	0
0	0	0	0	1	1
0	1	1	0	0	0

18.

0	1	1	0	0
0	0	0	1	0
0	0	0	0	1
1	0	1	0	0
0	0	1	1	0

19.

0	1	0	1	0	1
1	0	1	0	0	0
1	1	0	1	0	0
0	1	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0

20.

1	1	0	0	0
0	1	0	0	0
0	0	1	1	1
1	1	1	0	0
1	0	0	0	0

21.

0	1	0	1	0	0
1	0	1	0	1	0
0	1	0	0	0	1
1	0	1	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0

22.

0	1	0	1	0
1	0	1	0	0
0	0	0	1	1
0	0	1	0	0
1	0	0	0	0

23.

0	1	0	0	0	1
1	0	0	0	1	0
0	0	0	1	0	1
1	0	1	0	0	0
0	0	1	0	1	0
1	0	1	0	1	0

24.

0	1	1	0	0
0	0	0	1	0
1	1	0	0	0
0	1	0	1	0
0	0	0	0	1

25.

1	0	1	0	0	0
1	1	0	1	0	0
0	1	0	1	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	1	0	0	0	1

26.

0	0	0	1	0
0	1	0	1	0
0	1	0	0	0
1	0	1	0	1
0	1	1	1	0

27.

0	0	0	1	1	0
1	1	1	0	0	0
0	0	0	1	0	1
0	1	1	0	0	0
1	0	1	0	0	0
0	0	0	0	1	1

28.

0	0	0	1	1
0	1	0	1	0
1	0	1	0	0
0	0	1	0	0
1	0	0	0	0

29.

1	1	0	0	0	0
0	0	0	1	0	1
1	0	1	0	0	0
0	0	0	0	1	1
1	1	1	0	0	0
0	1	1	0	0	0

30.

0	0	0	1	0
0	1	1	0	0
0	0	0	0	1
1	0	1	0	0
0	0	1	1	0

31.

0	0	0	1	0	1
0	1	0	1	0	1
1	0	1	0	0	0
0	1	0	0	0	1
1	1	0	1	0	0
0	0	1	0	1	0

32.

0	1	0	0	0
1	1	0	0	0
0	0	1	1	1
1	1	1	0	0
1	0	0	0	0

33.

0	1	0	1	0	0
1	0	1	0	1	0
0	1	0	0	0	1
0	0	0	1	0	1
1	0	1	0	0	0
0	0	1	0	1	0

34.

1	0	1	0	0
0	0	0	1	1
0	0	1	0	0
1	0	0	0	0
0	1	0	1	0

35.

1	0	1	0	1	0
0	1	0	0	0	1
1	0	0	0	1	0
0	0	0	1	0	1
1	0	1	0	0	0
0	0	1	0	1	0

36.

0	1	1	0	0
1	1	0	0	0
0	1	0	1	0
0	0	0	0	1
0	0	0	1	0

37.

0	1	0	1	0	1
1	0	1	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	0	1
1	1	0	1	0	0

38.

0	0	0	1	0
0	1	0	1	0
0	1	0	0	0
1	0	1	0	1
0	1	1	1	0

39.

0	0	0	1	1	0
1	0	1	0	0	0
0	0	0	1	0	1
1	1	1	0	0	0
0	1	1	0	0	0
0	0	0	0	1	1

40.

0	1	0	1	0
1	0	1	0	0
0	0	0	1	1
0	0	1	0	0
1	0	0	0	0

41.

1	1	0	0	0	0
0	0	0	1	0	1
1	0	1	0	0	0
1	1	1	0	0	0
0	0	0	0	1	1
0	1	1	0	0	0

42.

0	0	0	1	0
1	0	1	0	0
0	1	1	0	0
0	0	0	0	1
0	0	1	1	0

43.

0	0	0	0	1	1
1	1	1	0	0	0
0	0	0	1	0	1
1	0	1	0	0	0
0	1	1	0	0	0
1	1	0	0	0	0

44.

0	1	0	0	0
1	1	0	0	0
0	0	1	1	1
1	0	0	0	0
1	1	1	0	0

45.

0	1	0	1	0	0
1	0	1	0	1	0
0	0	0	1	0	1
1	0	1	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1

46.

1	0	1	0	0
0	0	1	0	0
0	0	0	1	1
1	0	0	0	0
0	1	0	1	0

47.

1	0	1	0	1	0
0	1	0	0	0	1
1	0	0	0	1	0
0	0	0	1	0	1
0	0	1	0	1	0
1	0	1	0	0	0

48.

1	1	0	0	0
0	1	0	1	0
0	0	0	0	1
0	1	1	0	0
0	0	0	1	0

49.

0	1	0	1	0	1
0	1	0	0	0	1
0	0	0	1	0	1
1	1	0	1	0	0
0	0	1	0	1	0
1	0	1	0	0	0

50.

0	1	0	1	0
0	0	0	1	0
0	1	0	0	0
1	0	1	0	1
0	1	1	1	0

51.

0	1	0	1	0	1
0	1	0	0	0	1
1	0	1	0	0	0
0	0	0	1	0	1
1	1	0	1	0	0
0	0	1	0	1	0

52.

0	1	1	0	0
0	0	0	1	0
0	1	0	1	0
0	0	0	0	1
1	1	0	0	0

53.

0	1	0	1	0	1
0	0	1	0	1	0
0	1	0	0	0	1
1	0	1	0	0	0
0	0	0	1	0	1
1	1	0	1	0	0

54.

0	0	0	1	0
0	1	0	1	0
1	0	1	0	1
0	1	1	1	0
0	1	0	0	0

55.

0	0	0	1	1	0
1	1	1	0	0	0
0	1	1	0	0	0
0	0	0	0	1	1
1	0	1	0	0	0
0	0	0	1	0	1

56.

0	0	0	1	1
1	0	1	0	0
0	1	0	1	0
0	0	1	0	0
1	0	0	0	0

57.

1	1	0	0	0	0
1	1	1	0	0	0
0	0	0	1	0	1
1	0	1	0	0	0
0	0	0	0	1	1
0	1	1	0	0	0

58.

0	1	1	0	0
0	0	0	1	0
0	0	0	0	1
1	0	1	0	0
0	0	1	1	0

59.

0	1	0	1	0	1
1	0	1	0	0	0
1	1	0	1	0	0
0	1	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0

60.

1	1	0	0	0
0	1	0	0	0
0	0	1	1	1
1	1	1	0	0
1	0	0	0	0

61.

0	1	0	1	0	0
1	0	1	0	1	0
0	1	0	0	0	1
1	0	1	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0

62.

0	1	0	1	0
1	0	1	0	0
0	0	0	1	1
0	0	1	0	0
1	0	0	0	0

63.

0	1	0	0	0	1
1	0	0	0	1	0
0	0	0	1	0	1
1	0	1	0	0	0
0	0	1	0	1	0
1	0	1	0	1	0

64.

0	1	1	0	0
0	0	0	1	0
1	1	0	0	0
0	1	0	1	0
0	0	0	0	1

21. Литература

1. Берж К. Теория графов и ее приложения. – М. : Иностранная литература, 1962. – 320 с.
2. Оре О. Теория графов. – М. : Наука, 1968. – 352 с.
3. Харари Ф. Теория графов. – М. : Мир, 1973. – 302 с.
4. Басакер Р.. Саати Т. Конечные графы и сети. – М. : Наука, 1974. – 368 с.
5. Ху Т. Целочисленное программирование и потоки в сетях. – М. : Мир, 1974. – 520 с.
6. Адельсон-Вельский Г. М., Диниц Е. А., Карзанов А. В. Потоковые алгоритмы. – М. : Наука, 1975. – 120 с.
7. Харари Ф., Палмер Э. Перечисление графов. – М. : Мир, 1977. – 326 с.
8. Кристофидес Н. Теория графов. Алгоритмический подход. – М. : Мир, 1978. – 432 с.

9. Свами М., Тхурасиламан К. Графы, сети и алгоритмы. – М. : Мир, 1984. – 456 с.

10. Филлипс Д., Гарсиа-Диас А. Методы анализа сетей. – М. : Мир, 1984. – 496 с.

11. Татт У. Теория графов. – М. : Мир, 1988. – 424 с.

12. Евстигнеев В. А., Касьянов В. Н. Алгоритмы на деревьях. – Новосибирск : АН СССР, СО, ВЦ, 1989. – 312 с.

13. Евстигнеев В. А., Касьянов В. Н. Алгоритмы обработки деревьев. – Новосибирск : АН СССР, СО, ВЦ, 1989. – 209 с.

14. Алгоритмы и программы решения задач на графах и сетях / Нечепуренко В. К. [и др.]. – Новосибирск : Наука, СО АН СССР, ВЦ. – 515 с.

15. Зыков А. А. Основы теории графов. – М. : Вузовская книга, 2004. – 663 с.

16. Алексеев В. Е., Таланов В.А. Графы. Модели вычислениййю Алгоритмы. – Нижний Новгород : ННГУ, 2005. – 308 с.

17. Звонкин А. К., Ландо С. К. Графы на поверхностях и их приложения. – М. : МЦМНО, 2010. – 480 с.

Учебное издание

Рублев Вадим Сергеевич

ЭЛЕМЕНТЫ ТЕОРИИ ГРАФОВ

(индивидуальные работы № 8–10 по дисциплине
«Дискретная математика»)

Учебно-методическое пособие

Редактор, корректор Л. Н. Селиванова
Компьютерная верстка В. С. Рублев

Подписано в печать 28.01.2020 Формат 60×84/16.
Усл. печ. л. 10,5. Уч.-изд. л. 10,0.
Тираж 3 экз. Заказ .

Оригинал-макет подготовлен
в редакционно-издательском отделе ЯрГУ.

Ярославский государственный университет им. П. Г. Демидова
150003, Ярославль, ул. Советская, 14.