БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра информационных систем управления

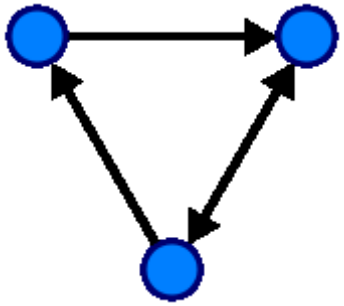
ИССЛЕДОВАНИЕ ОПЕРАЦИЙ В ЗАДАЧАХ

Учебно-методическое пособие для студентов факультета прикладной математики и информатики

В трёх частях

Часть II

СЕТЕВЫЕ ЗАДАЧИ



МИНСК
2011

УДК 519.8 (075)
ББК 22.18
И85

А в т о р ы - с о с т а в и т е л и:
А. Н. Исаченко, Л. Ф. Дробушевич

Рекомендовано ученым советом
факультета прикладной математики и информатики
6 декабря 2011 г., протокол № 3

Исследование операций в задачах: учеб.-метод. пособие для И85 студентов факультета прикладной математики и информатики.
В 3 ч. Ч. II: Сетевые задачи / авт.-сост.: А. Н. Исаченко, Л. Ф. Дробушевич. – Минск: БГУ, 2011. – 63 с.

Излагаются основные понятия, методика, алгоритмы и методы исследования операций, касающиеся построения сетевых моделей и решения задач на сетях. Даются задачи для самостоятельной работы студентов.

Предназначено для студентов факультета прикладной математики и информатики.

УДК 519.8(075)
ББК 22.18

ВВЕДЕНИЕ

Сеть – это ориентированный или неориентированный граф, дугам, ребрам, вершинам которого приписаны некоторые параметры, называемые весами.

Сеть как математическая модель возникает при рассмотрении объектов или явлений, которые можно представить в виде элементов, между которыми существует связь или взаимодействие, причем наличие связи или взаимодействия является определяющим.

Примером объектов, моделируемых сетью, являются транспортные сети (наземные, водные, воздушные), коммуникационные сети, электрические схемы, проекты работ с четкой иерархией предшествования, управляющие системы, информационные системы, компьютерные вычислительные сети и так далее. При этом веса, приписываемые элементам сети, имеют физическую интерпретацию, связанную с предметной областью в которой рассматривается объект (явление).

Так при моделировании транспортных систем вес дуги может интерпретироваться как длина участка дороги, соединяющего соответствующие пункты, или его пропускная способность. Аналогично, при моделировании нефте, газо, водопроводов, вес может интерпретироваться как длина или пропускная способность участка проводящей системы. При рассмотрении систем связи веса могут указывать количество каналов между абонентами. При моделировании управления назначениями – время или эффективность выполнения исполнителем работ. При моделировании проекта выполняемых работ веса имеют значения длительностей выполнения работ.

Сетевые задачи часто встречаются в приложениях и отличаются постановкой и методами решения. Моделируемый сетью объект может допускать математические модели, содержащие целевую функцию, систему ограничений в виде равенств и неравенств. Наличие таких моделей только расширяет совокупность математических методов, привлекаемых для решения рассматриваемой задачи.

К классическим сетевым задачам можно отнести задачу об остовных деревьях, задачу о кратчайших путях, задачу о максимальном потоке в сети, задачу коммивояжера, задача о максимальном паросочетании двудольного графа, задачи сетевого планирования и управления.

Существует ряд задач, в которых сетевые модели используются как вспомогательные. Например, в задаче о назначениях алгоритм поиска максимального потока используется в качестве одного из этапов получения решения.

1. ЗАДАЧА О МИНИМАЛЬНОМ ОСТОВНОМ ДЕРЕВЕ НЕОРИЕНТИРОВАННОГО ГРАФА

Напомним, что дерево это ациклический связный граф. Подграф графа называется остовным, если он содержит все вершины исходного графа. Пусть дан связный неориентированный граф $G(V, E)$, с числом вершин $|V| = n$, числом ребер $|E| = m$, и каждому его ребру $e \in E$ приписан некоторый вес $w(e)$. Вес подграфа $G'(V', E')$ графа $G(V, E)$ определим как сумму весов его ребер, то есть

$$w(G') = \sum_{e \in E'} w(e).$$

Задача о минимальном остовном дереве состоит в поиске остовного дерева исходного графа с минимальным весом.

Приведем базовые алгоритмы решения задачи. Обозначим через I_j совокупность ребер, включенное в искомое дерево на j-ой итерации алгоритма.

Алгоритм Краскала. Начальный шаг: $j = 0$, $I_0 = \varnothing$.

Шаг 1. Упорядочиваем ребра множества E в порядке неубывания их весов. Пусть $w(e_1) \le w(e_2) \le \ldots \le w(e_m)$.

Шаг 2.

$$I_{j+1} = \begin{cases} I_j \cup e_j, \text{ если } I_j \cup e_j \text{ не содержит цикла}, \\ I_j, \text{ в противном случае}. \end{cases}$$

Шаг 3. Если $|I_{j+1}| = n$-1, стоп. I_{j+1} – совокупность ребер искомого дерева. В противном случае $j := j$+1, идем к шагу 2.

Алгоритм Краскала выполняет не более чем m итераций, сложность алгоритма O($m \log m$).

Обозначим через O_e (I_j) множество ребер из $E \setminus I_j$, в котором каждое ребро инцидентно с некоторым ребром из I_j . Полагаем O_e ($\varnothing$) = E.

Алгоритм Прима. Начальный шаг: $j = 0$, $I_0 = \varnothing$.

Шаг 1. Строим множество O_e (I_j) и упорядочиваем его ребра по неубыванию весов. Пусть O_e (I_j) = $\{ e_1^j, e_2^j, \ldots, e_{l_j}^j \}$ и $w(e_1^j) \le w(e_2^j) \le \ldots \le w(e_{l_j}^j)$.

Шаг 2.1. $k = 1$.

Шаг 2.2.

$$I_{j+1} = \begin{cases} I_j \cup e_k^j, \text{ если } I_j \cup e_k^j \text{ не содержит цикл}, \\ I_j, \text{ в противном случае}. \end{cases}$$

Шаг 2.3. Если $I_{j+1} \neq I_j$, то идем к шагу 3. В противном случае $k := k + 1$ и возвращаемся к шагу 2.2.

Шаг 3. Если $|I_{j+1}| = n-1$, стоп. I_{j+1} – совокупность ребер искомого дерева. В противном случае $j := j+1$, идем к шагу 1.

Сложность алгоритма Прима составляет $O(m + n \log n)$.

Для подграфа $G'(V', E')$ графа $G(V, E)$ обозначим через $O_v(G')$ множество ребер из $E \setminus E'$ инцидентных ровно с одной вершиной множества V'. Через $V(I_j)$ обозначим множество вершин графа $G(V, E)$ инцидентных ребрам из I_j. При этом положим $V(\varnothing) = V$.

Алгоритм Борувки. Начальный шаг: $j = 0$, $I_0 = \varnothing$.

Шаг 1. Упорядочиваем ребра множества E в порядке неубывания их весов.

Шаг 2. Выделяем компоненты связности $K_1^j, K_2^j, ..., K_{p_j}^j$ подграфа $G'(V(I_j), I_j)$, образованного ребрами I_j.

Шаг 3. Для каждого K_i^j, $i = \overline{1, p_j}$, находим первое в упорядочении, полученном на шаге 1, ребро e_i^j из $O_v K_i^j$. Очевидно, $w(e_i^j) = \min\{w(e) | e \in O_v(K_i^j)\}$.

Шаг 4. Полагаем $I_{j+1} = I_j \cup \{e_i^j,\ i = \overline{1, p_j}\}$.

Шаг 5. Если $|I_{j+1}| = n-1$, стоп. I_{j+1} – совокупность ребер искомого дерева. В противном случае $j := j+1$, идем к шагу 2.

Сложность алгоритма Борувки составляет $O(m \log n)$.

Пример. Найти минимальное остовное дерево для графа, представленного на рисунке 1. Числа указывают вес ребра, в скобках указано обозначение ребра с номером соответствующим упорядочению по неубыванию.

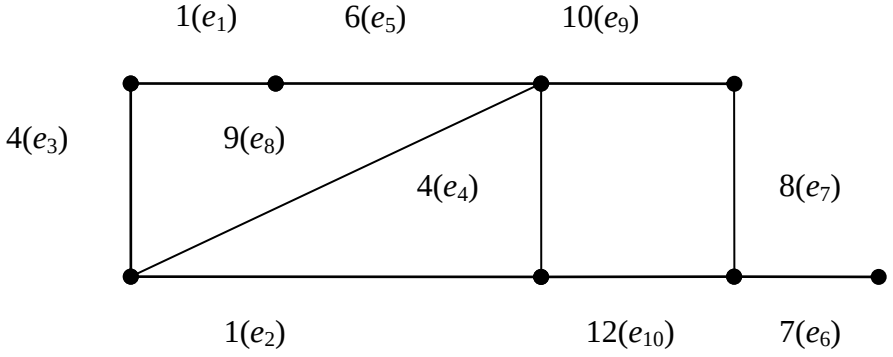


Рис. 1.

Схема алгоритма Крускала представлена на рисунках 2.1. – 2.4.

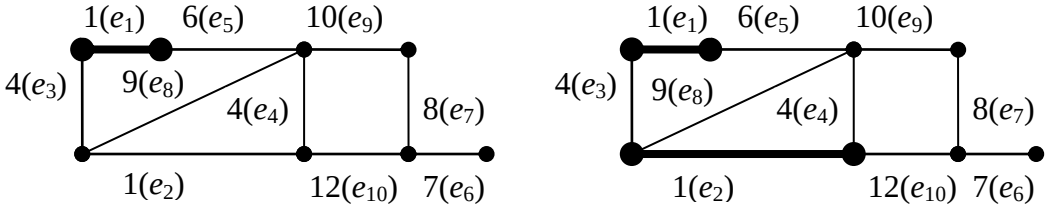


Рис. 2.1. Первые две итерации алгоритма Краскала

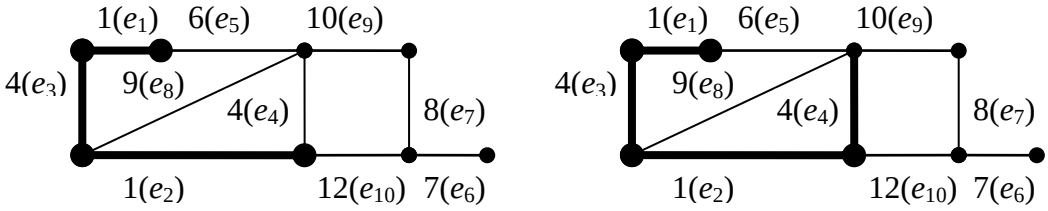


Рис. 2.2. Третья и четвертая итерации алгоритма Краскала

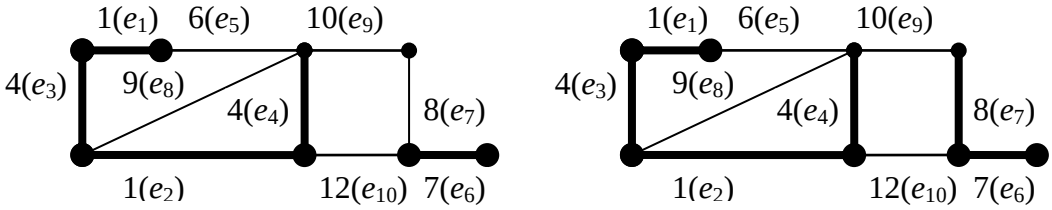


Рис. 2.3. Пятая и шестая итерации алгоритма Краскала

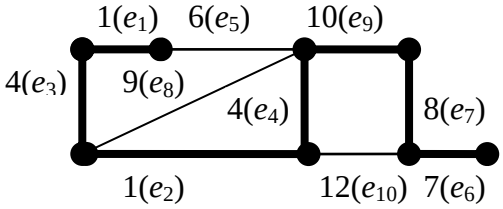


Рис. 2.4. Завершающая итерация алгоритма Крускала.

Схема алгоритма Прима представлена на рисунках 3.1 – 3.7.

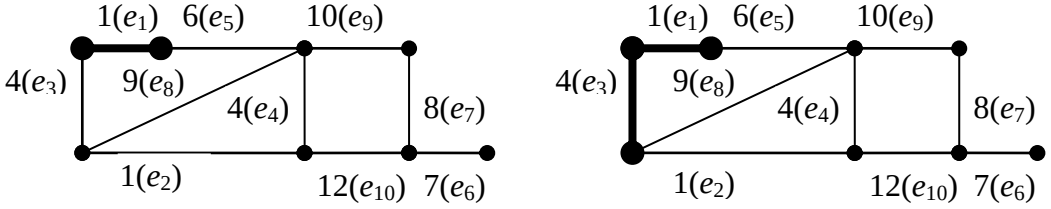


Рис. 3.1. Первые две итерации алгоритма Прима

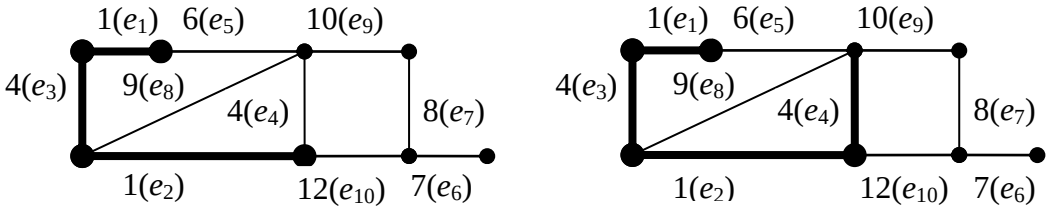


Рис. 3.2. Третья и четвертая итерации алгоритма Прима

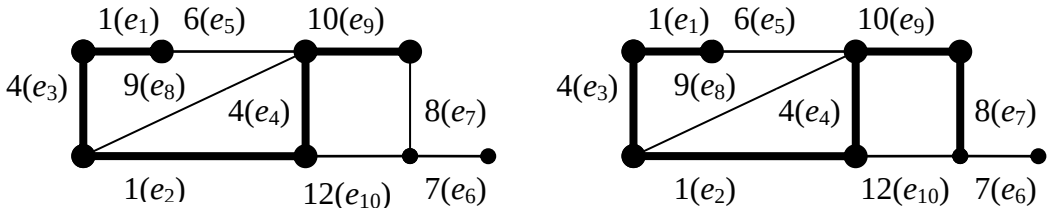


Рис. 3.3. Пятая и шестая итерации алгоритма Прима

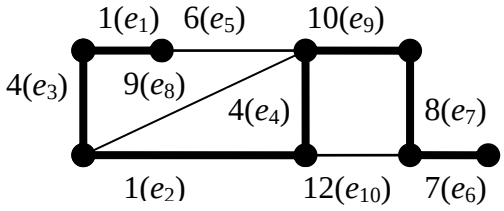


Рис. 3.4. Завершающая итерация алгоритма Прима

Схема алгоритма Борувки представлена на рисунке 4.

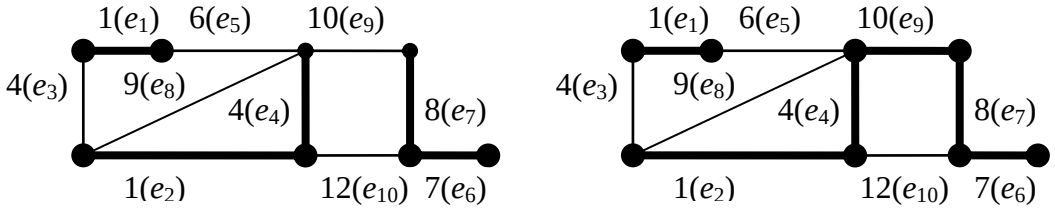


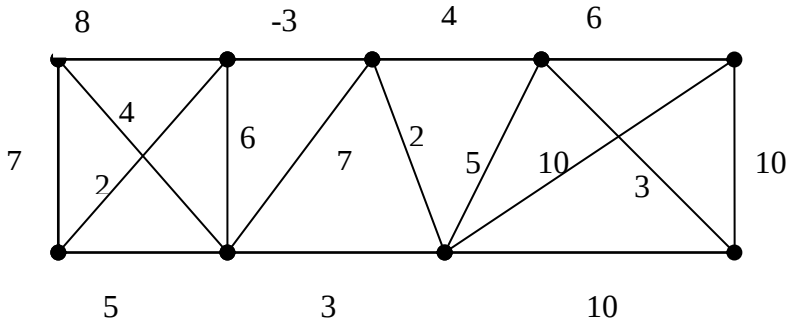
Рис. 4. Первая и завершающая итерации алгоритма Борувки

Указанные алгоритмы Краскала, Прима и Борувки можно применять и для поиска максимального остовного дерева. Для этого необходимо заменить упорядочение по неубыванию на упорядочение по невозрастанию в соответствующих шагах алгоритмов.

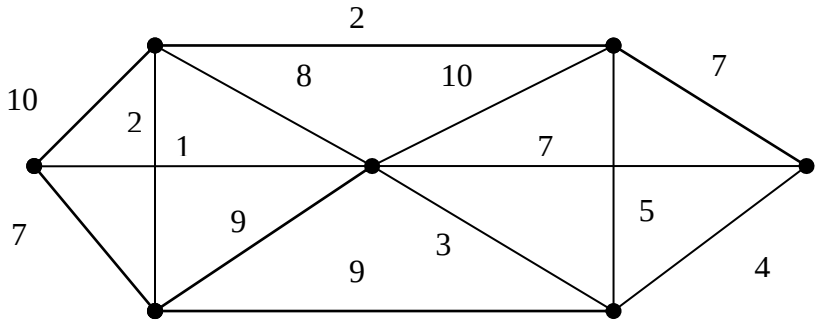
Задачи для самостоятельного решения

1. Для приведенных ниже неориентированных связных графов найти минимальное и максимальное остовные деревья:

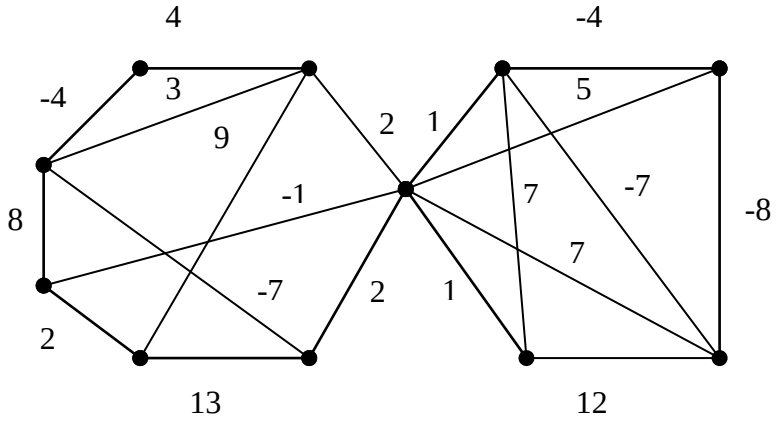
а)



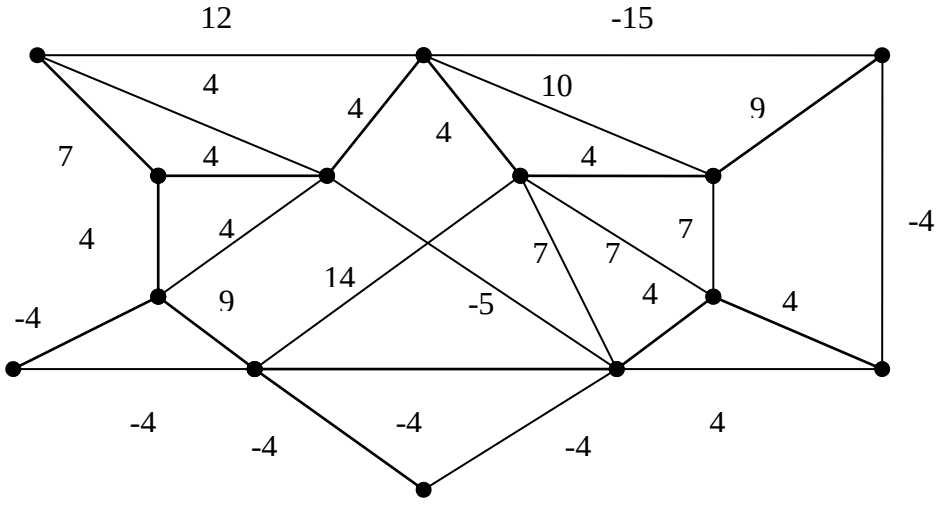
b)



c)



d)



2. КРАТЧАЙШИЕ ПУТИ

Пусть G (V, E) ориентированный граф, дугам которого приписаны веса $w(e)$, $e \in E$, интерпретируемые как длины дуг. Если между вершинами существует путь, то его длина определяется как сумма длин составляю-

щих его дуг. Рассмотрим следующие задачи: 1) для двух выделенных вершин s, $t \in V$ требуется выбрать путь из s в t минимальной длины; 2) для выделенной вершины $s \in V$ необходимо найти пути минимальной длины до всех остальных вершин графа; 3) для каждой пары вершин графа требуется найти кратчайший путь.

Задача 1 не имеет решения, если в графе существует контур отрицательной длины и путь из s в t , содержащий вершину контура. Задача 2 не имеет решение при наличии контура отрицательной длины и существования пути из s в какую-либо вершину контура. Задача 3 не имеет решения при наличии контура отрицательной длины. Поэтому алгоритмы решения задач должны либо обнаруживать наличие контуров отрицательной длины, либо применяться для графов заведомо не содержащих таких контуров.

Все алгоритмы нахождения кратчайших путей основаны на принципе динамического программирования, который применительно к данной задаче означает, что путь s , … , x , … , t является кратчайшим от вершины s к вершине t тогда и только тогда, когда путь s , … , x является кратчайшим от вершины s к вершине x, а путь x , … , t является кратчайшим от вершины x к вершине t.

Алгоритм Дейкстры применяется для поиска решения в задачах 1,2 при $w(e) \geq 0$, $e \in E$. Алгоритм основан на присвоении вершинам меток. Первая часть метки $l\,(u)$ вершины u указывает текущее кратчайшее расстояние от вершины s до вершины u, вторая – предшествующую вершину в текущем пути от вершины s до вершины u. Метки могут быть временными или постоянными. Постоянная метка не может изменяться в процессе выполнения алгоритма.

Для вершины $p \in V$ пусть $O^+(p) = \{ x \in V \mid \exists\, (p, x) \in E \}$.

Шаг 1. Положить метку вершины s равной $(0, s)$ и считать эту метку постоянной. Для всех вершин $u \neq s$ положить метки равными (∞, s) и считать эти метки временными. За текущую рассматриваемую вершину с постоянной меткой взять вершину $p = s$.

Шаг 2. Для каждой вершины $u \in O^+(p)$ с временной меткой изменить метку в соответствии со следующим выражением:

$$l(u) = \min [\, l(u), l(p) + w(p, u) \,].$$

При этом, если первая часть метки изменилась, то изменить вторую часть метки, положив ее равной p.

Шаг 3. Среди всех вершин с временными метками найти вершину u с минимальной первой частью $l(u)$ метки.

Шаг 4. Считать метку вершины u постоянной и положить $p = u$.

Шаг 5(а). (При нахождении пути от s к t). Если $p = t$, то l (t) является длиной кратчайшего пути от s к t. Алгоритм завершает работу.

Если $p \neq t$, перейти к шагу 2.

Шаг 5(б). (При нахождении путей от s ко всем вершинам). Если все вершины помечены постоянными метками, то эти метки дают длины кратчайших путей. Алгоритм завершает работу.

Если некоторые метки являются временными, то перейти к шагу 2.

По завершении алгоритма первые части меток дают искомые кратчайшие расстояния. Сами кратчайшие пути можно получить с помощью рекурсивной процедуры, начиная с вершины t , для которой ищется кратчайший путь от s . Вторая часть метки вершины t указывает вершину u непосредственно предшествует ей в кратчайшем пути от s к t. Берем вторую часть метки вершины u и повторяем действия, пока не достигнем вершины s.

Сложность алгоритма Дейкстры равна O ($|V|^2$).

Пример. Найти кратчайшее расстояние от вершины s до вершины t в следующем графе:

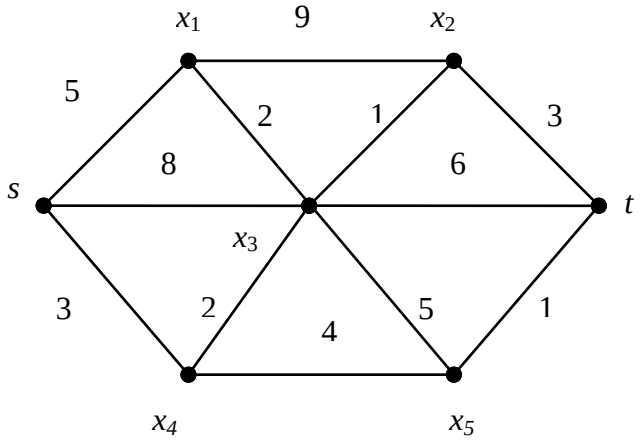


Таблица 1

s	x_1	x_2	x_3	x_4	x_5	t
(0, s)*	(∞, s)	(∞, s)	(∞, s)	(∞, s)	(∞, s)	(∞, s)
	(5, s)	(∞, s)	(8, s)	(3, s)*	(∞, s)	(∞, s)
	(5, s)*	(∞, s)	(5, x_4)		(7, x_4)	(∞, s)
		(14, x_1)	(5, x_4)*		(7, x_4)	(∞, s)
		(6, x_3)*			(7, x_4)	(11, x_3)
					(7, x_4)*	(9, x_2)
						(8, x_5)*

Граф является неориентированным, поэтому рассматриваем его ребра, как пары противоположно направленных дуг.

Метки вершин в процессе выполнения алгоритма приведены в таблице 1. Символом * отмечены постоянные метки.

В данном примере все вершины графа получими постоянные метки. Поэтому найдены кратчайшие расстояния от вершины *s* до всех остальных вершин графа.

Длина кратчайшего пути от *s* к *t* равна 8. Сам кратчайший путь от *s* к *t* строим по вторым частям меток. Получим путь $s \to x_4 \to x_5 \to t$.

Алгоритм Форда-Беллмана применяется для решения задачи 2. Ограничений на длину дуг не накладывается. Алгоритм позволяет определить наличие контура отрицательной длины в графе.

Пронумеруем вершины исходного графа номерами от 1 до $n = |V|$, причем вершине *s* присвоим первый номер. Составим матрицу расстояний $D = |d_{ij}|_{nxn}$ для исходного графа, положив

$$d_{ij} = \begin{cases} 0, i = j, \\ w(x_i, x_j), (x_i, x_j) \in E, \\ \infty, (x_i, x_j) \notin E . \end{cases}$$

Так же как в алгортме Дейкстры будем присваивать вершине x_i метку $(l_k(x_i), x_k)$, где $l_k(x_i)$ – кратчайшее расстояние от вершины x_1 до вершины x_i , по всем путям содержащим не более чем *k* дуг, x_k – вершина предшествующая x_i в кратчайшем пути, содержащим не более чем *k* дуг.

Шаг 1. Положить $k = 1$, $l_1(x_i) = d_{1i}$, вторую часть метки для всех вершин равной x_1.

Шаг 2. Найти $l_{k+1}(x_i) = \min\{ l_k(x_i), \min\limits_{1 \le j \le n} (l_k(x_j) + d_{ji}) \}$ для всех $i = 1, \ldots, n$. При $l_{k+1}(x_i) \neq l_k(x_i)$ вторая часть метки вершины x_i полагается равной вершине на которой достигается $\min\limits_{1 \le j \le n} (l_k(x_j) + d_{ji})$.

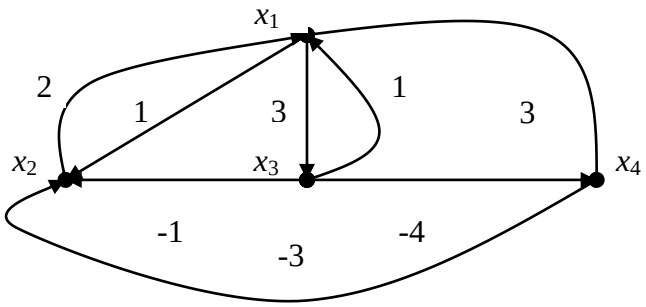
Шаг 3. Если $l_{k+1}(x_i) \neq l_k(x_i)$ для некоторой вершины x_i и $k + 1 \le n$, то палагаем $k := k + 1$ и возвращаемся к шагу 2.

Если $l_{k+1}(x_i) \neq l_k(x_i)$ для некоторой вершины x_i и $k + 1 = n + 1$, то граф имеет контур отрицательной длины. Задача не имеет решения.

Если $l_{k+1}(x_i) = l_k(x_i)$ для всех $i = 1, \ldots, n$, и $k + 1 \le n + 1$, то метки $l_{k+1}(x_i)$ дают кратчайшие расстояния.

Сами кратчайшие пути строятся так же, как в алгоритме Дейкстры по вторым частям меток. Сложность алгоритма Форда-Беллмана равна $O(|V|^3)$.

Пример. Найти кратчайшее расстояние от вершины x_1 до всех остальных вершин в следующем графе:



Матрица расстояний для данного графа имеет вид

$$D=\begin{vmatrix} 0 & 1 & 3 & 3 \\ 2 & 0 & \infty & \infty \\ 1 & -1 & 0 & -4 \\ 3 & -3 & \infty & 0 \end{vmatrix}.$$

Метки вершин в процессе выполнения алгоритма приведены в таблице 2.

Таблица 2.

	0	1	3	3	0	0	0	-2
	2	0	∞	∞	1	0	-4	-4
	1	-1	0	-4	3	3	3	3
	3	-3	∞	0	3	-1	-1	-1
1	(0, x_1)	(1, x_1)	(3, x_1)	(3, x_1)				
2	(0, x_1)	(0, x_4)	(3, x_1)	(-1, x_3)				
3	(0, x_1)	(-4, x_4)	(3, x_1)	(-1, x_3)				
4	(-2, x_2)	(-4, x_4)	(3, x_1)	(-1, x_3)				
5	(-2, x_2)	(-4, x_4)	(1, x_1)	(-1, x_3)				

В таблице первый столбец содержит номера итераций. Верхняя часть таблицы содержит элементы матрицы D. Строки с пятой по девятую в столбцах со второго по пятый содержат метки вершин после выполнения соответствующей итерации. Столбцы с шестого по девятый содержат отображение первых частей меток относительно главной диагонали таблицы. Они предназначены для поиска минимума на втором шаге алгоритма. Суммируя отображения, соответствующие некоторой итерации, с элементами столбца матрицы расстояний, выбираем минимальное значение, которое сравниваем с имеющейся меткой вершины.

Метка вершины x_3 изменилась на пятой итерации. Следовательно, граф имеет контур отрицательной длины. Его легко построить по вторым частям меток: $x_3 \to x_4 \to x_2 \to x_1 \to x_3$. Задача не имеет решения.

Алгоритм Флойда применяется для решения задачи 3. Ограничений на длины дуг не накладывается. Алгоритм обнаруживает контур отрицательной длины.

Нумеруем вершины графа номерами от 1 до $n = |V|$ и строим матрицу расстояний $D = |d_{ij}|_{nxn}$ для исходного графа, по аналогии с алгоритмом Форда-Беллмана.

Пусть d_{ij}^k значение кратчайшего расстояния от вершины x_i к вершине x_j по всем путям, содержащим вершины с номерами не более k.

Шаг 1. Положить $k = 0$, и построить матрицы $D^0 = |d_{ij}^0|_{nxn}$, $T^0 = |t_{ij}^0|_{nxn}$ по правилу: $d_{ij}^0 = d_{ij}$, $t_{ij}^0 = j$, $i, j = \overline{1,n}$. Здесь $T^0 = |t_{ij}^0|_{nxn}$ – вспомогательная матрица, предназначенная для построения матрицы кратчайших путей.

Шаг 2. Построить матрицы $D^{k+1} = |d_{ij}^{k+1}|_{nxn}$, $T^{k+1} = |t_{ij}^{k+1}|_{nxn}$ по матрицам $D^k = |d_{ij}^k|_{nxn}$ и $T^k = |t_{ij}^k|_{nxn}$, вычисляя их элементы по формулам:

$$d_{ij}^k = \begin{cases} d_{ij}^k,\ i = k+1,\ j = \overline{1,n}, \\ d_{ij}^k\ ,\ j = k+1,\ i = \overline{1,n}, \\ \min\{d_{ij}^k, d_{ik+1}^k + d_{k+1j}^k\}, i \neq k+1,\ j \neq k+1, \end{cases} \qquad t_{ij}^{k+1} = \begin{cases} t_{ij}^k, \text{если } d_{ij}^k = d_{ij}^k, \\ j, \text{если } d_{ij}^k \neq d_{ij}^k \end{cases}.$$

Шаг 3. Если $k + 1 \leq n$ и для некоторого i элемент $d_{ii}^{k+1} < 0$, то граф имеет контур отрицательной длины. Задача не имеет решения.

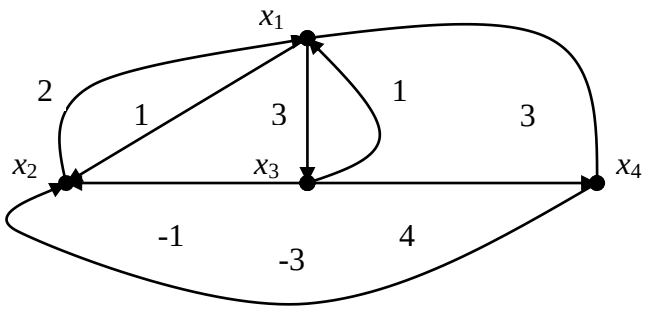
Если $k + 1 < n$ и $d_{ii}^{k+1} \geq 0$, $i = \overline{1,n}$, то положить $k := k + 1$ и перейти к шагу 2.

Если $k+1 = n$ и $d_{ii}^{k+1} \geq 0$, $i = \overline{1,n}$, то матрица D^n дает кратчайшие расстояния между парами вершин.

Кратчайший путь от вершины x_i к вершине x_j строится по элементам матрицы T^n. Элемент $t_{ij}^n = m$ указывает промежуточную вершину x_m в кратчайшем пути от вершины x_i к вершине x_j. Находим t_{im}^n и t_{mj}^n, которые указывают очередные промежуточные вершины. Если $t_{im}^n = m$ $t_{mj}^n = j$, то вершина x_m непосредственно следует за вершиной x_i (непосредственно предшествует вершине x_j). Процесс завершаем при получении номера вершины непосредственно следующей за x_i и номера вершины непосредственно предшествующей x_j.

Сложность алгоритма Флойда равна $O(|V|^3)$.

Пример. Найти кратчайшие расстояния между каждой парой вершин следующего графа:



Получим:

$$D^0=\begin{vmatrix}0&1&3&3\\2&0&\infty&\infty\\1&-1&0&4\\3&-3&\infty&0\end{vmatrix},\ T^0=\begin{vmatrix}1&2&3&4\\1&2&3&4\\1&2&3&4\\1&2&3&4\end{vmatrix},\quad D^1=\begin{vmatrix}0&1&3&3\\2&0&5&5\\1&-1&0&4\\3&-3&6&0\end{vmatrix},\ T^1=\begin{vmatrix}1&2&3&4\\1&2&1&1\\1&2&3&4\\1&2&1&4\end{vmatrix},$$

$$D^2=\begin{vmatrix}0&1&3&3\\2&0&5&5\\1&-1&0&4\\-1&-3&2&0\end{vmatrix},\ T^2=\begin{vmatrix}1&2&3&4\\1&2&1&1\\1&2&3&4\\2&2&2&4\end{vmatrix},\quad D^3=\begin{vmatrix}0&1&3&3\\2&0&5&5\\1&-1&0&4\\-1&-3&2&0\end{vmatrix},\ T^3=\begin{vmatrix}1&2&3&4\\1&2&1&1\\1&2&3&4\\2&2&2&4\end{vmatrix},$$

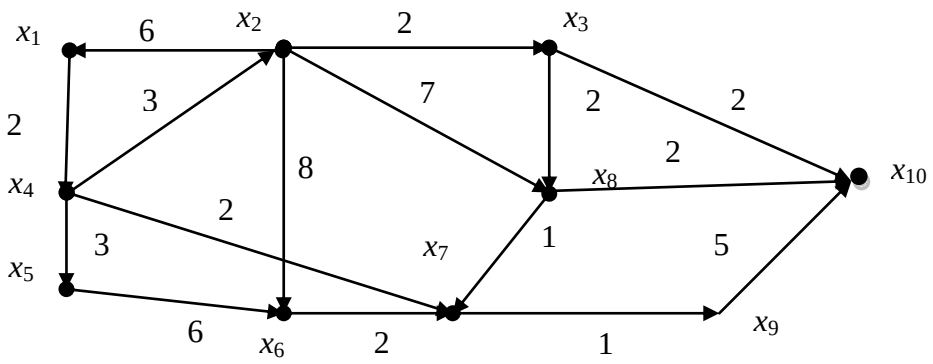
$$D^4=\begin{vmatrix}0&0&3&3\\2&0&5&5\\1&-1&0&4\\-1&-3&2&0\end{vmatrix},\ T^4=\begin{vmatrix}1&4&3&4\\1&2&1&1\\1&2&3&4\\2&2&2&4\end{vmatrix}.$$

Матрица D^4 дает значения кратчайших расстояний между вершинами. Например, кратчайшее расстояние между четвертой и третьей вершинами равно 2. Сам кратчайший путь строим по матрице T^4. Получим $x_4 \to x_2 \to x_1 \to x_3$.

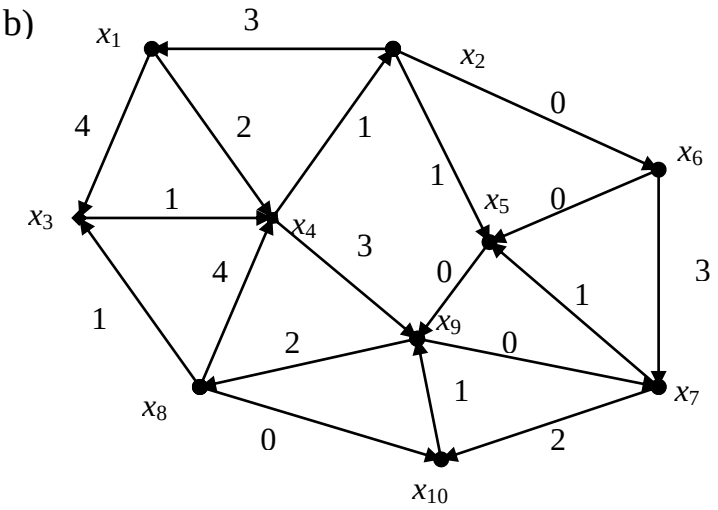
Задачи для самостоятельного решения

1. Найти кратчайший путь от вершины x_1 до вершины x_7 в следующих графах:

а)

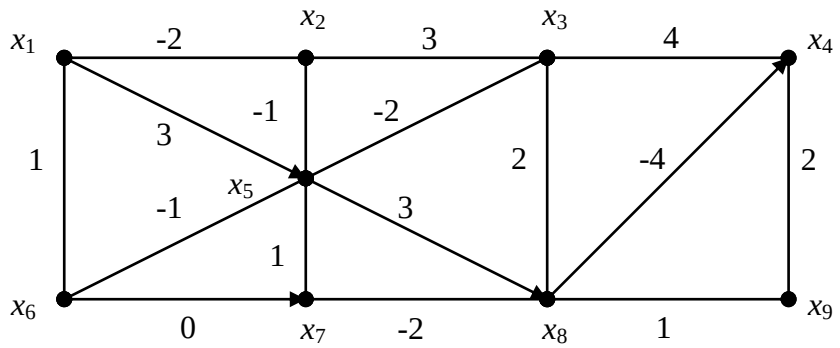


b)

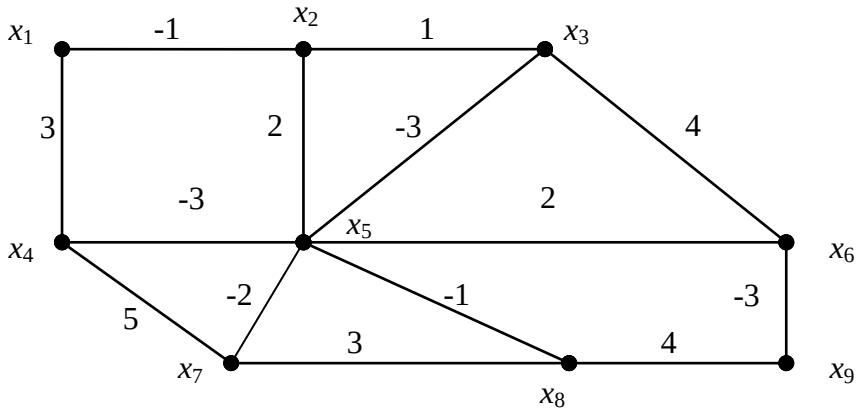


2. Найти кратчайшие расстояния от вершины x_1 до всех остальных вершин в следующих графах:

a)



b)



3. Для графов примеров 1 a), 1 b) найти кратчайшие расстояния от вершины x_2 до всех остальных вершин.

4. Определить кратчайшие расстояния между каждой парой вершин для графов со следующими матрицами расстояний:

a) $\begin{vmatrix} 0 & -15 & 15 & \infty & \infty \\ 20 & 0 & 7 & 1 & \infty \\ 8 & \infty & 0 & -10 & -3 \\ \infty & 2 & \infty & 0 & 6 \\ \infty & \infty & 14 & 4 & 0 \end{vmatrix}$ b) $\begin{vmatrix} 0 & 11 & 2 & 8 & 11 & 11 \\ 11 & 0 & 5 & \infty & \infty & 1 \\ \infty & 5 & 0 & \infty & 2 & 1 \\ 2 & \infty & \infty & 0 & 2 & \infty \\ \infty & 9 & \infty & 2 & 0 & 7 \\ \infty & 1 & \infty & \infty & 7 & 0 \end{vmatrix}$

c) $\begin{vmatrix} 0 & 1 & \infty & 5 & 9 & -3 \\ 1 & 0 & -1 & 7 & 6 & \infty \\ \infty & -1 & 0 & \infty & 7 & 5 \\ 5 & 7 & \infty & 0 & 6 & -2 \\ 9 & 6 & 7 & 6 & 0 & -4 \\ -3 & \infty & 5 & -2 & -4 & 0 \end{vmatrix}$ d) $\begin{vmatrix} 0 & 3 & 2 & 7 & 2 & \infty & \infty \\ 3 & 0 & 1 & -1 & 1 & 7 & \infty \\ 2 & 5 & 0 & 9 & \infty & \infty & -3 \\ 7 & 1 & \infty & 0 & 1 & \infty & 3 \\ 2 & 1 & \infty & 1 & 0 & \infty & \infty \\ \infty & \infty & 2 & \infty & 5 & 0 & 2 \\ \infty & \infty & 3 & \infty & \infty & 2 & 0 \end{vmatrix}$

3. ПОТОКИ В СЕТЯХ

Пусть каждой дуге (x,y) графа $G=(V,E)$ поставлено в соответствие положительное число $c(x,y)$, интерпретируемое как пропускная способность дуги. Зафиксируем две вершины $s,t\in E$. Вершину s назовем источником, а вершину t – стоком.

Стационарным потоком величины v из вершины s в вершину t на сети $G=(V,E)$ называется функция $f(x,y)$, заданная на всех дугах $(x,y)\in E$ и удовлетворяющая следующим условиям:

$$0\le f(x,y)\le c(x,y),\quad \forall (x,y)\in E, \tag{1}$$

$$\sum_{y\in O^+(x)} f(x,y)-\sum_{y\in O^-(x)} f(y,x)=\begin{cases} v, & \text{если } x=s,\\ 0, & \text{если } x\ne s,t,\\ -v, & \text{если } x=t. \end{cases} \tag{2}$$

Здесь $O^+(x)=\{y\in V\mid \exists\,(x,y)\in E\}$, $O^-(x)=\{y\in V\mid \exists\,(y,x)\in E\}$.

Условия (1) означают, что поток по каждой дуге не должен превышать ее пропускную способность. Условия (2) показывают, что для источника s суммарное количество входящего и выходящего потока должно быть равно v – величине потока в сети. Аналогично для стока t суммарное количество выходящего и входящего потока также равно v.

Для всех промежуточных вершин сети должно выполняться равенство $\sum_{y\in O^+(x)} f(x,y) = \sum_{y\in O^-(x)} f(x,y)$.

В сети с пропускными способностями дуг $c(x,y)>0$ всегда существует стационарный поток (например, величины 0), причем не единственный.

3.1. Задача о максимальном потоке

Задача о максимальном потоке состоит в нахождении для данной сети стационарного потока максимально возможной величины.

Назовем цепь из *s* в *t* увеличивающей поток, если на всех ее дугах (*x*, *y*) совпадающих по направлению от *s* к *t* (прямые дуги), выполняется неравенство $f(x,y)<c(x,y)$, а на всех дугах, не совпадающих по направлению от *s* к *t* (обратные дуги) – неравенство $f(x,y)>0$.

Поток является максимальным, тогда и только тогда, когда в сети не существует цепи, увеличивающей поток.

Пусть $S\subset V$, $S\neq\varnothing$ $\overline{S}=V\setminus S$. Разрезом назовем множество $<S,\overline{S}>=\{(x,y)\in E | x\in S, y\in\overline{S}\}$. Если $s\in S, t\in\overline{S}$, то говорим, что разрез $<S,\overline{S}>$ отделяет источник *s* от стока *t*. Пропускная способность разреза $<S,\overline{S}>$ определяется следующим образом $c(S)=\sum_{x\in S, y\in\overline{S}} c(x,y)$.

Разрез, отделяющий источник от стока, с минимальной пропускной способностью называют миннимальным разрезом. Теорема Форда-Фалкерсона о максимальном потоке и минимальном разрезе устанавливает, что в сети величина максимального потока равна пропускной способности минимального разреза.

Алгоритм Форда-Фалкерсона для нахождения максимального потока начинает свою работу с произвольного начального потока в сети. Например, нулевого потока.

Алгоритм на каждой итерации состоит из двух этапов.

Этап 1 (расстановка меток). На этом этапе ищется цепь, увеличивающая поток. Поиск такой цепи осуществляется с помощью расстановки меток. В процессе расстановки меток каждая вершина может находиться в одном из трех состояний: непомеченная, помеченная и непросмотренная, помеченная и просмотренная.

В начале все вершины непомечены.

Источник *s* получает метку вида $(-,\infty)$. После этого *s* переходит в состояние «помечен и непросмотрен».

Пусть существует несколько помеченных и непросмотренных вершин. Выберем среди них вершину *x*. Просматриваем непомеченные вершины $y \in O^+(x)$ с проверкой условия $f(x,y) < c(x,y)$. Если для дуги (x,y), неравенство имеет место, то вершина *y* получает метку $(x^+, \varepsilon(y))$, где

$$\varepsilon(y) = \min[\varepsilon(x), c(x,y) - f(x,y)] \quad .$$

Затем просматриваем непомеченные вершины $y \in O^-(x)$ и проверяем условие $f(y,x) > 0$. Если для дуги (y, x) неравенство выполняется, то вершина *y* получает пометку $(x^-, \varepsilon(y))$, где

$$\varepsilon(y) = \min[\varepsilon(x), f(y,x)] \ .$$

После этого вершина *x* переходит в состояние «помеченная и просмотренная».

Далее выбирается очередная помеченная и непросмотренная вершина, и для нее повторяются все описанные выше действия. Причем придерживаемся правила: «первый помечен − первый просмотрен».

Расстановка пометок продолжается до тех пор, пока или будет помечена вершина *t*, или сток *t* никоим образом пометить нельзя.

В первом случае найден увеличивающий путь. Переходим к этапу *2*.

Во втором случае алгоритм заканчивает свою работу и это означает, что максимальный поток найден на предыдущей итерации.

При этом определяется соответствующий полученному максимальному потоку минимальный разрез сети. Разрез будет состоять из дуг, идущих из помеченных вершин в непомеченные на последней итерации вершины, при этом все дуги минимального разреза насыщены.

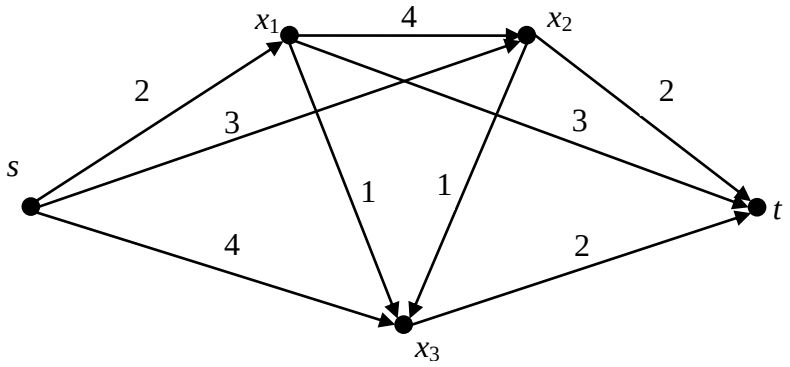
Этап 2 (увеличение потока). Сток *t* может получить одну из двух пометок $(x^+, \varepsilon(t))$ или $(x^-, \varepsilon(t))$. Если *t* имеет пометку $(x^+, \varepsilon(t))$, поток по дуге (x,t) увеличивается на значение $\varepsilon(t)$. Если *t* имеет пометку $(x^-, \varepsilon(t))$, поток по дуге (t,x), уменьшается на значение $\varepsilon(t)$.

В любом из этих случаев переходим к вершине *x*, которая указана в пометке вершины *t*. Эта вершина имеет пометку $(y^+, \varepsilon(x))$ или $(y^-, \varepsilon(x))$. В первом случае по дуге (y,x) увеличивает поток на $\varepsilon(t)$, во втором – поток по дуге (x,y) уменьшается на $\varepsilon(t)$. Изменение потоков на величину $\varepsilon(t)$ по дугам повторяется до тех пор, пока не будет достигнута вершина *s*.

Стираем у вершин все метки и возвращаемся к этапу 1 с новым увеличенным потоком.

В изложенном варианте сложность алгоритма Форда-Фалкерсона равна $O(|V|\,|E|^2)$.

Пример. Для следующей сети (на дугах заданы пропускные способности) найти максимальный поток.



Начинаем с нулевого потока. Метки вершин на итерациях приведены в таблице 3.

Изменение потока на дугах сети приведены на рисунке 5. Номер итерации, на которой проводится увеличение потока, указан у величины изменения нижним индексом.

Таблица 3

№	s	x_1	x_2	x_3	t	ν
1	$(-, \infty)$	$(s^+, 2)$	$(s^+, 3)$	$(s^+, 4)$	$(x_1^+, 2)$	0+2
2	$(-, \infty)$		$(s^+, 3)$	$(s^+, 4)$	$(x_2^+, 2)$	2+2
3	$(-, \infty)$		$(s^+, 1)$	$(s^+, 4)$	$(x_3^+, 2)$	4+2
4	$(-, \infty)$		$(s^+, 1)$	$(s^+, 2)$		

По последней (четвертой) итерации строим минимальный разрез. Получаем $S = \{s, x_2, x_3\}$. Следовательно, $<S, \overline{S}>=\{(s, x_1), (x_2, t), (x_3, t)\}$. Величина максимального потока $\nu = 6$.

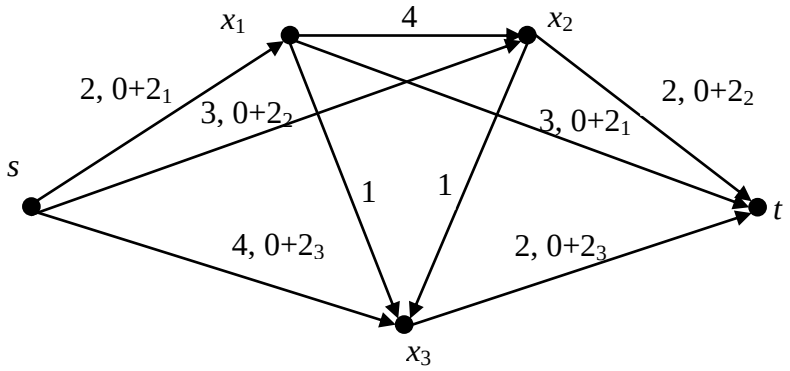


Рис. 5

Алгоритм Форда-Фалкерсона можно применять для решения задачи, при наличия дополнительных условий.

1) Заданы ограничения на пропускные способности вершин. Тогда вершина x с заданной пропускной способностью $c(x) > 0$ заменяется дугой (x', x''), пропускная способность которой полагается равной $c(x)$. При этом каждая дуга исходной сети (y, x), входящая в вершину x, заменяется дугой (y, x'), а каждая дуга (x, y), выходящая из x заменяется дугой (x'', y). Пропускные способности новых дуг принимаются равными пропускным способностям заменяемых дуг.

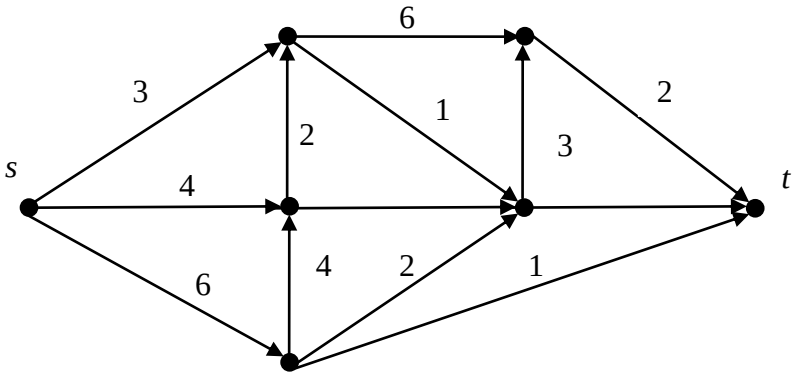
2) Сеть имеет несколько источников $s_1, \dots, s_k$ и стоков $t_1, \dots, t_r$. Необходимо найти максимальный поток из всех источников во все стоки. Вводим фиктивный источник s и фиктивный сток t. В сеть добавляем дуги (s, s_i), $i=\overline{1,k}$, (t_j, t), $j=\overline{1,r}$, с пропускными способностями $c(s, s_i) = \sum_{x \in O^+(s_i)} c(s_i, x)$, $c(t_j, t) = \sum_{x \in O^-(t_j)} c(x, t_j)$.

3) Для дуг заданы нижние и верхние пропускные способности. То есть величина потока по дуге (x, y) не может быть меньше нижней пропускной способности дуги $h(x, y)$ и не может превышать верхнюю пропускную способность $c(x, y)$. В предположении существования потока и найденного какого-либо начального потока, модификация алгоритма Форда-Фалкерсона состоит в вычислении $\varepsilon(y)$ для вершины x по входящей дуге (y, x) с $f(y, x) > h(y, x)$ по формуле $\varepsilon(y) = \min[\varepsilon(x), f(y, x) - h(y, x)]$.

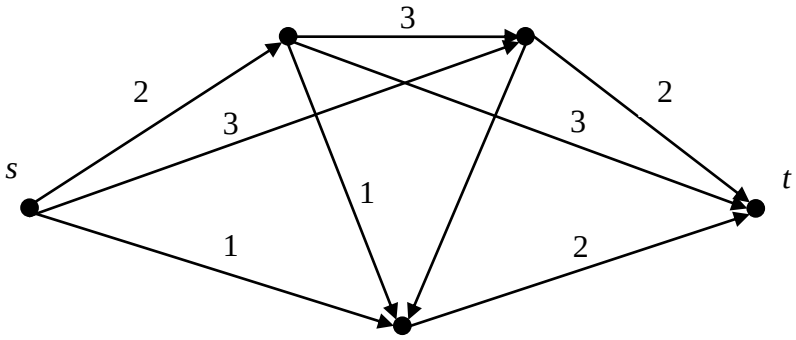
Задачи для самостоятельного решения

1. В следующих сетях найти максимальный поток из источника s в сток t и сответствующий минимальный разрез.

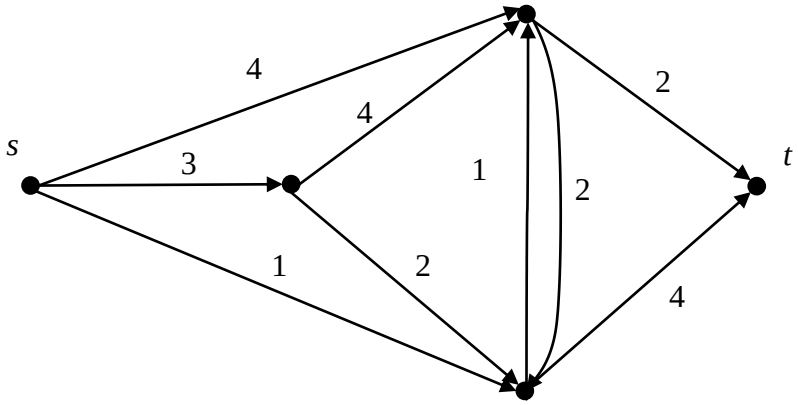
а)



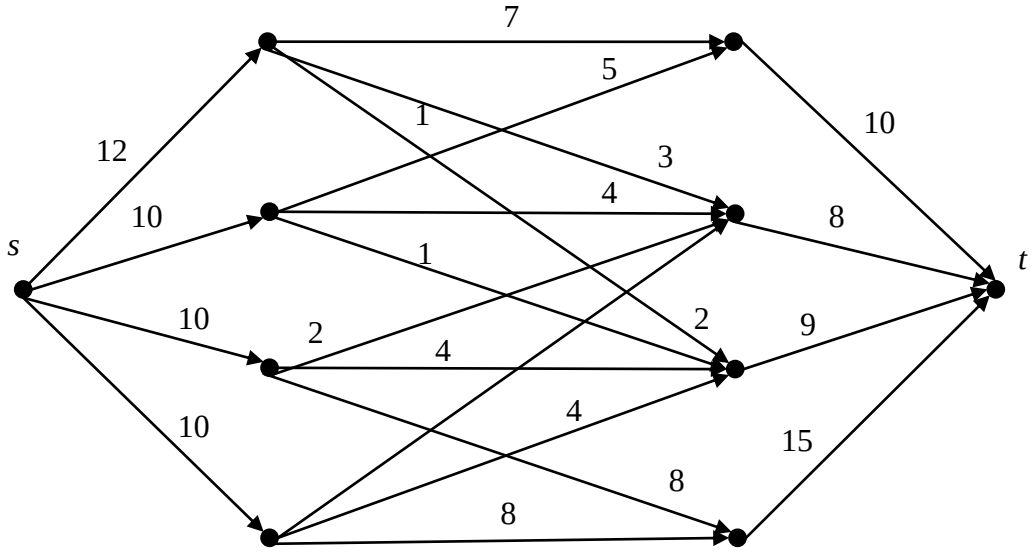
b)



c)

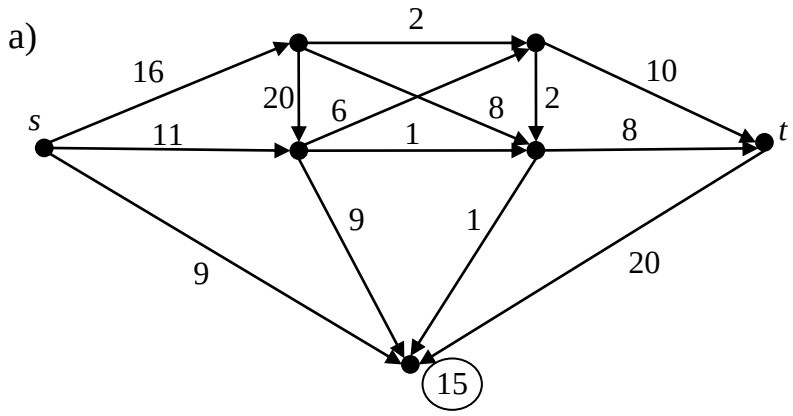


d)

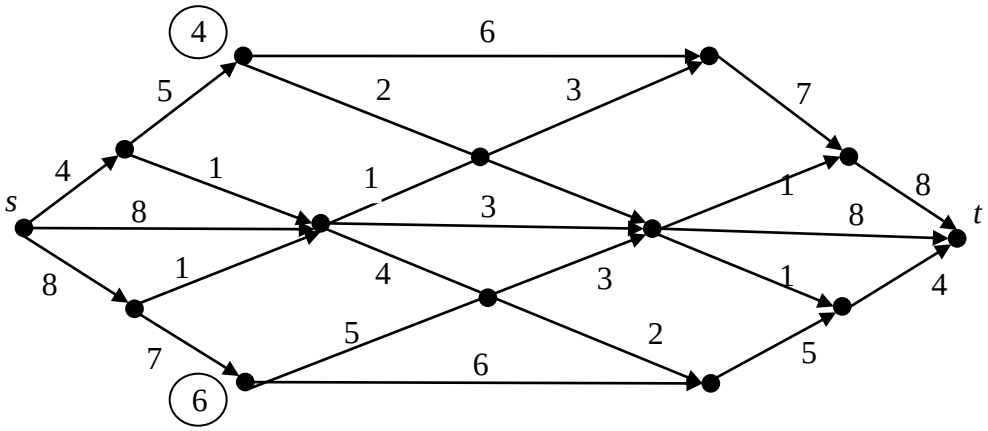


2. Найти максимальный поток из источника s в сток t и сответствующий минимальный разрез при заданных пропускных способностях для некоторых вершин (указаны в кружочках).

a)

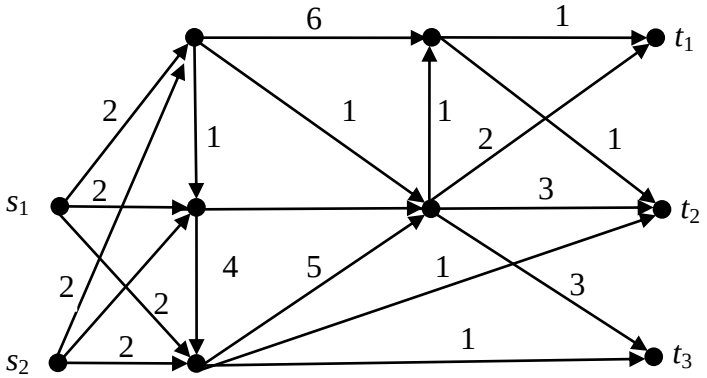


b)

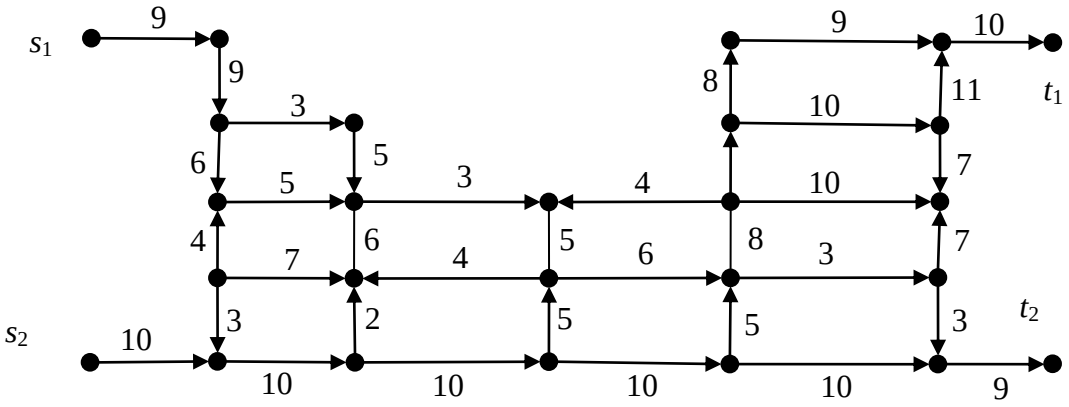


3. В следующих сетях найти максимальный поток из источников в стоки и сответствующий минимальный разрез.

a)

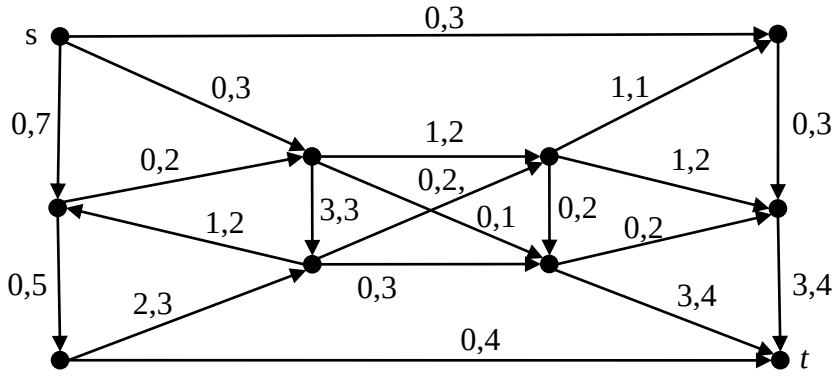


b)

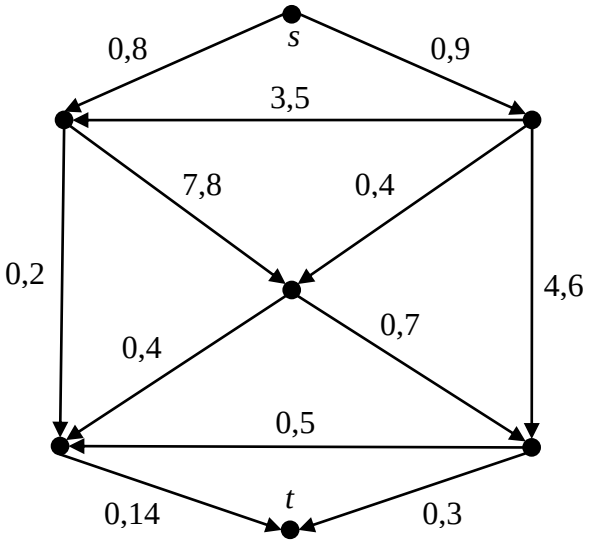


4. В следующих примерах на дугах первой цифрой указана нижняя граница пропускной способности, второй − верхняя граница пропускной способности. Найти максимальный поток из источника *s* в сток *t* или показать, что поток не существует.

a)



b)



3.2. Задача о многополюсном максимальном потоке

Пусть дана неориентированная сеть $G=(V,E)$ с пропускными способностями ребер $c(x,y) \geq 0$, $(x, y)\in E$. Задача о многополюсном максимальном потоке состоит в поиске максимального потока для всех пар вершин данной неориентированной сети. Для ее решения можно заменить ребра парами противоположно направленных дуг и применить алгоритм Форда-Фалкерсона для каждой пары вершин. При этом общее число задач о максимальном потоке, которые необходимо будет решить, равно $n(n\text{-}1)/2$ ($n = |V|$). Более эффективным методом, в котором задача о максимальном потоке решается n-1 раз, является алгоритм Гомори-Ху.

Для описания алгоритма введем следующую операцию над вершинами сети. Пусть $S \subseteq V$. Конденсацией множества вершин S назовем замену S одной вершиной $\{S\}$, с сохранением в сети всех ребер $(x, y) \in E$, для которых $x, y \in V \setminus S$, и их пропускных способностей, и замещением для каждой вершины $y \in V \setminus S$, всех ребер $(x, y) \in E$, для которых $x \in S$, одним ребром (s, y), с пропускной способностью равной сумме пропускных способностей замещаемых ребер.

Построим на множестве вершин исходной сети полный неориентированный граф $G' = (V, E')$ и положим вес ребра $(x, y) \in E'$ равным $\nu(x, y)$ пропускной способности минимального разреза отделяющего x от y в исходной сети (эквивалентно, величине максимального потока между x и y). Пусть $G'' = (V, E'')$ максимальное остовное дерево для $G' = (V, E')$. Если $x, u_1, u_2, \ldots, u_k, y$ единственный путь в $G'' = (V, E'')$ от x к y, то в силу максимальности остовного дерева,

$$\nu(x, y) \leq \min [\ \nu(x, u_1),\ \nu(u_1, u_2), \ldots,\ \nu(u_k, y)\ ].$$

С другой стороны, из свойств минимального разреза, получим

$$\nu(x, y) \geq \min [\nu(x, u_1),\ \nu(u_1, u_2), \ldots,\ \nu(u_k, y)].$$

Следовательно,

$$\nu(x, y) = \min [\nu(x, u_1),\ \nu(u_1, u_2), \ldots,\ \nu(u_k, y)].$$

Идея алгоритма Гомори-Ху состоит в итеративном построении максимального остовного дерева $G'' = (V, E'')$. Если требуется определить величину максимального потока между двумя произвольными узлами, надо в дереве найти путь, соединяющий эти два узла, и выбрать в этом пути дугу с минимальным весом. Вес этой дуги равен величине максимального потока между рассматриваемыми узлами.

Алгоритм Гомори-Ху. Шаг 1. В качестве множества ребер E'' дерева выбрать пустое множество. Объединить все узлы в одно множество-вершину $\{U\} = \{V\}$.

Шаг 2. k=1. Выбрать произвольную пару узлов s и t.

Шаг 3. Определить максимальный поток из s в t с помощью алгоритма Форда-Фалкерсона

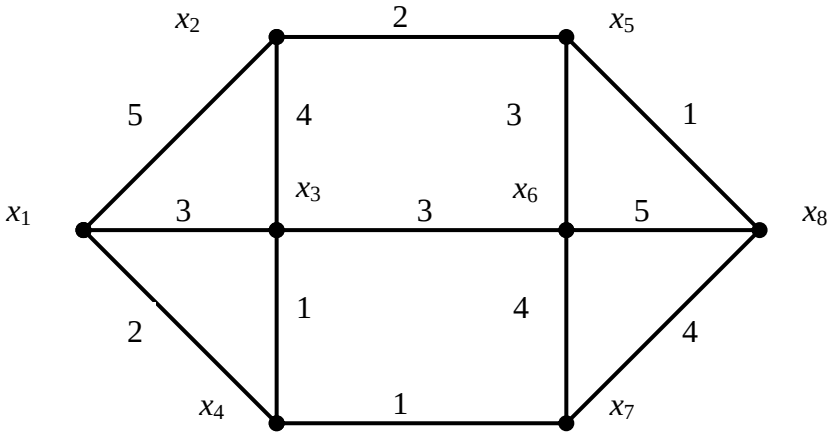
Шаг 4. Найти минимальный разрез $<S, \overline{S}>$, отделяющий s от t. Разбить множество-вершину $\{U\}$, которой принадлежат s и t, на два множества-вершин $\{U_1\}$, $\{U_2\}$ отнеся к первой вершины из $\{U\}$, принадлежащие S, ко второй – вершины из $\{U\}$ не принадлежащие S. Данный разрез представить ребром $(\{U_1\}, \{U_2\})$ и поместить в дерево. Вес ребра положить равным пропускной способности минимального разреза. Ребро $(\{X\}, \{U\}) \in E''$ текущего дерева, заменяем ребром $(\{X\}, \{U_1\})$, если $X \subseteq S$, и ребром $(\{U_2\}, \{X\})$, если $X \subseteq \overline{S}$.

Шаг 5. Если $k = n$-1, то закончить работу алгоритма. В противном случае, перейти к шагу 6.

Шаг 6. Выбрать произвольную пару узлов x и y, принадлежащих некоторому множеству-вершине $\{U\}$, то есть еще не отделенных друг от друга ребром в строящемся дереве. Положить $s = x$ и $t = y$.

Шаг 7. Сконденсировать в один узел каждую связную ветвь дерева, соединенную с множеством-вершиной $\{U\}$. Перейти к шагу 3.

Пример. Для следующей сети найти максимальный поток между каждой парой вершин.



Итерация 1. Возьмем $s = x_1$, $t = x_8$. Минимальный разрез, отделяющий x_1 от x_8, есть $<\{x_1, x_2, x_3, x_4\}, \{x_5, x_6, x_7, x_8\}>$. Его пропускная способность равна $\nu(x_1, x_8) = 6$. По минимальному разрезу получим, что дерево на первой итерации состоит из двух множеств-вершин: $\{x_1, x_2, x_3, x_4\}$, $\{x_5, x_6, x_7, x_8\}$ и единственного ребра с весом равным 6 (рис. 6).

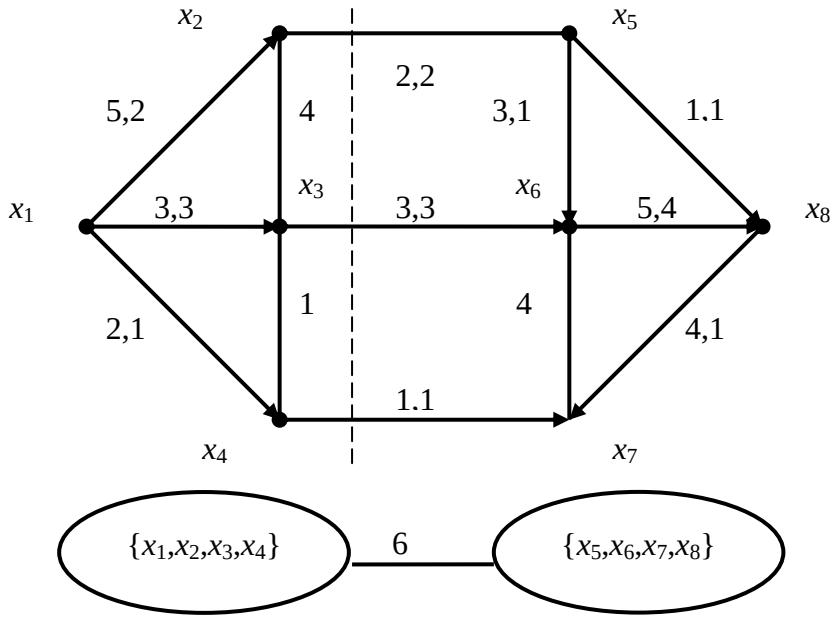


Рис. 6. Первая итерация алгоритма Гомори-Ху

Итерация 2. Берем вершины, принадлежащие одному множеству-вершине, например $s = x_1$, $t = x_4$. В исходной сети конденсируем множество вершин $\{x_5, x_6, x_7, x_8\}$, как принадлежащее одной связной ветви текущего дерева. В результате получим сеть, представленную на рис. 7.

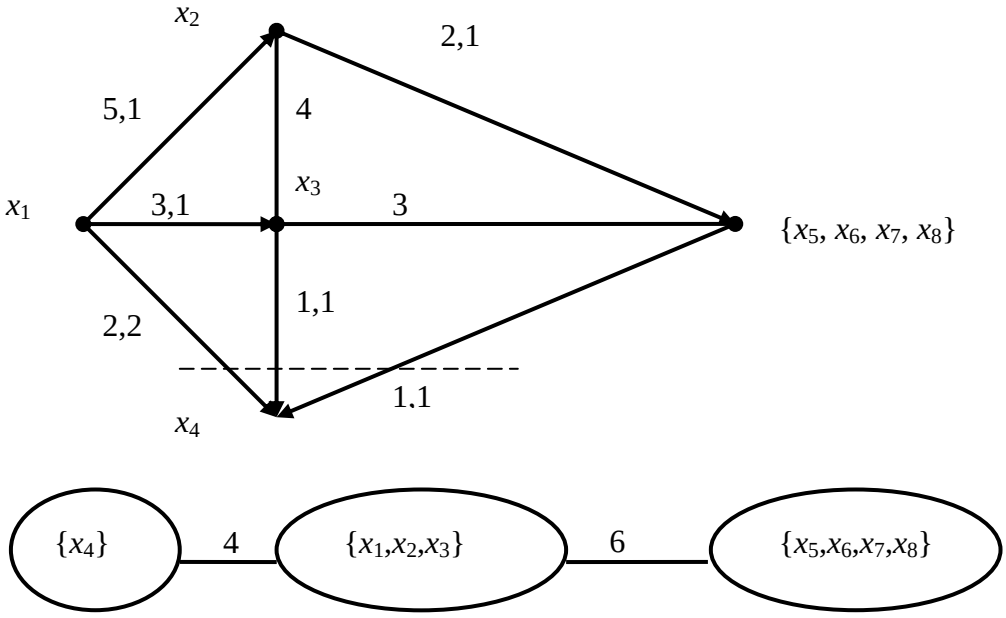


Рис. 7. Вторая итерация алгоритма Гомори-Ху

Минимальный разрез, отделяющий x_1 от x_4, есть $<\{x_1, x_2, x_3, \{x_5, x_6, x_7, x_8\}\}, \{x_4\} >$. Его пропускная способность равна $\nu(x_1, x_4) = 4$. По мини-

мальному разрезу получим, что дерево на второй итерации состоит из множеств-вершин: $\{x_4\}$, $\{x_1, x_2, x_3\}$, $\{x_5, x_6, x_7, x_8\}$ и ребер с весами равными 4 и 6 (рис. 7).

Итерация 3. Возьмем $s = x_5$, $t = x_8$. В исходной сети конденсируем множество вершин $\{x_1, x_2, x_3, x_4\}$, как принадлежащее одной связной ветви текущего дерева. В результате получим сеть, представленную на рис. 8. Минимальный разрез, отделяющий x_5 от x_8, есть $<\{\{x_5\}, \{x_6, x_7, x_8\}, \{x_1, x_2, x_3, x_4\}\}>$. Его пропускная способность равна $\nu(x_5, x_8) = 6$. По минимальному разрезу получим, что дерево на третьей итерации состоит из множеств-вершин: $\{x_4\}$, $\{x_1, x_2, x_3\}$, $\{x_6, x_7, x_8\}$, $\{x_5\}$ и ребер с весами равными 4, 6, 6 (рис. 8).

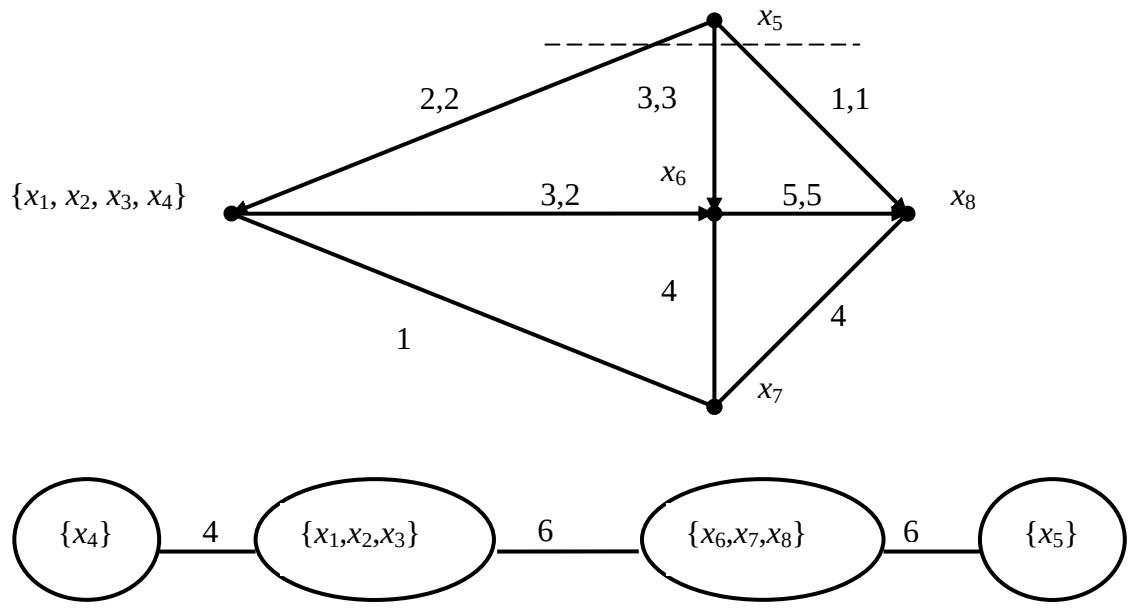


Рис. 8. Третья итерация алгоритма Гомори-Ху

Итерация 4. Возьмем $s = x_1$, $t = x_2$. В исходной сети конденсируем множество вершин $\{x_5, x_6, x_7, x_8\}$, как принадлежащее одной связной ветви текущего дерева. В результате получим сеть, представленную на рис. 9. Минимальный разрез, отделяющий x_1 от x_2, есть $<\{x_1\}, \{x_2, x_3, x_4\}, \{x_5, x_6, x_7, x_8\}>$. Его пропускная способность равна $\nu(x_1, x_2) = 10$. По минимальному разрезу получим, что дерево на четвертой итерации состоит из множеств-вершин: $\{x_4\}$, $\{x_1\}$, $\{x_2, x_3\}$, $\{x_6, x_7, x_8\}$, $\{x_5\}$ и ребер с весами равными 4, 10, 6, 6 (рис. 9).

Итерация 5. Возьмем $s = x_2$, $t = x_3$. Получим сеть, представленную на рис. 10. Минимальный разрез, отделяющий x_2 от x_3, есть $<\{x_2\}, \{x_1, x_3, x_4\}, \{x_5, x_6, x_7, x_8\}>$. Его пропускная способность равна $\nu(x_1, x_2) = 11$. По минимальному разрезу получим, что дерево на пятой итерации состоит

из множеств-вершин: $\{x_4\}$, $\{x_1\}$, $\{x_2\}$, $\{x_3\}$, $\{x_6, x_7, x_8\}$, $\{x_5\}$ и ребер с весами равными 4, 10, 11, 6, 6 (рис. 10).

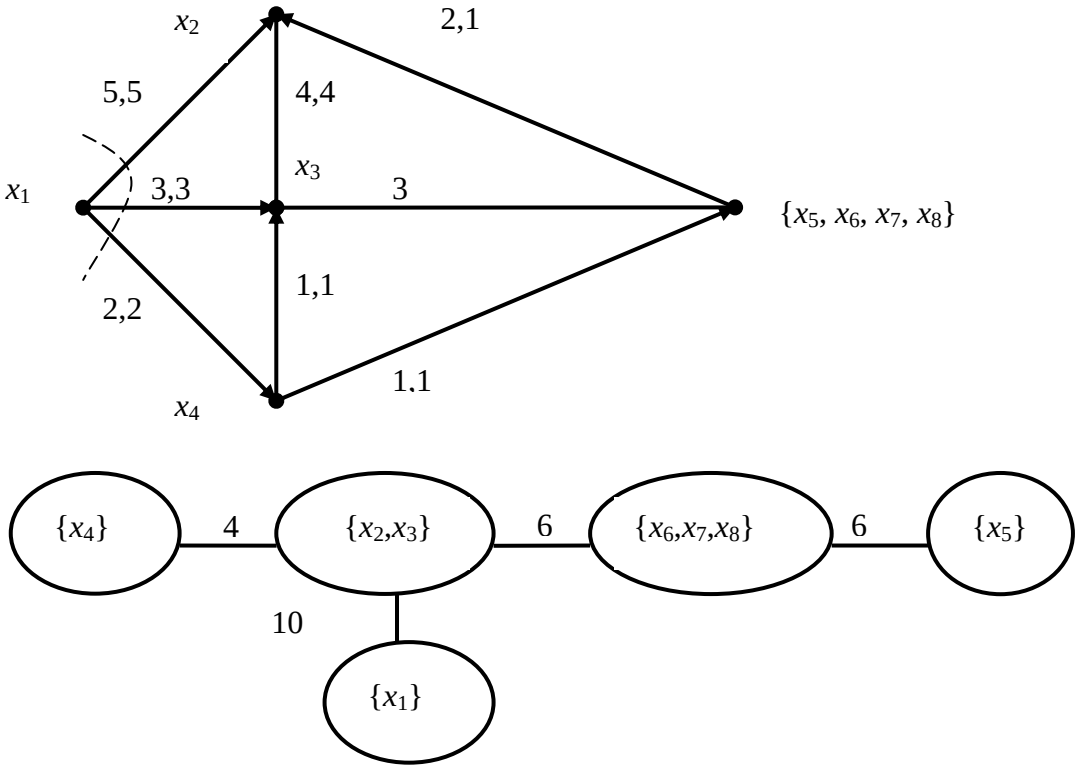


Рис. 9. Четвертая итерация алгоритма Гомори-Ху

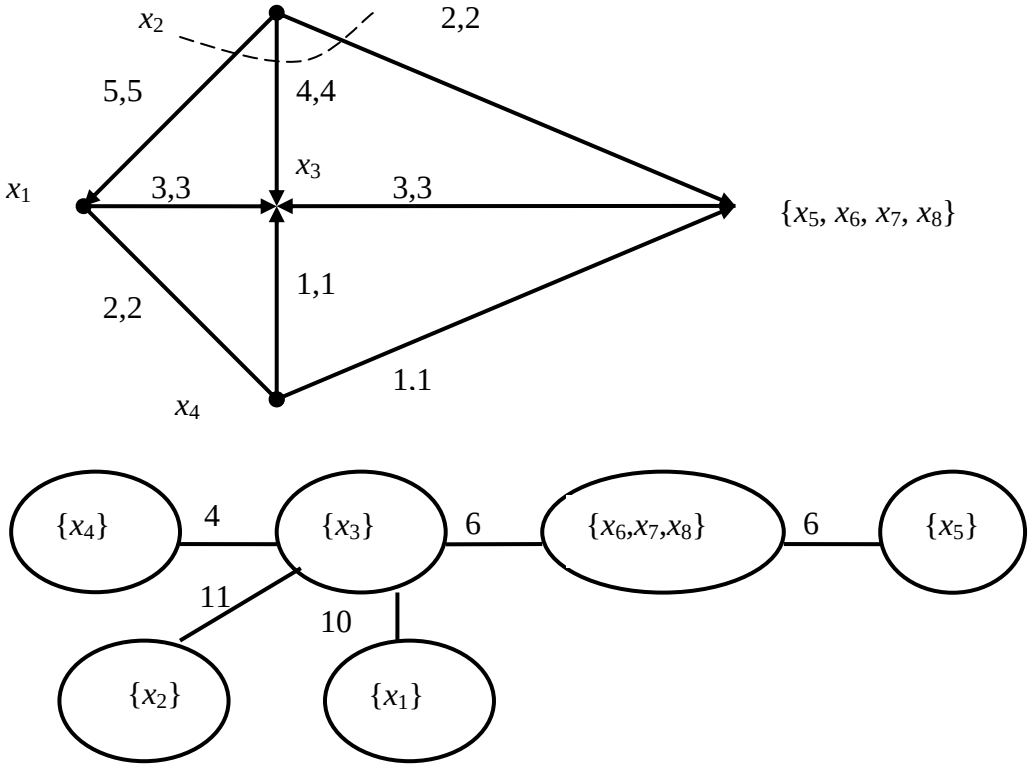


Рис. 10. Пятая итерация алгоритма Гомори-Ху.

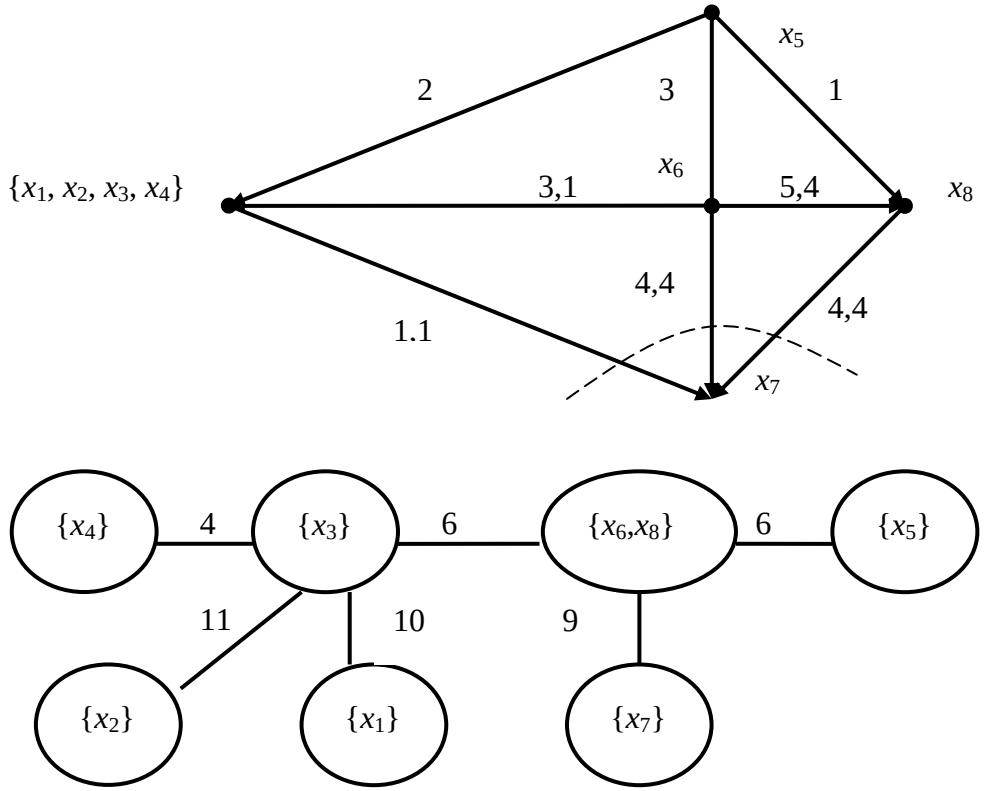


Рис. 11. Шестая итерация алгоритма Гомори-Ху

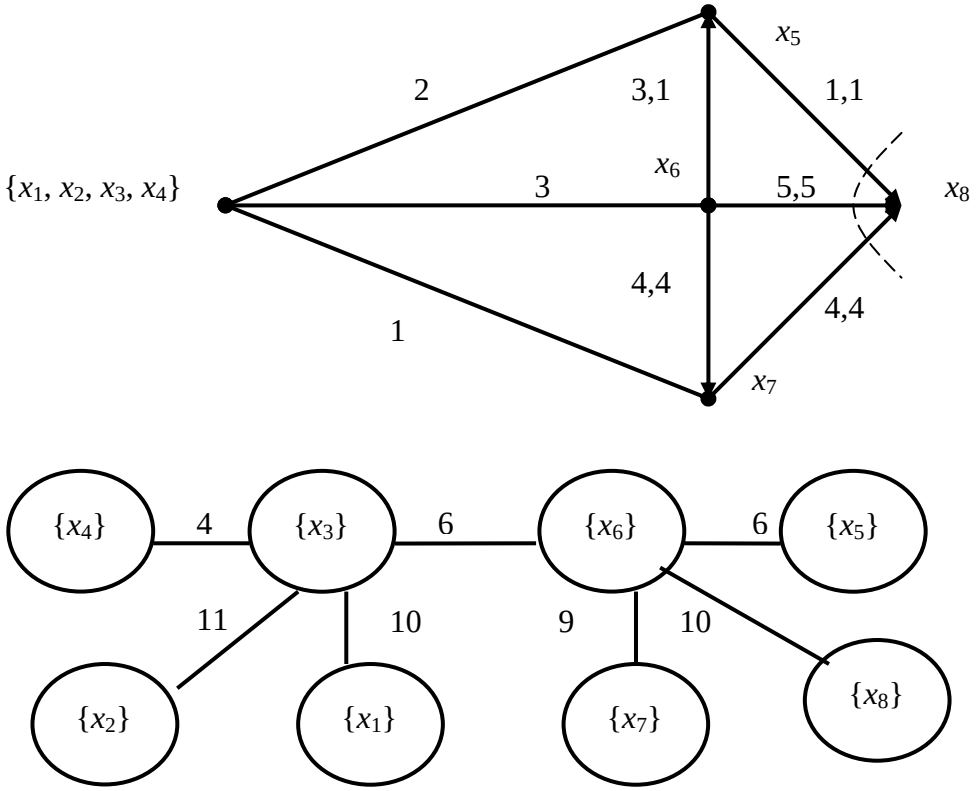


Рис. 12. Седьмая итерация алгоритма Гомори-Ху

Итерация 6. Возьмем $s = x_6$, $t = x_7$. Получим сеть, представленную на рис. 11. Минимальный разрез, отделяющий x_6 от x_7, есть $<\{x_7\}, \{\{x_1, x_2, x_3, x_4\}, x_5, x_6, x_8\}\}>$. Его пропускная способность равна $\nu(x_1, x_2) = 9$. По минимальному разрезу получим, что дерево на пятой итерации состоит из множеств-вершин: $\{x_4\}$, $\{x_1\}$, $\{x_2\}$, $\{x_3\}$, $\{x_7\}$, $\{x_6, x_8\}$, $\{x_5\}$ и ребер с весами равными 4, 10, 11, 9, 6, 6 (рис. 11).

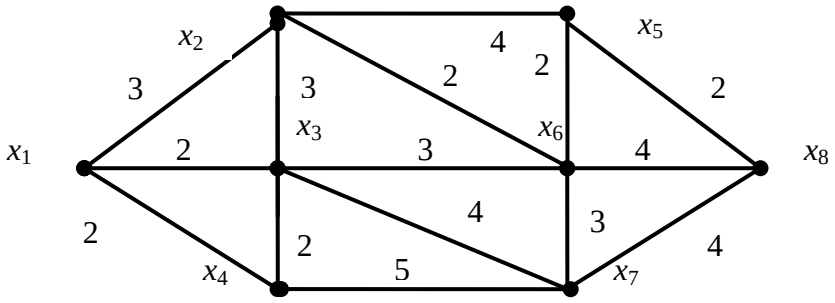
Итерация 7. $s = x_6$, $t = x_8$. Получим сеть, представленную на рис. 12. Минимальный разрез, отделяющий x_6 от x_8, есть $<\{\{x_1, x_2, x_3, x_4\}, x_5, x_6, x_7\}\}, \{x_8\}>$. Его пропускная способность равна $\nu(x_1, x_2) = 10$. По минимальному разрезу получим, что дерево на седьмой итерации состоит из вершин x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 и ребер с весами равными 4, 10, 10, 11, 9, 6, 6 (рис. 12). Дерево стало полным, и алгоритм завершает работу.

Для определения величины максимального потока, например, между вершинами x_2, x_7, находим единственный путь, соединяющий эти вершины в результирующем дереве. Это путь (x_2, x_3), (x_3, x_6), (x_6, x_7). Ребро с минимальным весом (x_3, x_6). Следовательно, искомый поток равен 6.

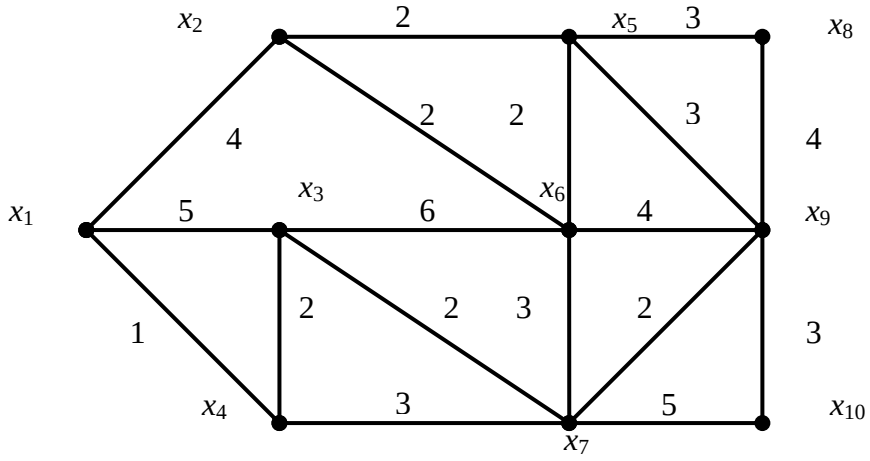
Задачи для самостоятельного решения

1. Найти максимальные потоки для всех пар вершин в сетях.

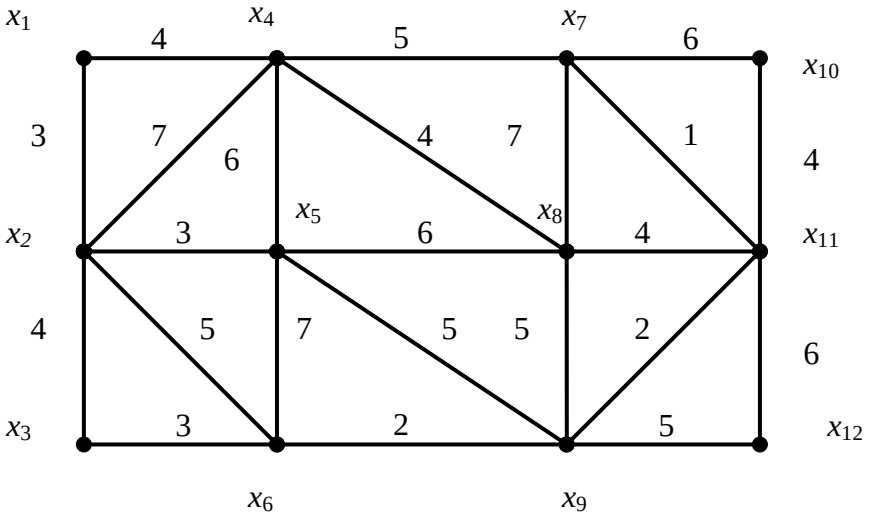
a)



b)



c)



3.3. Задача о многополюсных путях с максимальными пропускными способностями

Пусть дана ориентированная сеть $G = (V, E)$ с пропускными способностями дуг $c(x, y) \geq 0$, $(x, y) \in E$. В задаче о многополюсных путях с максимальными пропускными способностями необходимо для каждой пары вершин $x, y \in V$ найти соединяющий x и y путь (если он существует), имеющий максимальную пропускную способность.

Пронумеруем вершины сети и построим матрицу пропускных способностей $C = |c_{ij}|_{nxn}$ ($n = |V|$) с элементами

$$c_{ij} = \begin{cases} 0, i = j, \\ c(x_i, x_j), (x_i, x_j) \in E, \\ -\infty, i \neq j, (x_i, x_j) \notin E, \end{cases}$$

и матрицу путей $T = |t_{ij}|_{nxn}$ с элементами $t_{ij} = j$.

Алгоритм решения задачи о многополюсных путях с максимальными пропускными способностями подобен алгоритму Флойда поиска минимальных путей для каждой пары вершин сети.

Шаг 1. Положить $k = 0$, и положить матрицы $C^0 = |c_{ij}^0|_{nxn}$, $T^0 = |t_{ij}^0|_{nxn}$ с $c_{ij}^0 = c_{ij}$, $t_{ij}^0 = t_{ij}$, $i, j = \overline{1, n}$.

Шаг 2. Построить матрицы $C^{k+1} = |c_{ij}^{k+1}|_{nxn}$, $C^{k+1} = |c_{ij}^{k+1}|_{nxn}$ по матрицам $C^k = |c_{ij}^k|_{nxn}$ и $T^k = |t_{ij}^k|_{nxn}$, вычисляя их элементы по формулам:

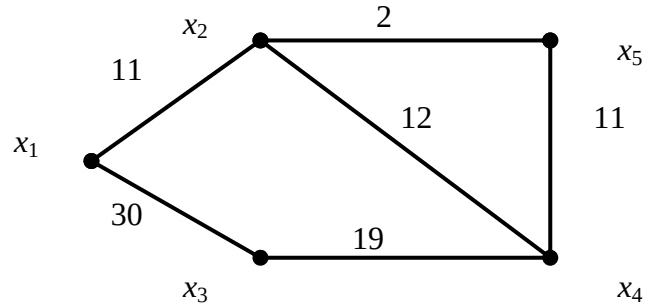
$$c_{ij}=\begin{cases} c_{ij}^{k}, i=k+1,\ j=\overline{1,n}, \\ c_{ij}^{k},\ j=k+1, i=\overline{1,n}, \\ \max\{c_{ij}^{k};\min[c_{ik+1}^{k}, c_{k+1}^{k}\}, i\neq k+1,\ j\neq k+1, \end{cases} \qquad t_{ij}^{k+1}=\begin{cases} t_{ij}^{k}, \text{если } c_{ij}^{k+1}=c_{ij}^{k}, \\ j, \text{если } c_{ij}^{k+1}\neq c_{ij}^{k}. \end{cases}$$

Шаг 3. Если $k + 1 < n$, то положить $k := k + 1$ и перейти к шагу 2.

Если $k+1 = n$, то матрица C^n дает искомые максимальные пропускные способности для пар вершин сети.

Путь с максимальной пропускной способностью от вершины x_i к вершине x_j строится по элементам матрицы T^n. Элемент $t_{ij}^{n}=m$ указывает промежуточную вершину x_m в пути от вершины x_i к вершине x_j. Находим t_{im}^{n} и t_{mj}^{n}, которые указывают очередные промежуточные вершины. Если $t_{im}^{n}=m$ $t_{mj}^{n}=j$, то вершина x_m непосредственно следует за вершиной x_i (непосредственно предшествует вершине x_j). Процесс завершаем при получении номера вершины непосредственно следующей за x_i и номера вершины непосредственно предшествующей x_j.

Пример. Решить задачу о многополюсных путях с максимальными пропускными способностями для следующей сети.



Поскольку сеть является неориентированной, то заменяем каждое ребро парой противоположно направленных дуг. В результате получим матрицу пропускных способностей и матрицу путей:

$$C=C^0=\begin{vmatrix} 0 & 11 & 30 & -\infty & -\infty \\ 11 & 0 & -\infty & 12 & 2 \\ 30 & -\infty & 0 & 19 & -\infty \\ -\infty & 12 & 19 & 0 & 11 \\ 2 & -\infty & -\infty & 11 & 0 \end{vmatrix} \qquad T=T^0=\begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{vmatrix}.$$

На последующих итерациях получим:

$$C^1=\begin{vmatrix} 0 & 11 & 30 & -\infty & -\infty \\ 11 & 0 & 11 & 12 & 2 \\ 30 & 11 & 0 & 19 & -\infty \\ -\infty & 12 & 19 & 0 & 11 \\ -\infty & 2 & -\infty & 11 & 0 \end{vmatrix} \quad T^1=\begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 1 & 4 & 5 \\ 1 & 1 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{vmatrix}$$

$$C^2=\begin{vmatrix} 0 & 11 & 30 & 11 & 2 \\ 11 & 0 & 11 & 12 & 2 \\ 30 & 11 & 0 & 19 & 2 \\ 11 & 12 & 19 & 0 & 11 \\ 2 & 2 & 2 & 11 & 0 \end{vmatrix} \quad T^2=\begin{vmatrix} 1 & 2 & 3 & 2 & 2 \\ 1 & 2 & 1 & 4 & 5 \\ 1 & 1 & 3 & 4 & 2 \\ 2 & 2 & 3 & 4 & 5 \\ 2 & 2 & 2 & 4 & 5 \end{vmatrix}$$

$$C^3=\begin{vmatrix} 0 & 11 & 30 & 19 & 2 \\ 11 & 0 & 11 & 12 & 2 \\ 30 & 11 & 0 & 19 & 2 \\ 19 & 12 & 19 & 0 & 11 \\ 2 & 2 & 2 & 11 & 0 \end{vmatrix} \quad T^3=\begin{vmatrix} 1 & 2 & 3 & 3 & 2 \\ 1 & 2 & 1 & 4 & 5 \\ 1 & 1 & 3 & 4 & 2 \\ 3 & 2 & 3 & 4 & 5 \\ 2 & 2 & 2 & 4 & 5 \end{vmatrix}$$

$$C^4=\begin{vmatrix} 0 & 12 & 30 & 19 & 11 \\ 12 & 0 & 12 & 12 & 11 \\ 30 & 12 & 0 & 19 & 11 \\ 19 & 12 & 19 & 0 & 11 \\ 11 & 11 & 11 & 11 & 0 \end{vmatrix} \quad T^4=\begin{vmatrix} 1 & 4 & 3 & 3 & 4 \\ 4 & 2 & 4 & 4 & 4 \\ 1 & 4 & 3 & 4 & 4 \\ 3 & 2 & 3 & 4 & 5 \\ 4 & 4 & 4 & 4 & 5 \end{vmatrix}$$

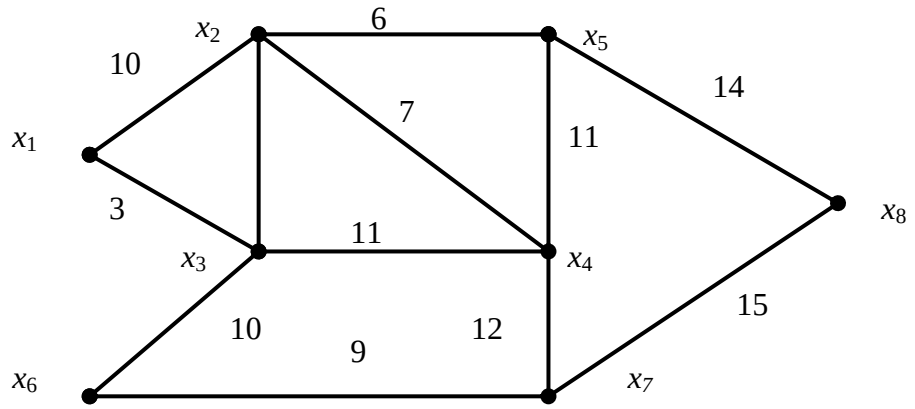
$$C^5=\begin{vmatrix} 0 & 12 & 30 & 19 & 11 \\ 12 & 0 & 12 & 12 & 11 \\ 30 & 12 & 0 & 19 & 11 \\ 19 & 12 & 19 & 0 & 11 \\ 11 & 11 & 11 & 11 & 0 \end{vmatrix} \quad T^5=\begin{vmatrix} 1 & 4 & 3 & 3 & 4 \\ 4 & 2 & 4 & 4 & 4 \\ 1 & 4 & 3 & 4 & 4 \\ 3 & 2 & 3 & 4 & 5 \\ 4 & 4 & 4 & 4 & 5 \end{vmatrix}.$$

Матрица C^5 дает значения максимальных пропускных способностей путей между вершинами. Например, максимальная пропускная способность путей между третьей и пятой вершинами равно 11. Сам путь с указанной пропускной способностью строим по матрице T^4. Получим $x_3 \to x_4 \to x_5$.

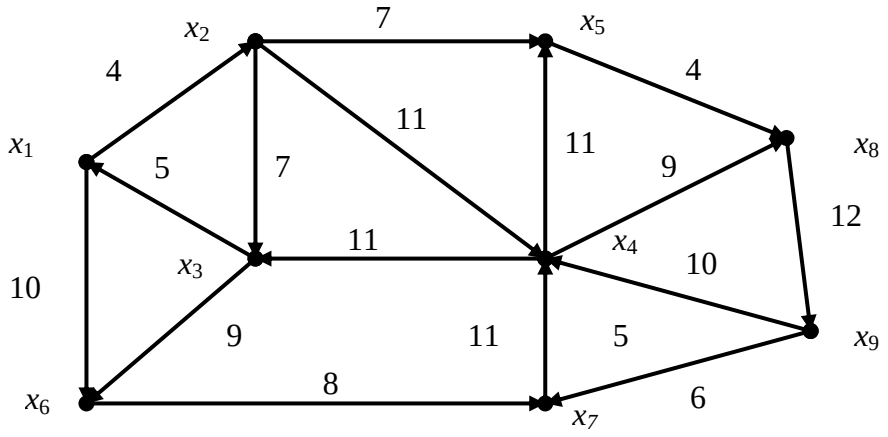
Задачи для самостоятельного решения

Решить задачу о многополюсных путях с максимальными пропускными способностями для следующих сетей:

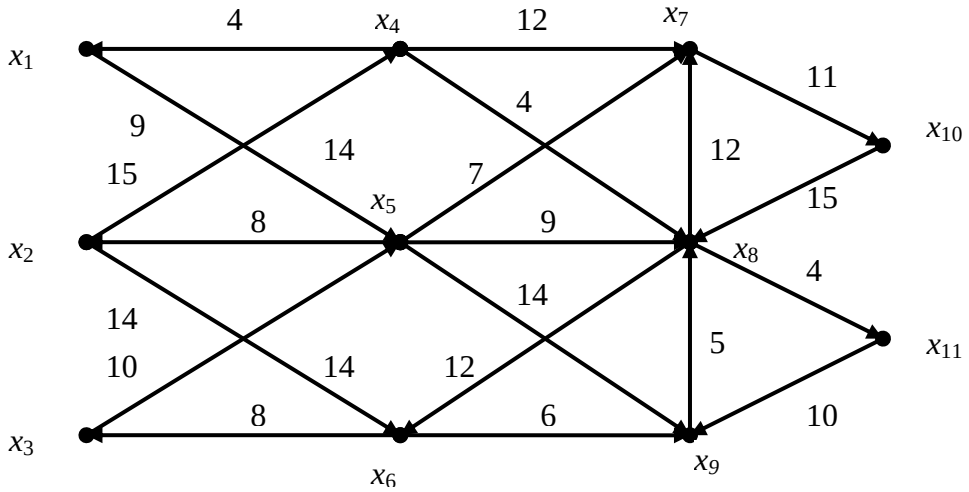
a)



b)



c)



3.4. Потоки минимальной стоимости

Дана сеть $G = (V,E)$, на каждой дуге $e \in E$ задана пропускная способность $c(e) > 0$ и удельная стоимость $d(e)$, выделен источник s и сток t. Если в сети имеется поток f, то его стоимость определяется как

$$S(f) = \sum_{e \in E} d(e) f(e).$$

Задача о потоке минимальной стоимости состоит в поиске допустимого потока f, имеющего заданную величину m и минимальную стоимость $S(f)$.

Граф модифицированных стоимостей $G_f = (V_f, E_f)$ имеет $V_f = V$. Множество дуг E_f строятся по следующему правилу:

1) если $e \in E$ и $f(e) = 0$, то дуга e строится в G_f с той же пропускной способностью $c_f(e) = c(e)$ и стоимостью $d_f(e) = d(e)$;

2) если $e = (x, y) \in E$ и $f(e) = c(e)$, то в G_f строится обратная дуга $\bar{e} = (y, x)$ с пропускной способностью $c_f(\bar{e}) = c(e)$ и удельной стоимостью $d_f(e) = -d(e)$;

3) если $e = (x, y) \in E$ и $0 < f(e) < c(e)$, то в G_f строятся две дуги $e = (x, y), \bar{e} = (y, x)$ с $c_f(e) = c(e) - f(e)$, $d_f(e) = d(e)$, $c_f(\bar{e}) = f(e)$, $d_f(\bar{e}) = -d(e)$.

Легко показать, что каждому простому ориентированному пути в сети G_f, ведущему из источника в сток, соответствует увеличивающая цепь в сети G.

Удельной стоимостью пути (или цикла) в сети будем называть сумму удельных стоимостей входящих в него дуг.

Критерием оптимальности потока фиксированной мощности является следующее утверждение. Допустимый поток f в сети G имеет минимальную стоимость среди всех допустимых потоков той же величины в том и только том случае, когда в графе G_f нет контуров с отрицательной удельной стоимостью.

Алгоритм Басакера − Гоуэна. На каждом шаге алгоритма находится наиболее дешевый (по удельной стоимости) ориентированный путь из s в t в графе модифицированных стоимостей G_f. Этому пути соответствует увеличивающая цепь минимальной удельной стоимости в сети G. Затем по этой цепи пропускается максимальный поток, который можно добавить к имеющемуся потоку f.

Шаг 1. В сети G пропускаем нулевой поток f. Его величина $\gamma(f)$ и стоимость $S(f)$ равны нулю.

Шаг 2. Строим граф модифицированных стоимостей G_f.

Шаг 3. Если в G_f нет ни одного ориентированного пути из s в t, то мощность потока f не может быть увеличена, и задача не имеет решения. В противном случае среди всех путей из s в t в графе G_f находим путь с минимальной удельной стоимостью. Обозначим его P_f.

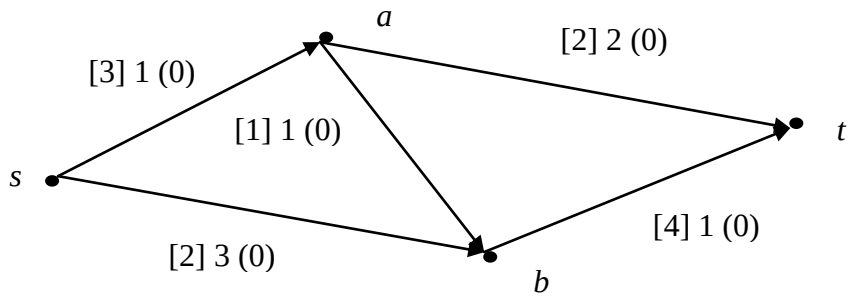
Шаг 4. В исходной сети G определяем неориентированную цепь P, соответствующую пути P_f. Для всех дуг e из цепи P вычисляем числа $\varepsilon(e)$: на прямых дугах $\varepsilon(e) = c(e) - f(e)$, а на обратных дугах $\varepsilon(e) = f(e)$. Затем находим $\varepsilon = \min [\ \varepsilon(e),\ m - \gamma(f) \mid e \in P\ ]$. На прямых дугах цепи P увеличиваем поток f на ε, а на обратных дугах уменьшаем поток f на ε. При этом величина потока увеличивается на ε.

Шаг 5. Если величина нового потока меньше m, то переходим к шагу 2. Если же она равна m, то в сети построен оптимальный поток мощности m.

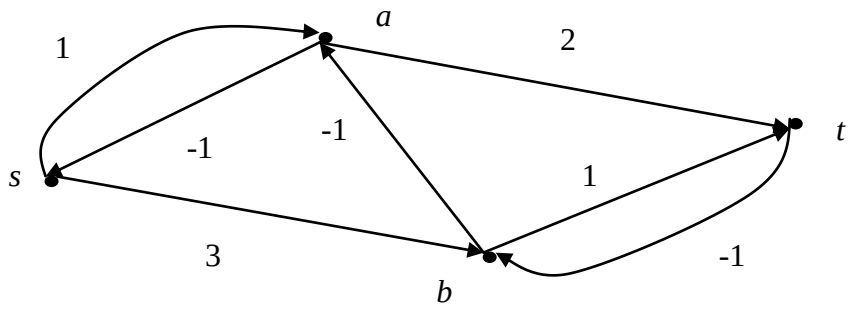
Алгоритм Басакера - Гоуэна имеет смысл применять только к таким сетям G, в которых нет контуров отрицательной удельной стоимости. Выполнение или невыполнение этого условия можно проверять с помощью алгоритма Форда-Беллмана или алгоритма Флойда.

Для поиска самого дешевого пути из s в t в графе G_f (шаг 3 алгоритма Басакера - Гоуэна) тоже можно использовать алгоритмы Форда-Беллмана и Флойда.

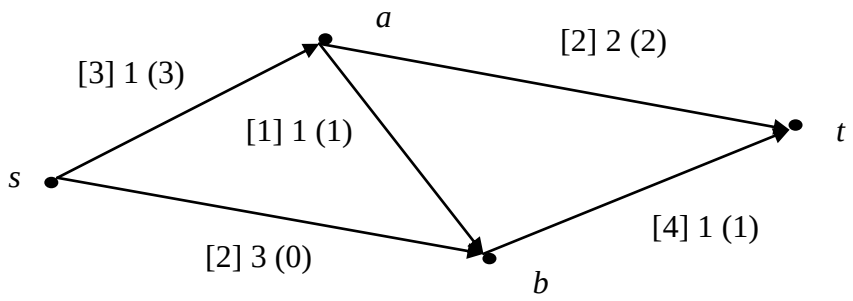
Пример. Построить в следующей сети поток величиной $m = 3$ с минимальной стоимостью. Пропускные способности дуг указаны в квадратных скобках, стоимости без скобок, величина потока в круглых скобках.



Пропускаем поток $f(e) = 0$, $e \in E$. Строим граф модифицированных стоимостей. Он будет совпадать с исходной сетью. Находим наиболее дешевый простой путь из s в t. Он состоит из дуг (s,a), (a,b), (b,t) и имеет удельную стоимость равную 3. Соответсвующая цепь в исходной сети дает $\varepsilon = \min [3, 1, 4, 3 - 0] = 1$. Все дуги цепи прямые. Увеличиваем поток по дугам на единицу. Получим новый поток в сети величины 1. Мощность потока меньше 3. Поэтому строим граф модифицированных стоимостей.



Находим путь из источника в сток с минимальной удельной стоимостью. Он состоит из дуг (*s*,*a*) , (*a*,*t*) и имеет удельную стоимость равную 3. В исходной сети для соответсвующая цепи получим ε = min [3–1, 2, 3–1] = 2. Увеличиваем поток по дугам (*s*,*a*), (*a*,*t*) на 2. Величина нового потока равна 3.



Необходимая величина потока достигнута. Искомый поток построен. Его стоимость $S(f) = 9$.

Алгоритм Клейна. Его сущность заключается в том, что вначале нужно найти какой-нибудь поток f величины m, а затем последовательно уменьшать его стоимость, перестраивая вдоль имеющихся циклов с отрицательной удельной стоимостью. При этих перестройках величина потока f будет сохраняться. В тот момент, когда циклы с отрицательной удельной стоимостью исчезнут, поток f станет оптимальным.

Шаг 1. Находим любой допустимый поток f величины $M(f) = m$.

Это можно сделать с помощью алгоритма Форда-Фалкерсона (в котором вычисления нужно проводить до тех пор, пока поток не достигнет величины m). Если в сети не существует допустимого потока величины m, то задача не имеет решения.

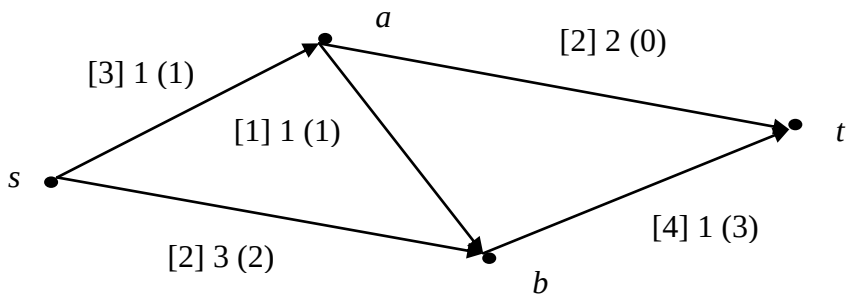
Шаг 2. Строим граф модифицированных стоимостей G_f. Если в нем нет контуров с отрицательной удельной стоимостью, то задача решена: построенный поток f имеет минимальную стоимость среди потоков величины m. В противном случае находим в графе G_f контур P_f с отрица-

тельной удельной стоимостью (например, с помощью алгоритма Флойда).

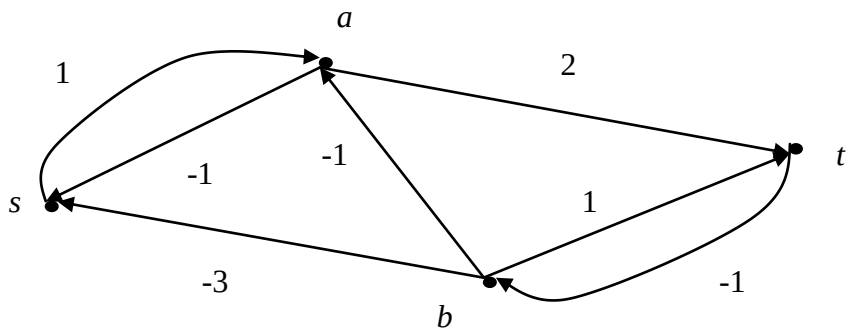
Шаг 3. В исходной сети G определяем неориентированный цикл P, соответствующий контуру P_f. Все дуги $e \in P$ разбиваются на два класса: прямые, для которых $e \in P_f$, и обратные, для которых $\bar{e} \in P_f$ - (где $\bar{e}$ – дуга, противоположная e) . Вычисляем $\varepsilon = \min [\varepsilon(e) \mid e \in P\}$, где $\varepsilon(e) = c(e) - f(e)$ на прямых дугах и $\varepsilon(e) = f(e)$ на обратных дугах.

Шаг 4. На прямых дугах цикла P увеличиваем поток f на ε, а на обратных дугах цикла P уменьшаем поток f на ε. Переходим к шагу 2.

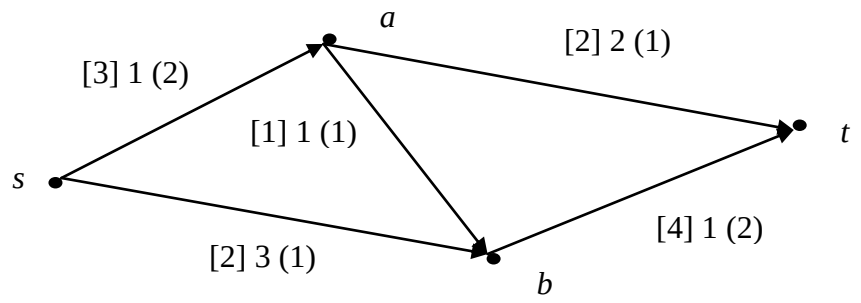
Пример. Построить в следующей сети поток величиной m=3 с минимальной стоимостью. Пропускные способности дуг указаны в квадратных скобках, стоимости без скобок, величина потока в круглых скобках.



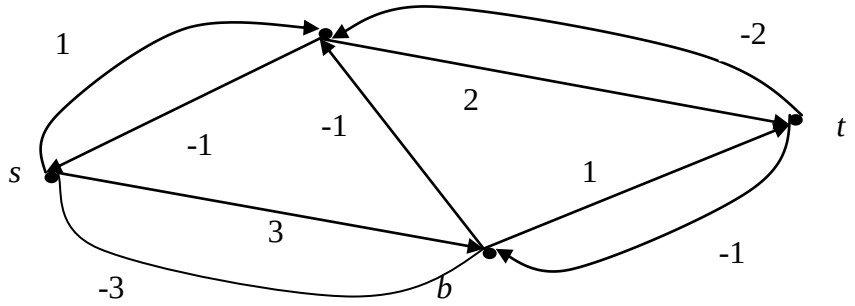
В сети уже имеется поток величины 3. Строим граф модифицированных стоимостей.



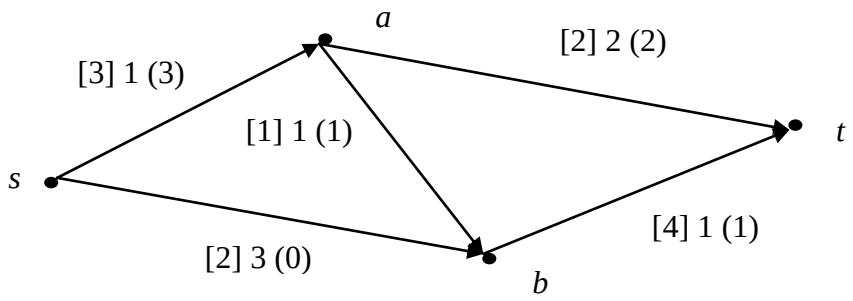
Находим контур с отрицательной удельной стоимостью: (s,a), (a,t), (t,b), (b,s) . Ему соответсвует цикл (s,a), (a,t), (b,t), (s,b) в исходной сети. Вычисляем $\varepsilon = \min [3 - 1, 2 - 0, 4 - 3, 2] = 1$. На дугах (s,a), (a,t) увеличиваем поток на 1, на дугах (b,t), (s,b) уменьшаем на 1. Получим сеть с новым потоком.



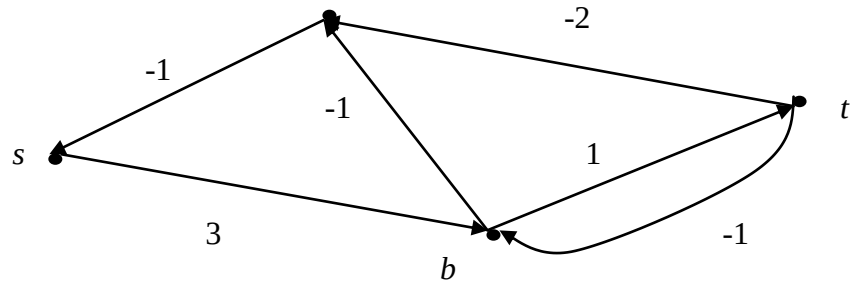
Для нового потока строим граф модифицированных стоимостей.



Опять находим контур с отрицательной удельной стоимостью: (*s*,*a*), (*a*,*t*), (*t*,*b*), (*b*,*s*) . Ему соответсвует цикл (*s*,*a*), (*a*,*t*), (*b*,*t*), (*s*,*b*) в исходной сети. Вычисляем ε = min [3 – 2, 2 – 1, 4 – 2, 2-1] = 1. На дугах (*s*,*a*), (*a*,*t*) увеличиваем поток на 1, на дугах (*b*,*t*), (*s*,*b*) уменьшаем на 1. Получим сеть с новым потоком.



Строим граф модифицированных стоимостей.

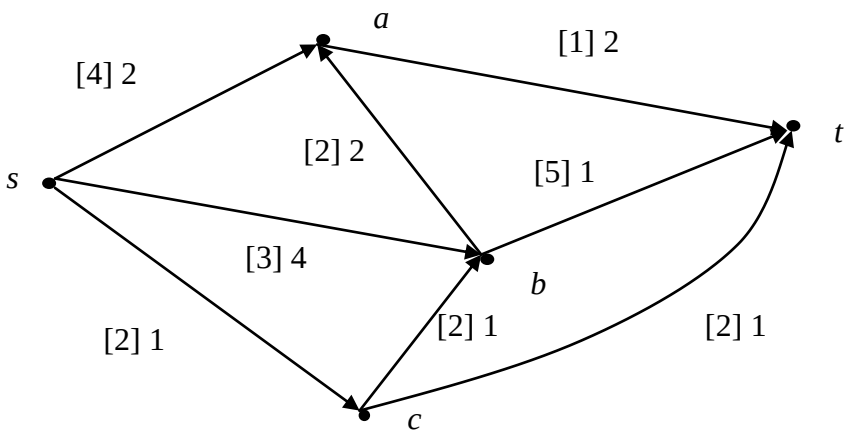


В графе отсутствуют контуры с отрицательной удельной стоимостью. Следовательно, построенный поток оптимален. Его стоимость $S(f) = 9$.

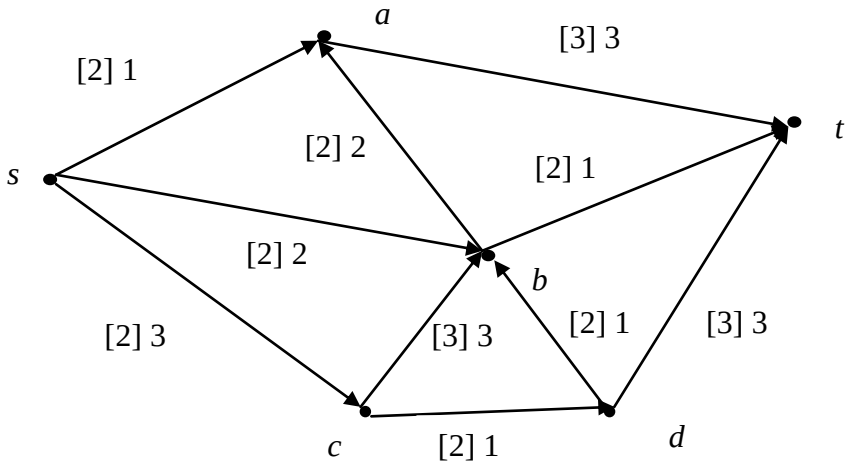
Задачи для самостоятельного решения

1. Найти с помощью алгоритма Басакера-Гоуэна и алгоритма Клейна поток величины $m = 4$ с минимальной стоимостью.

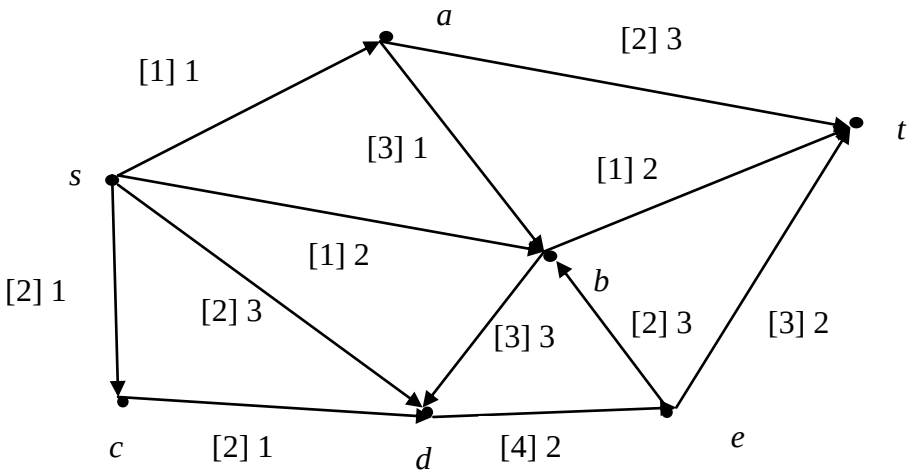
a)



b)



c)



4. ЗАДАЧИ О НАЗНАЧЕНИЯХ

4.1. Классическая задача о назначениях

Задано n работ и n исполнителей, причем каждую из работ может выполнять любой из исполнителей. Стоимость выполнения i-ой работы j-ым исполнителем определяется величиной $c_{ij} \geq 0$. Требуется распределить исполнителей на выполнение работ так, чтобы все работы были выполнены, каждый исполнитель выполнял только одну работу, и стоимость выполнения всех работ была бы минимальной.

Построим математическую модель для данной задачи. Пусть

$$x_{ij} = \begin{cases} 1, \text{ если } i \text{ -й исполнитель назначен на } j\text{-ю работу;} \\ 0, \text{ в противном случае.} \end{cases}$$

Тогда задача может быть сформулирована в виде следующей задачи булевого программирования:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\, x_{ij} \to \min ,$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ j = \overline{1,n} , \quad \sum_{j=1}^{n} x_{ij} = 1,\ i = \overline{1,n} ,$$

$$x_{ij} = 0 \vee 1 .$$

Можно привести еще одну интерпритацию постановки задачи. Построим полный двудольный граф $G=(V_1 \cup V_2 , E)$, вершины первой доли которого соответсвуют исполнителям, второй – работам. Припишем ребру (v_i , u_j), $v_i \in V_1$, $u_j \in V_2$, вес c_{ij} . Тогда задача заключается в поиске совершенного (покрывающего все вершины графа) паросочетания с минимальным весом.

Приведем алгоритм решения задачи, использующий в качестве вспомогательного алгоритм Форда-Фалкерсона нахождения максимального потока в сети.

Начальный шаг. Находим в каждом строке матрицы стоимостей $C = [c_{ij}]_{n \times n}$ минимальный элемент v_i и вычитаем его из всех элементов этой строки. Затем находим минимальный элемент u_i в каждом столбце и вычитаем его из элементов данного столбца. Проделанные выше процедуры называются *приведением матрицы*, а полученная матрица $C' = \left[c'_{ij} \right]_{n \times n}$ – *приведенной*.

Легко показать, что решения задач для исходной и для приведенной матриц совпадают.

Нулевые элементы приведенной матрицы называются *допустимыми*. Если бы можно было выбрать по одному нулевому элементу в каждом столбце и каждой строке, то это и было бы оптимальным назначением.

Задачу выбора нулевых клеток можно решить с помощью алгоритма нахождения максимального потока.

Шаг 1. Для приведенной матрицы строим сеть с $(2n + 2)$-мя вершинами: источником s, множеством вершин $S = \{s_1, \ldots, s_n\}$, соответсвующими строкам матрицы стоимостей, множеством вершин $T = \{t_1, \ldots, t_n\}$, соответсвующими столбцам матрицы стоимостей, и стоком t. Источник s соединяем дугой с каждой вершиной s_i, каждую вершину t_j соединяем дугой со стоком t. Добовляем дугу (s_i, t_j) тогда и только тогда, когда $c'_{ij} = 0$. Пропускные способности всех дуг полагаем равными 1.

Шаг 2. Находим максимальный поток в сети из s в t. Если величина найденного потока равна n, то решение задачи получено. В оптимальном назначении $x_{ij} = 1$ огда и только тогда, когда дуга (s_i, t_j) существует и поток по ней равен 1.

Если максимальный поток меньше n, то переходим на шаг 3.

Шаг 3. Необходимо преобразовать сеть, добавив какую-либо новую дугу (или дуги). Естественно, следует добавить такую дугу, которая увеличивала бы поток, а соответсвующий ей элемент c'_{ij} был бы наименьшей.

Пусть при нахождении максимального потока на последнем шаге алгоритма вершины множества $S' \subset S$ оказались *помеченными*, а вершины множества $T' \subset T$ – *непомеченными*. Пусть S'' (T'') номера вершин, входящих в S' (T'). Находим $\bar{c} = \min_{i \in S'', j \in T''} (c'_{ij})$.

Шаг 4. Вычитаем элемент $\bar{c}$ из всех элементов в строках, соответсвующих вершинам из S' и добавляем к элементам столбцов соответсвующих вершинам из $T \backslash T'$.

Шаг 5. Переходим к шагу 1 с матрицей, полученной на шаге 4.

В результате работы алгоритма в матрице стоимостей на каждой итерации появляется хотя бы один новый ноль (на месте элементов равных $\bar{c}$), т. е. в сети появляется хотя бы одна новая дуга.

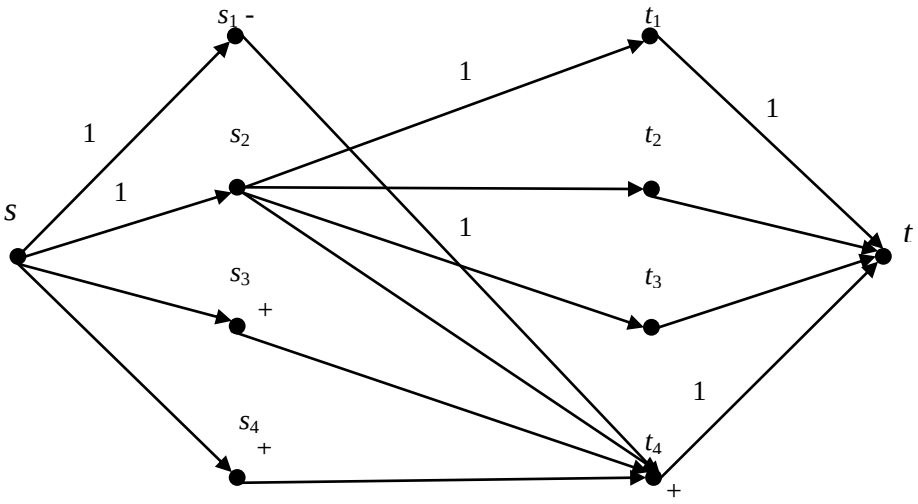
Пример. Решить задачу о назначении со следующей матрицей стоимостей:

$$C=\begin{bmatrix} 7 & 5 & 6 & 3 \\ 2 & 1 & 2 & 1 \\ 5 & 5 & 5 & 2 \\ 4 & 4 & 5 & 2 \end{bmatrix}.$$

Начальный шаг. Приводим матрицу C сначала по строкам, затем − по столбцам. Приведенная матрица C' имеет вид

$$C'=\begin{bmatrix} 3 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 \\ 2 & 3 & 2 & 0 \\ 1 & 2 & 2 & 0 \end{bmatrix}.$$

Итерация 1. Строим для приведенной матрицы C' сеть



Пропускные способности всех дуг полагаем равными 1. Находим максимальный поток в полученной сети. На последней итерации алгоритма Форда-Фалкерсона получим:

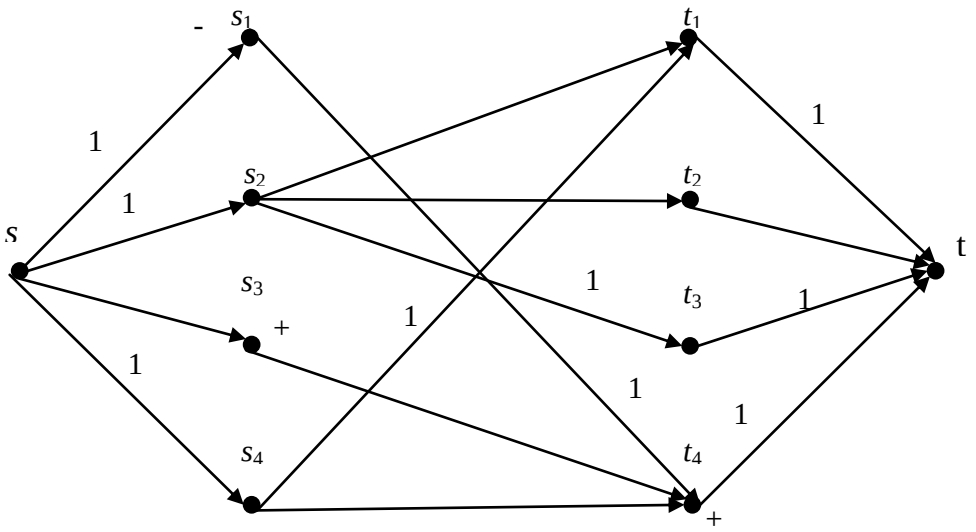
s	s_1	s_2	s_3	s_4	t_1	t_2	t_3	t_4	t	v
(-,∞)	(t_4^-,1)		(s^+,1)	(s^+,1)				(s_3^+,1)		2

Значение максимального потока $v = 2$. Это меньше 4, поэтому необходимо преобразовать сеть. Имеем: $S'' = \{1, 3, 4\}$, $T'' = \{1, 2, 3\}$. Находим $\overline{c} = \min\limits_{i \in S'', j \in T''} (c_{ij}) = 1$. Преобразуем приведенную матрицу, вычитая элемент

$\bar{c}$ из всех элементов в строках с номерами из S'' и добавляя к элементам столбцов с номерами, не вошедшими в T''. Матрица C' примет вид

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

Итерация 2. Получим следующую сеть:



Находим максимальный поток в сети. На последней итерации алгоритма Форда-Фалкерсона получим:

s	s_1	s_2	s_3	s_4	t_1	t_2	t_3	t_4	t	v
(-,∞)	(t_4^-,1)		(s^+,1)					(s_3^+,1)		3

Имеем: $v = 3 < 4$, $S'' = \{1,3\}$, $T'' = \{1,2,3\}$. Находим $\bar{c} = \min_{i \in S'', j \in T''} (c_{ij}) = 1.$

После преобразования получим матрицу

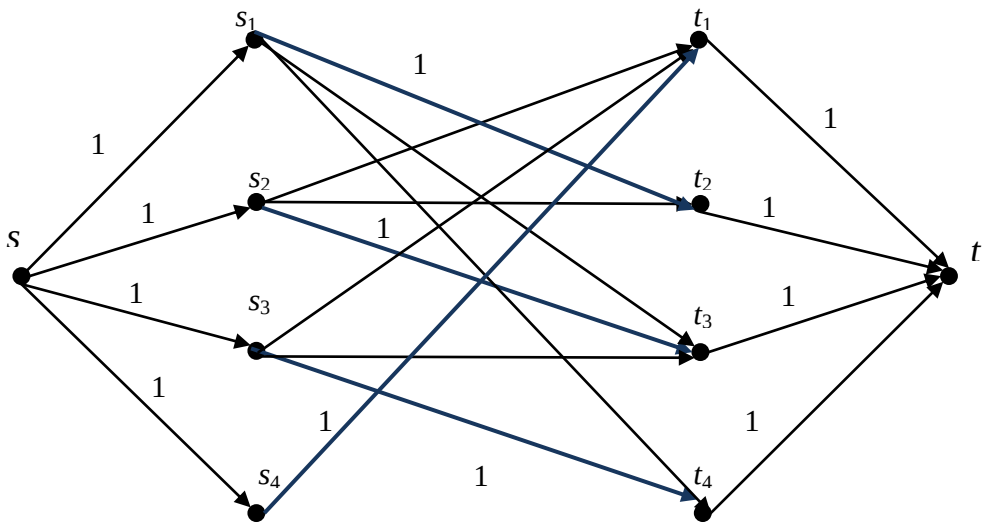
$$C' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

Итерация 3. Строим сеть. Максимальный поток в сети равен 4. Следовательно, решение получено. На сети соответствующие дуги с еди-

ничным потоком выделены. Получаем назначение: $x_{12} = 1$, $x_{23} = 1$, $x_{34} = 1$, $x_{41} = 1$. Значение целевой функции равно 13.

Примечания. 1. Можно рассматривать задачу о назначениях, в которой количество исполнителей больше числа работ. В этом случае матрицу дополняют нулями до квадратной матрицы и решают задачу обычным методом.

2. В некоторых случаях задачу о назначении необходимо решать на максимум. Тогда поступают следующим образом: находят $c_0 = \max\limits_{ij} c_{ij}$; переходят к матрице с элементами $c'_{ij} = c_0 - c_{ij}$; решают для этой матрицы задачу минимизации.



Задачи для самостоятельного решения

В следующих задачах найти оптимальное назначение, соответствующее минимальному значению целевой функции.

1.

4	0	8	6	5
3	0	2	3	3
0	0	0	0	0
4	0	7	6	8
5	0	1	4	3

2.

0	0	0	0	0	0
4	2	0	1	7	6
3	2	0	3	4	1
8	7	0	9	6	7
9	8	0	5	6	7
2	8	0	3	5	8

3.

$$\begin{vmatrix} 2 & 7 & 5 & 0 & 2 & 1 \\ 3 & 4 & 6 & 0 & 3 & 2 \\ 7 & 6 & 7 & 0 & 3 & 1 \\ 5 & 7 & 5 & 0 & 2 & 4 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 6 & 4 & 2 & 0 & 6 & 7 \end{vmatrix}$$

4.

$$\begin{vmatrix} 8 & 9 & 7 & 5 & 0 & 4 \\ 5 & 8 & 4 & 4 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 7 & 6 & 5 & 0 & 4 \\ 8 & 4 & 3 & 7 & 0 & 5 \\ 2 & 4 & 6 & 8 & 3 & 7 \end{vmatrix}$$

5.

$$\begin{vmatrix} 21 & 10 & 25 & 27 & 10 \\ 13 & 11 & 32 & 14 & 12 \\ 23 & 12 & 10 & 15 & 20 \\ 0 & 0 & 0 & 0 & 0 \\ 15 & 11 & 24 & 30 & 16 \\ 17 & 12 & 25 & 27 & 28 \end{vmatrix}$$

6.

$$\begin{vmatrix} 8 & 11 & 0 & 2 & 6 & 1 \\ 12 & 13 & 0 & 7 & 9 & 5 \\ 9 & 7 & 0 & 5 & 5 & 4 \\ 3 & 16 & 0 & 9 & 6 & 3 \\ 4 & 8 & 0 & 1 & 2 & 4 \\ 7 & 12 & 9 & 4 & 5 & 2 \end{vmatrix}$$

7.

$$\begin{vmatrix} 9 & 4 & 2 & 5 & 6 & 8 & 4 \\ 1 & 3 & 5 & 7 & 9 & 11 & 13 \\ 12 & 3 & 1 & 8 & 2 & 6 & 4 \\ 11 & 8 & 2 & 4 & 3 & 2 & 5 \\ 21 & 8 & 6 & 1 & 8 & 2 & 4 \\ 15 & 7 & 2 & 6 & 3 & 8 & 2 \\ 17 & 2 & 6 & 3 & 8 & 2 & 4 \end{vmatrix}$$

8.

$$\begin{vmatrix} 4 & 8 & 2 & 9 & 1 & 9 & 6 \\ 1 & 5 & 5 & 4 & 5 & 1 & 3 \\ 2 & 6 & 3 & 9 & 9 & 2 & 7 \\ 2 & 2 & 6 & 4 & 6 & 7 & 2 \\ 6 & 3 & 4 & 4 & 3 & 6 & 7 \\ 4 & 2 & 4 & 8 & 9 & 3 & 1 \\ 7 & 1 & 2 & 9 & 4 & 6 & 3 \end{vmatrix}$$

9.

$$\begin{vmatrix} 5 & 2 & 9 & 6 & 9 & 5 & 6 & 7 \\ 7 & 2 & 8 & 4 & 4 & 8 & 7 & 9 \\ 6 & 3 & 5 & 5 & 3 & 1 & 5 & 8 \\ 1 & 4 & 4 & 2 & 1 & 2 & 7 & 6 \\ 7 & 2 & 3 & 6 & 2 & 3 & 1 & 8 \\ 2 & 1 & 7 & 2 & 4 & 1 & 2 & 5 \\ 3 & 4 & 5 & 8 & 3 & 3 & 7 & 4 \\ 6 & 7 & 9 & 8 & 4 & 1 & 3 & 7 \end{vmatrix}$$

10.

$$\begin{vmatrix} 2 & 4 & 5 & 7 & 4 & 0 & 8 & 1 & 6 \\ 5 & 6 & 8 & 3 & 4 & 4 & 8 & 2 & 9 \\ 5 & 5 & 8 & 1 & 4 & 7 & 8 & 9 & 2 \\ 1 & 3 & 5 & 7 & 9 & 0 & 2 & 4 & 6 \\ 7 & 6 & 5 & 4 & 3 & 7 & 7 & 9 & 3 \\ 1 & 3 & 6 & 7 & 9 & 6 & 7 & 5 & 8 \\ 5 & 6 & 6 & 7 & 8 & 9 & 3 & 2 & 4 \\ 1 & 5 & 7 & 8 & 9 & 8 & 6 & 5 & 5 \\ 7 & 8 & 5 & 3 & 3 & 2 & 4 & 7 & 5 \end{vmatrix}$$

В следующих задачах найти оптимальное назначение, соответствующее максимальному значению целевой функции

1.

20	10	25	27	10
1	12	10	15	2
8	9	28	20	12
14	11	24	21	16
10	8	25	27	28

2.

32	14	28	29	40
14	25	27	16	25
41	27	36	57	23
34	52	57	16	19
37	42	18	27	37

3.

20	31	15	19	8	55
19	55	22	31	7	35
25	43	23	53	57	16
5	50	49	30	39	9
24	34	33	5	45	14
34	26	6	3	36	37

4.

1	31	15	8	8	18
19	55	22	23	7	1
5	43	23	7	43	16
8	50	2	30	23	9
4	34	22	5	17	14
34	26	26	4	36	29

4.2. Задача о назначениях на узкие места

Имеется n лиц и n работ. Заданы эффективности a_{ij}, $i, j = \overline{1,n}$, исполнения каждым лицом каждой работы. Каждый исполнитель должен назначаться на выполнение только одной работы и все работы должны быть выполнены. Требуется найти такое назначение исполнителей на работы, при котором наименьшая эффективность выполнения работ максимальна.

Если, например, на конвейере имеется n рабочих мест и n работников, каждый из которых может выполнять любую работу за некоторое заданное время, то для достижения максимальной скорости движения конвейера надо распределить работников на конвейере так, чтобы минимальное из времен выполнения работ было максимально.

То назначение исполнителя на работу, на котором реализуется минимальная эффективность, называют *узким местом в назначении.*

Нетрудно видеть, что каждое назначение задается взаимно однозначным отображением множества $\{1,2,\ldots,n\}$ на себя, т. е. образует подстановку

$$P = \begin{pmatrix} 1 & 2 & \ldots & n \\ p(1) & p(2) & \ldots & p(n) \end{pmatrix},$$

где $p(i)$ указывает номер назначаемой i-у исполнителю работы. Количество всевозможных назначений работников на работы равно $n!$

Рассмотрим алгоритм Гросса для решения задачи.

Начальный шаг. Фиксируем любое назначение P_0 , что соответсвует выбору в строках и столбцах матрицы эффективностей $A=\left| a_{ij} \right|_{n\times n}$ ровно по одному элементу. При этом назначении вычисляется значение целевой функции, т. е. вычисляется величина $F(P_0)=\min\{a_{1p_0(1)},...,a_{np_0(n)}\}$. Обозначим эту величину через *s*.

Шаг 1. Строим двудольный граф $G = (V_1 \cup V_2 , E)$, вершины первой доли которого соответсвуют исполнителям, второй – работам. Ребро $(v_i,u_j) \in E$ тогда и только тогда, когда $a_{ij} > s$.

Шаг 2. Ищем максимальное (по числу ребер) паросочетание в графе $G = (V_1 \cup V_2 , E)$.

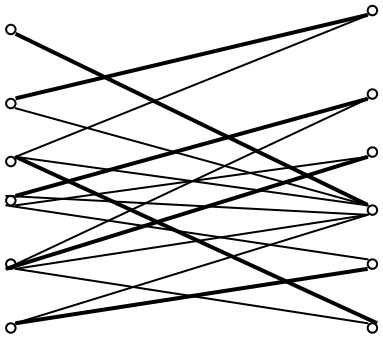
Если паросочетание имеет ровно *n* ребер, то по ним строим новое назначение с более высокой минимальной эффективностью (следует из способа построения двудольного графа). Обозначим ее снова через *s* и вернемся к шагу 1.

Если же число ребер в паросочетании окажется меньше *n*, то имеющееся назначение оптимально.

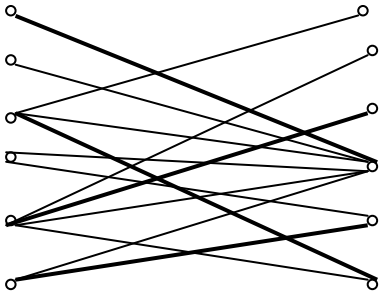
Пример. Пусть задана матрица эффективностей:

$$A=\begin{vmatrix} 1 & 3 & 2 & 6 & 0 & 1 \\ 4 & 2 & 3 & 8 & 3 & 1 \\ 8 & 1 & 1 & 5 & 0 & 9 \\ 3 & 4 & 4 & 8 & 8 & 3 \\ 2 & 9 & 9 & 5 & 2 & 9 \\ 3 & 3 & 3 & 6 & 7 & 1 \end{vmatrix}$$

Возьмем назначение $P_0=\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 1 & 4 & 3 & 6 & 5 \end{pmatrix}$. Имеем $F(P_0) = 3$. Строим двудольный граф:



Количество ребер в максимальном паросочетании равно 6, поэтому находим новую подстановку $P_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 3 & 6 & 2 & 3 & 5 \end{pmatrix}$. Для этой подстановки имеем $F(P_1) = 4$. Двудольный граф имеет максимальное паросочетание с 4 ребрами. Следовательно, назначение P_1 оптимально.



Для построения максимального паросочетания в двудольном графе можно использовать алгоритм Форда-Фалкерсона для сети, полученной из исходного двудольного графа добавлением источника, связанного с каждой вершиной первой доли, и стока, связанного с вершинами второй доли. Пропускные способности дуг полагаем равными единице. Очевидно, что величина максимального потока в сети будет равна мощности максимального паросочетания в исходном графе, причем насыщенные ребра от вершин первой доли к вершинам второй, дают ребра этого паросочетания.

Приведем алгоритм Кенига-Эгервари, построения максимального паросочетания в двудольном графе. Пусть $G = (V_1 \cup V_2, E)$ двудольный граф с $|V_1| = n$, $|V_2| = m$.

Начальный шаг. Строим таблицу размером nxm , строки которой соответсвуют вершинам первой, а столбцы – вершинам второй долей. В клетку (i, j) ставим символ * и называем ее недопустимой, если в графе нет ребра (v_i, u_j) для вершин $v_i \in V_1$, $u_j \in V_2$. Если $(v_i, u_j) \in E$, то клетку оставляем свободной и называем допустимой. Множество допустимых клеток называем независимым, если среди них нет двух, стоящих в одной строке или одном столбце. Очевидно, что между множествами независимых допустимых клеток построенной таблицы и паросочетаниями исходного двудольного графа существует взаимно однозначное соответсвие. И максимальным паросочетаниям будет соответсвовать множества независимых допустимых клеток с максимальным числом клеток.

Шаг 1. Строим произвольное множество независимых допустимых клеток, помещая в вошедшие в него клетки символ «1». Например, просмотром в порядке возрастания номеров строк слева направо и фиксаци-

ей «1» в первой по ходу просмотра допустимой клетке, которая является независимой по отношению к допустимым клеткам, отмеченных ранее.

Если окажется, что во всей таблице все клетки недопустимы, то это означает, что в графе нет ребер.

Шаг 2. Находим в таблице строки без символа «1» и помечаем их символом «-» и переходим к следующему шагу. Если в каждой строке таблицы окажется символ «1», то ребра соответсвующего паросочетания составляют наибольшее паросочетание, и действия окончены.

Шаг 3. Просмотрим все получившие пометки строки в порядке возрастании их номеров. Просмотр очередной строки состоит в следующем: в строке отыскиваются допустимые клетки, и столбцы, в которых эти клетки расположены, помечаются меткой (+*p*), где *p* - номером просматриваемой строки. При этом соблюдается принцип: если пометка уже стоит, то на ее место вторая не ставится.

Если в результате ни один из не имевших метку столбцов не будет помечен, то это означает, что уже имеющееся паросочетание является искомым наибольшим и все действия прекращаются. Если некоторые из непомеченных столбцов получат метки, то переходим к следующему шагу.

Шаг 4. Просмотрим помеченные на шаге 3 столбцы в порядке возрастания их номеров. Просмотр столбца состоит в следующем: в столбце отыскивается символ «1» и строка, в которой он расположен, помечается меткой (-*h*), где *h* - номер просматриваемого столбца. При этом соблюдается прежний принцип: если столбец уже помечен, то метка не изменяется.

Если возникает ситуация, когда в просматриваемом столбце нет символа «1», то действия на данном шаге прерываются, и осуществляется переход к следующему шагу 5.

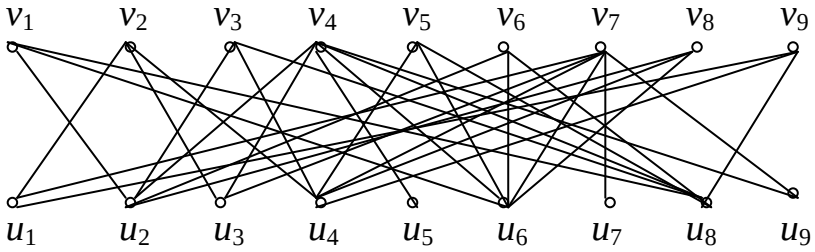
Если же в результате действий шага 4 будут просмотрены все помеченные столбцы и, возникнет набор новых помеченных строк, то следует вернуться к Шагу 3.

Наконец, если в результате действий шага 4 не возникнет новых помеченных строк, то это означает, что имеющееся паросочетание является искомым.

Шаг 5. Производим изменение множества независимых допустимых клеток в таблице. Рассматриваем столбец, имеющий пометку и не содержащий символа «1». В нем ставим символ «1» в строку, номером которой помечен этот столбец. Затем в этой строке ищем «старый» символ «1» и перемещаем его по столбцу в строку, номер которой равен пометке при этом столбце. Далее в строке, куда попал последний символ «1» ищем «старый» символ «1» и с ним проделываем то же самое. В конце концов очередной перемещаемый «старый» символ «1» окажется в стро-

ке, где больше символов «1» нет, то есть в строке, помеченную символом «-». Возник новый набор независимых допустимых клеток, в котором элементов на один больше, чем в прежнем. Все метки уничтожаются и осуществляется переход на шаг 2.

Пример. Найти максимальное паросочетание для следующего двудольного графа:



Начальный и первый шаг. Соответсвующая таблица с первоначальным множеством независимых допустимых клеток имеет вид:

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

Первая итерация выполнения шагов 2 – 4 даст таблицу с метками:

	1	2	3	4	5	6	7	8	9	
1	*	1	*	*	*		*		*	-2
2	1	*			*	*	*	*	*	-1
3	*		*	1	*	*	*		*	-4
4	*		1				*			-3
5	*	*	*		*	1	*		*	-6
6	*		*	*	*		*	1	*	-8
7		*			*		1	*		
8	*		*		*		*	*	*	-
9		*	*		*	*	*		*	-
	+9	+8	+2	+8	+4	+8		+9	+4	

Выполнение шага 5 увеличивает количество независимых допустимых клеток до 8.

Выполнение второй итерации приведет к следующей таблице с метками:

	1	2	3	4	5	6	7	8	9	
1	*	1	*	*	*		*		*	-2
2		*	1		*	*	*	*	*	
3	*		*	1	*	*	*		*	-4
4	*				1		*			
5	*	*	*		*	1	*		*	-6
6	*		*	*	*		*	1	*	-8
7		*			*		1	*		
8	*		*		*		*	*	*	-
9	1	*	*		*	*	*		*	
		+8		+8		+8		+1		

Поскольку больше меток установить нельзя и столбец без символа «1» не помечен, то алгоритм завершает работу.

Задачи для самостоятельного решения

Найти решения задач о назначениях на узкие места для следующих матриц эффективностей:

1.

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

2.

7	6	8	3	2	1	4
9	5	8	7	4	3	2
10	4	7	8	9	6	5
3	9	8	2	5	7	4
2	7	3	4	2	1	8
4	8	4	5	8	3	9
5	9	2	3	7	6	8

3.

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

4.

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

5. ЗАДАЧА КОММИВОЯЖЕРА

Имеется n городов, между которыми существует определенная сеть дорог. Коммивояжер, выходя из некоторого города, должен обойти все остальные города, побывав в каждом из них по одному разу, и вернуться в исходный город. При этом пройденное коммивояжером расстояние должно быть минимально. Расстояния между городами задаются матрицей $C = | c_{ij} |_{nxn}$, у которой элемент $c_{ij} \geq 0$ равен длине дороги, непосредственно соединяющей i-ый и j-ый города, и равен ∞ при отсутствии дороги, связывающей непосредственно i-ый и j-ый города. Кроме этого $c_{ii} = \infty$, $i=\overline{1,n}$, для запрещения коммивояжеру возвращаться сразу же в город.

Для графа $G = (V, E)$ гамильтоновым называется цикл, содержащий все вершины графа. Если построить граф, вершины которого соответствуют городам, а дуги (ребра) непосредственно связывающим города дорогам, то приписав дугам длины, совпадающие с длинами дорог, мы получим задачу нахождения гамильтонова цикла минимальной длины.

Введем для дуг графа G в рассмотрение переменные

$$x_e = \begin{cases} 1, \text{если дуга } e \text{ включена в гамильтонов цикл}, \\ 0, \text{ в противном случае}. \end{cases}$$

Тогда задачу коммивояжера можно сформулировать в виде задачи линейного булевого программирования:

$$\sum_{e \in E} c_e x_e \to \min,$$

$$\sum_{e \in \vartheta} x_e = n, \text{ для любого гамильтонова цикла } \vartheta,$$

$$\sum_{e \in \vartheta} x_e \leq |\vartheta| - 1, \text{для любого не гамильтонова цикла } \vartheta.$$

Здесь c_e - длина дуги $e \in E$.

Для решения задачи коммивояжера применяется одна из реализаций метода ветвей и границ.

5.1. Общая схема метода ветвей и границ

Метод ветвей и границ - общий метод для нахождения решений задач дискретной и комбинаторной оптимизации. Метод является алгоритмом перебора с отсевом подмножеств допустимых решений, не содержащих оптимальных решений.

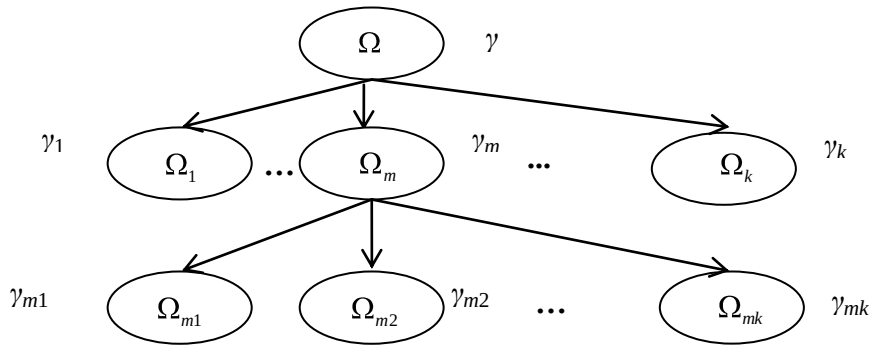
Опишем идею метода на примере поиска минимума функции $f(x)$ на конечном множестве допустимых значений Ω. Метод ветвей и границ

основан на трех процедурах: *ветвление, нахождение оценок* (границ), *отсев вариантов*.

Ветвление состоит в разбиении по некоторому правилу *A* множества допустимых решений на подмножества $\Omega_i,\ i=\overline{1,k} : \Omega=\bigcup\limits_{i=1}^{k}\Omega_i,$ $\Omega_i \cap \Omega_j = \emptyset,\ i \neq j$. Процедура нахождения оценок заключается в поиске по некоторому правилу *B* нижних границ для минимальных значений функции *f*(*x*) на Ω и $\Omega_i,\ i=\overline{1,k}$. Пусть $\gamma,\ \gamma_i,\ i=\overline{1,k},$ полученные нижние границы. Очевидно, $\gamma \le \gamma_i,\ i=\overline{1,k}$.

Из полученных подмножеств $\Omega_i,\ i=\overline{1,k}$, выбираем подмножество Ω_m, у которого $\gamma_m = \min\limits_{i=\overline{1,k}} \gamma_i$. По правилу *A* разбиваем Ω_m на подмножества $\Omega_{mj},\ j=\overline{1,k} : \Omega_m = \bigcup\limits_{j=1}^{k}\Omega_{mj},\ \Omega_{mj} \cap \Omega_{mp} = \emptyset,\ j \neq p$, и вычисляем по правилу *B* нижние границы для минимальных значений функции *f*(*x*) на $\Omega_{mj},\ j=\overline{1,k}$.

Описанный процесс можно изобразить в виде дерева поиска, корень которого соответсвует исходному множеству допустимых решений Ω, вершины - подмножествам, полученным в результате разбиений по правилу *A*, дуги идут от вершины-подмножества к вершинам-подмножествам, состовляющим его разбиение.



Общий шаг метода ветвей и границ состоит в выборе из всех висячих вершин дерева поиска вершины с наименьшей границей, разбиении соответствующего подмножества допустимых решений по правилу *A*, вычислении по правилу *B* для вновь полученных подмножеств нижних гра-

ниц с добавлением в дерево поиска вершин и дуг, соответсвующих вновь полученным подмножествам.

В силу конечности множества Ω, за конечное число шагов хотя бы одна из висячих вершин будет соответствовать множеству, состоящему из одного допустимого решения $\{x^*\}$. Значение функции $f^* = f(x^*)$ называют *рекордом*.

Процедура отсева основана на следующем простом принципе: если нижняя граница γ' для подмножества Ω' дерева поиска не меньше f^*, то все допустимые решения из множества Ω' могут быть исключены из дальнейшего рассмотрения (вершина Ω' в дереве поиска далее не ветвиться).

Если нижние границы для всех висячих вершин дерева поиска не меньше рекорда, то решение найдено: $\min\limits_{x\in\Omega} f(x) = f(x^*)$.

Если мы получаем висячую вершину, соответствующую одному допустиму решению $\{x^0\}$, и $f(x^0) < f^*$, то рекорд обновляется $f^* = f(x^0)$.

Чтобы использовать метод ветвей и границ для решения конкретного класса задач, необходимо указать правило ветвления и правило вычисления нижних границ.

5.2. Алгоритм Литтла

Алгоритм Литтла является реализацией метода ветвей и границ для задачи коммивояжера. Правило ветвления состоит в разбиении множества рассматриваемых гамильтоновых циклов на два подмножества, одно из которых состоит из циклов, содержащих выбранную дугу *e*, а другое из циклов, не содержащих этой дуги. Дуга *e* выбирается среди дуг минимальной длины по условию, что запрет на использование этой дуги должен приводить к максимальному увеличению длины гамильтонова цикла. Правило вычисления нижних границ основано на процедуре приведения матриц расстояний, соответствующих вновь полученным висячим вершинам дерева поиска.

Опишем алгоритм Литтла.

Шаг 1. Приведение матрицы расстояний. Находим в каждой *i*-ой строке матрицы $C = \left| c_{ij} \right|_{n\times n}$ минимальный элемент α_i и вычитаем его из всех элементов этой строки. В полученной матрице в каждом *j*-ом столбце находим минимальный элемент β_j и вычитаем его из всех элементов данного столбца. После проделанных операций получим матрицу C', каждая строка и каждый столбец которой содержит, по крайней мере, один нуль.

Шаг 2. Вычисляем сумму приводящих констант $\gamma = \sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{n} \beta_j$.

Очевидно, что γ является нижней границей для всего множества решений Ω, которое берется в качестве корня дерева поиска и текущего множества решений.

Шаг 3. Для каждого нулевого элемента $c'_{ij} = 0$ матрицы C' находим штраф за неиспользование $\theta_{ij} = \min_{k \neq j} c'_{ik} + \min_{s \neq i} c'_{sj}$ (сумма минимальных элементов в строке и столбце, на пересечении которых стоит нуль, без учета самого нулевого элемента).

Шаг 4. Выбираем нулевой элемент с максимальным штрафом $\theta_{ij} = \max_{c'_{sk}=0} \theta_{sk}$. Разбиваем текущее множество всех гамильтоновых циклов Ω на два подмножества: Ω_1 - "не включающие дугу (*i, j*)" и Ω_2 - "включающие дугу (*i,j*)". Присоединяем соответсвующие вершины к дереву поиска.

Шаг 5. Вычисляем нижнюю границу $\gamma_1 = \gamma + \theta_{ij}$ для гамильтоновых циклов подмножества Ω_1 .Строим соответсвующую Ω_1 матрицу расстояний C'_1. Для этого значение элемента c'_{ij} заменяем на ∞ и приводим полученную матрицу (неприведенными могут быть только *i*-ая строка и *j*-ый столбец, поэтому сумма приводящих констант равна θ_{ij}) .

Шаг 6. Вычисляем нижнюю границу для гамильтоновых циклов подмножества Ω_2. Для этого удаляем из матрицы C' *i*-ю строку и *j*-й столбец, сохраняя исходную нумерацию для оставшихся строк и столбцов, и заменяем на ∞ значение элементов, соответсвующих дугам, использование каждой из которых с уже включенными в гамильтонов цикл дугами, приводит к образованию цикла с числом дуг меньше *n*. Приводим полученную матрицу, обозначаем ее C'_2 и добавляем сумму приводящих констант к нижней границе γ множества решений Ω. Получаем нижнюю границу γ_2 .

Шаг 7. Если в результате шага 6 получаем матрицу C'_2 порядка два и ее нижняя граница не превышает границ висячих вершин дерева поиска, то процесс заканчивается, решение найдено, преходим к шагу 11. В противном случае переходим к шагу 8.

Шаг 8. Среди висячих вершин построенного дерева поиска выбираем вершину с наименьшей границей (если таких вершин несколько, выбираем любую из них).

Шаг 9. Если выбранная на шаге 8 вершина соответствует свойству "включающие дугу (*i,j*)", то соответствующую ей матрицу расстояний

C'_2, полученную на шаге 6, берем за C' и переходим к шагу 3. В противном случае переходим к шагу 10.

Шаг 10. Выбранная на шаге 8 вершина соответствует свойству “не включающие дугу (i, j)”). Соответствующую ей матрицу расстояний C'_1, полученную на шаге 5, берем за C' и переходим к шагу 3.

Шаг 11. Строим гамильтонов цикл минимальной длины. Для этого включаем в гамильтонов цикл дуги соответствующие нулевым элементам 2x2-матрицы расстояний C'_2, полученной на шаге 7. Далее двигаемся от висячей вершины к корню дерева поиска по единственному обратному пути. При прохождении обратной дуги дерева поиска, соответствующей переходу от множества решений к его подмножеству по свойству “ включающие дугу (i, j)”, дугу (i, j) включаем в гамильтонов цикл.

Пример. Решить задачу коммивояжера со следующей матрицей расстояний C

$$\begin{vmatrix} \infty & 5 & 2 & 4 & 5 \\ 3 & \infty & 3 & 5 & 8 \\ 4 & 2 & \infty & 3 & 7 \\ 3 & 5 & 3 & \infty & 2 \\ 1 & 4 & 2 & 5 & \infty \end{vmatrix}$$

Приведем матрицу по строкам. Имеем: $\alpha_1 = 2$, $\alpha_2 = 3$, $\alpha_3 = 2$, $\alpha_4 = 2$, $\alpha_5 = 1$. Получим матрицу

$$\begin{vmatrix} \infty & 3 & 0 & 2 & 3 \\ 0 & \infty & 0 & 2 & 5 \\ 2 & 0 & \infty & 1 & 5 \\ 1 & 3 & 1 & \infty & 0 \\ 0 & 3 & 1 & 4 & \infty \end{vmatrix}$$

Приведем матрицу по столбцам. Имеем: $\beta_1 = 0$, $\beta_2 = 0$, $\beta_3 = 0$, $\beta_4 = 1$, $\beta_5 = 0$. Получим матрицу C'

$$\begin{vmatrix} \infty & 3 & 0 & 1 & 3 \\ 0 & \infty & 0 & 1 & 5 \\ 2 & 0 & \infty & 0 & 5 \\ 1 & 3 & 1 & \infty & 0 \\ 0 & 3 & 1 & 3 & \infty \end{vmatrix}$$

Текущая нижняя граница $\gamma = 11$.

Находим штрафы для нулевых элементов: $\theta_{13} = 1$, $\theta_{21} = 0$, $\theta_{23} = 0$, $\theta_{32} = 3$, $\theta_{34} = 1$, $\theta_{45} = 4$, $\theta_{51} = 1$. Максимальный штраф $\theta_{45} = 4$.

Разбиваем множество всех гамильтоновых циклов Ω на два подмножества Ω_1 - “не включающие дугу (4,5)” и Ω_2 - “включающие дугу (4,5)”. Для первого подмножества нижняя граница $\gamma_1 = 15$, а соответствующую матрицу расстояний C'_1 получим из матрицы C', положив $c'_{45} = \infty$ и приведя результат.

$$\begin{vmatrix} \infty & 3 & 0 & 1 & 0 \\ 0 & \infty & 0 & 1 & 2 \\ 2 & 0 & \infty & 0 & 2 \\ 0 & 2 & 0 & \infty & \infty \\ 0 & 3 & 1 & 3 & \infty \end{vmatrix}$$

Для второго подмножества матрица расстояний C'_2 получается из C' удалением 4-ой строки и пятого столбца, причем для запрещения образования цикла 4->5->4, полагаем $c'_{54} = \infty$, полученный результат приводим.

	1	2	3	4
1	∞	3	0	1
2	0	∞	0	1
3	2	0	∞	0
5	0	3	1	∞

Сумма приводящих констант равна 0, следовательно, $\gamma_2 = 11$.

Минимальную нижнюю границу имеет множество Ω_2. Поэтому в матрице C'_2 вычисляем штрафы для нулевых элементов: $\theta_{13} = 1$, $\theta_{21} = 0$, $\theta_{23} = 0$, $\theta_{32} = 3$, $\theta_{34} = 1$, $\theta_{51} = 1$. Максимальный штраф $\theta_{32} = 3$. Разбиваем множество Ω_2 на два подмножества Ω_{21} - “не включающие дугу (3,2)” и Ω_{22} - “включающие дугу (3,2)”. Для подмножества Ω_{21} нижняя граница $\gamma_{21} = 14$. Матрицу расстояний C'_{21} получим из матрицы C'_2, положив $c'_{32} = \infty$ и проведя процедуру приведения.

	1	2	3	4
1	∞	0	0	1
2	0	∞	0	1
3	2	∞	∞	0
5	0	0	1	∞

Для подмножества Ω_{22} матрицу расстояний C'_{22} получаем из C'_2, удаляя третью строку и второй столбец, затем для запрещения образования цикла 3->2->3, полагаем $c'_{23} = \infty$, полученный результат приводим.

	1	3	4
1	∞	0	0
2	0	∞	0
5	0	1	∞

Сумма приводящих констант равна 1, следовательно, $\gamma_{22} = 12$.

В дереве поиска висячие вершины соответсвуют подмножествам Ω_1, Ω_{21}, Ω_{22}. Минимальную нижнюю границу имеет множество Ω_{22}. Поэтому в матрице C'_{22} вычисляем штрафы для нулевых элементов: $\theta_{13} = 1$, $\theta_{14} = 0$, $\theta_{21} = 0$, $\theta_{24} = 0$, $\theta_{51} = 1$. В результате сравнения мы получили два одинаковых максимальных штрафа равных 1.

Возьмем $\theta_{13} = 1$. Разбиваем множество Ω_{22} на два подмножества Ω_{221} - “не включающие дугу (1,3)” и Ω_{222} - “включающие дугу (1,3)”. Для подмножества Ω_{221} нижняя граница $\gamma_{221} = 13$. Матрицу расстояний C'_{221} получим из матрицы C'_{22}, положив $c'_{13} = \infty$ и проведя процедуру приведения.

	1	3	4
1	∞	∞	0
2	0	∞	0
5	0	0	∞

Для подмножества Ω_{222} матрицу расстояний C'_{222} получаем из C'_{22}, удаляя первую строку и третий столбец, затем для запрещения образования цикла 1->3->2->1, полагаем $c'_{21} = \infty$, полученный результат приводим. Сумма приводящих констант равна 0, следовательно, $\gamma_{222} = 12$.

	1	4
2	∞	0
5	0	∞

В дереве поиска висячие вершины соответсвуют подмножествам Ω_1, Ω_{21} , Ω_{221} , Ω_{222} . Минимальную нижнюю границу имеет множество Ω_{222}. Соответсвующая матрица C'_{222} имеет размерность 2x2. Следовательно, решение найдено.

Переходим к постоению гамильтонового цикла. Включаем в гамильтонов цикл дуги (2,4), (5,1), как соответствующие нулевым элементам матрицы C'_{222}. Затем, двигаясь по дереву поиска к корню, включаем дуги (1,3), (3,2), (4,5). Дерево поиска приведено на рисунке 13.

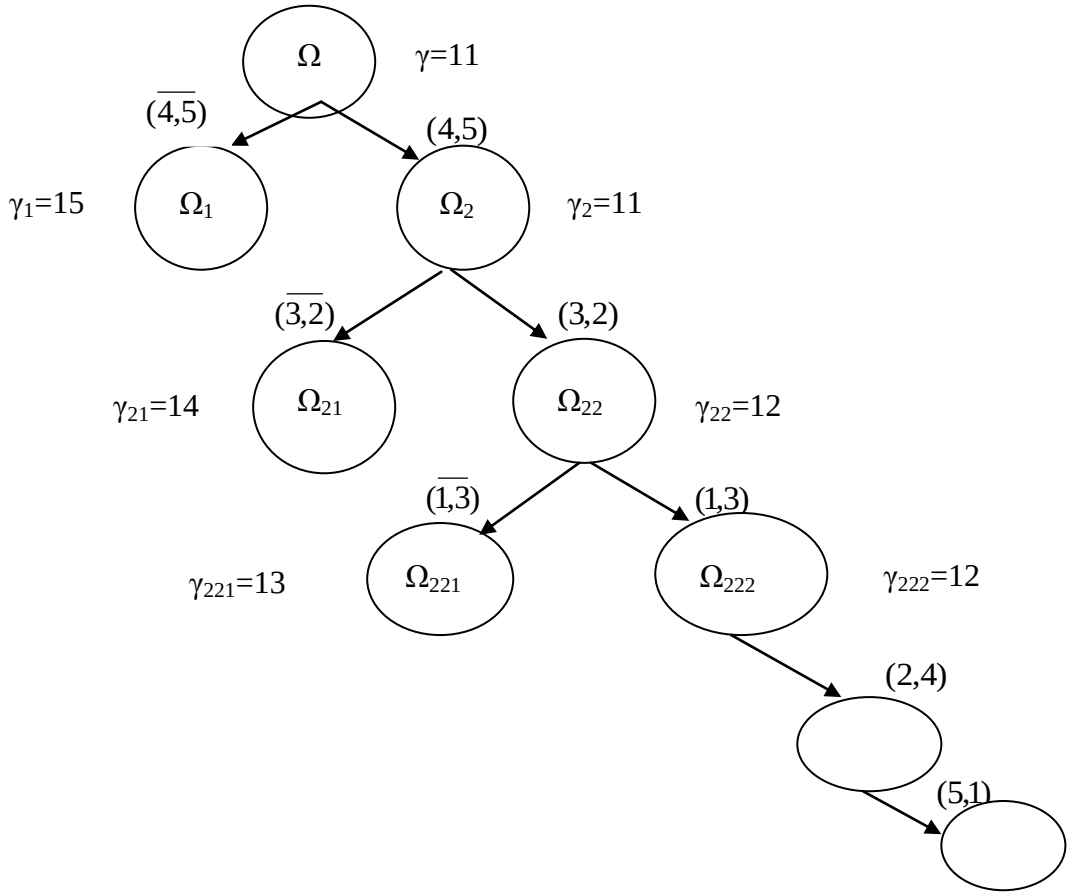


Рис. 13.

Гамильтонов цикл минимальной длины состоит из дуг (5,1), (1,3), (3,2), (2,4), (4,5).

Задачи для самостоятельного решения

Найти оптимальные маршруты для следующих задач коммивояжера.

1.

∞	1	0	4	5
2	∞	0	3	2
0	0	∞	0	0
3	4	0	∞	2
1	3	3	2	∞

2.

∞	0	2	3	4
0	∞	0	0	0
3	0	∞	2	1
2	0	3	∞	2
1	0	4	1	∞

3.

∞	7	0	9	4	5
1	∞	0	7	8	2
0	0	∞	0	0	0
2	9	0	∞	8	7
9	3	0	3	∞	4
4	8	0	2	4	∞

4.

∞	5	2	7	9
3	∞	8	6	2
7	3	∞	9	9
8	8	9	∞	5
3	6	4	6	∞

5.

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

6.

∞	0	5	4	0	6
5	∞	3	0	1	6
3	2	∞	0	3	2
2	5	0	∞	6	0
0	4	2	0	∞	2
0	1	6	4	6	∞

7.

∞	68	73	0	70	9
58	∞	16	0	11	2
63	9	∞	0	13	18
0	0	0	∞	0	0
4	17	12	0	∞	12
14	18	9	0	48	∞

8.

∞	2	4	0	3	7	1
6	∞	5	0	4	9	8
0	0	∞	0	0	0	0
3	1	2	∞	8	7	9
7	6	4	0	∞	5	8
6	6	9	0	7	∞	4
2	7	8	0	6	5	∞

9.

∞	2	4	0	3	7	1	6
6	∞	5	0	4	8	9	7
3	1	∞	0	2	4	5	9
4	6	2	∞	1	8	7	5
0	0	0	0	∞	0	0	0
8	3	6	0	7	∞	1	5
7	5	3	0	9	6	∞	3
9	5	2	0	1	4	0	∞

10.

∞	5	3	2	4	5	2	7
7	∞	1	6	7	8	9	6
3	2	∞	3	2	4	5	7
4	6	2	∞	1	3	6	3
5	5	1	2	∞	4	3	2
3	8	9	6	4	∞	1	5
1	6	6	4	2	1	∞	0
7	8	7	2	8	3	0	∞

ЛИТЕРАТУРА

1. *Филипс, Д.* Методы анализа сетей / Д. Филипс, А. Гарсиа-Диас. М. : Мир, 1984. 496 с.
2. *Морозов, В. В.* Исследование операций в задачах и упражнениях / В. В. Морозов, А. Г. Сухарев, В. В. Федоров. М. : Высш. шк., 1986. 287 с.
3. *Давыдов, Э. Г.* Исследование операций: учеб. пособие для студентов вузов / Э. Г. Давыдов. М. : Высш. шк., 1990. 383 с.
4. *Бахтин, В. И.* Исследование операций : курс лекций / В. И. Бахтин, А. П. Коваленок, А. В. Лебедев, Ю. В. Лысенко. Минск : БГУ, 2003. 199 с.
5. *Таха, Хемди А.* Введение в исследование операций. 7-е издание / Хемди А. Таха. М. : Издательский дом «Вильямс», 2005. 912 с.
6. *Морозов, В. В.* Исследование операций / В. В. Морозов, А. А. Васин, П. С. Краснощеков. М. : Academia, 2008. 464 с.
7. *Костевич, Л. С.* Исследование операций. Теория игр : учеб. пособие / Л. С. Костевич, А. А. Лапко. 2-е изд., прераб. и доп. Минск : Выш. шк., 2008. 368 с.

СОДЕРЖАНИЕ

Учебное издание

ИССЛЕДОВАНИЕ ОПЕРАЦИЙ В ЗАДАЧАХ

Учебно-методическое пособие для студентов факультета прикладной математики и информатики

В трёх частях

Часть II

СЕТЕВЫЕ ЗАДАЧИ

А в т о р ы - с о с т а в и т е л и

Исаченко Александр Николаевич

Дробушевич Любовь Федоровна

В авторской редакции

Ответственный за выпуск *А. Н. Исаченко*

Подписано в печать 21.12.2011. Формат 60×84/16. Бумага офсетная.
Гарнитура Таймс. Усл. печ. л. 3,72. Уч.-изд. л. 3,04. Тираж 50 экз. Зак.

Белорусский государственный университет.
ЛИ № 02330/0494425 от 08.04.2009.
Пр. Независимости, 4, 220030, Минск.

Отпечатано с оригинала-макета заказчика
на копировально-множительной технике
факультета прикладной математики и информатики
Белорусского государственного университета.
Пр. Независимости, 4, 220030, Минск.