МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ, УКРАИНЫ

ОДЕССКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ ИМЕНИ И. И. МЕЧНИКОВА

ИНСТИТУТ МАТЕМАТИКИ, ЭКОНОМИКИ И МЕХАНИКИ

А. В. Арсирий, Б. Ф. Трофимов, Е.М. Страхов

Сетевые модели

Методические указания и варианты заданий
для самостоятельной работы

Одеса
2013

УДК 519.87(075.8)
ББК 22.183.5я73
А855

Методические указания и варианты заданий для самостоятельной работы по курсу «Математическое моделирование и системный анализ» для студентов факультета прикладной математики и компьютерных систем, а также факультета информационных технологий.

Составители: **А.В. Арсирий**, к.ф.-м.н., доцент кафедры оптимального управления и экономической кибернетики Одесского национального университета имени И.И. Мечникова;
Б.Ф. Трофимов, к.т.н., старший преподаватель кафедры информационных управляющих систем и технологий Одесского национального политехнического университета;
Е.М. Страхов, ассистент кафедры оптимального управления и экономической кибернетики Одесского национального университета имени И.И. Мечникова.

Рецензенты: **А.В. Плотников**, д. ф.-м. н., профессор, заведующий кафедрой прикладной, вычислительной математики и САПР Одесской государственной академии строительства и архитектуры;
О.Д. Кичмаренко, к. ф.-м. н., доцент, заведующий кафедрой оптимального управления и экономической кибернетики Одесского национального университета имени И.И. Мечникова.

Рекомендовано к печати Ученым советом ИМЕМ ОНУ имени И. И. Мечникова. Протокол № 4 от 21.03.2013.

СОДЕРЖАНИЕ

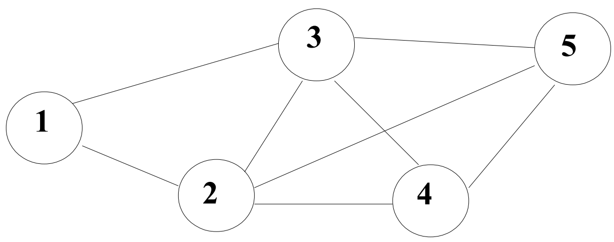
ЛИТЕРАТУРА

1. Таха Хемди А. Введение в исследование операций. 7 издание: Пер. с англ. – М.: Издательский дом «Вильямс», 2006. – 912 с.
2. Таха Хемди А. Введение в исследование операций: В двух книгах. Кн.1.: Пер. с англ. – М.: «Мир», 1985. – 479 с.
3. Вагнер Г. Основы исследования операций. В трех томах. Том 1. Пер. С англ. – М.: Мир, 1973 – 336 с.
4. Аронович А.Б., Афанасьев М.Ю., Суворов Б.П. Сборник задач по исследованию операций. – М. Изд-во МГУ. – 1997. – 256 с.
5. Кормен Томас Х., Лейзерсон Чарльз И., Риверс Рональд Д., Штайн Клиффорд. Алгоритмы: построение и анализ. 2 издание: Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 1296 с.
6. Басакер Р., Саати Т. Конечные графы и сети. Пер. с англ. – М.: Наука, 1974 – 368 с.
7. Кирсанов М.Н. Графы в Maple. Задачи, алгоритмы, программы. – М.: Издательство ФИЗМАТЛИТ, 2007. – 168 с.

Графы и сети. Основные определения

Граф G(V,E) – совокупность двух множеств: вершин V и ребер E, между которыми определено отношение инцидентности. Каждое ребро e из E *инцидентно* ровно двум вершинам v и w, которые оно соединяет. При этом вершина v и ребро e называются инцидентными, а вершины v и w называют *смежными*. Число вершин графа называют *порядком графа* и обозначают n=|V(G)|. Число ребер графа называют *размером графа* и обозначают m=|E(G)|. Если V и E — конечные множества, то G называется *конечным графом*. *Вырожденным* называется граф, если он не имеет ребер.

Если все ребра (v,w) графа не ориентированы, т. е. пары вершин, определяющие элементы множества E, не упорядочены, то такой граф называется *неориентированным графом* или *неорграфом*.



V = {1,2,3,4,5}

E = {(1,2),(1,3),(2,3),(2,4),(2,5),(3,4),(3,5),(4,5)}

Маршрут — это последовательность ребер, в которой каждые два соседних ребра имеют общую вершину. Маршрут в неорграфе, в котором все ребра разные, называется *цепью*. Граф *связен*, если любые две вершины соединены хотя бы одним маршрутом. Если e – это ребро, соединяющее вершины v и w, то v и w — граничные точки e. Если v=w, то e называют *петлей*. Если ребра e_1 и e_2 оба соединяют вершины v и w, то такие ребра называют *кратными* или *параллельными*. Маршрут, в котором начало и конец совпадают, называют *циклическим*. Циклический маршрут называют *циклом*, если он цепь. *Остовом* графа G называют граф, не содержащий циклов и состоящий из ребер графа G и всех его вершин.

Теорема 1. Граф G(V,E) связен тогда и только тогда, когда множество его вершин нельзя разбить на два непустых подмножества V_1 и V_2 так, чтобы обе граничные точки каждого ребра находились в одном и том же подмножестве.

Степень вершины deg(v) – число ребер, инцидентных вершине v. Если deg(v)=0, то вершина v называется *изолированной*. Таким образом, вырожденный граф состоит из изолированных вершин.

Теорема 2. В конечном графе число вершин нечетной степени четно.

Лемма о рукопожатиях. В неорграфе сумма степеней всех вершин равна удвоенному числу ребер.

$$\sum_{i=1}^{n} \deg(v_i) = 2m$$

Матрица смежности графа — квадратная матрица A порядка n, где элемент $a_{i,j}$ равен числу ребер, соединяющих вершины i и j. С графом связывают также *матрицу инцидентности* I. Число строк этой матрицы равно числу вершин, число столбцов — числу ребер.

$I_{v,e}=1$ - если вершина v инцидентна ребру e,

$I_{v,e}=0$ - в противном случае.

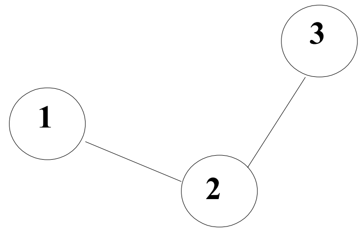
В каждом столбце матрицы инцидентности простого графа (без петель и кратных ребер) содержится по две единицы. Число единиц в строке равно степени соответствующей вершины.

Граф $H(V_1, E_1)$ называется *подграфом* графа $G(V,E)$, если $V_1 \subseteq V$, $E_1 \subseteq E$. Если $V_1 = V$, то подграф называется *остовным*.

Компонента связности графа — это максимальный по включению вершин и ребер связный подграф.

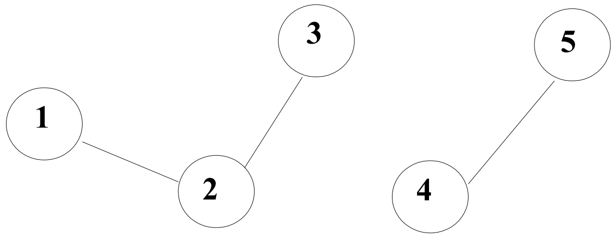
Ранг графа ν = n - k, где n — число вершин, k — число компонент связности.

Граф называется *деревом*, если он связен и не имеет циклов.

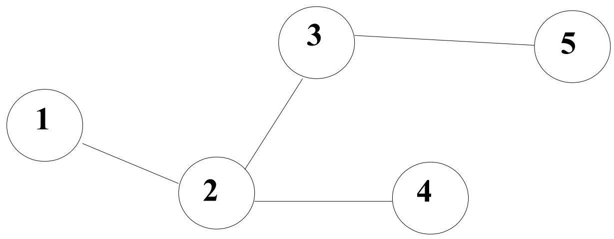


Теорема 3. Каждое дерево с n вершинами имеет в точности n-1 ребро.

Граф, который не содержит циклов, но не обязательно является связным, называется *лесом*. Каждая компонента связности леса является деревом.



Остовом графа называется граф, не содержащий циклов и состоящий из ребер графа и всех его вершин. Остов графа является деревом. Число ребер остова равно рангу графа.



Ребро (v,w) в графе G может быть ориентированным, т.е. его граничные точки представляют собой упорядоченную пару. Такое ребро называется *дугой*, а граф, содержащий дуги, называется *ориентированным* или *орграфом*. При изображении дуга представляется стрелкой.

Маршрут в орграфе — последовательность вершин, соединенных дугами, направленными в одну сторону. Маршрут, в котором все дуги разные есть *путь*. Путь, в котором начало и конец совпадают, называется *контуром*.

Если дугам в орграфе поставлено в соответствие некоторое число (вес), то такой орграф называется *взвешенным*. Взвешенным может быть также и неорграф. Это может быть расстояние между вершинами или какая либо другая характеристика (цена, время, энергия). Вес дуг может быть положительным, отрицательным, нулевым или бесконечным (что соответствует отсутствию дуги).

Взвешенный ориентированный или неориентированный граф часто называют *сетью*, а его вершины — *узлами*. Термины «сеть» и «узел» чаще встречаются в прикладных областях.

Сетевое моделирование

Сетевые оптимизационные модели являются частным случаем моделей линейного программирования. Используя особенности математических характеристик сетевых моделей можно существенно повысить эффективность процесса отыскания оптимального решения задач, которые удается описать на «сетевом языке». В реальных примерах сетевые модели часто содержат тысячи операций (переменных) и сотни ограничений в связи с чем применение эффективных сетевых алгоритмов становится не только выгодным, а просто необходимым. Наконец, исследуя сети, можно убедиться, что разнообразные на первый взгляд совершенно непохожие модели допускают применение общего метода анализа, что несомненно обеспечивает существенные преимущества.

Примерами задач, которые можно представить в виде сетевых моделей являются нахождение кратчайшего маршрута между двумя городами по существующей сети дорог, создание оптимальной кабельной сети между различными районами города и телецентром, определение максимальной пропускной способности трубопровода для транспортировки нефти от пунктов нефтедобычи к нефтеперерабатывающим заводам, составление временного графика строительных работ и многие другие.

Для перечисленных задач и для многих других разработаны различные сетевые оптимизационные алгоритмы. В данном издании будут рассмотрены следующие:

1) Алгоритмы нахождения минимального остовного дерева.
2) Алгоритмы определения максимального потока.

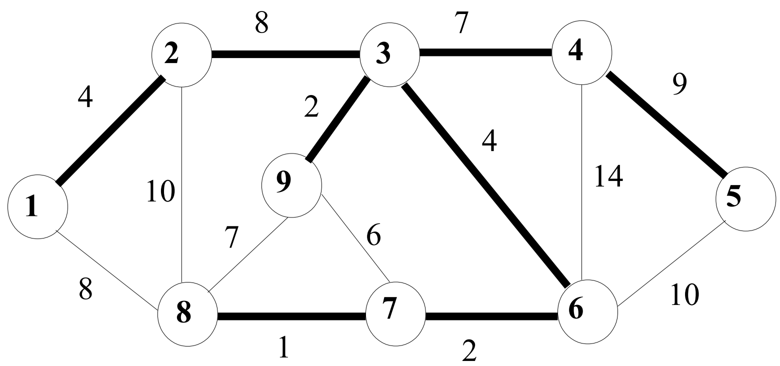
Задача построения минимального остовного дерева

При разработке электронных схем зачастую необходимо электрически соединить контакты нескольких компонентов. Для соединения множества из n контактов мы сможем использовать некоторую компоновку, которая использует минимальное количество провода.

Можно сформулировать эту задачу при помощи неориентированной сети G=(V,E), где V – множество контактов, E – множество возможных соединений между парами контактов. Для каждого ребра $(u,v) \in E$ задан вес $w(u,v)$, определяющий стоимость (количество необходимого провода) соединения u и v. Нужно найти ациклическое подмножество $T \subseteq E$, которое соединяет все вершины и чей общий вес

$$w(T) = \sum_{(u,v) \in T} w(u,v)$$

минимален. Множество T представляет собой остовное дерево. Таким образом задачу поиска дерева T будем называть *задачей поиска минимального остовного дерева* (*minitum spanning-tree problem*).

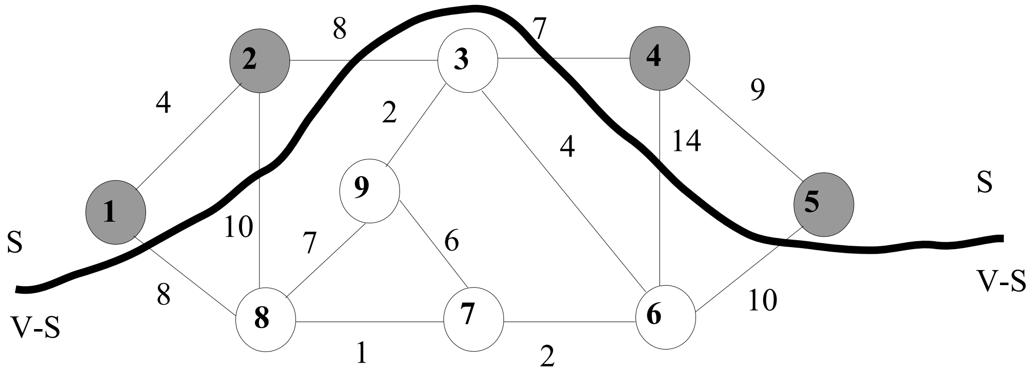


Также типичной задачей является создание (проектирование) сети дорог с твердым покрытием, соединяющих населенные пункты в сельской местности, где дороги, соединяющие два каких-либо пункта, могут проходить через другие населенные пункты. Наиболее экономичный проект дорожной системы должен минимизировать общую длину дорог с твердым покрытием.

Рассмотрим два алгоритма решения задачи поиска минимального остовного дерева: *алгоритм Крускала* (Kruskal) и *алгоритм Прима* (Prim). Оба эти алгоритмы жадные. На каждом шаге алгоритма мы выбираем один из возможных вариантов. Жадная стратегия предполагает выбор варианта, наилучшего в данный момент. В общем случае такая стратегия не гарантирует глобального оптимального решения задачи, однако для задачи поиска минимального остовного дерева доказана эффективность жадных стратегий

Оба алгоритма имеют общую схему, согласно которой минимальное остовное дерево растет путем добавления к нему ребер по одному. Алгоритм работает с множеством ребер А, и инвариант цикла алгоритма выглядит следующим образом: перед каждой очередной итерацией А представляет собой подмножество некоторого минимального остовного дерева. На каждом шаге алгоритма мы определяем ребро (u,v), которое можно добавить к А без нарушения этого инварианта, в том смысле, что $A \cup (u, v)$ также является подмножеством минимального остовного дерева. Такое ребро назовем *безопасным*.

Разрезом (S,V-S) неориентированной сети G(V,E) называется разбиение S, что проиллюстрировано на следующем рисунке,который приведен ниже (вершины в S окрашены в серый цвет, а в V-S – в белый). Будем говорить, что ребро $(u, v) \in E$ *пересекает* разрез (S,V-S), если один из его концов оказывается во множестве S, а второй в V-S. Разрез *согласован* со множеством А *по ребрам*, если ни одно ребро из А не пересекает разрез. Ребро, пересекающее разрез, является легким, если оно имеет минимальнiй вес среди всех ребер, пересекающих разрез.



Теорема 4. (Правило распознавания безопасных ребер.)

Пусть G=(V,E) – связная неориентированная сеть с действительной весовой функцией w, определенной на E. Пусть A — подмножество E, которое входит в некоторое минимальное остовное дерево сети G, (S,V-S) – разрез сети G, согласованный с A по ребрам, а (u,v) – легкое ребро, пересекающее разрез (S,V-S). Тогда ребро (u,v) – является безопасным для A.

Следствие. Пусть G=(V,E) – связная неориентированная сеть с действительной весовой функцией w, определенной на E. Пусть A — подмножество E, которое входит в некоторое минимальное остовное дерево сети G, и пусть $C=(V_A, E_A)$ - связный компонент (дерево) в лесу $G_A=(V, A)$. Если (u,v) – легкое ребро, соединяющее C с некоторым другим деревом в G_A , то ребро (u,v) – безопасно для A.

<u>**Алгоритм Крускала**</u>

Введем обозначения:

E_k – множество ребер сети, которые входят в минимальное остовное дерево, после выполнения k-й итерации,

$\overline{E_k}$ – множество ребер сети, которые не входят в минимальное остовное дерево, после выполнения k-й итерации,

Шаг 0. Полагаем $E_0=\emptyset$, $\overline{E_0}$ = E, k = 1.

…

Основной шаг k.

В множестве $\overline{E_{k-1}}$ выбираем ребро (u,v), которое имеет минимальный вес. Если добавление этого ребра к множеству E_{k-1} не вызовет появления цикла, то ребро (u,v) присоединяется к множеству E_{k-1} и удаляется из множества $\overline{E_{k-1}}$. Если же добавление ребра (u,v) вызывает появление цикла, то такое ребра просто удаляется из множества $\overline{E_{k-1}}$. Таким образом, получаем новые множества E_k и $\overline{E_k}$.

Если множество $\overline{E_k}$ пусто, то выполнение алгоритма заканчивается. В противной случае полагаем k=k+1 и повторяем последний шаг.

<u>Алгоритм Прима</u>

Введем обозначения:

V_k – множество узлов сети, соединенных алгоритмом после выполнения k-й итерации алгоритма,

$\overline{V_k}$ – множество узлов сети, не соединенных с узлами множества V_k после выполнения k-й итерации алгоритма.

Шаг 0. Полагаем $V_0 = \varnothing$, $\overline{V_0} = V$, k = 1.

Шаг 1. Выбираем любой узел i из множества и $\overline{V_0}$ определяем $V_1 = \{i\}$, тогда $\overline{V_1} = V - \{i\}$. Полагаем k = 2.

…

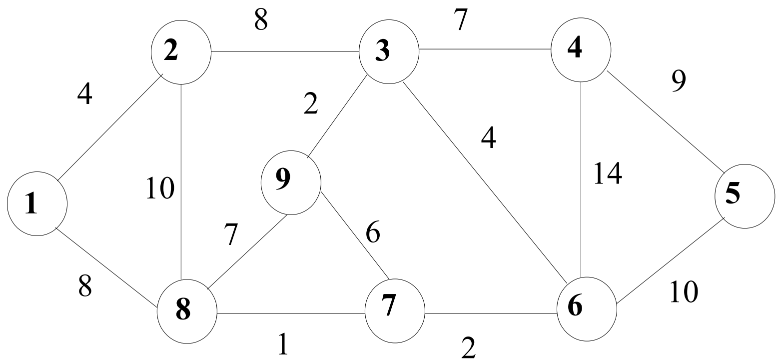
Основной шаг k.

В множестве $\overline{V_{k-1}}$ выбираем узел j^*, который соединен самой короткой дугой с каким-либо узлом из множества V_{k-1} . Узел j^* присоединяется к множеству V_{k-1} и удаляется из множества $\overline{V_{k-1}}$. Таким образом, V_k = V_{k-1} +{ j^*}, $\overline{V_k}$ = $\overline{V_{k-1}}$ -{ j^*}.

Если множество $\overline{V_k}$ пусто, то выполнение алгоритма заканчивается. В противной случае полагаем k=k+1 и повторяем последний шаг.

Пример 1.

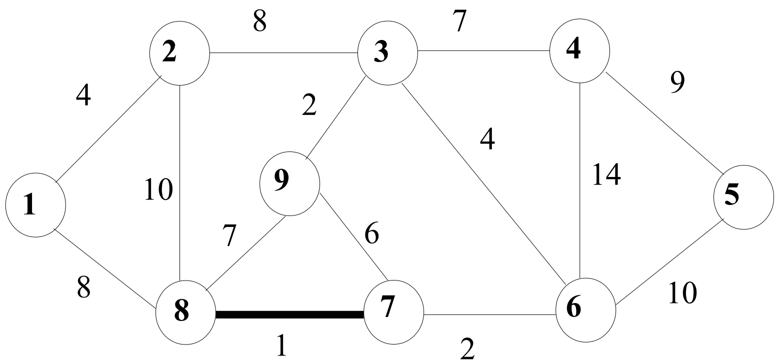
Телевизионная компания планирует подключение к своей кабельной сети семи новых районов. На рисунке показана структура планируемой сети и расстояния (в километрах) между районами и телецентром. Необходимо спланировать наиболее экономичную кабельную сеть.



Решение задачи при помощи алгоритма Крускала.

Шаг 0.

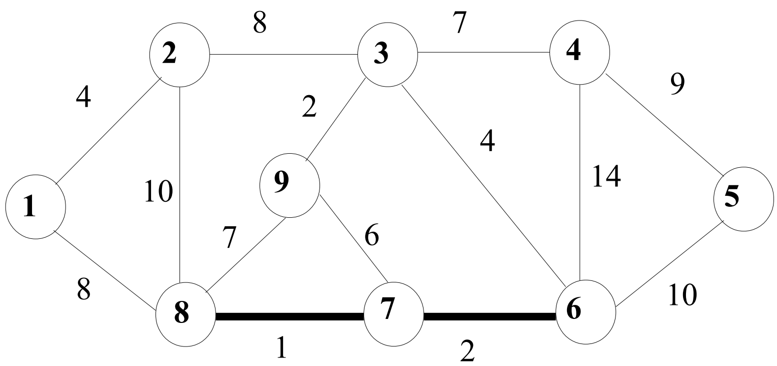
$E_0 = \varnothing$, $\overline{E_0}$ = {(1,2),(1,8),(2,3),(2,8),(3,4),(3,6),(3,9),(4,5),(4,6),(5,6),(6,7),(7,8),(7,9),(8,9)}



Шаг 1.

E_1 = {(7,8)}

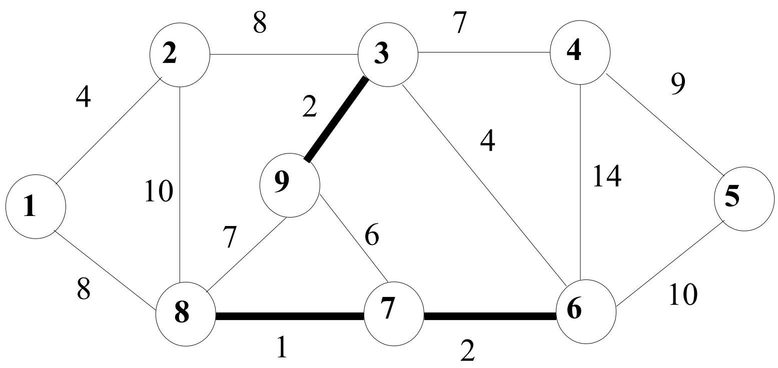
$\overline{E_1}$ = {(1,2),(1,8),(2,3),(2,8),(3,4),(3,6),(3,9),(4,5),(4,6),(5,6),(6,7),(7,9),(8,9)}



Шаг 2.

E_2 = {(7,8),(6,7)}

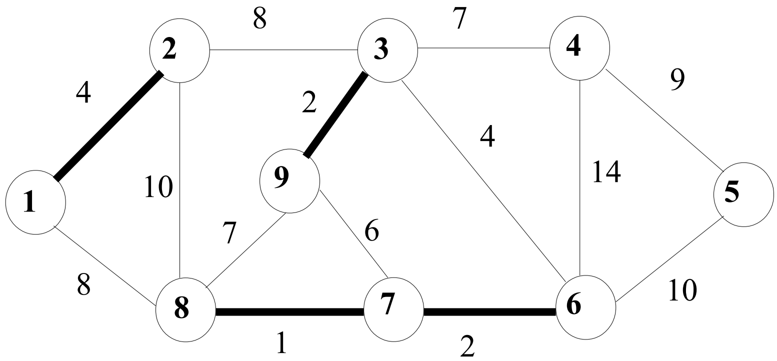
$\overline{E_2}$ = {(1,2),(1,8),(2,3),(2,8),(3,4),(3,6),(3,9),(4,5),(4,6),(5,6),(7,9),(8,9)}



<u>Шаг 3.</u>

$E_3 = \{(7,8),(6,7),(3,9)\}$

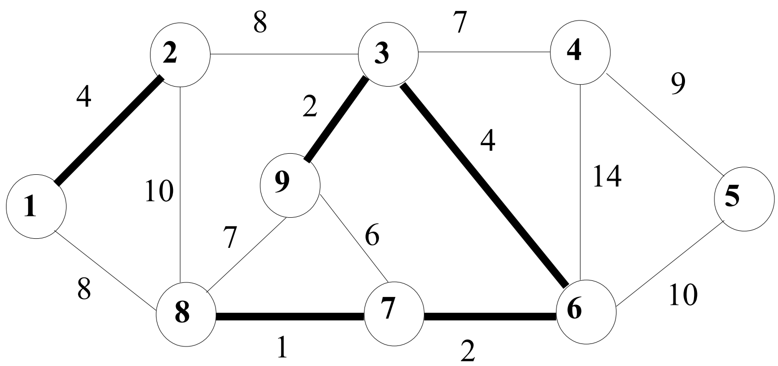
$\overline{E}_3 = \{(1,2),(1,8),(2,3),(2,8),(3,4),(3,6),(4,5),(4,6),(5,6),(7,9),(8,9)\}$



<u>Шаг 4.</u>

$E_4 = \{(7,8),(6,7),(3,9),(1,2)\}$

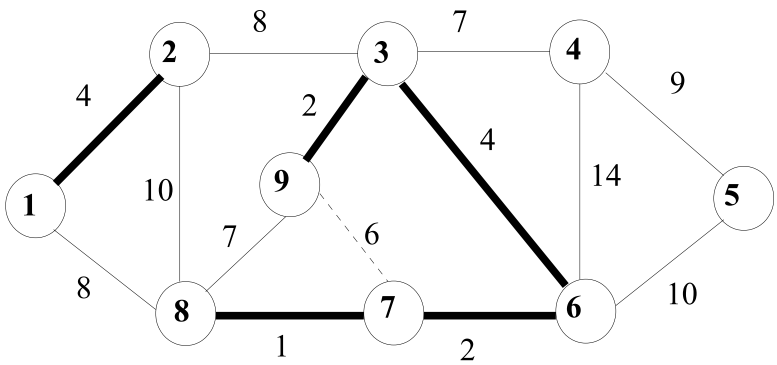
$\overline{E}_4 = \{(1,8),(2,3),(2,8),(3,4),(3,6),(4,5),(4,6),(5,6),(7,9),(8,9)\}$



<u>Шаг 5.</u>

$E_5 = \{(7,8),(6,7),(3,9),(1,2),(3,6)\}$

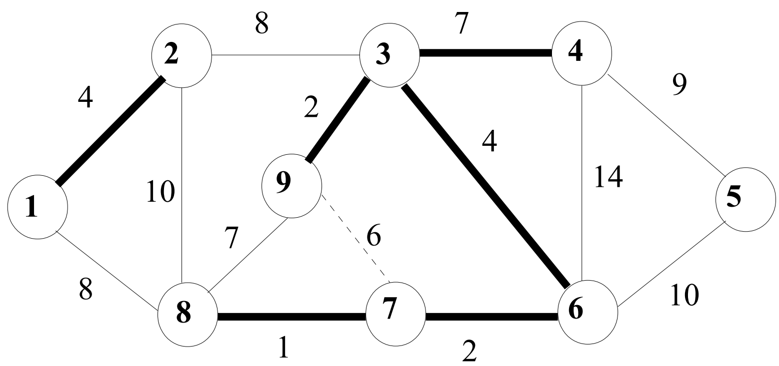
$\overline{E}_5 = \{(1,8),(2,3),(2,8),(3,4),(4,5),(4,6),(5,6),(7,9),(8,9)\}$



<u>Шаг 6.</u>

$E_6 = \{(7,8),(6,7),(3,9),(1,2),(3,6)\}$

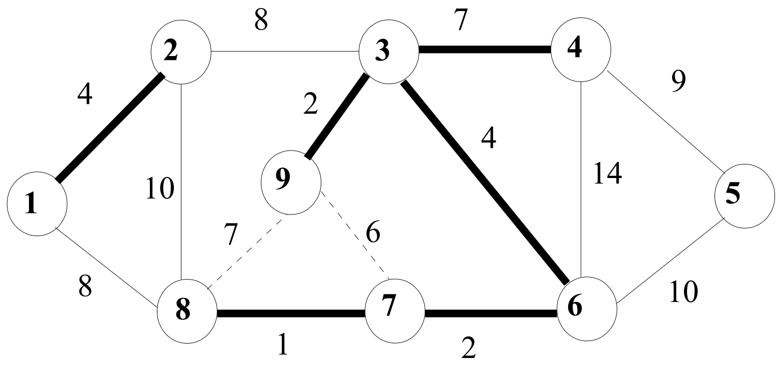
$\overline{E}_6 = \{(1,8),(2,3),(2,8),(3,4),(4,5),(4,6),(5,6),(8,9)\}$



<u>Шаг 7.</u>

$E_7 = \{(7,8),(6,7),(3,9),(1,2),(3,6),(3,4)\}$

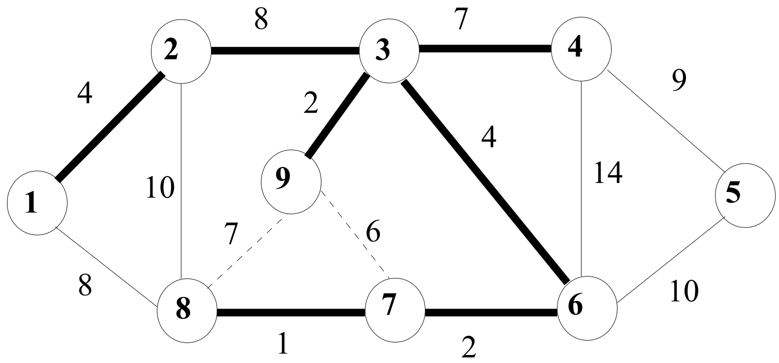
$\overline{E}_7 = \{(1,8),(2,3),(2,8),(4,5),(4,6),(5,6),(8,9)\}$



Шаг 8.

$E_8 = \{(7,8),(6,7),(3,9),(1,2),(3,6),(3,4)\}$

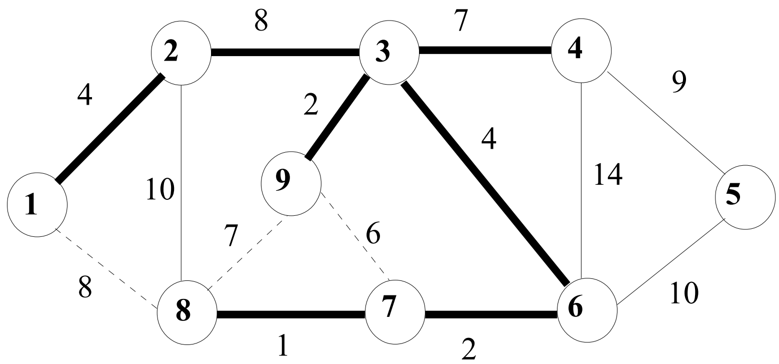
$\overline{E}_8 = \{(1,8),(2,3),(2,8),(4,5),(4,6),(5,6)\}$



Шаг 9.

$E_9 = \{(7,8),(6,7),(3,9),(1,2),(3,6),(3,4),(2,3)\}$

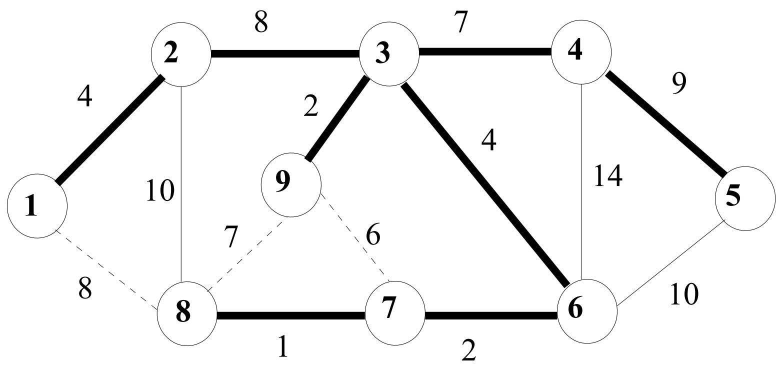
$\overline{E}_9 = \{(1,8),(2,8),(4,5),(4,6),(5,6)\}$



Шаг 10.

$E_{10} = \{(7,8),(6,7),(3,9),(1,2),(3,6),(3,4),(2,3)\}$

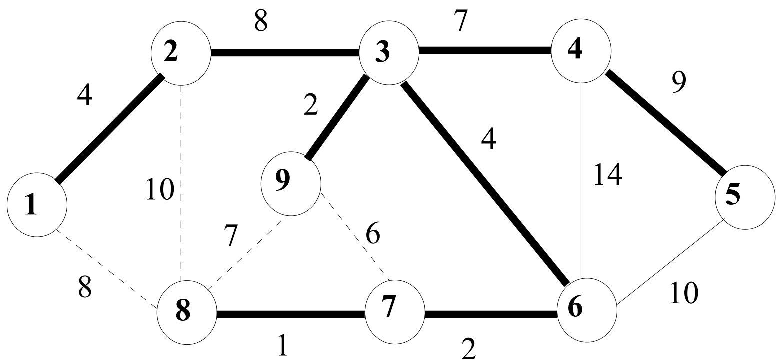
$\overline{E}_{10} = \{(2,8),(4,5),(4,6),(5,6)\}$



Шаг 11.

$E_{11} = \{(7,8),(6,7),(3,9),(1,2),(3,6),(3,4),(2,3),(4,5)\}$

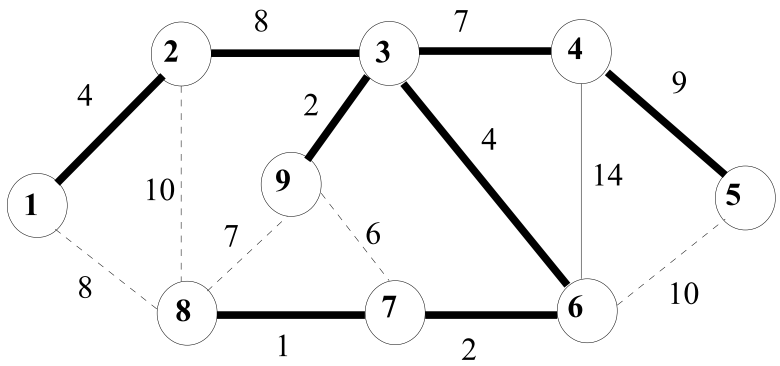
$\overline{E}_{11} = \{(2,8),(4,6),(5,6)\}$



Шаг 12.

$E_{12} = \{(7,8),(6,7),(3,9),(1,2),(3,6),(3,4),(2,3),(4,5)\}$

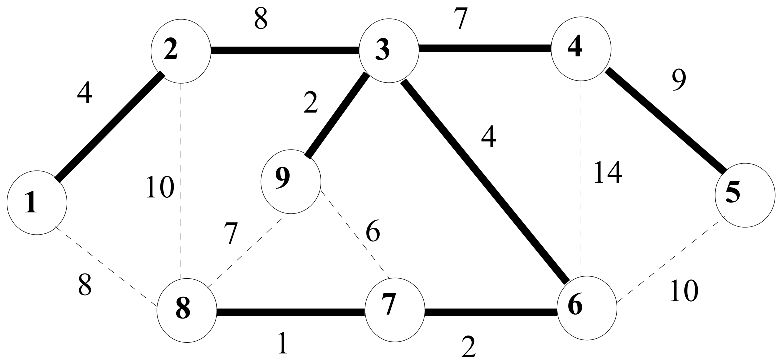
$\overline{E}_{12} = \{(4,6),(5,6)\}$



Шаг 13.

$E_{13} = \{(7,8),(6,7),(3,9),(1,2),(3,6),(3,4),(2,3),(4,5)\}$

$\overline{E_{13}} = \{(4,6)\}$



Шаг 14.

$E_{14} = \{(7,8),(6,7),(3,9),(1,2),(3,6),(3,4),(2,3),(4,5)\}$

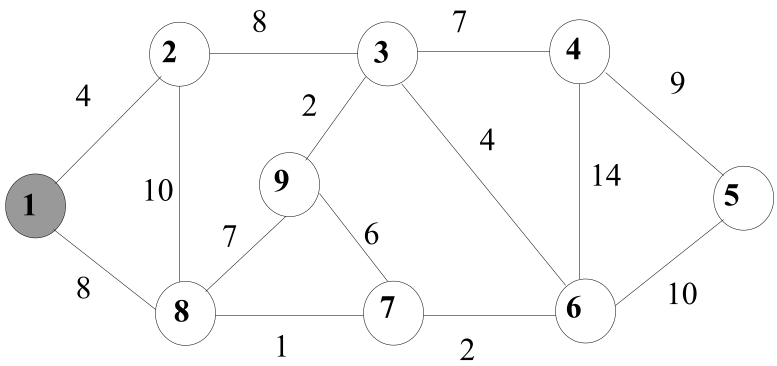
$\overline{E_{14}} = \varnothing$

Решение задачи при помощи алгоритма Прима.

Шаг 0.

$V_0 = \varnothing$,

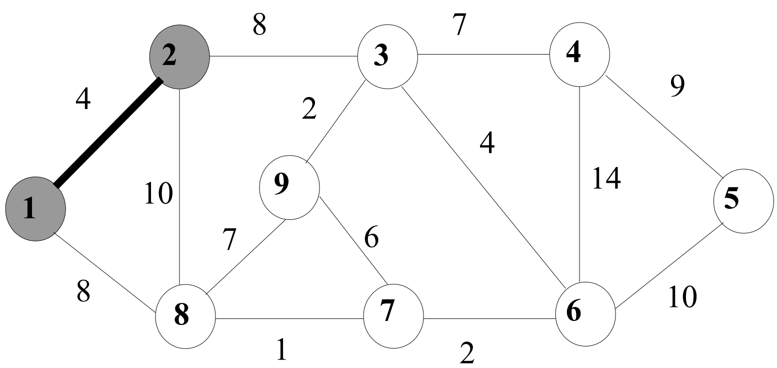
$\overline{V_0} = \{1,2,3,4,5,6,7,8,9\}$



Шаг 1.

$V_1 = \{1\}$

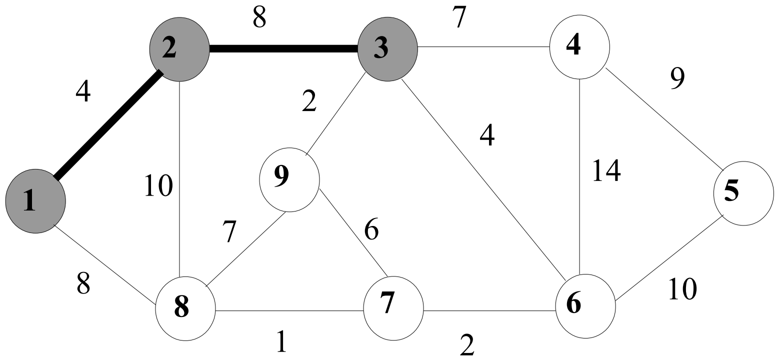
$\overline{V_1} = \{2,3,4,5,6,7,8,9\}$



Шаг 2.

$V_2 = \{1,2\}$

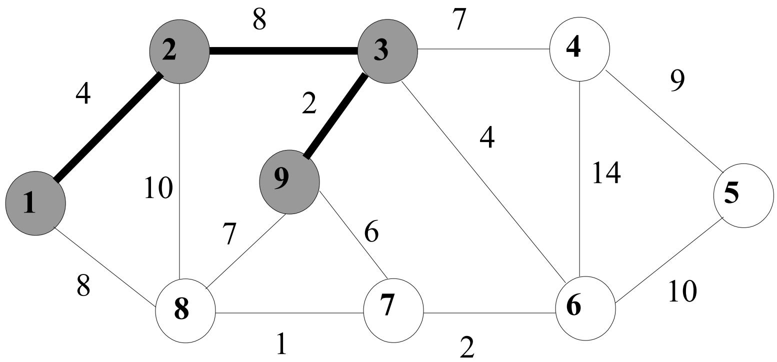
$\overline{V_2} = \{3,4,5,6,7,8,9\}$



Шаг 3.

$V_3 = \{1,2,3\}$

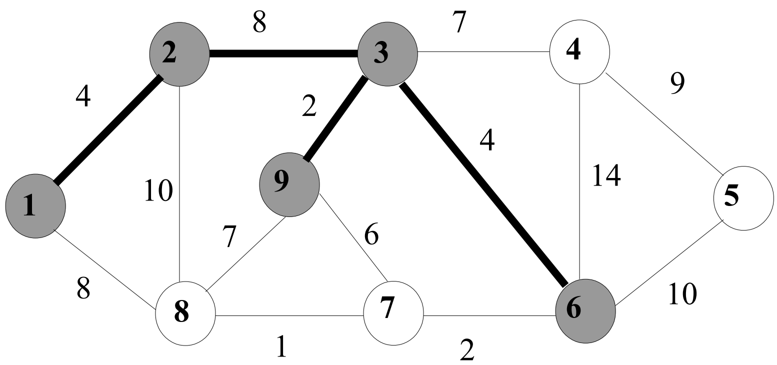
$\overline{V}_3 = \{4,5,6,7,8,9\}$



Шаг 4.

$V_4 = \{1,2,3,9\}$

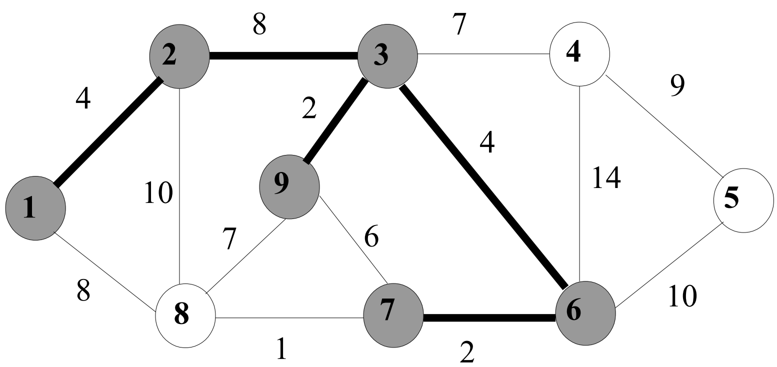
$\overline{V}_4 = \{4,5,6,7,8\}$



Шаг 5.

$V_5 = \{1,2,3,9,6\}$

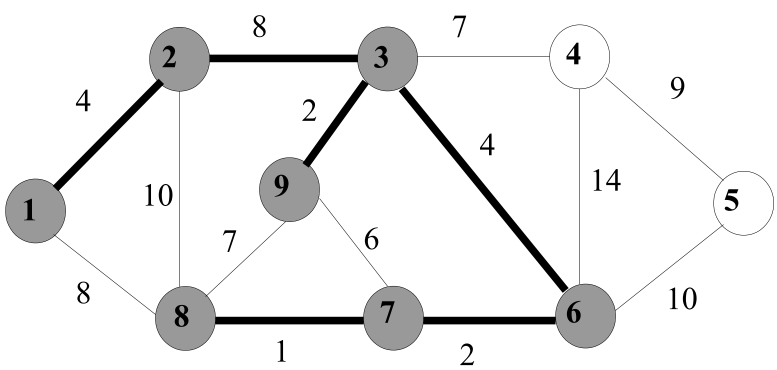
$\overline{V}_5 = \{4,5,7,8\}$



Шаг 6.

$V_6 = \{1,2,3,9,6,7\}$

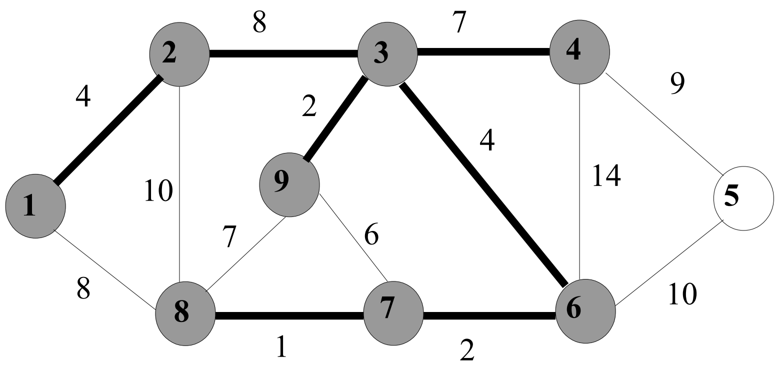
$\overline{V}_6 = \{4,5,8\}$



Шаг 7.

$V_7 = \{1,2,3,9,6,7,8\}$

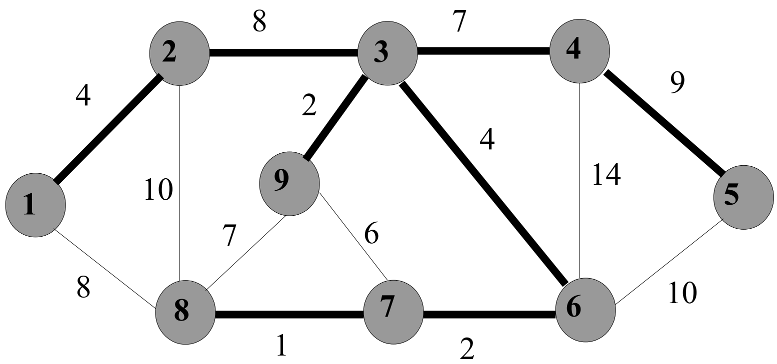
$\overline{V}_7 = \{4,5\}$



Шаг 8.

V_8 = {1,2,3,9,6,7,8,4}

$\overline{V_8}$ = {5}



Шаг 9.

V_9 = {1,2,3,9,6,7,8,4,5}

$\overline{V_9} = \varnothing$

Задача о максимальном потоке в сети

Ориентированную сеть можно интерпретировать как некоторую транспортную сеть и использовать ее для решения задач о потоках вещества в системе трубопроводов. Представим, что некоторый продукт передается по системе от источника, где данный продукт производится, к стоку, где он потребляется. Источник производит продукт с некоторой максимальной скоростью, а сток с той же скоростью потребляет продукт. Потоком продукта в любой точке системы является скорость движения продукта.

С помощью транспортных сетей можно моделировать течение жидкостей по трубопроводам, движение деталей на сборочных линиях, передачу тока по электрическим сетям, информации — по информационным сетям и т. д. Каждое ориентированное ребро сети можно рассматривать как канал, по которому движется продукт. Каждый канал имеет заданную пропускную способность, которая характеризует максимальную скорость перемещения продукта по каналу. Вершины являются точками пересечения каналов. Через вершины, отличные от источника и стока, продукт проходит не накапливаясь.

В задаче о максимальном потоке мы хотим найти максимальную скорость пересылки продукта от источника к стоку, при которой не будут нарушаться ограничения пропускной способности.

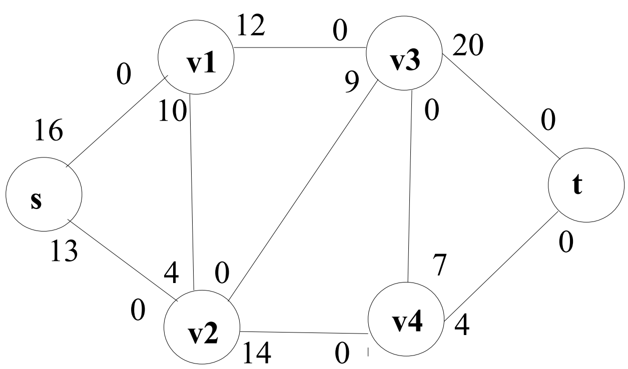
Транспортная сеть (flow network) $G=(V,E,)$ представляет собой ориентированную сеть, в которой каждое ребро $(u,v)\in E$ имеет неотрицательную *пропускную способность (capacity)* $c(u,v)>0$. Если $(u,v)\notin E$, предполагается, что $c(u,v)=0$. В транспортной сети выделяются две вершины: *источник (sourse)* s и *сток (sink)* t. Предполагается, что каждая вершина лежит на неком пути из источника стоку, т. е. Для любой вершины $v\in V$ существует путь $s\sim v\sim t$. Таким образом сеть является связной и $|E|>|V|-1$.

Пример 1.

С помощью транспортной сети можно моделировать задачу о грузовых перевозках. Итак, у некоторой компании «Аврора» в Измаиле есть фабрика (источник s), производящая стулья, а в Одессе - склад (сток t), где эти стулья хранятся. Компания арендует место на грузовиках других фирм для доставки стульев с фабрики на склад. Поскольку грузовики ездят по определенным маршрутам (ребрам) между городами (вершинами) и имеют ограниченную грузоподъемность, компания может перевозить не более $c(u,v)$ ящиков в день между каждой парой городов u и v.

Компания не может повлиять на маршруты и пропускную способность, то есть она не может менять транспортную сеть. Ее задача определить, какое наибольшее количество p ящиков в день можно отгружать, а затем производить именно такое количество, поскольку не имеет смысла производить больше стульев, чем можно отправить на склад.

Транспортная сеть для задачи о грузоперевозках компании «Аврора»



Товары доставляются через промежуточные города v1,v2,v3,v4, но из города u в город v можно перевезти не более, чем c(u,v) ящиков товара. Например, из города s в город v1 можно перевезти не более, чем c(s,v1)=16 ящиков, а из города v2 в город v1 - не более, чем c(v2,v1)=4 ящика.

Потоком (flow) в сети G является действительная функция $f: V \times V \rightarrow R$, удовлетворяющая трем условиям:

1) *ограничение пропускной способности (capacity constraint)*

$\forall u, v \in V \; f(u,v) \leq c(u,v)$

Поток из одной вершины в другую не должен превышать заданную пропускную способность.

2) *антисимметричность (skew symmetry)*

$$f(u,v)=-f(u,v)\,\forall\, u,v\in V$$

Поток из вершины u в вершину v противоположен потоку в обратной направлении.

3) *сохранение потока (flow conservation)*

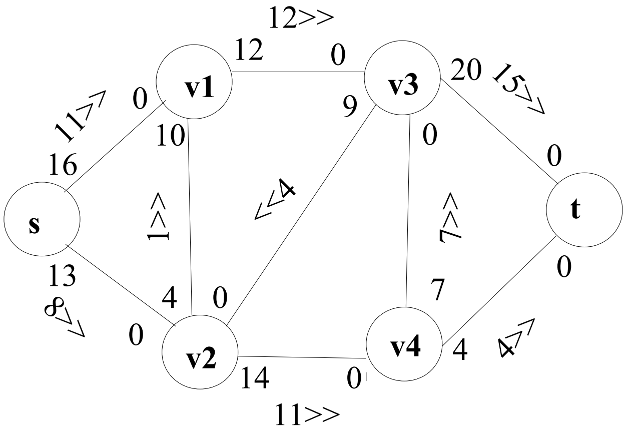
$$\forall\, u\in V/s,t\ \sum_{v\in V} f(u,v)=0$$

Суммарный поток, выходящий из вершины, не являющейся источником или стоком, равен нулю. *Величина потока (flow value)* определяется как суммарный поток, выходящий из источника

$$|f|=\sum_{v\in V} f(s,v)$$

Пример 2. Поток f в сети G величины 19.

Если f(u,v)>0, ребро (u,v) снабжается меткой f(u,v)>> (знак «>>» используется чтобы отделить направление потока).



Для работы с потоками введем неявное обозначение суммирования

$$f(X,Y)=\sum_{x\in X}\sum_{y\in Y} f(x,y)$$ X,Y – множества вершин

Лемма 1. Пусть G=(E,V) – транспортная сеть, а f — некоторый поток в сети. Тогда справедливы следующие неравенства:

1) $\forall\, X\subseteq V, f(X,X)=0;$

2) $\forall\, X,Y\subseteq V, f(X,Y)=-f(Y,X);$

3) $\forall X, Y, Z \subseteq V$ таких что, $X \cap Y = \varnothing$,

$$f(X \cap Y, Z) = f(X, Z) + f(Y, Z) \quad , \quad f(Z, X \cap Y) = f(Z, X) + f(Z, Y).$$

В *задаче максимальном потоке (minitimum flow problem)* дана некоторая транспортная сеть с источником s и стоком t, и необходимо найти поток максимальной величины.

Метод Форда-Фалкерсона

Это именно метод, а не алгоритм, поскольку он допускает несколько реализаций с различным временем выполнения. Метод Форда-Фалкерсона базируется на трех важных концепциях. Это — остаточные сети, увеличивающие пути и разрезы.

Метод Форда-Фалкерсона является итеративным. Вначале величине потока присваивается значение ноль. ($f(u, v) = 0 \, \forall u, v \in V$) На каждой итерации величина потока увеличивается посредством поиска «увеличивающего пути» (т. е. Некоторого пути от источника s к стоку t, вдоль которого можно послать больший поток) и последующего увеличения потока. Этот процесс повторяется до тех пор, пока уже невозможно отыскать увеличивающий путь.

Остаточные сети

Пусть задана транспортная сеть G(V,E) с источником s и стоком t. Пусть f – некоторый поток в G. Рассмотрим пару вершин $u, v \in V$. Величина дополнительного потока, который мы можем направить из u в v, не превысив пропускную способность c(u,v), является *остаточной пропускной способностью (residual capacity)* ребра (u,v) и задается формулой:

$$c_f(u, v) = c(u, v) - f(u, v)$$

Для заданной транспортной сети G(V,E) и потока f, *остаточной сетью (residual network)* в G, порожденной потоком f, является сеть $G_f = (V, E_f)$, где

$$E_f = (u, v) \in V \times V : c_f(u, v) > 0.$$

Таким образом, по каждому ребру остаточной сети или *остаточному ребру (residual edge)* можно направить поток больший нуля.

Лемма 2. Пусть G=(V,E) – транспортная сеть с источником s и стоком t, а f — поток в G. Пусть G_f - остаточная сеть в G, порожденная потоком f, а f' – поток в G_f. Тогда сумма потоков f+f', определяемая как

$$(f+f')(u,v)=f(u,v)+f'(u,v),$$

является потоком в G и величина этого потока равна

$$|f+f'|=|f|+|f'|$$

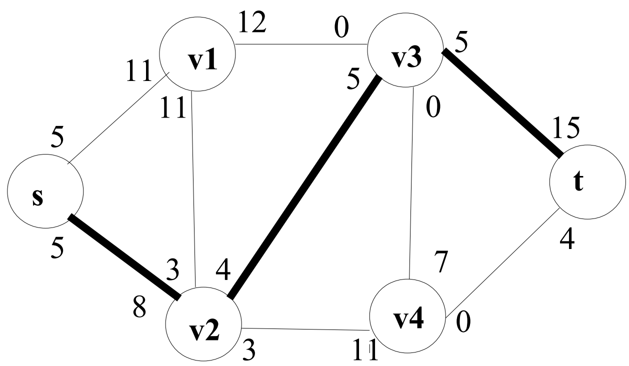
Увеличивающие пути

Для заданной транспортной сети G=(V,E) и потока f *увеличивающим путем (augmenting path)* p является простой путь из s в t в остаточной сети G_f. Максимальная величина, на которую можно увеличить поток вдоль каждого ребра увеличивающего пути p, *называется остаточной пропускной способностью пути p (residual capacity of p)* и задается формулой

$$c_f(p)=\min\{c(u,v):(u,v)\in p\}.$$

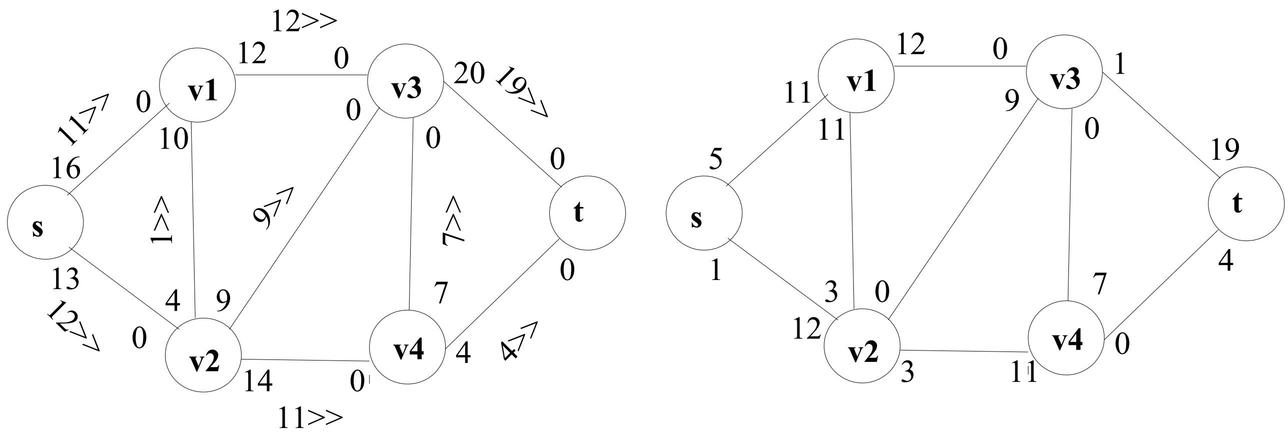
Пример 3.

Остаточная сеть G_f для сети G, представляющую транспортную сеть компании «Аврора», порожденная потоком f величины 19, с выделенным увеличивающим путем p.



$$p=\{s,v2,v3,t\},c_f(p)=\min\{c(s,v2),c(v2,v3),c(v3,t)\}=\min\{5,4,5\}=4$$

Поток в сети, полученный в результате увеличения потока вдоль пути p на величину его остаточной пропускной способности, и остаточная сеть.



Лемма 3. Пусть G=(V,E) – транспортная сеть, а f – некоторый поток в G и пусть p – некоторый увеличивающий путь в G_f. Определим функцию $f_p(u,v): V \times V \to R$ следующим образом:

$$f_p(u,v) = \begin{Bmatrix} c_f(p), \text{если}\,(u,v) \in p \\ -c_f(p), \text{если}\,(v,u) \in p \\ 0, \text{в противном случае} \end{Bmatrix}$$

Тогда f_p является потоком в G и его величина составляет $|f_p| = c_f(p) > 0$.

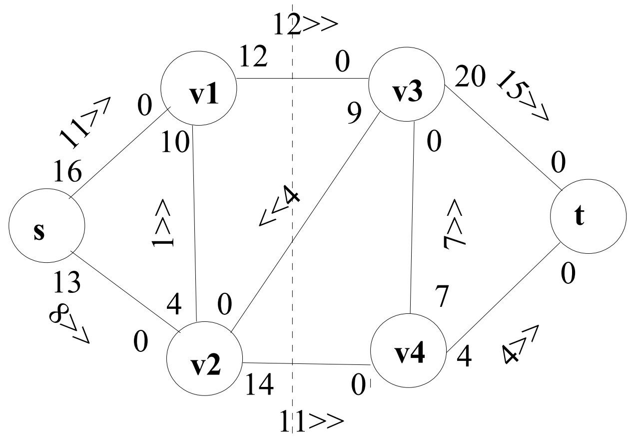
Следствие. Пусть G=(V,E) – транспортная сеть, а f — некоторый поток в G, и пусть p — некоторый увеличивающий путь в G_f. Пусть функция f_p определена как в лемме. Определим $f': V \times V \to R$ как $f' = f + f_p$. Тогда f' является потоком в G и имеет величину $|f'| = |f| + |f_p| > |f|$.

<u>Разрезы транспортных сетей</u>

Разрезом (cut) (S,T) транспортной сети G=(V,E) называется разбиение множества вершин на множества S и T=V-S такие, что $s \in S$, а $t \in T$.

Если f — поток, то чистый поток через разрез (S,T) обозначим как f(S,T). Пропускную способность разреза (S,T) обозначим, соответственно, c(S,T).

Пример 4.



Чистый поток через разрез:

f(S,T) =

$= \ f(v_{1,} v_3) + f(v_{2,} v_3) + f(v_{2,} v_4) \ =$

$= \ 12 - 4 + 11 = 19$

Пропускная способность разреза:

$c(S,T) = c(v_{1,} v_3) + c(v_{2,} v_4) \ =$

$= \ 12 + 14 = 26$

Минимальным разрезом является разрез, пропускная способность которого среди всех разрезов минимальна. Как видно, поток через разрез, в отличие от пропускной способности разреза, может включать и отрицательные слагаемые.

Лемма 4. Пусть f – поток в сети G с истоком s и стоком t, а (S,T) — разрез сети G. Тогда поток F(S,T) через разрез (S,T) равен $|f|$.

Следствие. Величина любого потока f в сети G не превосходит пропускной способности любого разреза сети G.

Теорема 1 (О максимальном потоке и минимальном разрезе). Пусть f – поток в сети G=(V,E). Тогда следующие утверждения равносильны:

1) поток f – максимален в сети G (является потоком максимальной величины);
2) остаточная сеть G_f не содержит увеличивающих путей;
3) для некоторого разреза (S,T) сети G выполнено равенство $|f| = c(S,T)$.

Базовый алгоритм Форда-Фалкерсона

Шаг 0. Пусть дана транспортная сеть G=(V,E), где V — множество вершин, а E — множество ребер. Обнуляем все потоки: положим величину потока в сети равна нулю $|f|=0$ и потоки по все ребрам также положим равными нулю $\forall u,v \in V f(u,v)=0$. Остаточная сеть изначально совпадает с исходной сетью $G_f=G$.

Шаг 1. В остаточной сети G_f находим любой путь из источника в сток или увеличивающий путь p. Если такого пути нет, то переходим к шагу 4, иначе переходим к шагу 2.

Шаг 2. Пускаем через найденный увеличивающий путь максимально возможный поток, то есть ищем пропускную способность увеличивающего пути: ищем ребро с минимальной пропускной способностью $c_p=\min\{c(u,v),(u,v)\in p\}$. Увеличиваем величину потока в сети на найденную пропускную способность увеличивающего пути $|f|=|f|+c_p$.

Шаг 3. Модифицируем остаточную сеть G_f , вычисляя новую пропускную способность. Для всех рёбер (u,v) на найденном увеличивающем пути пропускную способность уменьшаем на c_p , а также для противоположных им рёбер увеличиваем на c_p . Переходим к шагу 1.

Шаг 4. Максимальный поток в сети равен последней найденной величине потока в сети. Потоки по ребрам для достижения общего максимального потока могут быть найдены путем сравнения конечных пропускных способностей ребер с начальными. Если конечная пропускная способность меньше начальной, то поток проходит по ребру с величиной равной разности между первоначальной и конечной пропускной способностью.

Алгоритм Эдмондса-Карпа

Алгоритм Эдмондса — Карпа — это вариант алгоритма Форда — Фалкерсона, при котором на каждом шаге выбирают кратчайший увеличивающий путь из s в t в остаточной сети (полагая, что каждое ребро имеет единичную длину). Кратчайший путь находится поиском в ширину.

Алгоритм поиска в ширину

Шаг 0. Создаём очередь вершин О. Вначале очередь состоит из единственной вершины s.

Шаг 1. Отмечаем вершину s как посещённую, без предка, а все остальные как не посещённые.

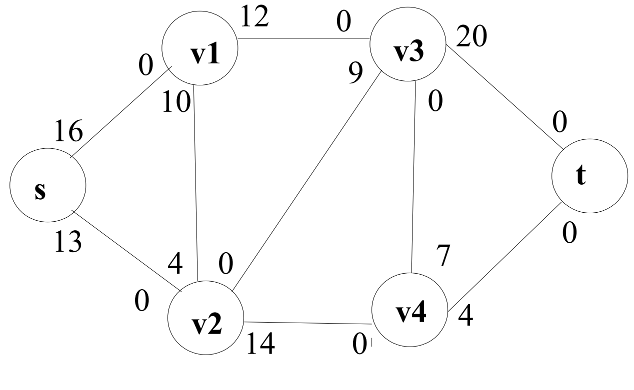
Шаг 2. Если очередь не пуста, переходим к шагу 3 иначе переходим к шагу 4.

Шаг 3. Удаляем первую в очереди вершину и обозначаем ее u. Для всех рёбер (u, v), исходящих из вершины u, таких что вершина v ещё не посещена, выполняем следующие шаги: отмечаем вершину v как посещённую, с предком u, добавляем вершину v в конец очереди. Если v=t, то переходим к шагу 5. Иначе переходим к шагу 2.

Шаг 4. Если очередь пуста, возвращаем ответ, что кратчайший путь не может быть. Работа алгоритма заканчивается.

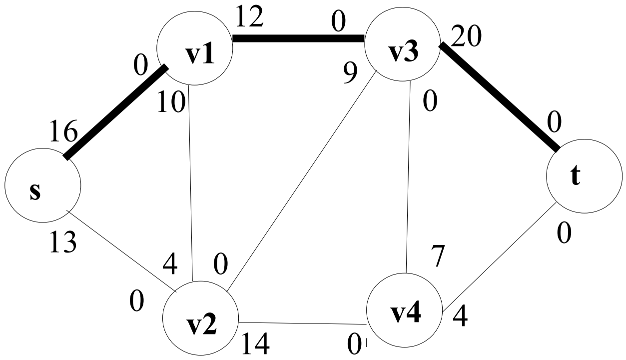
Шаг 5. Работа алгоритма заканчивается - кратчайший путь найден. Возвращаем путь в обратном порядке, идя от t к s, каждый раз переходя к предку.

Решение задачи при помощи алгоритма Форда-Фалкерсона.



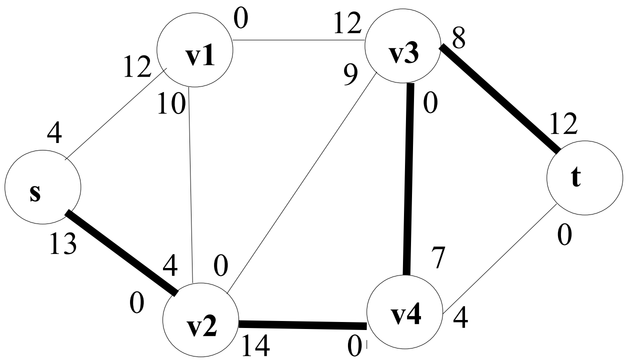
Шаг 0. Дана транспортная сеть G. Величина потока в сети равна нулю $|f|=0$. Остаточная сеть G_f, порожденная потоком в сети, совпадает с данной.

Остаточная сеть G_f :



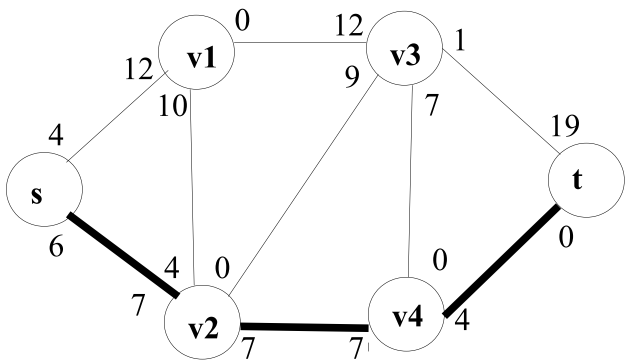
Шаг 1. Увеличивающий путь
$p = \{s, v1, v3, t\}$.
Шаг 2. Пропускная способность увеличивающего пути
$c_p = \min\{16, 12, 20\} = 12$.
Величина потока в сети
$|f| = 0 + 12 = 12$.
Шаг 3. Модифицируем остаточную сеть G_f и переходим к шагу 1.

Остаточная сеть G_f :



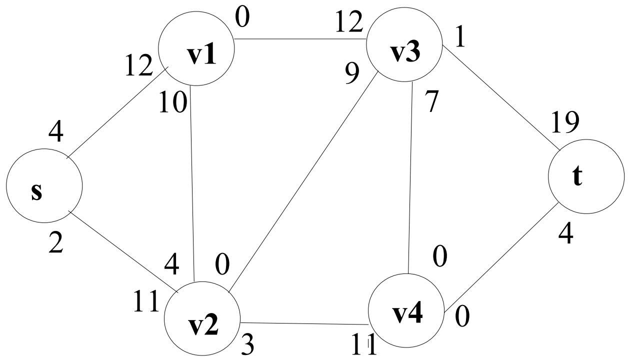
Шаг 1. Увеличивающий путь
$p = \{s, v2, v4, v3, t\}$.
Шаг 2. Пропускная способность увеличивающего пути
$c_p = \min\{13, 14, 7, 8\} = 7$.
Величина потока в сети
$|f| = 12 + 7 = 19$.
Шаг 3. Модифицируем остаточную сеть G_f и переходим к шагу 1.

Остаточная сеть G_f :



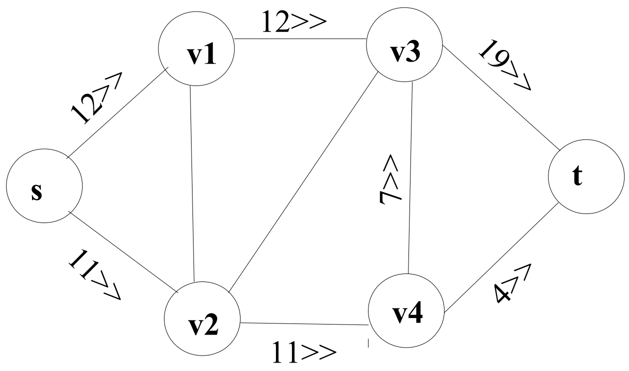
Шаг 1. Увеличивающий путь
$p = \{s, v2, v4, t\}$.
Шаг 2. Пропускная способность увеличивающего пути
$c_p = \min\{6, 7, 4\} = 4$.
Величина потока в сети
$|f| = 19 + 4 = 23$.
Шаг 3. Модифицируем остаточную сеть G_f и переходим к шагу 1.

Остаточная сеть G_f :

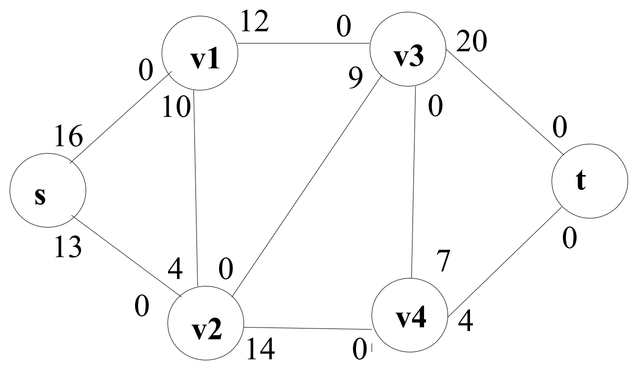


Шаг 1. Увеличивающий путь построить нельзя. Переходим к шагу 4.
Шаг 4. Максимальный поток в сети $|f|=23$. Вычисляем потоки по ребрам, которые обеспечивают максимальный поток в сети.

Потоки по ребрам:

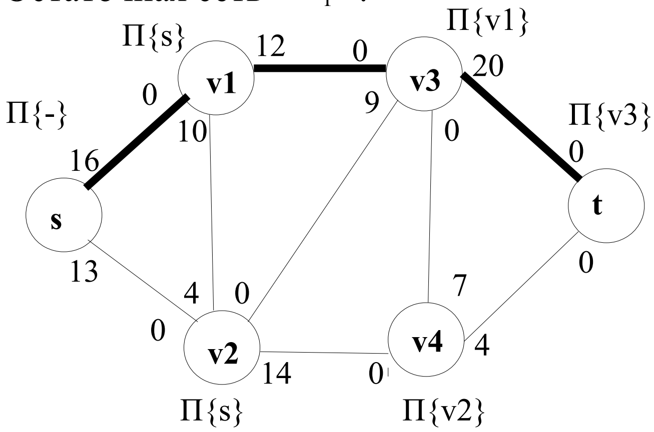


Решение задачи при помощи алгоритма Эдмондса-Карпа.



Шаг 0. Дана транспортная сеть G. Величина потока в сети равна нулю $|f|=0$. Остаточная сеть G_f , порожденная потоком в сети, совпадает с данной.

Остаточная сеть G_f :



Шаг 1. Строим увеличивающий путь при помощи алгоритма поиска в ширину: $p=\{s, v1, v3, t\}$.

Шаг 2. Пропускная способность увеличивающего пути $c_p=\min\{16,12,20\}=12$.

Величина потока в сети $|f|=0+12=12$.

Шаг 3. Модифицируем остаточную сеть G_f и переходим к шагу 1.

Поиск в ширину:

Шаг 0. O = {s}.

Шаг 1. s — посещенная вершина без предка (П{-}).

Шаг 2. O не пуста.

Шаг 3. u=s. O = {}.

v1,v2 – не посещенные вершины, связанные с u. v1 – посещенная вершина с предком s (П{s}). O = {v1}. v2 – посещенная вершина с предком s (П{s}). O = {v1,v2}.

Шаг 2. O не пуста.

Шаг 3. u=v1. O = {v2}.

v3 – не посещенная вершина, связанные с u. v3 – посещенная вершина с предком v1 (П{v1}). O = {v2,v3}.

u=v2. O = {v3}.

v4 – не посещенная вершина, связанные с u. v4 – посещенная вершина с предком v2 (П{v2}). O = {v3,v4}.

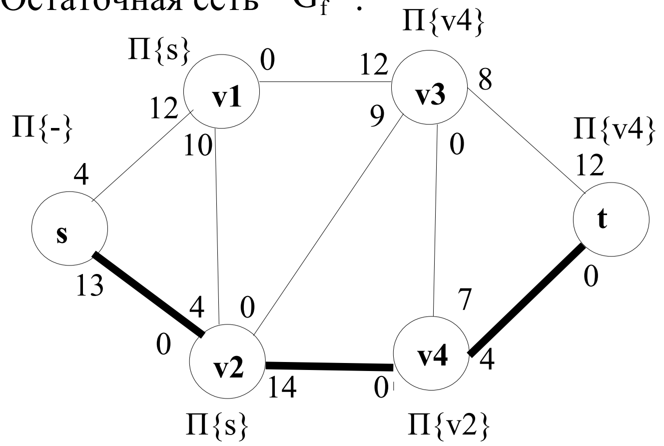
Шаг 2. O не пуста.

Шаг 3. u=v3. O = {v4}.

t – не посещенная вершина, связанные с u. t – посещенная вершина с предком v3 (П{v3}). O = {v4,t}.

Шаг 5. Найденный путь: {t,v3,v1,s}.

Остаточная сеть G_f :



Шаг 1. Строим увеличивающий путь при помощи алгоритма поиска в ширину: $p=\{s, v2, v4, t\}$.

Шаг 2. Пропускная способность увеличивающего пути $c_p=\min\{13,14,4\}=4$.

Величина потока в сети $|f|=12+4=16$.

Шаг 3. Модифицируем остаточную сеть G_f и переходим к шагу 1.

Поиск в ширину:

Шаг 0. O = {s}.

Шаг 1. s — посещенная вершина без предка (П{-}).

Шаг 2. O не пуста.

Шаг 3. u=s. O = {}.

v1,v2 – не посещенные вершины, связанные с u. v1 – посещенная вершина с предком s (П{s}). O = {v1}.

v2 – посещенная вершина с предком s (П{s}). O = {v1,v2}.

Шаг 2. O не пуста.

Шаг 3. u=v1. O = {v2}. Нет не посещенных связанных с u вершин.

u=v2. O = {}.

v4 – не посещенная вершина, связанная с u. v4 – посещенная вершина с предком v2 (П{v2}). O = {v4}.

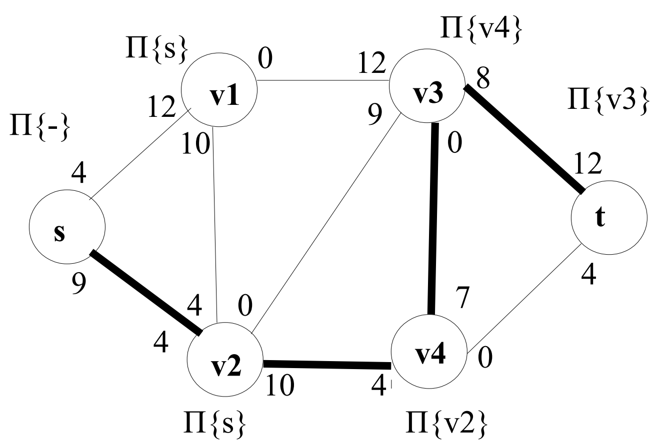
Шаг 2. O не пуста.

Шаг 3. u=v4. O = {}.

v3,t – не посещенные вершины, связанные с u. v3 – посещенная вершина с предком v4 (П{v4}). O = {v3}. t – посещенная вершина с предком v4 (П{v4}). O = {v3,t}.

Шаг 5. Найденный путь: {t,v4,v2,s}.

Остаточная сеть G_f :



Шаг 1. Строим увеличивающий путь при помощи алгоритма поиска в ширину: $p=\{s, v2, v4, v3, t\}$.

Шаг 2. Пропускная способность увеличивающего пути

Поиск в ширину:

Шаг 0. O = {s}.

Шаг 1. s — посещенная вершина без предка (П{-}).

Шаг 2. O не пуста.

Шаг 3. u=s. O = {}.

v1,v2 – не посещенные вершины, связанные с u. v1 – посещенная вершина с предком s (П{s}). O = {v1}.

v2 – посещенная вершина с предком s (П{s}). O = {v1,v2}.

Шаг 2. O не пуста.

Шаг 3. u=v1. O = {v2}. Нет не посещенных связанных с u вершин.

u=v2. O = {}.

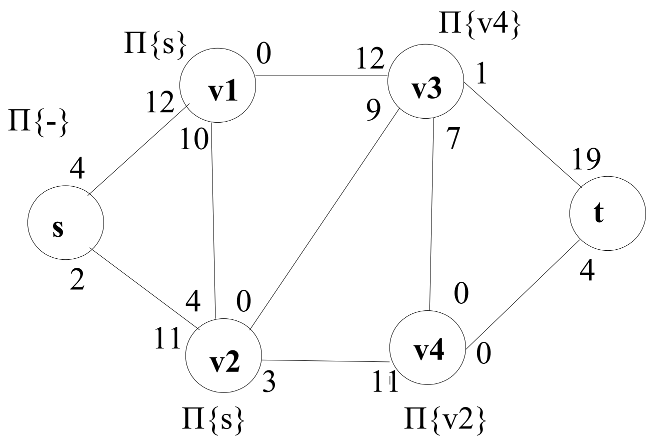
v4 – не посещенная вершина, связанная с u. v4 – посещенная верши-

$c_p = \min\{9,10,7,8\} = 7$.

Величина потока в сети $|f| = 16 + 7 = 23$.

<u>Шаг 3.</u> Модифицируем остаточную сеть G_f и переходим к шагу 1.

Остаточная сеть G_f :



<u>Шаг 1.</u> Строим увеличивающий путь при помощи алгоритма поиска в ширину: путь построить нельзя.

<u>Шаг 4.</u> Максимальный поток в сети $|f| = 23$. Вычисляем потоки по ребрам, которые обеспечивают максимальный поток в сети.

на с предком v2 (П{v2}). O = {v4}.

<u>Шаг 2.</u> O не пуста.

<u>Шаг 3.</u> u=v4. O = {}.

v3 – не посещенная вершина, связанная с u. v3 – посещенная вершина с предком v4 (П{v4}). O = {v3}.

<u>Шаг 2.</u> O не пуста.

<u>Шаг 3.</u> u=v3. O = {}.

t – не посещенная вершина, связанная с u. t – посещенная вершина с предком v3 (П{v3}). O = {t}.

<u>Шаг 5.</u> Найденный путь: {t,v3,v4,v2,s}.

<u>Поиск в ширину:</u>

<u>Шаг 0.</u> O = {s}.

<u>Шаг 1.</u> s — посещенная вершина без предка (П{-}).

<u>Шаг 2.</u> O не пуста.

<u>Шаг 3.</u> u=s. O = {}.

v1,v2 – не посещенные вершины, связанные с u. v1 – посещенная вершина с предком s (П{s}). O = {v1}. v2 – посещенная вершина с предком s (П{s}). O = {v1,v2}.

<u>Шаг 2.</u> O не пуста.

<u>Шаг 3.</u> u=v1. O = {v2}. Нет не посещенных связанных с u вершин.

u=v2. O = {}.

v4 – не посещенная вершина, связанная с u. v4 – посещенная вершина с предком v2 (П{v2}). O = {v4}.

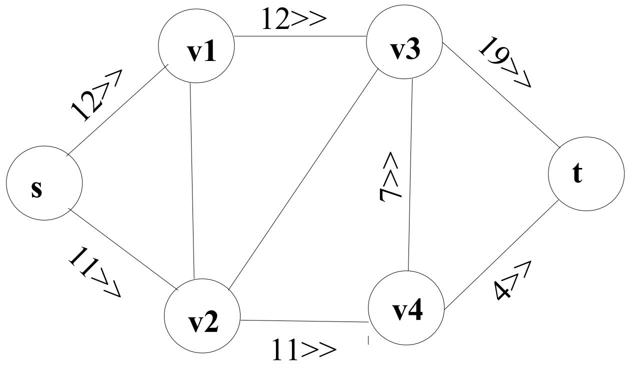
<u>Шаг 2.</u> O не пуста.

<u>Шаг 3.</u> u=v4. O = {}. Нет не посещенных связанных с u вершин.

<u>Шаг 2.</u> O пуста.

<u>Шаг 4.</u> Путь построить нельзя.

Потоки по ребрам:

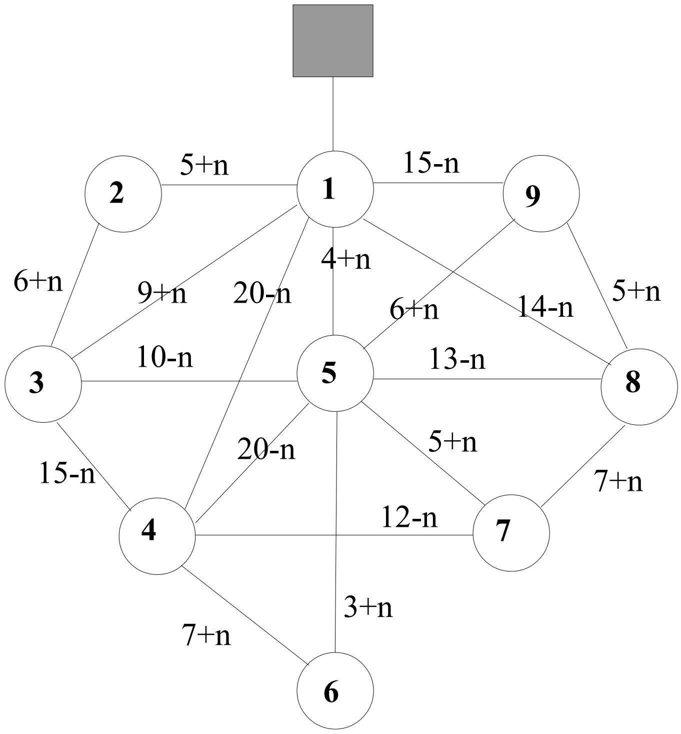


Задание для самостоятельной работы №1.
Построение минимального остовного дерева

Вариант	Задача	Алгоритм	n	Вариант	Задача	Алгоритм	n
1	1	Крускала	1	**26**	1	Прима	6
2	2	Прима	1	**27**	2	Крускала	6
3	3	Крускала	1	**28**	3	Прима	6
4	4	Прима	1	**29**	4	Крускала	6
5	5	Крускала	1	**30**	5	Прима	6
6	1	Прима	2	**31**	1	Крускала	7
7	2	Крускала	2	**32**	2	Прима	7
8	3	Прима	2	**33**	3	Крускала	7
9	4	Крускала	2	**34**	4	Прима	7
10	5	Прима	2	**35**	5	Крускала	7
11	1	Крускала	3	**36**	1	Прима	8
12	2	Прима	3	**37**	2	Крускала	8
13	3	Крускала	3	**38**	3	Прима	8
14	4	Прима	3	**39**	4	Крускала	8
15	5	Крускала	3	**40**	5	Прима	8
16	1	Прима	4	**41**	1	Крускала	9
17	2	Крускала	4	**42**	2	Прима	9
18	3	Прима	4	**43**	3	Крускала	9
19	4	Крускала	4	**44**	4	Прима	9
20	5	Прима	4	**45**	5	Крускала	0
21	1	Крускала	5	**46**	1	Прима	0
22	2	Прима	5	**47**	2	Крускала	0
23	3	Крускала	5	**48**	3	Прима	0
24	4	Прима	5	**49**	4	Крускала	0
25	5	Крускала	5	**50**	5	Прима	0

Задача №1.

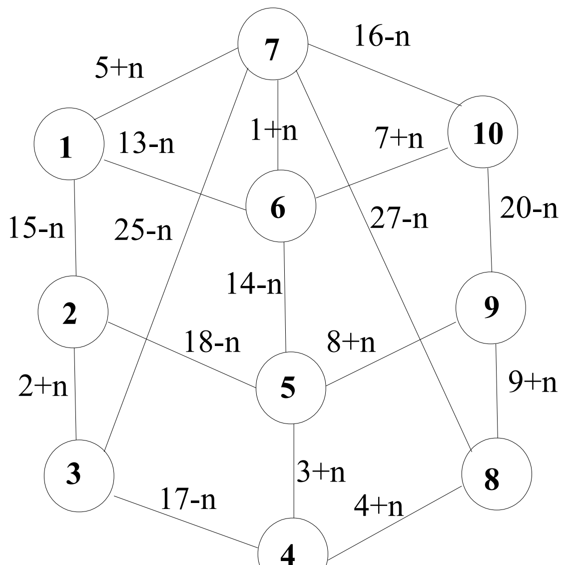
На рисунке показаны расстояния между платформами, добывающими газ в открытом море, и приемным пунктом, расположенным на берегу. Поскольку платформа 1 ближе остальных к берегу, она оснащена необходимым оборудованием для перекачки газа от остальных платформ к приемному пункту. Необходимо спроектировать сеть трубопроводов минимальной длины, соединяющую приемный пункт со всеми добывающими платформами.



Какова минимальная общая длина трубопроводов?

Задача №2.

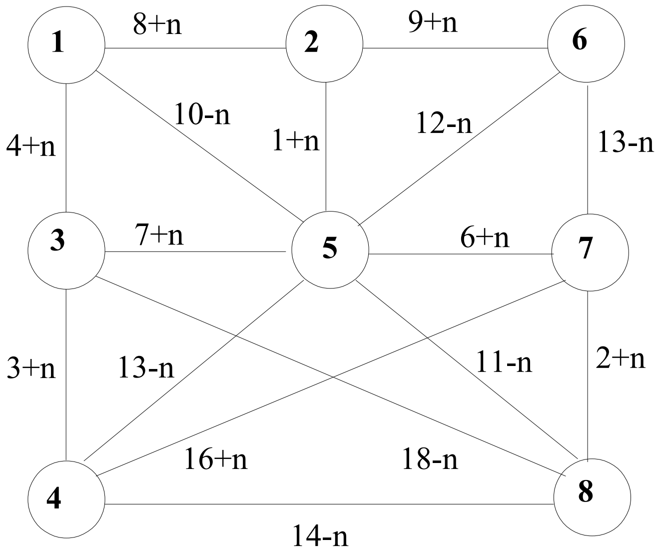
Постройте дерево кратчайших расстояний для коммуникационной сети службы пожарной безопасности.



Какова минимальная общая протяженность сети связи?

Задача №3.

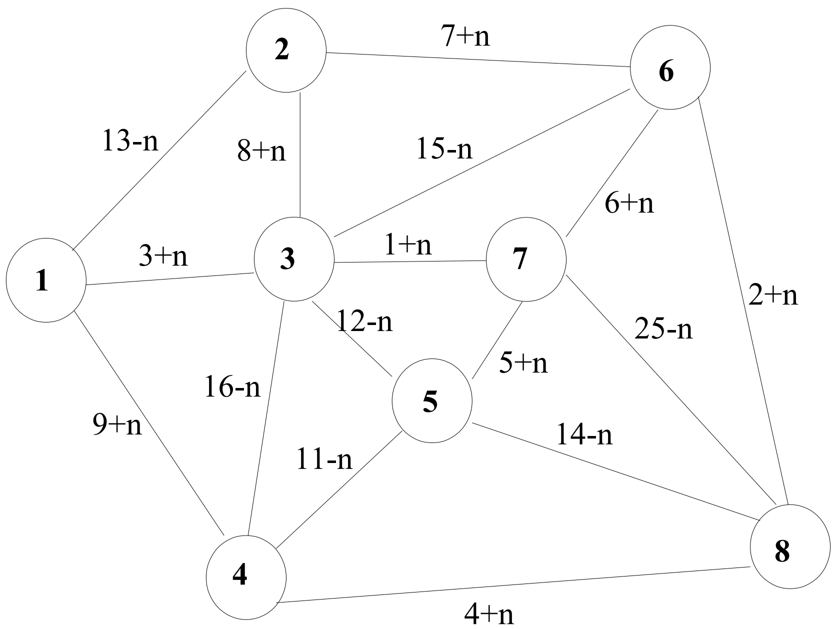
В городе возможно строительство нового парка. Определены места расположения павильонов, торговых палаток, пикников, лодочных станций, открытых сцен и т.д. Эти места представлены узлами на сети, приведенной ниже. Дуги сети отражают возможные альтернативы дорожек в парке. Необходимо минимизировать общую протяженность дорожек, которые требуется провести в парке и при этом обеспечить доступ ко всем узлам.



Какова минимальная общая протяженность дорожек?

Задача №4.

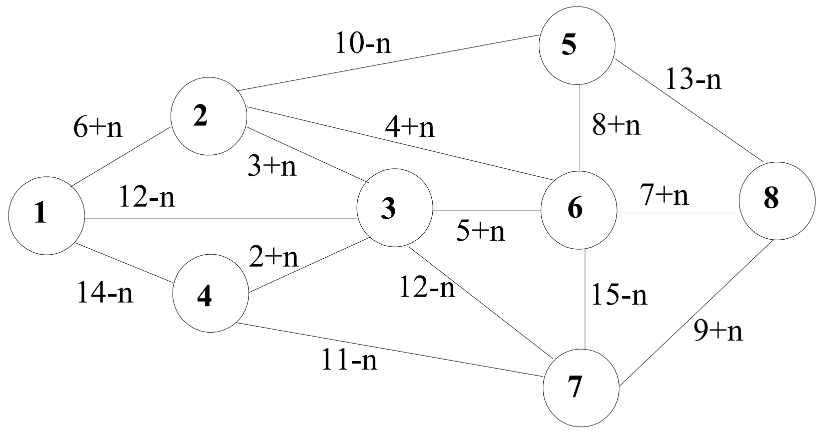
Разрабатывается электронная схема. Для соединения из восьми контактов можно использовать некоторую компоновку из 7 проводов, каждый из которых соединяет два контакта. Необходимо получить компоновку, которая использует минимальное количество провода.



Какова минимальная общая длина провода?

Задача №5.

На предприятии по производству мыла инспекторы осуществляют контроль качества на различных производственных участках, а затем возвращаются в лабораторию с образцами продукции для анализа. Процесс инспектирования слишком затянут, т.к. инспекторы тратят значительное время на доставку образцов с производственных участков в лабораторию. Руководство рассматривает возможность установки пневматических труб, которые можно было бы использовать для доставки образцов в лабораторию. Ниже представленная сеть отражает расположение лаборатории (узел 1) и производственных участков (узлы 2..8), на которых берутся образцы. Дуги представляют собой альтернативы конвейерной системы. Расстояния на дугах сети показаны в сотнях метров. Какова минимальная по длине конвейерная система, которая позволит посылать образцы продукции со всех участков в лабораторию?



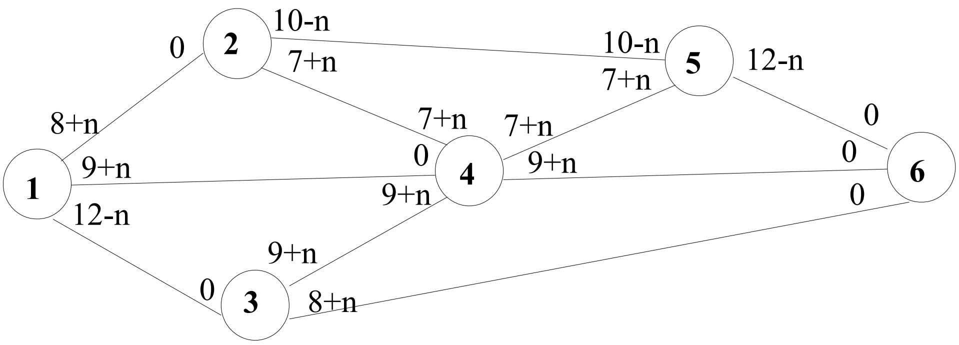
Какова минимальная общая протяженность конвейера?

Задание для самостоятельной работы №2
Вычисление максимального потока в сети.

Вари-ант	Зада-ча	Алгоритм	n	Вари-ант	Зада-ча	Алгоритм	n
1	1	Форда-Фалкерсона	1	**26**	1	Эдмондса-Карпа	6
2	2	Эдмондса-Карпа	1	**27**	2	Форда-Фалкерсона	6
3	3	Форда-Фалкерсона	1	**28**	3	Эдмондса-Карпа	6
4	4	Эдмондса-Карпа	1	**29**	4	Форда-Фалкерсона	6
5	5	Форда-Фалкерсона	1	**30**	5	Эдмондса-Карпа	6
6	1	Эдмондса-Карпа	2	**31**	1	Форда-Фалкерсона	7
7	2	Форда-Фалкерсона	2	**32**	2	Эдмондса-Карпа	7
8	3	Эдмондса-Карпа	2	**33**	3	Форда-Фалкерсона	7
9	4	Форда-Фалкерсона	2	**34**	4	Эдмондса-Карпа	7
10	5	Эдмондса-Карпа	2	**35**	5	Форда-Фалкерсона	7
11	1	Форда-Фалкерсона	3	**36**	1	Эдмондса-Карпа	8
12	2	Эдмондса-Карпа	3	**37**	2	Форда-Фалкерсона	8
13	3	Форда-Фалкерсона	3	**38**	3	Эдмондса-Карпа	8
14	4	Эдмондса-Карпа	3	**39**	4	Форда-Фалкерсона	8
15	5	Форда-Фалкерсона	3	**40**	5	Эдмондса-Карпа	8
16	1	Эдмондса-Карпа	4	**41**	1	Форда-Фалкерсона	9
17	2	Форда-Фалкерсона	4	**42**	2	Эдмондса-Карпа	9
18	3	Эдмондса-Карпа	4	**43**	3	Форда-Фалкерсона	9
19	4	Форда-Фалкерсона	4	**44**	4	Эдмондса-Карпа	9
20	5	Эдмондса-Карпа	4	**45**	5	Форда-Фалкерсона	0
21	1	Форда-Фалкерсона	5	**46**	1	Эдмондса-Карпа	0
22	2	Эдмондса-Карпа	5	**47**	2	Форда-Фалкерсона	0
23	3	Форда-Фалкерсона	5	**48**	3	Эдмондса-Карпа	0
24	4	Эдмондса-Карпа	5	**49**	4	Форда-Фалкерсона	0
25	5	Форда-Фалкерсона	5	**50**	5	Эдмондса-Карпа	0

Задача №1.

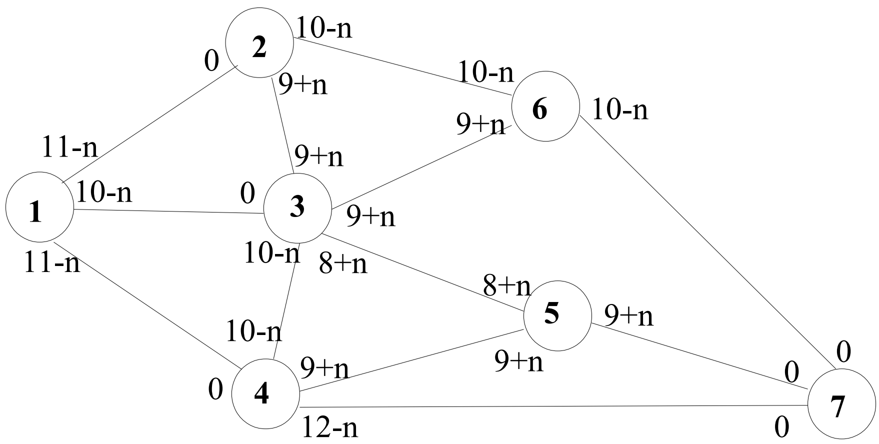
Система автодорог, проходящих через Одесскую область, может обеспечить пропускные способности, показанные на приводимом ниже рисунке (тысячи автомашин в час).



1) Каков максимальный поток через эту систему (тыс автомашин в час)?
2) Сколько автомашин в час должно проехать по дороге 5-6 чтобы обеспечить максимальный поток?

Задача №2.

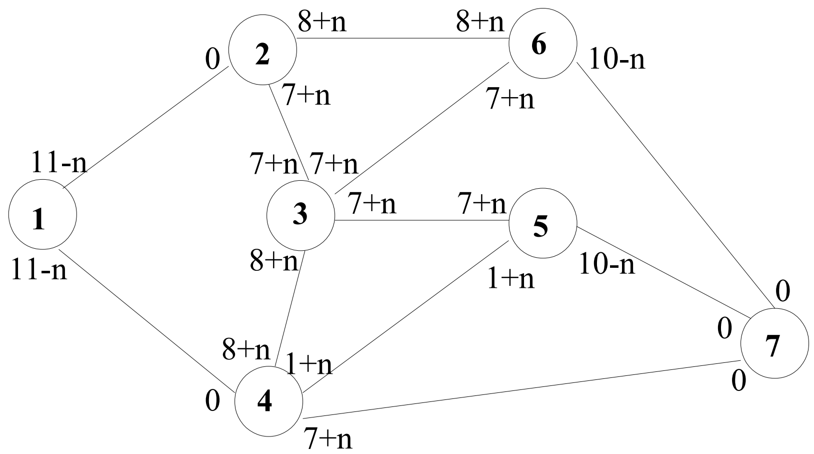
Телефонная компания использует подземную кабельную сеть линии связи для обеспечения качественной аудио связи между большими городами (узлы 1 и 7 сети). Переговоры осуществляются через серию кабельных линий и соединяющие их узлы сети, как это показано на рисунке. На нем показано также число телефонных переговоров (тыс.), которое допускается одновременно в любой точке времени.



1) Каково максимальное число телефонных переговоров между двумя городами, которое может быть допущено одновременно (тыс.шт.)?
2) Какое число телефонных переговоров должно обеспечиваться кабелем 4-7?

Задача №3.

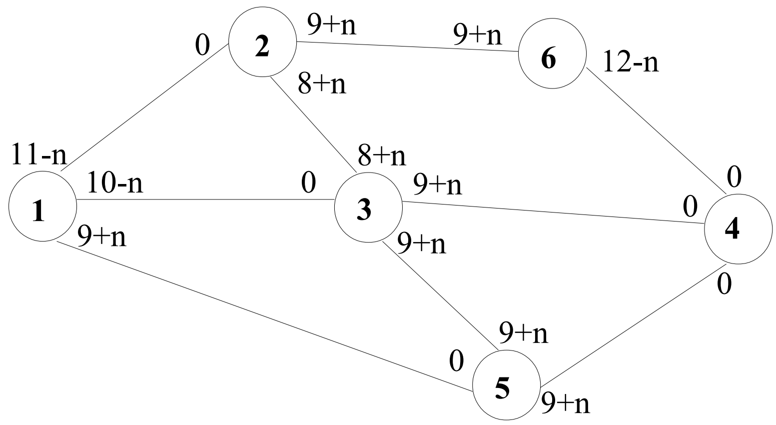
Нефтяная компания «Лукойл» владеет сетью нефтепроводов, через которые нефть перекачивается от месторождений до нефтехранилищ. Часть этой сети представлена на рисунке (пропускная способность нефтепроводов показана в тыс. т/ч).



1) Если компания хочет поставить нефть в хранилище 7 и полностью использовать пропускную способность системы, то сколько времени займет поставка 10 тыс. т. нефти?
2) Какой максимальный поток может обеспечит линия 2-3?

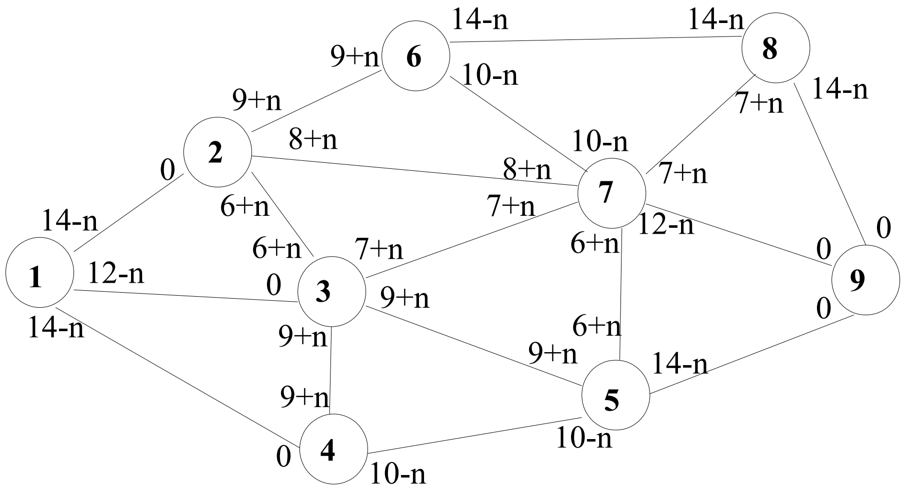
Задача №4.

1) Чему равен максимальный поток автомашин (количество автомашин в час) для системы автодорог, представленной на рисунке?
2) Какой максимальный поток может обеспечит дорога 3-4?



Задача №5.

Химический завод имеет сеть труб, предназначенных для перемещения жикуих химических продуктов из одних частей предприятия в другие. Сеть труб и пропускные способности (т/мин) показаны на рисунке.



1) Каков максимальный поток для системы, если завод собирается перегнать из узла 1 в узел 9 столько жидких химикатов, сколько это возможно?
2) Сколько химикатов будет поступать через секцию 3-5?