ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Северо-Кавказский филиал

ордена Трудового Красного Знамени федерального государственного

бюджетного образовательного учреждения высшего образования

«Московский технический университет связи и информатики»

ОПТИМИЗАЦИЯ НА ГРАФАХ

Учебно-методическое пособие

для проведения практических занятий по дискретной математике

Ростов-на-Дону

2019 г.

УДК 519.1
ББК 22.17
К 64

Конева С.И.

Оптимизация на графах: учебно – методическое пособие для проведения практических занятий. – Ростов- на-Дону: Северо-Кавказский филиал МТУСИ, 2019. – 30 с.

Пособие предназначено для проведения занятий со студентами специальностей 11.03.02 и 09.03.01.

Составитель: Ст. преподаватель кафедры ИВТ Конева С. И.

Рецензент: Доцент кафедры ИВТ к.т.н. Чикалов А. Н.

Учебно – методическое пособие обсуждено и одобрено на заседании кафедры СПОИ.

Протокол №3 от 10.11.16

И з д а т е л ь с т в о С К Ф М Т У С И

Сдано в набор 10.11.16. Изд. № 262. Подписано в печать 23.01.17. Зак. 276.
Печ. листов 1,88. Учетно-изд. л. 1,5. Печать оперативная. Тир. 10 экз.
Отпечатано в Полиграфическом центре СКФ МТУСИ, Серафимовича, 62.

ОГЛАВЛЕНИЕ

Введение

Наиболее исследованным классом объектов, относящихся к графическим представлениям, являются так называемые *графы,* изучаемые в *теории графов.*

Название «граф» подразумевает наличие графической интерпретации. Теория графов имеет огромные приложения, так как ее язык, с одной стороны, нагляден и понятен, а с другой - удобен в формальном исследовании.

Теория графов многократно использовалась разными авторами при решении различных прикладных задач, таких как «Задача о Кёнигсбергских мостах», «Задача о трёх домиках и трёх колодцах», «Задача о четырёх красках».

На языке теории графов формулируются и решаются многие задачи управления, в том числе задачи сетевого планирования и управления. Теория графов предоставляет удобный *язык для описания программных моделей.*

1. ОСНОВНЫЕ ПОНЯТИЯ ТЕОРИИ ГРАФОВ

1.1 **Основные понятия**.

Графические представления в узком смысле - это описание исследуемой системы, процесса, явления средствами теории графов в виде совокупности двух классов объектов: *вершин* и соединяющих их линий - *ребер* или *дуг.*

Обычно граф изображают *диаграммой*: вершины точками или кружками, рёбра линиями.

Графом G называется совокупность двух множеств: *вершин V* и *рёбер E,* между элементами которых определено *отношение инцидентности* - каждое ребро *E* инциденно ровно двум вершинам v', v'' ∈ *V,* которые оно соединяет. При этом вершина v' (v'') и ребро *e* называются *инцидентными* друг другу, а вершины v' и v'', являющиеся для ребра *e* концевыми точками, называются *смежными.*

Ребро, соединяющее две вершины, может иметь направление от одной вершины к другой; в этом случае оно называется *направленным,* или *ориентированным,* или *дугой* и изображается стрелкой, направленной от вершины, называемой *началом,* к вершине, именуемой *концом.*

Граф, содержащий направленные ребра (дуги) с началом v' и концом *v''*, называется *ориентированным* (орграфом), а ненаправленные – *неориентированным* .

Ребра, инцидентные одной и той же паре вершин, называются *параллельными,* или *кратными.* Граф, содержащий кратные ребра, именуется *мультиграфом.* Ребро, концевые вершины которого совпадают, называется *петлей.*

Граф называется *конечным,* если множество его элементов (вершин и ребер) конечно, и *пустым,* если его множество вершин *V* (а значит и ребер *E)* пусто. Граф без петель и кратных ребер называется ***полным***, если каждая пара вершин соединена ребром.

Дополнением графа G называется граф G , имеющий те же вершины, что и граф *G,* и содержащий только те ребра, которые нужно добавить к графу *G,* чтобы получить полный граф.

Степень вершины – это удвоенное количество петель, находящихся у этой вершины плюс количество остальных прилегающих к ней рёбер.

Графы G_1, и G_2 *равны,* т.е. $G_1 = G_2$, если их множества вершин и ребер (выраженных через пары инцидентных им вершин)совпадают:$V_1 = V_2$ и $E_1=E_2$

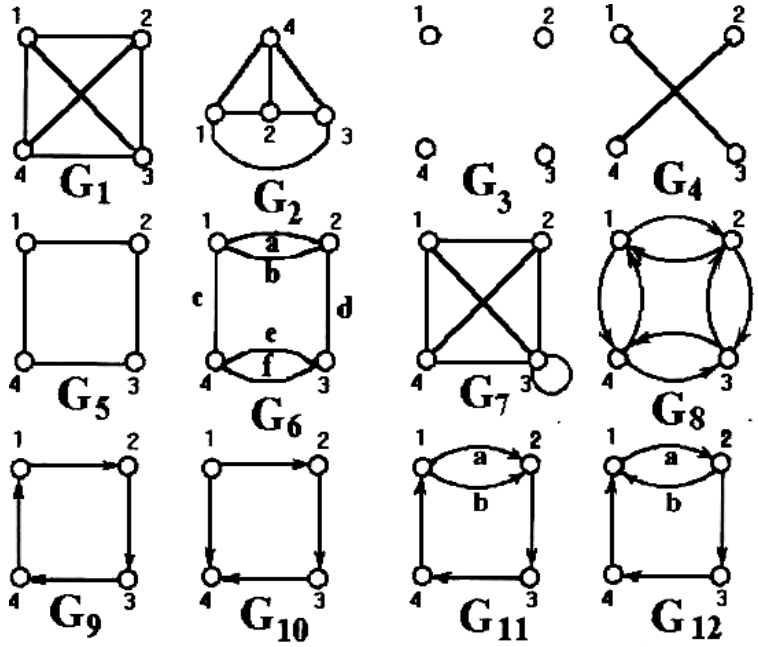


Рис. 1

Результаты сравнения графов таковы:

$G_1 – G_7$ — неориентированные; $G_8 – G_{12}$ - ориентированные;

G_7 - не является полным, хотя каждая пара вершин и соединена ребром, но имеется одна петля. (Иногда полным называют граф с петлями во всех вершинах, каждая пара которых соединена ребром). Граф G_7 не отвечает и этому определению. G_3 - все вершины этого графа являются изолированными (граф с пустым множеством ребер, т.е. $E = 0$); _

G_4 и G_5 ЯВЛЯЮТСЯ дополнением друг другу.

G_6 - мультиграф, так как содержит кратные рёбра.

Пример 1. Чему равны степени вершин графов приведенные ниже?

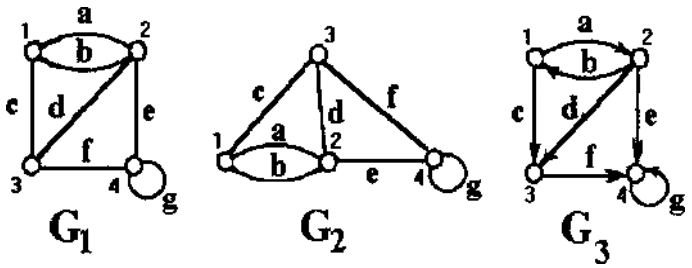


Рис. 2

Степени вершин неориентированного графа G_1 : p(1) = 3, p(2) = 4, p(3) = 3, p(4) = 4.

Степени вершин ориентированного графа G_3:

p(l) = 2, p (2) = 3, p,(3) = 1, p(4) = 1.

1.2 Способы задания графов

В общем виде задать граф - значит описать множества его вершин и ребер, а также отношение инцидентности. Для описания вершин и ребер достаточно их занумеровать. Отношение инцидентности задается:

• *матрицей инцидентности* размера *m* x *n* : по вертикали указываются вершины, а по горизонтали - ребра соответственно. На пересечении i-й вершины и j-го ребра в случае **неориентированного** графа проставляется 1, если они инцидентны, и 0 - в противном случае, т.е.

$$\begin{cases} 1, \text{если ребро } e_i \text{ инцидентно вершине } v_j, \\ 0\text{-в противном случае.} \end{cases}$$

в случае орграфа:

$$\begin{cases} -1, \text{ если вершина } v_i \text{ -начало дуге } e_j; \\ 1, \text{ если вершина vi- конец дуге } e_j; \\ 0\text{- если вершина vi не инцидентна дуге } e_j. \end{cases}$$

• ***списком ребер*** графа, представленным двумя столбцами: в левом перечисляются все ребра $e \in E$, а в правом - инцидентные ему вершины v_j.

• ***матрицей смежности*** - квадратной матрицей размера $n \times n$: по вертикали и горизонтали перечисляются все вершины $v_j \in V$ *для* ***неориентированного графа:***

$$\begin{cases} 1, \text{ если в графе вершины } v_i \text{ и } v_j \text{ соединены ребром;} \\ 0, \text{ иначе.} \end{cases}$$

Для *ориентированного графа:*

$$\begin{cases} 1, \text{ если в графе есть дуга из i–вершины в j – вершину;} \\ 0, \text{ иначе.} \end{cases}$$

• Если два графа равны, то их матрицы совпадают. Если в графе поменять нумерацию вершин, матрицы (и список ребер) в общем случае изменяются, т.е. вид матриц и списка ребер зависит от нумерации вершин и ребер графа.

Пример 2. Задать матрицами инцидентности и смежности графы G_1, G_3 (см. рис.2)

Матрицы смежности графов G_1 и G_3

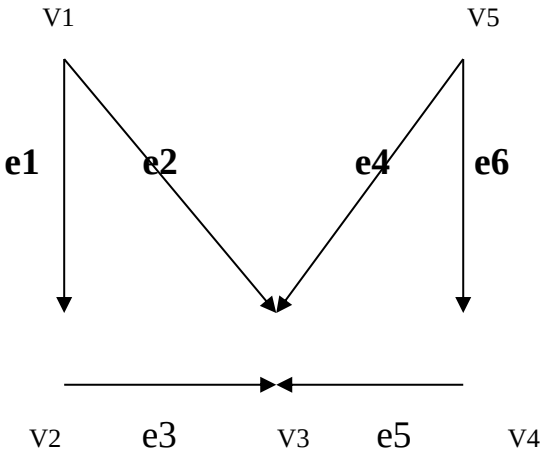
G_1

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

G_3

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

Пример 3. Построить матрицы смежности и инцидентности для орграфа D(V,X).



Матрица смежности имеет вид:

	V1	V2	V3	V4	V5
V1	**0**	**1**	**1**	**0**	**0**
V2	**0**	**0**	**1**	**0**	**0**
V3	**0**	**0**	**0**	**1**	**0**
V4	**0**	**0**	**0**	**0**	**1**
V5	**0**	**0**	**1**	**0**	**0**

Матрица инцидентности имеет вид:

	e1	e2	e3	e4	e5	e6
V1	**-1**	**-1**	**0**	**0**	**0**	**0**
V2	**1**	**0**	**-1**	**0**	**0**	**0**
V3	**0**	**1**	**1**	**1**	**-1**	**0**
V4	**0**	**0**	**0**	**0**	**1**	**-1**
V5	**0**	**0**	**0**	**-1**	**0**	**1**

Матрица смежности и списки смежности являются основными структурами данных, которые используются для представления графов в компьютерных программах.

1.3 **Маршруты, пути, цепи, циклы**.

Маршрутом в графе называется такая последовательность рёбер, в которой каждые два соседних ребра имеют общую вершину. В маршруте одно и то же ребро может встречаться несколько раз.

Маршрут, в котором совпадают его начало и конец, называется ***циклическим***.

Маршрут, в котором все рёбра разные, называется ***цепью***. Цепь, не пересекающая себя, т.е. не содержащая повторяющихся вершин, называется ***простой цепью***.

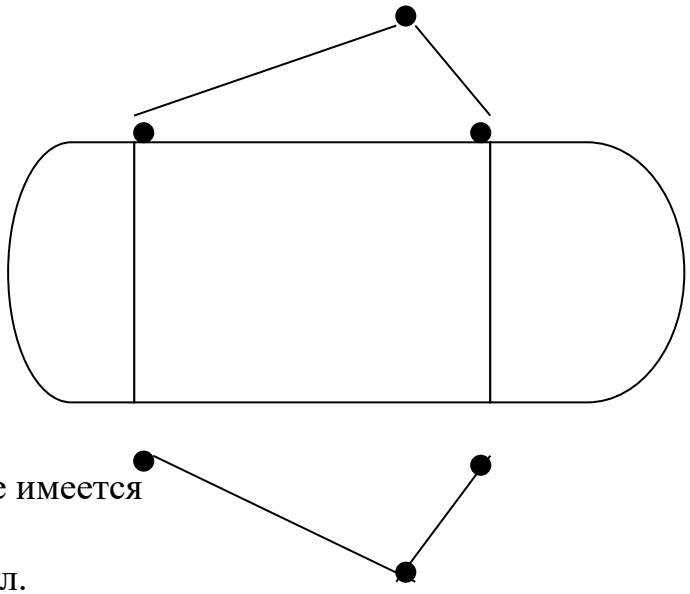
Путь в ориентированном графе - это последовательность дуг, в которой конечная вершина всякой дуги, отличной от последней, является начальной вершиной следующей.

Граф называется ***связным***, если любая пара его вершин связана. Орграф называется ***связным***, если он связан **без учёта ориентации дуг**, и сильно связан, если из любой вершины в любую существует путь.

Эйлеров цикл, цикл графа, содержащий все рёбра графа. Эйлеров граф – это граф, имеющий эйлеров цикл. *Теорема Эйлера*: **конечный** неориентированный граф **э*йлеров*** тогда и только тогда, когда **он связан** и степени всех его вершин **чётны**.

Гамильтонов цикл – простой цикл, проходящий через все вершины рассматриваемого графа. Гамильтонова цепь – простая цепь, проходящая через все вершины графа, **с началом и концом в разных вершинах.**

Пример 4. Наличие эйлерова цикла в следующем графе.

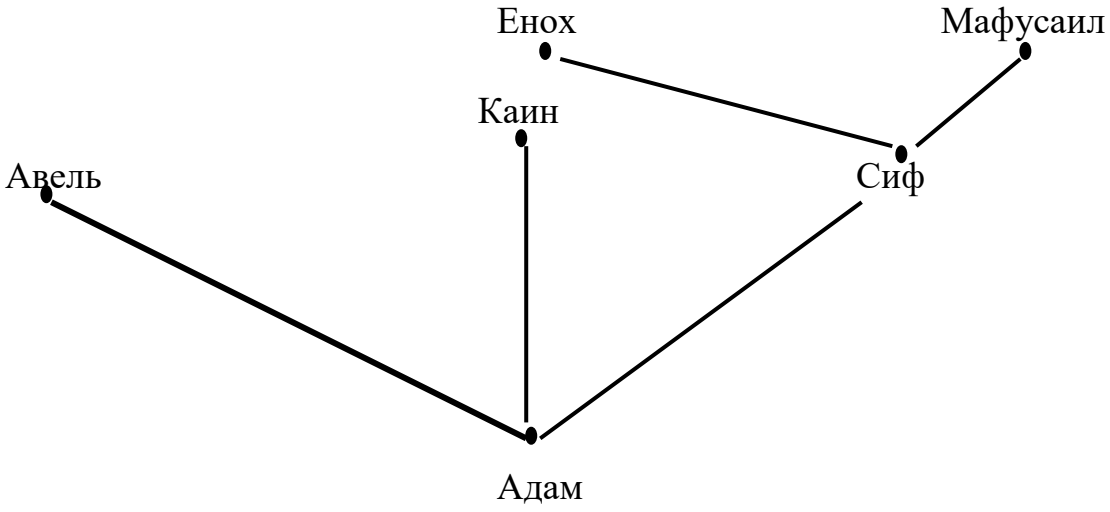


В этом графе имеется

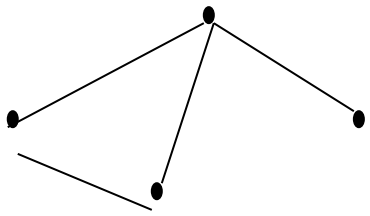
Эйлеров цикл.

1.4 **Деревья.**

Дерево – это **связный** граф **без циклов**. ***Дерево*** – это минимальный связный граф в том смысле, что при удалении хотя бы одного ребра он теряет связность. Деревья особенно часто возникают на практике при изображении различных иерархий. Примеры деревьев: генеалогический граф (родословное дерево), совокупность всех файлов на дискете.



Библейское генеалогическое дерево.



Этот граф не является деревом, так как содержит цикл.

При обходе дерева невозможны циклы. Это делает деревья очень удобной формой организации данных для различных алгоритмов. Таким образом, понятия дерева активно используется в информатике и программировании.

Вершина графа называется концевой, или висячей, если её степень p(v) = 1. Ребро, инцидентное концевой вершине, называется концевым.

Каждому дереву с ***m*** рёбрами можно взаимно однозначно сопоставить вектор длины ***2m из нулей и единиц,*** называемый ***кодом дерева.***

Обход дерева начинается с корня дерева. По каждому ребру нужно пройти дважды. Первый проход по ребру отмечается **нулём**. Повторному проходу по ребру соответствует **единица.** Из всех возможных вариантов продолжения обхода выбирается продолжение обхода по крайнему левому ребру. Заканчивается обход в корне дерева. Количество нулей в коде дерева равно количеству единиц.

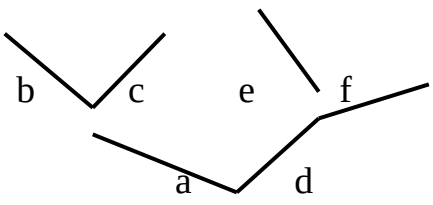


Рис.3

Пример 5.Обозначим рёбра дерева буквами латинского алфавита. Тогда обход дерева задаётся следующей последовательностью.

a, b, b, c, c, a, d, e, e, f, f, d

По этой последовательности построим код дерева.

Движение по последовательности начинаем слева направо. Если буква встречается в последовательности первый раз, то вместо неё присваиванием **0.** Буквам, которые повторяются в последовательности, соответствует **1.** Тогда код дерева равен:

001011001011

По коду дерева можно восстановить само дерево. Двигаемся по последовательности из нулей и единиц слева направо. Если очередной символ равен нулю, то рисуем новое ребро. Если очередной символ – это единица, то очередной шаг делаем в обратном направлении по последнему нарисованному ребру.

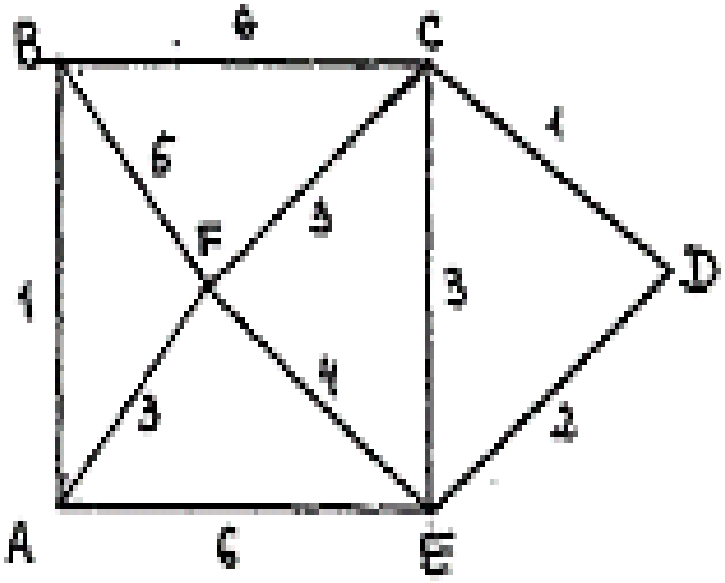
1.5 **Алгоритм Краскалла оптимизации на взвешенных графах**

Цель практического занятия: освоить алгоритм Краскалла оптимизации параметров на взвешенных графах.

Граф, каждому ребру которого сопоставлено некоторое число, называется **взвешенным** графом. Число, сопоставленное ребру, называется **весом** ребра. Одной из важнейших задач в теории взвешенных графов является задача о кратчайшем соединении, основанная на применении алгоритма **Краскалла**. Рассмотрим это на примере.

Пример 6.

A,B,C,D,E,F – населенные пункты, линии – проектируемые участки дорог, цифры – их стоимость. Найти, какие дороги надо построить, чтобы полученная схема дорог позволяла попасть из любого города в любой и из всех возможных схем имела наименьшую стоимость.



Решение.

Решаем задачу, используя алгоритм Краскалла.

Алгоритм Краскалла.

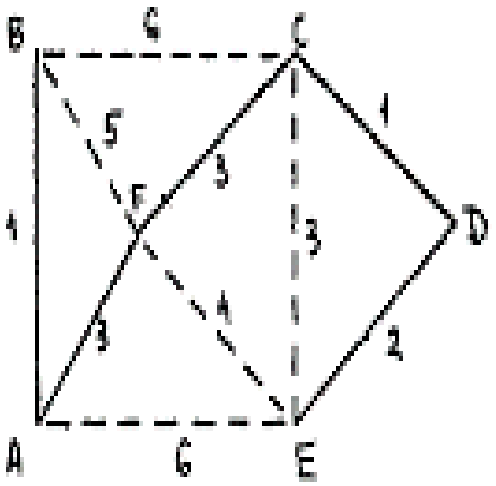
1. ν_i – дуга минимального веса из всех имеющихся дуг, не являющаяся петлей.

2. Если дуги ν_m,… $\nu_{к-m}$ уже выбраны, то $\nu_к$ выбираем из множества еще не выбранных дуг следующим образом:

   а) Добавление дуги $\nu_к$ не приводит к образованию циклов;

   б) Из дуг, удовлетворяющих условию а, дуга $\nu_к$ обладает наименьшим весом.

В нашем случае выбираем ν_1=CD, ν_2=AB, ν_3=ED и далее отпадает возможность выбора CE, т. к. это приводит к образованию цикла. ν_4=CF и отпадает возможность выбора EF. ν_5=AF, отпадает возможность выбора BC,BF, AE и процесс выбора дуг автоматически оборвался. Полученный путь изображен на рис. Сплошными линиями.



Задания для практического занятия.

1. Строится нефтепровод, соединяющий города x_1……$x_{11.}$ Стоимость возможных участков строительства задана таблицей

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
x_1	0	6		10							
x_2		0	6		8	6					
x_3			0				11				
x_4				0	3					5	
x_5					0	9		4			
x_6						0					

x_7							0		12		
x_8								0		7	
x_9									0	11	10
x_{10}										0	9
x_{11}											0

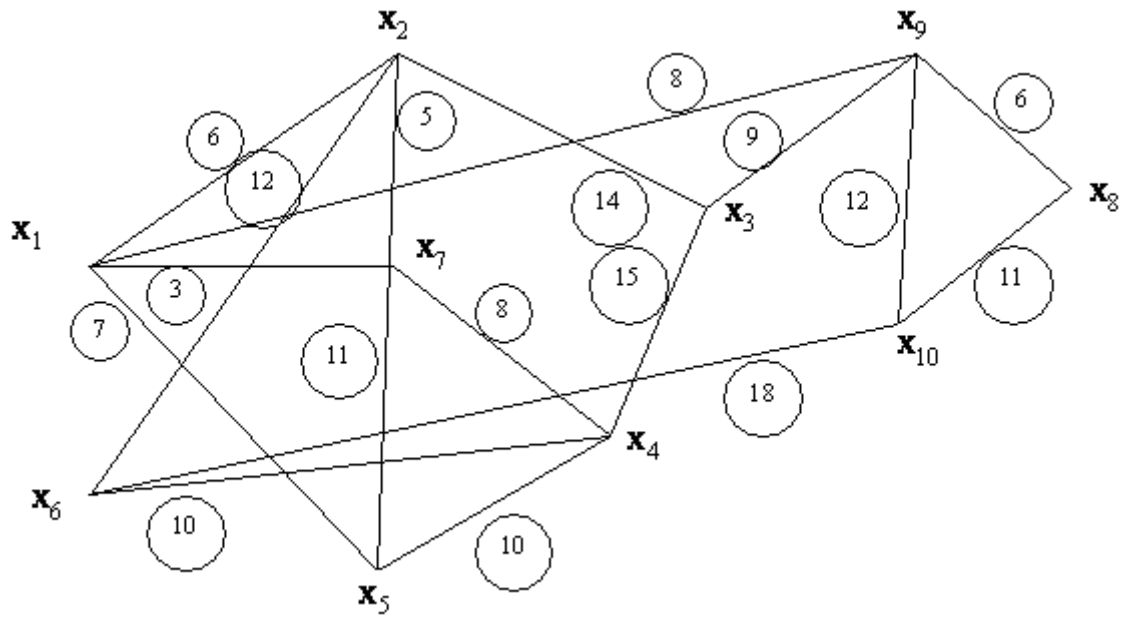
Изобразить схему нефтепровода, стоимость строительства которого минимальна, и подсчитать минимальные затраты.

2. Каждый из шести городов может быть соединен с другим участком газопровода, стоимость строительства которого указана в таблице

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	2	4	9	6	10
x_2		0	3	3	9	7
x_3			0	5	8	8
x_4				0	7	10
x_5					0	11
x_6						0

Как построить самый дешевый нефтепровод, какова стоимость его строительства?

3. 10 городов, обозначенных на графе вершинами $x_1 \ldots x_{10,}$ необходимо соединить электролинией. Возможные соединения обозначены ребрами. Стоимость строительства на участке (x_i,x_j) обозначена соответствующим числом.



Определить стоимость строительства самой дешевой электролинии. Как она должна проходить?

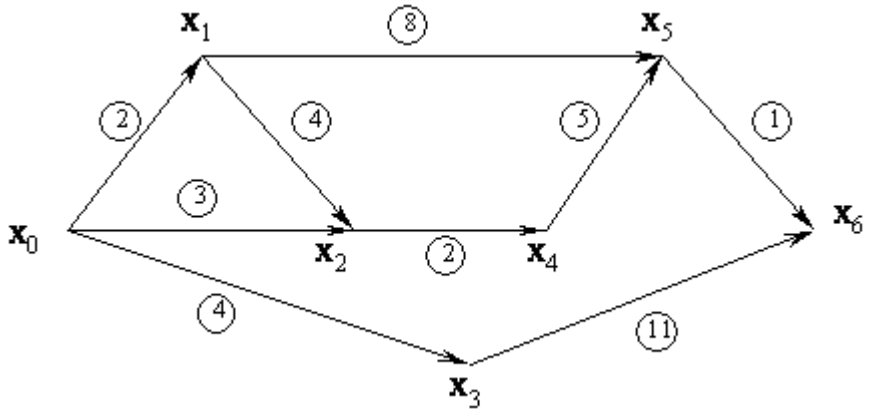
4. На строительство электролинии между городами x_1…..x_7 отпущено 250 тысяч рублей. Стоимость строительства на возможных участках между городами x_i,x_j $l(x_ix_j)$ в тыс. руб. задана следующим образом:

$l(x_1,x_2)=40$	$l(x_2,x_6)=20$	$l(x_4,x_5)=90$
$l(x_1,x_3)=50$	$l(x_2,x_7)=30$	$l(x_4,x_7)=70$
$l(x_1,x_6)=60$	$l(x_2,x_4)=90$	$l(x_5,x_6)=80$
$l(x_1,x_7)=50$	$l(x_3,x_4)=60$	$l(x_5,x_7)=100$
$l(x_2,x_3)=30$	$l(x_3,x_7)=60$	$l(x_6,x_7)=10$

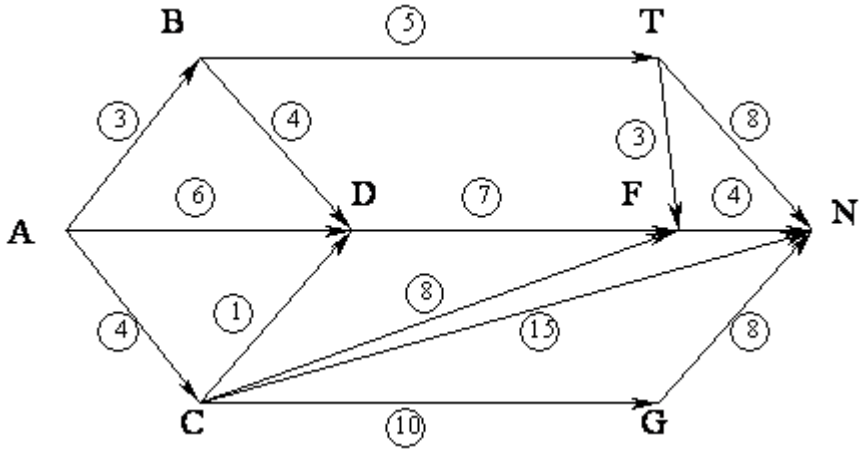
Как построить электролинию, чтобы уложиться в эту смету?

5. Определить наименьшие затраты при перевозке груза из пункта x_0 в пункт x_6через перевалочные пункты x_1,x_2,x_3,x_4,x_5. Стоимость перевозки

груза из пункта x_i в x_j указана на графе. Определить путь, соответствующий минимальной стоимости.

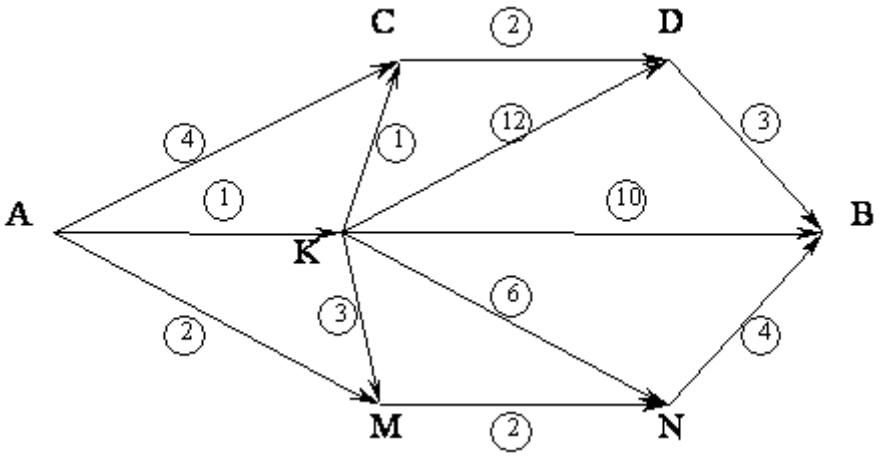


6.



Из пункта A в пункт N перевозят однородный груз, используя перевалочные пункты B,C,D,E,F,G. Расстояние между пунктами, соединенными дорогами, указаны на графе. Определить кратчайший путь и его длину, предварительно пронумеровав вершины графа.

7.

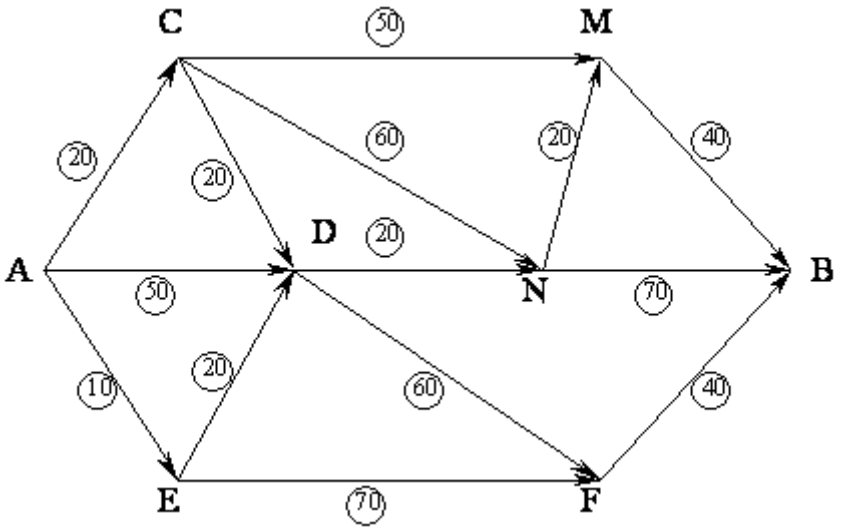


Пункты A и B связаны сетью дорог, проходящих через пункты C,D,E,M,N. Стоимость проезда из пункта x_i в x_j указана на графе. Какова минимальная стоимость проезда из A и B?. Как проходит путь, соответствующий минимальным затратам?

8. Для графов задач 6, 7 определить критический путь и критическое время.

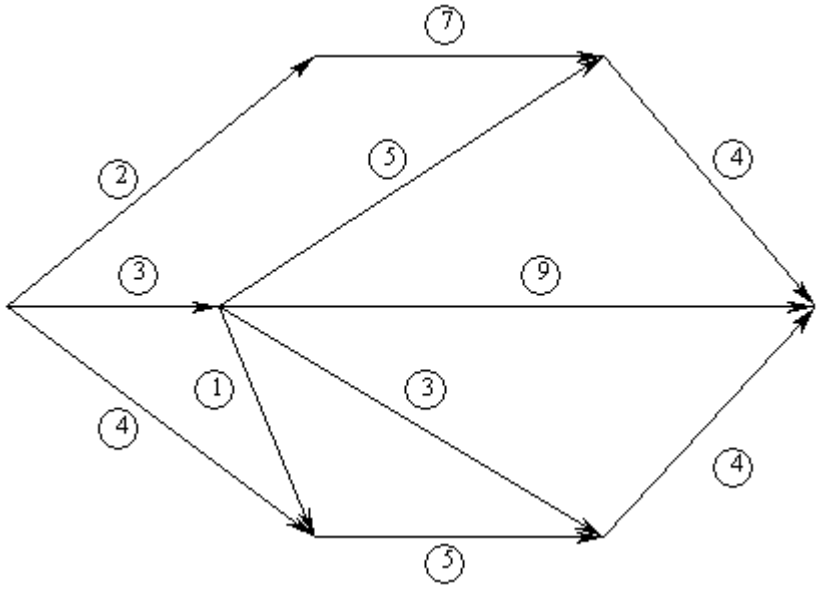
9.

Основу строительства объекта составляют 14 операций, последовательность выполнения которых задана графом. Продолжительность каждой из них указана на графе.



Определить скорейшее время завершения всего проекта. Какие операции не допускают запаздывания по времени?

10. Найти кратчайший и длиннейший пути, соединяющие вход и выход графа, предварительно правильно пронумеровав вершины.



1.6 Алгоритм Дейкстры нахождения кратчайшего пути в графах.

Цель практического занятия освоить использование алгоритма Дейкстры для нахождения кратчайшего пути в графах.

Алгоритм Дейкстры.

Рассмотрим неориентированный взвешенный граф G=(V,E). Назовем маршрутом в этом графе такую последовательность ребер графа, в которой начало каждого ребра является концом предыдущего ребра, а конец ребра – началом последующего ребра. Назовем длиной маршрута, соединяющего две вершины, число, равное сумме весов всех ребер, входящих в этот маршрут, а маршрут наименьшей длины – кратчайшим маршрутом между двумя вершинами.

Поставим задачу нахождения кратчайших маршрутов между некоторой вершиной s графа и другими его вершинами.

Одним из эффективных алгоритмов решения этой задачи является алгоритм Дейкстры, основанный на присвоении всем вершинам ν_i графа G временных меток $m(\nu_i)$, которые задают верхнюю границу длины маршрута от вершины s до рассматриваемой вершины. На каждой итерации одна (или несколько) временная метка становится постоянной (постоянными), определяя длину кратчайшего маршрута от вершины s до рассматриваемой вершины (вершин).

Приведем алгоритм Дейкстры.

Шаг 1. Начальная установка меток.

Принять метку вершины $m(s) = 0$ и считать эту метку постоянной. В дальнейшем постоянные метки будем обозначать *, т.е. $m(s) = 0^*$. Для всех $\nu_i \neq s$ положить $m(\nu_i)=\infty$ и считать эти метки временными. Пусть S – множество

вершин с постоянными метками. Положить S={ s }. Будем обозначать через Г(S) вершины графа, смежные по крайней мере с одной из вершин графа, входящих в множество S.

Шаг 2. Метки $\nu_i \notin \Gamma(S)$ изменить следующим образом:

$$m(\nu_i)= \min(m(\nu_i)); m(\nu)+c(\nu,\nu_i),$$

где ν - произвольная метка вершины графа, принадлежащая множеству S, $c(\nu,\nu_i)$ – вес ребра $\{\nu,\nu_i\}$.

Шаг 3. После пересчета меток, входящих в Г(S), в множестве этих вершин найти вершину, имеющую метку наименьшего веса, сделать ее постоянной и добавить эту вершину в множество S.

Шаг 4. Если $\Gamma(S)=\varnothing$, работа алгоритма закончена. Все вершины имеют постоянные метки, которые определяют длины кратчайших маршрутов до вершины s.

Если $\Gamma(S)\neq\varnothing$, то переход к шагу 2.

Замечание. Если необходимо решить задачу о нахождении длины кратчайшего пути до некоторой вершины $\nu_к$, то алгоритм должен быть прерван после присвоения вершине $\nu_к$ постоянной метки.

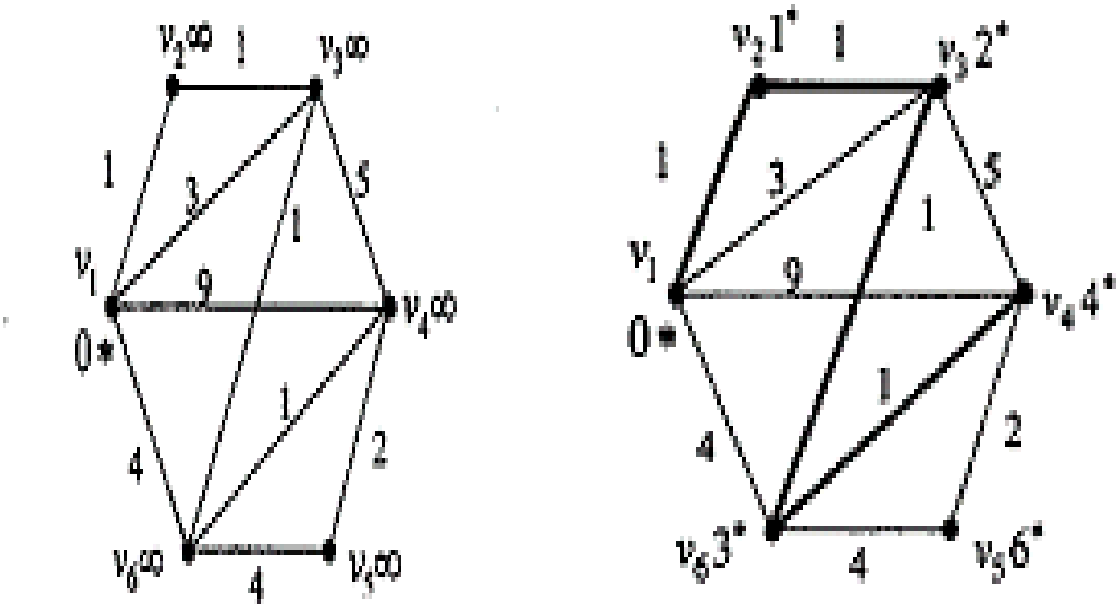
После нахождения длин кратчайших маршрутов, сами маршруты можно найти при помощи рекурсивной процедуры. Для любой вершины ν_i предыдущую вершину маршрута ν_i’ можно найти как вершину (или одну из вершин), смежную с вершиной ν_i, для которой выполняется условие:

$$m(\nu_i')+c(\nu_i',\nu_i)= m(\nu_i).$$

Так, начиная с последней вершины, можно восстановить кратчайший маршрут от вершины $\nu_к$ до вершины s, а значит и от s до $\nu_к$.

Заметим, что восстановление кратчайшего маршрута в некоторых случаях может быть неоднозначным.

Пример 7. Для графа, представленного на рисунке (слева), найти кратчайшие маршруты от вершины ν_1 ко всем остальным вершинам. Указать кратчайший маршрут, соединяющий вершины ν_1 и ν_4.



Решение.

Результаты пошагового выполнения алгоритма приведены в таблице:

	ν_1	ν_2	ν_3	ν_4	ν_5	ν_6
Шаг 1.	0*	∞	∞	∞	∞	∞
Шаг 2.1,3.1.	0*	1*	3	9	∞	4
Шаг 2.2,3.2.	0*	1*	2*	9	∞	4
Шаг 2.3,3.3.	0*	1*	2*	7	∞	3*
Шаг 2.4,3.4.	0*	1*	2*	4*	7	3*
Шаг 2.5,3.5.	0*	1*	2*	4*	6*	3*

Шаг 1. $m(\nu_1) = 0^*$, $m(\nu_i) = \infty$ при всех i=2;6, $S\{\nu_1\}$, $\Gamma(S)= \{\nu_2; \nu_3; \nu_4; \nu_6\}$.

Шаг 2.1. Обновление меток: $m(\nu_2)= \min(\infty;0^* +1)=1$,

$m(\nu_3)= \min(\infty;0^* +3)=3$,

$m(\nu_4)= \min(\infty;0^* +9)=9$,

$m(\nu_4)= \min(\infty;0^* +4)=4$.

Шаг 3.1. Выбираем вершину с наименьшим весом метки (это вершина ν_2) и делаем эту метку постоянной: $m(\nu_2)=1^*$. Добавляем метку ν_2 в множество S: $S= \{\nu_2; \nu_3\}$; $\Gamma(S)= \{\nu_3; \nu_4; \nu_6\}$.

Шаг 2.2. Обновляем метки: $m(\nu_3)= \min(3;1^* +1)=2$. Метки $m(\nu_4)=9$, $m(\nu_4)=4$ не изменяются.

Шаг 3.2. Выбираем вершину с наименьшим весом метки (это вершина ν_3) и делаем ее постоянной: $m(\nu_3)=2^*$. Полагаем $S= \{\nu_1;\nu_2;\nu_3\}$. Тогда $\Gamma(S)= \{\nu_4; \nu_6\}$.

Шаг 2.3. Обновление меток: $m(\nu_6)= \min(4;2^* +1)=3$, $m(\nu_4)= \min(9;2^* +5)=7$.

Шаг 3.3. Выбираем вершину с наименьшим весом метки (это вершина ν_6) и делаем ее постоянной: $m(\nu_6)=3^*$. Полагаем $S= \{\nu_1;\nu_2;\nu_3;\nu_6\}$. Тогда $\Gamma(S)= \{\nu_4; \nu_5\}$.

Шаг 2.4. Обновление меток: $m(\nu_5)= \min(\infty;3^* +4)=7$, $m(\nu_4)= \min(7;3 * +1)=4$.

Шаг 3.4. Выбираем вершину с наименьшим весом метки (это вершина ν_4) и делаем ее постоянной: $m(\nu_4)=4^*$. Полагаем $S= \{\nu_1;\nu_2;\nu_3;\nu_4;\nu_6\}$. Тогда $\Gamma(S)= \{\nu_5\}$.

Шаг 2.5. Обновление меток: $m(\nu_5)= \min(7;4^* +2)=6$.

Шаг 3.5. Выбираем вершину с наименьшим весом метки (это вершина ν_5) и делаем ее постоянной: $m(\nu_5)=6^*$. Полагаем $S=\{\nu_1;\nu_2;\nu_3;\nu_4;\nu_5;\nu_6\}$. Тогда $\Gamma(S)=\varnothing$. Выполнение алгоритма закончено.

Вершине ν_4 присвоена постоянная метка $m(\nu_4)=4^*$. Поэтому длина кратчайшего маршрута из вершины ν_1 в вершины ν_4 равна 4.

Теперь построим сам маршрут.

Начинаем с конца цепи: вершины ν_4. Среди смежных с ней вершин находим такую вершину ν_4', для которой:

$$m(\nu_4')+c(\nu_4',\nu_4)= m(\nu_4).$$

Видно, что это равенство выполняется для вершины ν_6: $3^* +2=4^*$ (для ν_1 : $0^* +9\neq4^*$, для ν_3 : $2^* +5\neq4^*$, для ν_5 : $6^* +2\neq4^*$). Значит, вершина ν_6 принадлежит искомому пути.

Далее ищем вершину ν_6', смежную ν_6 и удовлетворяющую условию

$m(\nu_6')+c(\nu_6',\nu_6)= m(\nu_6)$. Видно, что этому условию удовлетворяет вершина ν_3 : $2^* +1=3^*$, Т.о., получаем искомый маршрут: $\nu_1;\nu_2;\nu_3;\nu_6;\nu_4$.

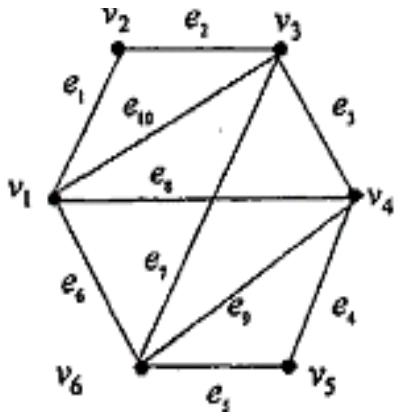
Результат работы алгоритма показан на рисунке (справа).

Задания.

Пользуясь алгоритмом Дейкстры, найти кратчайшие расстояния

из вершины υ_1 неориентированного взвешенного графа в другие вершины графа.

Указать кратчайший маршрут из вершины υ_1 в вершину υ_4.



1.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

2.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

3.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

4.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

5.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

6.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

7.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

8.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

9.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

10.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

11.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

12.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

13.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

14.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

15.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

16.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

17.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

18.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

19.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

20.

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
3	2	4	1	2	3	3	8	4	1

ЛИТЕРАТУРА.

1.	Ф. А. Новиков	Дискретная математика для бакалавров и магистров: стандарт 3-го поколения	СПб.: Питер, 2013.
2.	С. В. Микони	Дискретная математика для бакалавра: Учебное пособие	СПб.: Издательство «Лань», 2012.
3.	Г. И. Просветов	Дискретная математика: Задачи и решения: Учебно – практическое пособие. 2-е изд., доп	М.: Издательство «Альфа-Пресс», 2015.
4.	Г. И. Просветов	Математические методы в логистике: Задачи и решения. 3-е изд.	М.: Издательство «Альфа-Пресс», 2014.
5.	Д. В. Гринченков С. И. Потоцкий	Математическая логика и теория алгоритмов для программистов: Учебное пособие	М.: КНОРУС, 2014.
6.	В. И. Игошин	Математическая логика: Учебное пособие.	М.: Инфра-М, 2013
7.	Л. М. Лихтарников, Т. Г. Сукачева	Математическая логика: Задачник – практикум и решения: Учебное пособие	СПб.: Издательство «Лань», 2012.
8.	О. П. Кузнецов	Дискретная математика для инженера	СПб.: Издательство «Лань», 2007.