ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Б1.В.ДВ.03.01 Теория графов и её приложения

Специальность 10.05.03 Информационная безопасность автоматизированных систем

Специализация Информационная безопасность автоматизированных систем критически важных объектов

Форма обучения очная

СОДЕРЖАНИЕ

1. КОНСПЕКТ ЛЕКЦИЙ

1. 1 Лекция № 1 (2 часа).

Тема: «Определение графов, основные понятия теории графов. Виды графов. Способы задания графов. Матрицы смежности и инцидентности графа. Матрица Кирхгофа. Числовые характеристики графов»

1.1.1 Вопросы лекции:

1. Определение графов, основные понятия теории графов. Виды графов. Способы задания графов. Матрицы смежности и инцидентности графа. Матрица Кирхгофа.
2. Числовые характеристики графов.

1.1.2 Краткое содержание вопросов:

Наименование вопросов.

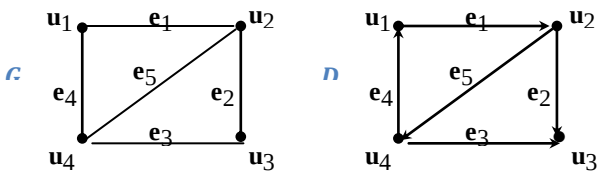
1. Определение графов, основные понятия теории графов. Виды графов. Способы задания графов. Матрицы смежности и инцидентности графа. Матрица Кирхгофа.
2. Числовые характеристики графов.

Представление графов в ЭВМ. Требования к представлению графов. Чтобы задать граф, нужно каким-либо способом описать множество его вершин, множество его ребер, а также указать, какие вершины и ребра инцидентны (или смежные), т.е. задать отношение инцидентности (смежности). Рассмотрим несколько способов представления графа в ЭВМ. Они различаются объемом занимаемой памяти и скоростью выполнения операций над графами. Представление выбирается по потребностям конкретной задачи.

Напомним: число вершин графа обозначаем через n, а число ребер – через m. Характеристика $M(n,m)$, приведенная для каждого представления, означает требуемый для него объем памяти.

Указанные представления пригодны для графов и орграфов, а после некоторой модификации – для псевдографов, мультиграфов и гиперграфов.

Все представления будем иллюстрировать на конкретных примерах графа G и орграфа D (см. рисунок.).



Способы представления графа

1) Матрица смежности.

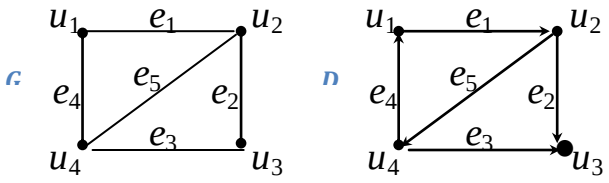
Матрица смежности A(G') графа (орграфа) – это квадратная матрица размера n×n, у которой для любых $i,j \in \{1,2,\ldots,n\}$ элемент в i-й строке и j-м столбце равен 1, если i-я и j-я вершины соединены ребром (дугой с началом в вершине i), и равен 0 в противном случае.

$$a_{ij} = \begin{cases} 1, & \text{если вершины } v_i \text{ и } v_j - \text{смежные (для орграфа дуга идет из } v_i \text{ в } v_j) \\ 0, & \text{иначе} \end{cases}$$

Память $M(n,m)=O(n^2)$.

Фактически это уже знакомая нам матрица бинарного отношения. Очевидно, что матрица смежности неориентированного графа является симметричной, элементы главной диагонали равны нулю, а количество единиц в каждой строке равно степени вершины, которой соответствует эта строка. По матрице смежности легко построить диаграмму графа.

Матрица смежности орграфа, не являющегося мультиграфом, не может быть симметричной, т.к. при ее составлении вершины орграфа играют различные роли.

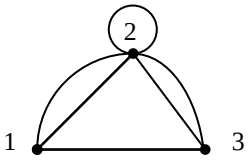


Матрицы смежности для заданных графа G и орграфа D

$$A(G) = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \quad A(D) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

В матрице смежности мультиграфа или псевдографа число, находящееся на пересечении i-й строки и j-го столбца, совпадает с числом ребер, соединяющих вершины i и j, при этом каждая петля считается двумя ребрами.

Псевдограф, изображенный на рисунке, имеет матрицу смежности следующего вида: $A(P) = \begin{pmatrix} 0 & 2 & 1 \\ 2 & 2 & 2 \\ 1 & 2 & 0 \end{pmatrix}$



2) Матрица инцидентности.

Другой способ задать граф – определить *матрицу инцидентности* (или *инциденций*) $I(G)$, имеющую n строк и m столбцов, элементы которой задаются следующим образом:

$$i_{kl} = \begin{cases} 1, & \text{если вершина } v_k \text{ инцидентна ребру } e_l \\ 0, & \text{иначе} \end{cases}$$

Для ориентированного графа:

$$i_{kl} = \begin{cases} 1, & \text{если вершина } v_k \text{ инцидентна ребру } e_l \text{ и является его концом} \\ 0, & \text{если вершина } v_k \text{ и ребро } e_l \text{ не инцидентны} \\ -1, & \text{если вершина } v_k \text{ инцидентна ребру } e_l \text{ и является его началом} \end{cases}$$

Матрицы инцидентности для заданных графа G и орграфа D

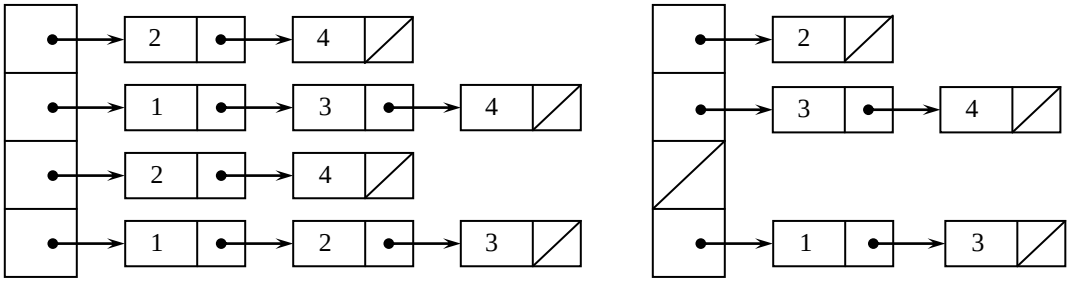
$$I(G) = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad I(D) = \begin{pmatrix} -1 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 1 \end{pmatrix}$$

Очевидно, что в каждом столбце матрицы инцидентности только два элемента отличны от 0 (или один, если ребро является петлей), т.к. ребро может быть инцидентно не более чем

двум вершинам (а столбец соответствует ребру). Поэтому матрица содержит много нулей и такой способ описания неэкономен. $M(n,m)=O(n\cdot m)$.

3) Списки смежности.

Граф представляется с помощью списочной структуры (списка смежности), отражающей смежность вершин и состоящей из массива указателей на списки смежных вершин. Элемент списка представлен структурой с двумя полями: номер вершины и указатель. Для неориентированных графов $M(n,m)=O(n+2m)$, для орграфов $M(n,m)=O(n+m)$.



Списки смежности для заданных графа G и орграфа D:

	кон	нач	кон
1	2	1	2
1	4	2	3
2	3	2	4
2	4	4	1
3	4	4	3

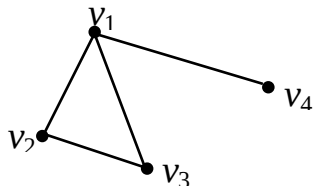
4) Массив ребер (дуг).

Отношение инцидентности можно задать также списком ребер графа. Каждая строка этого списка соответствует ребру, в ней записаны номера вершин, инцидентных ему. $M=O(2m)$.

По списку ребер графа легко построить матрицу инцидентности, т.к. каждое ребро этого списка соответствует столбцу матрицы, а номера вершин в каждом элементе списка – это номера строк матрицы инцидентности, элементы в которых равны 1. Для орграфа координата начала – номер строки, где стоит –1, а координата конца – номер строки, где стоит 1.

2. Числовые характеристики графов

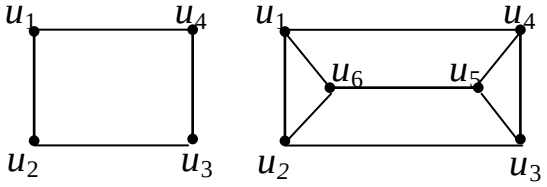
Степенью (или *валентностью*) *вершины* v называется число инцидентных ей ребер. Степень вершины обозначается $deg(v)$. Очевидно, что для любой вершины $v \in V$ справедливо: $0 \le deg(v) \le |V|-1$; $deg(v) = |\Gamma(v)|$. Вершина графа, имеющая степень 0, называется *изолированной*, а вершина со степенью 1 – *висячей*, или *концевой*.



В показанном на рисунке графе вершина v_4 является висячей: $deg(v_4)=1$. Степени остальных вершин: $deg(v_1)=3$; $deg(v_2) = deg(v_3) = 2$.

Если степени всех вершин графа одинаковы и равны некоторому числу k, то такой граф называется *регулярным* графом степени k. Степень регулярности является инвариантом

графа и обозначается *r(G)*. Для нерегулярных графов *r(G)* не определено. На рисунке показаны регулярные графы соответственно степени 2 и 3. Найдем степенную последовательность для графа *G*. Выпишем степени всех вершин графа в соответствии с их номерами (2,2,3,2,1).



1. 2 Лекция № 2 (2 часа).

Тема: «Маршруты, циклы, связность. Свойства связных графов, Эйлеровы и гамильтоновы графы. Ориентированные графы и деревья. Сети»

1.2.1 Вопросы лекции:

1. Маршруты, циклы, связность. Свойства связных графов, Эйлеровы и гамильтоновы графы.
2. Ориентированные графы и деревья. Сети.

1.2.2 Краткое содержание вопросов:

Наименование вопросов.

1. Маршруты, циклы, связность. Свойства связных графов, Эйлеровы и гамильтоновы графы.
2. Ориентированные графы и деревья. Сети.

Маршруты, цепи, циклы. Маршрутом *от вершины* u *к вершине* v *или (*u,v*)*-маршрутом *в графе* G *называется всякая последовательность вида* $u=v_0,e_1,v_1,e_2,\ldots,e_n,v_n=v$, *в которой любые два соседних элемента инцидентны, т.е.* e_k *– ребро, соединяющее вершины* v_{k-1} *и* v_k, k = *1,2,…,*n.

Это определение подходит также для псевдо-, мульти- и орграфов. В случае орграфа v_{k-1} *– начало* ребра e_k , а v_k – его *конец*. При этом вершину *u* называют *началом маршрута*, а вершину *v* – его *концом*. В маршруте некоторые вершины и ребра могут совпадать. Если *u* = *v*, то маршрут *замкнут*, а иначе *открыт*. Для «обычного» графа маршрут можно задавать только последовательностью вершин $v_0,v_1,\ldots,v_n$ или ребер $e_1,e_2,\ldots,e_n$.

Маршрут называется *цепью*, если в нем нет совпадающих ребер, и *простой цепью* – если дополнительно нет совпадающих вершин, кроме, может быть, начала и конца цепи. Про цепь $u=v_0,v_1,\ldots,v_n=v$ говорят, что она *соединяет* вершины *u* и *v* и обозначают ⟨*u*,*v*⟩.

Очевидно, что если есть цепь, соединяющая вершины *u* и *v*, то есть и простая цепь, соединяющая эти вершины.

Замкнутая цепь называется *циклом*; замкнутая простая цепь – *простым циклом*. Число циклов в графе *G* обозначается *z(G)*. Граф без циклов называется *ациклическим*.
Для орграфов цепь называется *путем*, а цикл – *контуром*.

Число ребер в маршруте M (возможно, с повторениями) называется его *длиной*, обозначается |M|.

Расстоянием между вершинами u и *v* (обозначается *d*(*u*,*v*)) называется

длина кратчайшей цепи $\langle u,v\rangle$, а сама кратчайшая цепь называется *геодезической*. Если не существует цепи, соединяющей вершины u и v, то по определению $d(u,v) = +\infty$.

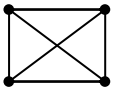
Диаметром графа G (обозначается $D(G)$) называется длина длиннейшей геодезической.

Максимальным удалением в графе G *от вершины* v называется $r(v) = max\ d(v,v'), \forall\ v' \in V$. Вершина v графа G является его *центром*, если максимальное удаление от нее до всех вершин принимает наименьшее значение.

Множество вершин, находящихся на одинаковом расстоянии n от вершины v, называется *ярусом* (обозначается $D(v,n)$): $D(v,n) = \{u \in V \mid d(v,u) = n\}$.

Граф, любая из вершин которого является его центром – максимальное удаление до всех вершин от любой =

Связность. *Если две вершины* u *и* v *в графе можно соединить цепью, то такие вершины связаны. Граф называется связным, если в нем связаны все вершины.*



Легко видеть, что отношение связности на множестве вершин является отношением эквивалентности. Данное отношение разбивает множество вершин графа на классы, объединяющие вершины, связанные друг с другом. Такие классы называются *компонентами связности*; число компонент связности обозначается $k(G)$.

Граф G является связным тогда и только тогда, когда он имеет одну компоненту связности: $k(G) = 1$. Если $k(G) > 1$, то это *несвязный* граф. Граф, состоящий только из изолированных вершин (в котором k(G)=|V|, r(G)=0), называется *вполне несвязным*.

Вершина графа, удаление которой увеличивает число компонент связности, называется *разделяющей* или *точкой сочленения*.

Ориентированный граф $G(V,E)$ является *слабо связным* (*слабым*), если симметричное замыкание множества E определяет связный граф (иными словами, если после замены всех дуг графа G ребрами полученный граф будет связным). Ориентированный граф является *сильно связным* (*сильным*), если для любой пары вершин $u,v \in V$ существует ориентированный путь из u в v (т.е. из любой вершины графа достижимы все его остальные вершины). Если для любой пары вершин по крайней мере одна достижима из другой, то такой граф является *односторонне связным*, или *односторонним*. Граф, состоящий из одной вершины, по определению считается сильно связным.

Множества вершин связных компонент образуют разбиение множества вершин графа.

1. 3 Лекция № 3 (2 часа).

Тема: «Нахождение экстремальных путей в сети: алгоритм Дейкстры и его прикладные аспекты. Нахождение экстремальных путей в сети с отрицательными весами: Алгоритм Беллмана – Мура. Компьютерные технологии реализации алгоритма Дейкстры.»

1.1.1 Вопросы лекции:

1. Нахождение экстремальных путей в сети: алгоритм Дейкстры и его прикладные аспекты.

2. Нахождение экстремальных путей в сети с отрицательными весами: Алгоритм Беллмана – Мура.
3. Компьютерные технологии реализации алгоритма Дейкстры.

1.1.2 Краткое содержание вопросов:

Наименование вопросов.

1. Нахождение экстремальных путей в сети: алгоритм Дейкстры и его прикладные аспекты.
2. Нахождение экстремальных путей в сети с отрицательными весами: Алгоритм Беллмана – Мура.
3. Компьютерные технологии реализации алгоритма Дейкстры.

Кратчайшие пути. *Задача поиска кратчайшего пути (наиболее дешевого? короткого?) «от пункта A до пункта B» имеет массу практических приложений и различные алгоритмы решения. Математическая постановка задачи имеет следующий вид.*

Рассматривается взвешенный граф (орграф) $G(V,E)$, ребрам (дугам) которого сопоставлены веса, обозначающие длину (или стоимость) пути из одного конца ребра в другой. Если из вершины v_i нет ребра (дуги) в вершину v_j, то вес ребра (v_i,v_j) считается равным ∞. Для ребер, являющихся петлями (диагональ матрицы смежности), их веса считаются равными 0. Все компоненты матрицы – веса ребер, соединяющих соответствующие вершины. Требуется определить кратчайший путь из одной вершины в другую.

Наиболее широко известны два алгоритма поиска кратчайших путей. Алгоритм Дейкстры находит кратчайшее расстояние от одной фиксированной вершины до другой и указывает сам путь, длина которого равна этому расстоянию. Алгоритм Флойда-Уоршалла позволяет найти кратчайшие расстояния между всеми парами вершин графа.

Алгоритм Дейкстры

Находит кратчайшее расстояние от вершины v_1 до вершины v_n

Идея: Идем по вершинам, постепенно удаляясь от старта, при этом для каждой вершины определяем самый короткий до нее путь из старта и запоминаем его.

Каждой вершине v_k ставится в соответствие упорядоченная пара (m,v_r), где первая координата означает наименьшее на текущий момент расстояние от старта – v_1 – до вершины v_k, а вторая координата указывает на предыдущую вершину на этом пути. Первоначально все вершины помечены парами $(\infty,0)$ и имеют статус временных. Постепенно по мере нахождения кратчайшего пути часть вершин становятся постоянными. Алгоритм заканчивает работу, когда постоянной станет вершина v_n.

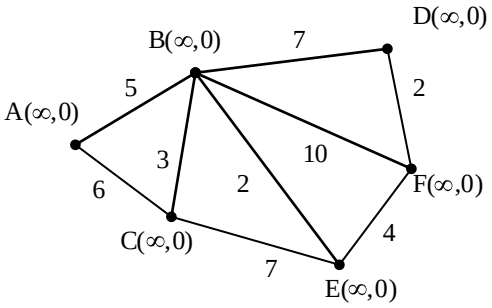
Алгоритм состоит из следующих шагов: Начать в вершине $v_1(\infty,0)$, заменить ее метку на $v_1(0,0)$ и сделать эту вершину
постоянной.

1. Пока v_n не станет постоянной вершиной, проделывать следующие шаги:

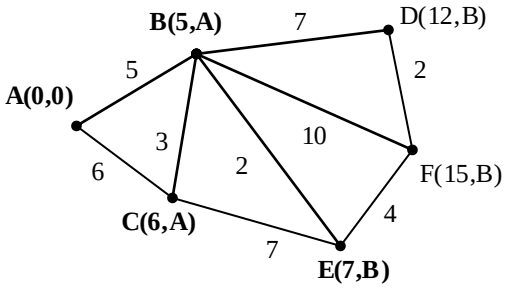
   а. Если вершина $v_k(m,v_r)$ стала постоянной, то для всех смежных с ней временных вершин v_j вычислить $m+d(v_k,v_j)$ и сравнить с текущим расстоянием, которым помечена вершина v_j. Если полученная сумма меньше, то текущее расстояние заменить ею, а вторую координату заменить на v_k.

b. Найти минимум из расстояний, приписанных временным вершинам. Первую из таких вершин сделать постоянной.

2. Когда v_n станет постоянной вершиной, то расстояние, приписанное ей – это и есть длина кратчайшего пути. Сам путь (в обратном порядке) можно отследить, если пройти в обратную сторону – от вершины v_n к v_1 – по вторым координатам меток вершин.

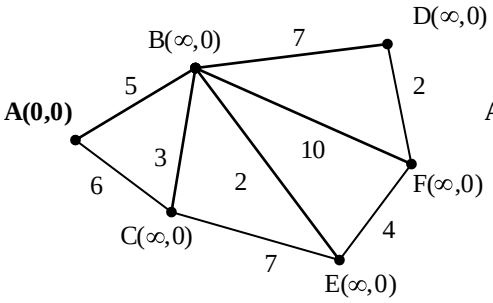


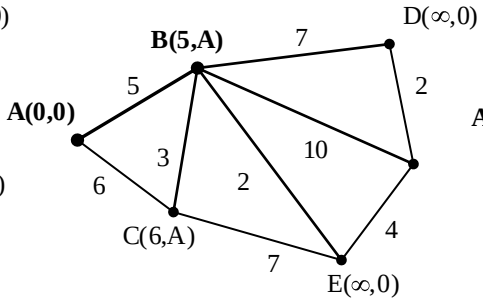
Найти кратчайший путь из вершны A в вершину F во взвешенном графе. Вершины снабжаем пометками, и в графе будут присутствовать метки (∞,0), пока не найден путь

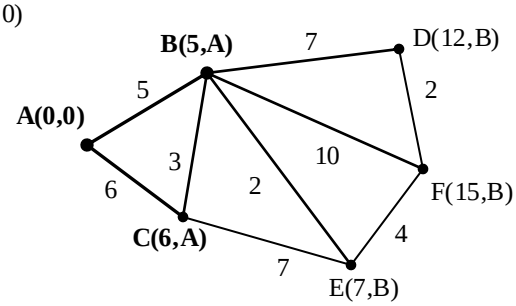


Вершины, которые будут становиться постоянными, выделяем жирным шрифтом.

F(∞,0)







а) б) в)

Вершина A стала постоянной (рис. а), с ней смежные B и C $\Rightarrow$ для них нашли расстояния (это 5 и 6), заменили метки на (5,A) и (6,A). Минимальное из расстояний 5 $\Rightarrow$ вершина B(5,A) становится постоянной (рис. б). Вычисляем расстояния от нее до смежных с ней C, E, F, D. Расстояние до C будет 5+3=8 $\Rightarrow$ оно больше текущего (6) $\Rightarrow$ метка вершины C не меняется. Все остальные расстояния меньше ∞ $\Rightarrow$ метки остальных вершин заменяются. Изо всех текущих меток расстояний min{12,15,7,6} = 6 $\Rightarrow$ вершину C(6,A) делаем постоянной (рис. в). Смежная с ней временная вершина E, расстояние до нее через C 6+7=13>7 $\Rightarrow$ изменений нет. Теперь из временных вершин min=7 $\Rightarrow$ вершина E(7,B) становится постоянной. 7+4=11<15 $\Rightarrow$ метка вершины F меняется на F(11,E) и она становится постоянной. Алгоритм завершен, расстояние от A до F равно 11, а кратчайший путь – A,B,E,F.

Тема: «Нахождение кратчайших путей в сети с отрицательными весами: Алгоритм Беллмана – Мура»

1.1.1 Вопросы лекции:

1. Нахождение кратчайших путей в сети с отрицательными весами.
2.Алгоритм Беллмана - Мура

1.1.2 Краткое содержание вопросов:

Наименование вопросов.

1. Нахождение кратчайших путей в сети с отрицательными весами.
2. Алгоритм Беллмана - Мура

Если веса – произвольные числа и граф G не содержит ориентированных циклов отрицательного веса, то минимальный путь может быть найден алгоритмом Беллмана – Мура, похожим на алгоритм Дейкстры. Этот алгоритм часто называют алгоритмом корректировки меток. Как и в алгоритме Дейкстры всем вершинам приписываются метки, однако здесь нет деления меток на временные и постоянные. Корректировка меток осуществляется по старому правилу (4.7.1), однако выполнение условия (4.7.2) теперь не гарантирует, что длина кратчайшего пути от s до x_j найдена, так как наличие в графе G дуг с отрицательными весами может уменьшить эту метку на последующих шагах. Поэтому в алгоритме Беллмана – Мура вместо процедуры превращения временной метки в постоянную формируется очередь вершин, которые нужно просмотреть

для анализа возможности уменьшения по правилу (4.7.1) меток вершин, не находящихся в данный момент в очереди, но достижимых из нее за один шаг. В процессе работы алгоритма одна и та же вершина может несколько раз вставать в очередь, а затем покидать ее.

Алгоритм Беллмана – Мура состоит из двух этапов. На первом этапе находятся длины кратчайших путей от вершины s до всех остальных вершин графа. Этот этап заканчивается при отсутствии вершин в очереди.

Второй этап – построение кратчайшего пути от s до t совпадает с соответствующим этапом в алгоритме Дейкстры и выполняется по формуле (4.7.3). Опишем подробно все шаги алгоритма.

Этап 1. Нахождение длин кратчайших путей от вершины s до всех остальных вершин графа.

Шаг 1. Присвоение начальных значений. $d(x_j)=0,\ d(x_j)=\infty,\ x_j \in S,\ \tilde{x}=s,\ Q=\tilde{x}$ - множество вершин в очереди.

Шаг 2. Корректировка меток и очереди.

2а). Удаляем из очереди Q вершину, находящуюся в самом начале очереди.

2б). Для каждой вершины x_i, непосредственно достижимой из $\tilde{x}$, корректируем ее метку по формуле (4.7.1).

2бб). Если при этом $d_{нов.}(x_i) \geq d_{стар}(x_i)$, то переходим к подшагу 2бв), иначе продолжаем перебор вершин по шагу 2б).

2бв). Корректировка очереди. Если x_i не была ранее в очереди и не находится в ней в данный момент, то вершину x_i ставим в конец очереди. Если же x_i уже была когда-нибудь ранее в очереди или находится там в данный момент, то переставляем ее в начало очереди.

Шаг 3. Проверка на завершение первого этапа. Если $Q \neq \varnothing$, то возвращаемся к началу второго шага, если же $Q=\varnothing$, то первый этап закончен, то есть минимальные расстояния от s до всех вершин графа найдены.

1. 4 Лекция № 4 (2 часа).

Тема: «Построение остовного дерева графа (сети): задача об остове минимального веса. Алгоритмы Краскала и Прима. Компьютерные технологии реализации алгоритма Краскал»

1.4.1 Вопросы лекции:

1. Построение остовного дерева (леса) графа.
2. алгоритмы Краскала и Прима; задача об остове экстремального веса.

1.4.2 Краткое содержание вопросов:

1. Построение остовного дерева (леса) графа.
2. алгоритмы Краскала и Прима; задача об остове экстремального веса.

1. Наименование вопроса № 1.

Деревья являются простейшим классом графов. Для них выполняются многие свойства, которые не всегда выполняются для обычных графов. Кроме того, деревья широко применяются в программировании при различного рода обработке данных, в частности, в алгоритмах сортировки, кодирования и т.п. Подробно алгоритмы работы с деревьями будут рассматриваться позднее в других курсах, а сейчас только краткое знакомство.

Дерево – это связный граф без циклов. Несколько деревьев (или несвязный граф без циклов) составляют *лес*. Таким образом, дерево является компонентой связности леса.

3.2. Вопрос №2: «Свойства деревьев»

Пусть $G = (S, U)$ и $|S| = n,\ |U| = m.$ Тогда справедлива эквивалентность следующих утверждений:

1). G - дерево;

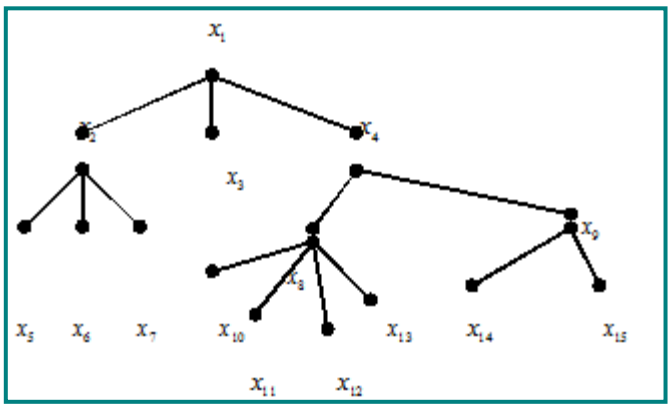
2). G - связный граф и $m = n - 1;$

3). G - ациклический граф и $m = n - 1;$

4). любые две несовпадающие вершины графа соединяет единственная простая цепь;

5). G - ациклический граф, обладающий тем свойством, что если какую- либо пару его несмежных вершин соединить ребром, то полученный граф будет содержать ровно один цикл.

Ориентированный граф называется ориентированным деревом (ордеревом), если:
1). существует ровно одна вершина $x_1 \in S$, называемая корнем, кото-рая не имеет предшествующих вершин, то есть $P(x_1) = 0;$
2). любой вершине $x_j \neq x_1$ в графе G непосредственно предшествует ровно одна вершина, то есть $P(x_j) = 1.$



Неориентированное дерево можно превратить в ориентированное, выбрав в качестве корня произвольную вершину. Пусть $G = (S, U)$. Граф $G' = (S', U')$ называется подграфом графа G, если $S' \subset S$ и $U' \subset U$. Подграф G' графа G называется остовным подграфом, если $S' = S$. Подграф G' графа графа G называется остовным поддеревом (остовом, каркасом), если $S' = S$ и G - дерево.

Теорема Кэли*. *Число различных деревьев, которые можно построить на* n *различных вершинах, равно* $t_n = n^{n-2}$.

В этой формуле подсчитывается число всех деревьев с данными n вершинами. Многие из этих деревьев изоморфны, и возникает вопрос о числе не изоморфных деревьев среди них. Это более трудная задача, она решается для каждого конкретного случая по алгоритму теории Пойа.

Вернемся к произвольным графам. Матрицей Кирхгофа** графа G называется матрица $B_{n\times n}$, $n = |S|$, если $b_{ij} = \begin{cases} -1, & x_i \text{ и } x_j \text{ смежны,} \\ 0, & x_i \text{ и } x_j \text{ несмежны,} \\ P(x_i), & i = j. \end{cases}$ Сумма элементов в каждой строке и каждом столбце этой матрицы равна нулю, то есть $\sum_{i=1}^{n} b_{ij} = 0,\ j = \overline{1,n}$, $\sum_{j=1}^{n} b_{ij} = 0,\ i = \overline{1,n}$.

Кроме того, из этого следует, что алгебраические дополнения всех элементов матрицы B равны между собой. Матрица Кирхгофа используется для подсчета числа остовов в графе.

* Артур Кэли(Кэйли) (1821-1895 г.г.) - английский математик.

** Густав Роберт Кирхгоф (1824-1887 г.г.) - немецкий физик

Теорема Кирхгофа. *Число остовных деревьев в связном графе* G *порядка* $n \geq 2$ *равно алгебраическому дополнению любого элемента матрицы Кирхгофа* $B(G)$.

3.3 Вопрос №3: «Задача об остове минимального веса»

Пусть $G = (S, U)$ - связная сеть. В приложениях часто возникает задача о построении остова графа G, имеющего наименьший вес. Пусть, например, $G = (S, U, \Omega)$ служит моделью железнодорожной сети, соединяющей пункты $x_1, x_2, \ldots, x_n \in S$, а $\omega(x_i, x_j)$ - расстояние между пунктами x_i и x_j. Требуется проложить сеть телеграфных линий вдоль линий железнодорожной сети так, чтобы все пункты $x_1, x_2, \ldots, x_n$ были связаны между собой телеграфной сетью и общая протяженность линий телеграфной сети была наименьшей.

Известно несколько алгоритмов построения кратчайшего остовного дерева. Рассмотрим алгоритм Прима*, представляющий собой итерационную процедуру, состоящую из двух шагов и выполняющуюся $n-1$ раз на графе G с n вершинами.

Пусть $S' \subset S$, $S'' \subset S$ и $S = S' \cup S''$, $S' \cap S'' = \varnothing$, то есть S' и S'' - разбиение множества узлов сети G на два непересекающихся подмножества. Определим пошаговое расстояние между множествами S' и S'' следующим образом:

$$d(S', S'') = \min\left\{ \omega(x_i, x_j) \,/\, x_i \in S',\ x_j \in S'' \right\},$$

где (x_i, x_j) - дуга, соединяющая вершины x_i и x_j.

В алгоритме Прима остовное дерево строится в результате последовательного расширения исходного поддерева. На каждой итерации число вершин и ребер поддерева увеличивается на единицу. Основные шаги алгоритма таковы.

Шаг 1. (Присвоение начальных значений).

Полагают $S' = \{x_1\}$, где x_1 - произвольная вершина, $S'' = S / S'$, $U' = \varnothing$.

Шаг 2. (Обновление данных).

Находим ребро (x_i, x_j) такое, что $x_i \in S', x_j \in S''$ и

$$\omega(x_i, x_j) = \min\ \omega(x_i, x_j) : x_i \in S', x_j \in S'' .$$

Полагают $S' = S' \cup \{x_j\}$, $S'' = S / S'$, $U' = U' \cup (x_i, x_j)$.

Шаг 3. (Проверка на завершение).

Если $S' = S$, то $G' = (S', U')$ - искомый остов. В противном случае переходят ко второму шагу.

Минимальный остов, алгоритм Крускала

Эта задача возникает при проектировании линий электропередач, трубопроводов, дорог и т.д., когда требуется заданные центры (вершины графа) соединить некоторой системой каналов связи таким образом, чтобы любые два центра были связаны непосредственно или через другие каналы, и чтобы общая длина (стоимость) каналов связи была минимальной.

Для постановки и решения задачи дадим некоторые определения.

Вес остовного дерева взвешенного графа равен сумме весов, приписанных его ребрам. *Минимальным остовным деревом* называется остовное дерево графа с минимальным весом.

Математическая формулировка задачи: во взвешенном связном графе G(V,E) найти минимальное остовное дерево T(V,E').

Рассмотрим алгоритм Крускала (Краскала?) решения этой задачи. Идея алгоритма состоит в том, чтобы постепенно формировать дерево T(V,E'), выбирая ребра с наименьшим весом так, чтобы не возникало циклов.

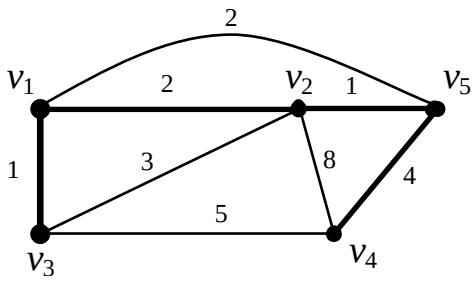
Первоначально множество E' пустое, V – множество вершин графа, T пустое. Два следующих действия выполняются до тех пор, пока это возможно.

1). Выбрать ребро минимального веса в исходном графе G, не принадлежащее множеству E', так, что его добавление в E' не создает цикла в дереве T.

2). Добавить это ребро во множество ребер E'.

Пример. Найти минимальный остов во взвешенном графе.

Построение остова начнем с ребра (v_1, v_3). Порядок присоединения ребер к остову: (v_1, v_3), (v_2, v_5), (v_1, v_2) (или (v_1, v_5)), (v_4, v_5). Вес остова W=1+2+1+4=8.



1. 5 Лекция № 5 (2 часа).

Тема: «Потоки в сетях, задача о максимальном потоке и минимальном разрезе. Теорема Форда – Фалкерсона. Компьютерные технологии реализации алгоритма Форда-Фалкерсона»

1.5.1 Вопросы лекции:

1. Потоки в сетях, задача о максимальном потоке и минимальном разрезе.
2. Теорема Форда - Фалкерсона

1.5.2 Краткое содержание вопросов:

1. Потоки в сетях, задача о максимальном потоке и минимальном разрезе.
2. Теорема Форда - Фалкерсона

1. Наименование вопроса № 1.

Потоки в сетях. Функциональное назначение большинства физически реализованных сетей состоит в том, что они служат носителями систем потоков, то есть систем, в которых некоторые объекты текут, движутся или транспортируются по системе каналов (дуг сети) ограниченной пропускной способности. Примерами могут служить поток автомобильного транспорта по сети автодорог, поток грузов по участку железнодорожной сети, поток воды в городской сети водоснабжения, поток электрического тока в электросети, поток телефонных или телеграфных сообщений по каналам связи, поток программ в вы-

числительной сети. Ограниченная пропускная способность означает, что интенсивность перемещения соответствующих предметов по каналу ограничена сверху определенной величиной.

Наиболее часто в сети решается задача о максимальном потоке и минимальном разрезе. При этом граф $G=(S,U)$ должен удовлетворять следующим условиям:

1). G - связный граф без петель;

2). существует ровно одна вершина, не имеющая предшествующих; эта вершина называется источником и обозначается s;

3). существует ровно одна вершина, не имеющая последующих; эта вершина называется стоком и обозначается t;

4). каждой дуге $(x_i,x_j)\in U$ поставлено в соответствие неотрицательное число $c(x_i,x_j)$, называемое пропускной способностью дуги.

Функция $\varphi(x_i,x_j)$, определенная на множестве дуг сети $G=(S,U,\Omega)$, называется потоком, если $0\le\varphi(x_i,x_j)\le c(x_i,x_j)\ \forall(x_i,x_j)\in U$ и $\sum_{x_j\in S_{np}(x_i)}\varphi(x_i,x_j)=\sum_{x_j\in S_{сл}(x_i)}\varphi(x_i,x_j)$ для любой вершины $x_i\in S$ и $x_i\notin\{s,t\}$.

Последнее условие называется условием сохранения потока, в промежуточных вершинах потоки не создаются и не исчезают.

Величина $\Delta(x_i,x_j)=c(x_i,x_j)-\varphi(x_i,x_j)$ называется остаточной пропускной способностью дуги (x_i,x_j). Если $\varphi(x_i,x_j)=c(x_i,x_j)$, то дуга называется насыщенной.

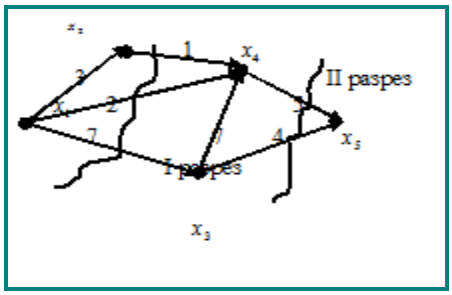
Максимальный поток определяется с помощью одного из основных понятий теории сетей – разреза. Разрез может быть определен как множество дуг, исключение которых из сети отделило бы некоторое множество узлов от остальной сети. Предположим, что множество вершин сети S разбито на два непустых непересекающихся подмножества $S=S'\cup S''$ и $S'\cap S''=\varnothing$.

Множество дуг, начала которых лежат в S', а концы в S'', называется ориентированным разрезом и обозначается $(S'\to S'')$. Следовательно,

$$(S'\to S'')=\left\{(x_i,x_j)\ \middle/\ x_i\in S',\ x_j\in S''\right\}.$$

Пропускной способностью или величиной разреза $(S'\to S'')$ называется сумма пропускных способностей входящих в него дуг, то есть

$$c(S'\to S'')=\sum_{x_i\in S',\,x_j\in S''}c(x_i,x_j).$$



На рисунке слева изображена сеть, на которой около каждого ребра указана его пропускная способность. Произведены два разреза: I и II. При разрезе I вершины ока-

зались разбиты на подмножества $S' = \{x_1, x_2\}$ и $S'' = \{x_3, x_4, x_5\}$, а ребрами, образующими разрез стали ребра $(x_1, x_3), (x_1, x_4), (x_2, x_4)$. При разрезе II $S' = \{x_1, x_2, x_3, x_4\}$, а $S'' = \{x_5\}$, разрез образуют ребра $(x_3, x_5), (x_4, x_5)$.

Теорема Форда-Фалкерсона. Для любой сети с одним источником и одним стоком величина максимального потока в сети от источника к стоку равна величине минимального разреза.

Поток минимальной стоимости.

Рассмотрим задачу определения потока, заданной величины θ от s к t в сети $G = (S, U, \Omega, D)$, в которой каждая дуга (x_i, x_j) характеризуется пропускной способностью $c(x_i, x_j) \in \Omega$ и неотрицательной стоимостью $d(x_i, x_j) \in D$ пересылки единичного потока из x_i в x_j вдоль дуги (x_i, x_j).

Если $\theta > \varphi_{\max}$, где $\varphi_{\max}$- величина максимального потока в сети G от s к t, то решения нет. Если же $\theta \le \varphi_{\max}$, то может быть определено несколько различных потоков величины θ от s к t. Математическая модель задачи выглядит следующим образом. Минимизировать

$$\sum_{(x_i, x_j) \in U} d(x_i, x_j) \cdot \varphi(x_i, x_j),$$

где $\varphi(x_i, x_j)$ - поток по дуге (x_i, x_j) при ограничениях

1). $0 \le \varphi(x_i, x_j) \le c(x_i, x_j) \; \forall (x_i, x_j) \in U,$

2). для начальной вершины $s \in S$ $\sum_{x_i \in S} \varphi(s, x_i) - \sum_{x_i \in S} \varphi(x_i, s) = \theta$ - уравнение истока,

3). для конечной вершины $t \in S$ $\sum_{x_i \in S} \varphi(t, x_i) - \sum_{x_i \in S} \varphi(x_i, t) = -\theta$ - уравнение стока,

4). для любой другой вершины $x_j \ne s, t$ $\sum_{x_i \in S} \varphi(x_j, x_i) - \sum_{x_i \in S} \varphi(x_i, x_j) = 0$ - условие сохранения потока.

1. 6 Лекция № 6 (2 часа).

Тема: «Построение остовного дерева графа (сети): задача об остове минимального веса. Алгоритмы Краскала и Прима. Компьютерные технологии реализации алгоритма Краскал»

1.6.1 Вопросы лекции:

1. Построение остовного дерева (леса) графа.
2. алгоритмы Краскала и Прима; задача об остове экстремального веса.

1.6.2 Краткое содержание вопросов:

1. Построение остовного дерева (леса) графа.
2. алгоритмы Краскала и Прима; задача об остове экстремального веса.

1. Наименование вопроса № 1.

Деревья являются простейшим классом графов. Для них выполняются многие свойства, которые не всегда выполняются для обычных графов. Кроме того, деревья широко приме-

няются в программировании при различного рода обработке данных, в частности, в алгоритмах сортировки, кодирования и т.п. Подробно алгоритмы работы с деревьями будут рассматриваться позднее в других курсах, а сейчас только краткое знакомство.

Дерево – это связный граф без циклов. Несколько деревьев (или несвязный граф без циклов) составляют *лес*. Таким образом, дерево является компонентой связности леса.

3.2. Вопрос №2: «Свойства деревьев»

Пусть $G = (S, U)$ и $|S| = n$, $|U| = m$. Тогда справедлива эквивалентность следующих утверждений:

1). G - дерево;

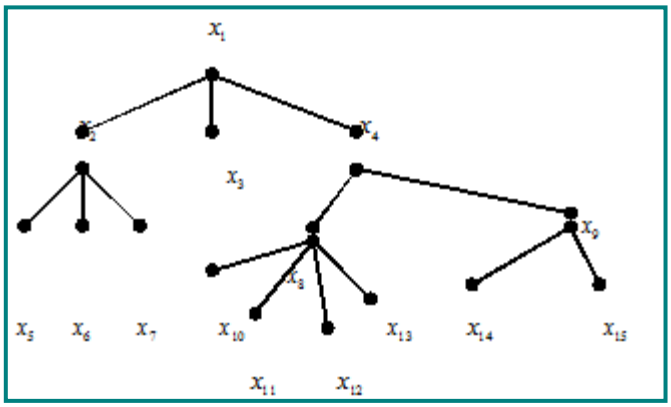
2). G - связный граф и $m = n - 1$;

3). G - ациклический граф и $m = n - 1$;

4). любые две несовпадающие вершины графа соединяет единственная простая цепь;

5). G - ациклический граф, обладающий тем свойством, что если какую- либо пару его несмежных вершин соединить ребром, то полученный граф будет содержать ровно один цикл.

Ориентированный граф называется ориентированным деревом (ордеревом), если:
1). существует ровно одна вершина $x_1 \in S$, называемая корнем, кото-рая не имеет предшествующих вершин, то есть $P(x_1) = 0$;
2). любой вершине $x_j \neq x_1$ в графе G непосредственно предшествует ровно одна вершина, то есть $P(x_j) = 1$.



Неориентированное дерево можно превратить в ориентированное, выбрав в качестве корня произвольную вершину. Пусть $G = (S, U)$. Граф $G' = (S', U')$ называется подграфом графа G, если $S' \subset S$ и $U' \subset U$. Подграф G' графа G называется остовным подграфом, если $S' = S$. Подграф G' графа графа G называется остовным поддеревом (остовом, каркасом), если $S' = S$ и G - дерево.

Теорема Кэли*. *Число различных деревьев, которые можно построить на* n *различных вершинах, равно* $t_n = n^{n-2}$.

В этой формуле подсчитывается число всех деревьев с данными n вершинами. Многие из этих деревьев изоморфны, и возникает вопрос о числе не изоморфных деревьев среди них. Это более трудная задача, она решается для каждого конкретного случая по алгоритму теории Пойа.

Вернемся к произвольным графам. Матрицей Кирхгофа** графа G называется матрица $B_{n\times n}$, $n=|S|$, если $b_{ij}=\begin{cases}-1, & x_i \text{ и } x_j \text{ смежны},\\ 0, & x_i \text{ и } x_j \text{ несмежны},\\ P(x_i), & i=j.\end{cases}$ Сумма элементов в каждой строке и каждом столбце этой матрицы равна нулю, то есть $\sum_{i=1}^{n} b_{ij}=0,\ j=\overline{1,n}$, $\sum_{j=1}^{n} b_{ij}=0,\ i=\overline{1,n}$.

Кроме того, из этого следует, что алгебраические дополнения всех элементов матрицы B равны между собой. Матрица Кирхгофа используется для подсчета числа остовов в графе.

* Артур Кэли(Кэйли) (1821-1895 г.г.) - английский математик.

** Густав Роберт Кирхгоф (1824-1887 г.г.) - немецкий физик

Теорема Кирхгофа. *Число остовных деревьев в связном графе G порядка $n\geq 2$ равно алгебраическому дополнению любого элемента матрицы Кирхгофа $B(G)$.*

3.3 Вопрос №3: «Задача об остове минимального веса»

Пусть $G=(S,U)$ - связная сеть. В приложениях часто возникает задача о построении остова графа G, имеющего наименьший вес. Пусть, например, $G=(S,U,\Omega)$ служит моделью железнодорожной сети, соединяющей пункты $x_1,x_2,\ldots,x_n\in S$, а $\omega(x_i,x_j)$ - расстояние между пунктами x_i и x_j. Требуется проложить сеть телеграфных линий вдоль линий железнодорожной сети так, чтобы все пункты $x_1,x_2,\ldots,x_n$ были связаны между собой телеграфной сетью и общая протяженность линий телеграфной сети была наименьшей.

Известно несколько алгоритмов построения кратчайшего остовного дерева. Рассмотрим алгоритм Прима*, представляющий собой итерационную процедуру, состоящую из двух шагов и выполняющуюся $n-1$ раз на графе G с n вершинами.

Пусть $S'\subset S,\ S''\subset S$ и $S=S'\cup S'',\ S'\cap S''=\varnothing$, то есть S' и S'' - разбиение множества узлов сети G на два непересекающихся подмножества. Определим пошаговое расстояние между множествами S' и S'' следующим образом:

$$d(S',S'')=\min\left\{\omega(x_i,x_j)\ /\ x_i\in S',\ x_j\in S''\right\},$$

где (x_i,x_j) - дуга, соединяющая вершины x_i и x_j.

В алгоритме Прима остовное дерево строится в результате последовательного расширения исходного поддерева. На каждой итерации число вершин и ребер поддерева увеличивается на единицу. Основные шаги алгоритма таковы.

Шаг 1. (Присвоение начальных значений).

Полагают $S^{/} = \{x_1\}$, где x_1 - произвольная вершина, $S^{//} = S / S^{/}$, $U^{/} = \varnothing$.

Шаг 2. (Обновление данных).

Находим ребро (x_i, x_j) такое, что $x_i \in S^{/}, x_j \in S^{//}$ и

$$\omega(x_i, x_j) = \min\{\omega(x_i, x_j) : x_i \in S^{/}, x_j \in S^{//}\}.$$

Полагают $S^{/} = S^{/} \cup \{x_j\}$, $S^{//} = S / S^{/}$, $U^{/} = U^{/} \cup \{(x_i, x_j)\}$.

Шаг 3. (Проверка на завершение).

Если $S^{/} = S$, то $G^{/} = (S^{/}, U^{/})$ - искомый остов. В противном случае переходят ко второму шагу.

1. 7 Лекция № 7 (2 часа).

Тема: «Алгоритмы обхода и поиска в графе: поиск в глубину и в ширину. Эйлеровы циклы в графах»

1.7.1 Вопросы лекции:

1. Задачи глобального анализа графа.
2. Алгоритмы обхода и поиска
3. Эйлеровы циклы в графах

1.7.2 Краткое содержание вопросов:

1. Задачи глобального анализа графа.
2. Алгоритмы обхода и поиска
3. Эйлеровы циклы в графах

1. Наименование вопроса № 1.

Маршруты, цепи, циклы. Маршрутом *от вершины* u *к вершине* v *или (*u,v*)*-маршрутом *в графе* G *называется всякая последовательность вида* $u = v_0, e_1, v_1, e_2, \ldots, e_n, v_n = v$, *в которой любые два соседних элемента инцидентны, т.е.* e_k *– ребро, соединяющее вершины* v_{k-1} *и* v_k, k = *1,2,…,*n.

Это определение подходит также для псевдо-, мульти- и орграфов. В случае орграфа v_{k-1} *– начало* ребра e_k , а v_k – его *конец*. При этом вершину *u* называют *началом маршрута*, а вершину *v* – его *концом*. В маршруте некоторые вершины и ребра могут совпадать. Если *u* = *v*, то маршрут *замкнут*, а иначе *открыт*. Для «обычного» графа маршрут можно задавать только последовательностью вершин $v_0, v_1, \ldots, v_n$ или ребер $e_1, e_2, \ldots, e_n$.

Маршрут называется *цепью*, если в нем нет совпадающих ребер, и *простой цепью* – если дополнительно нет совпадающих вершин, кроме, может быть, начала и конца цепи. Про цепь $u = v_0, v_1, \ldots, v_n = v$ говорят, что она *соединяет* вершины *u* и *v* и обозначают ⟨*u*,*v*⟩.

Очевидно, что если есть цепь, соединяющая вершины *u* и *v*, то есть и простая цепь, соединяющая эти вершины.

Замкнутая цепь называется *циклом*; замкнутая простая цепь – *простым циклом*. Число циклов в графе *G* обозначается *z(G)*. Граф без циклов называется *ациклическим*. Для орграфов цепь называется *путем*, а цикл – *контуром*.

Число ребер в маршруте M (возможно, с повторениями) называется его *длиной*, обозначается |M|.

Расстоянием между вершинами u и *v* (обозначается $d(u,v)$) называется длина кратчайшей цепи $\langle u,v\rangle$, а сама кратчайшая цепь называется *геодезической*. Если не существует цепи, соединяющей вершины *u* и *v*, то по определению $d(u,v) = +\infty$.

Диаметром графа *G* (обозначается *D(G)*) называется длина длиннейшей геодезической.

Максимальным удалением в графе *G от вершины v* называется $r(v) = max\ d(v,v'), \forall\ v' \in V$. Вершина *v* графа *G* является его *центром*, если максимальное удаление от нее до всех вершин принимает наименьшее значение.

Множество вершин, находящихся на одинаковом расстоянии *n* от вершины *v*, называется *ярусом* (обозначается $D(v,n)$): $D(v,n) = \{u \in V \mid d(v,u) = n\}$.

Граф, любая из вершин которого является его центром – максимальное удаление до всех вершин от любой =

Связность. *Если две вершины u и v в графе можно соединить цепью, то такие вершины связаны. Граф называется связным, если в нем связаны все вершины.*

Легко видеть, что отношение связности на множестве вершин является отношением эквивалентности. Данное отношение разбивает множество вершин графа на классы, объединяющие вершины, связанные друг с другом. Такие классы называются *компонентами связности*; число компонент связности обозначается *k(G)*.

Граф *G* является связным тогда и только тогда, когда он имеет одну компоненту связности: $k(G) = 1$. Если $k(G) > 1$, то это *несвязный* граф. Граф, состоящий только из изолированных вершин (в котором k(G)=|V|, r(G)=0), называется *вполне несвязным*.

Вершина графа, удаление которой увеличивает число компонент связности, называется *разделяющей* или *точкой сочленения*.

Ориентированный граф *G(V,E)* является *слабо связным* (*слабым*), если симметричное замыкание множества *E* определяет связный граф (т.е., если после замены всех дуг графа *G* ребрами полученный граф будет связным). Ориентированный граф является *сильно связным* (*сильным*), если для любой пары вершин $u,v \in V$ существует ориентированный путь из *u* в *v* (т.е. из любой вершины графа достижимы все его остальные вершины). Если для любой пары вершин по крайней мере одна достижима из другой, то такой граф является *односторонне связным*, или *односторонним*. Граф, состоящий из одной вершины, по определению считается сильно связным.

Множества вершин связных компонент образуют разбиение множества вершин графа.

Обходы графов. *Обход графа – это некоторое систематическое перечисление его вершин (ребер). Среди всех обходов наиболее известны поиск в глубину и в ширину. Алгоритмы такого поиска лежат в основе многих алгоритмов на графах.*

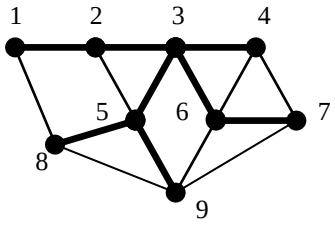
При поиске используется некоторая структура данных T, в которую помещаются вершины графа. Для обозначения пройденных вершин заводят дополнительный массив пометок этих вершин.

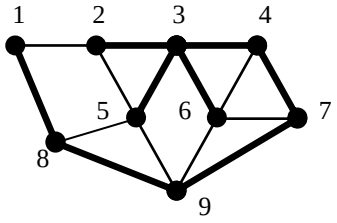
Поиск основывается на следующих действиях.

1.Сначала все вершины считаются неотмеченными.
2.Выбирается любая вершина (нач. поиска), заносится в структуру данных Т и помечается.
3.Следующие действия выполняются в цикле до тех пор, пока структура Т не станет пустой: из структуры данных Т выбирается вершина *u*; она выдается в качестве очередной пройденной вершины; перебираются все вершины из Г(u), и все те, которые не помечены, тоже заносятся в структуру Т и помечаются.

Если Т-это стек (LIFO), то обход называется поиском *в глубину* (т.е. первым делом из структуры Т извлекается вершина, попавшая туда последней). Если Т – это очередь (FIFO), то обход называется поиском *в ширину*. При поиске в глубину находятся более длинные пути.

Если граф G связен и конечен, то поиск в ширину или поиск в глубину обойдет ***все вершины графа по одному разу.***





Рассмотрим алгоритмы поиска в глубину и в ширину на примере. В качестве стартовой возьмем вершину с номером 3. Тогда поиск в ширину даст последовательность вершин: 3→T. T={3}⇒ **3**: {2,5,6,4}→T; T={2,5,6,4}⇒ **2**: {1}→T; T={5,6,4,1}⇒ **5**: {8,9}→T; T={6,4,1,8,9}⇒ **6**: {7}→T; T={4,1,8,9,7}. Окружения всех этих вершин уже отмечены ⇒ они будут выданы по порядку. Итак, выполнен обход всех вершин графа в следующем порядке: **3,2,5,6,4,1,8,9,7**. При поиске в глубину начало такое же: 3→T. T={3} ⇒ **3**: {2,5,6,4}→T; T={2,5,6,4}⇒ **4**: {7}→T; T={2,5,6,7}⇒ **7**: {9}→T; T={2,5,6,9} ⇒ **9**: {8}→T; T={2,5,6,8}⇒ **8**: {1}→T; T={2,5,6,1}. Оставшиеся вершины выдаются по порядку. В итоге последовательность вершин: **3,4,7,9,8,1,6,5,2.**

Любой из рассмотренных обходов позволяет построить остовное дерево исходного графа с корнем в исходной вершине (связный подграф без циклов).

1. 8 Лекция № 8 (2 часа).

Тема: «Поиск расстояния между всеми парами вершин. Алгоритм Уоршалла – Флойда»

1.8.1 Вопросы лекции:

1. Поиск расстояния между всеми парами вершин.
2. Алгоритм Уоршалла – Флойда

1.8.2 Краткое содержание вопросов:

1. Поиск расстояния между всеми парами вершин.
2. Алгоритм Уоршалла – Флойда

1. Наименование вопроса № 1.

Кратчайшие пути. *Задача поиска кратчайшего пути (наиболее дешевого? короткого?) «от пункта А до пункта В» имеет массу практических приложений и различные алгоритмы решения. Математическая постановка задачи имеет следующий вид.*

Рассматривается взвешенный граф (орграф) $G(V,E)$, ребрам (дугам) которого сопоставлены веса, обозначающие длину (или стоимость) пути из одного конца ребра в другой. Если из вершины v_i нет ребра (дуги) в вершину v_j, то вес ребра (v_i,v_j) считается равным ∞. Для ребер, являющихся петлями (диагональ матрицы смежности), их веса считаются равными 0. Все компоненты матрицы – веса ребер, соединяющих соответствующие вершины. Требуется определить кратчайший путь из одной вершины в другую.

Наиболее широко известны два алгоритма поиска кратчайших путей. Алгоритм Дейкстры находит кратчайшее расстояние от одной фиксированной вершины до другой и указывает сам путь, длина которого равна этому расстоянию. Алгоритм Флойда-Уоршалла позволяет найти кратчайшие расстояния между всеми парами вершин графа.

Алгоритм Флойда-Уоршалла **находит** кратчайшие расстояния между всеми парами вершин графа

Идея: Строится специальная матрица D, элементы которой – кратчайшие пути между всевозможными парами вершин графа G. При определении кратчайшего пути выбирается минимум из «прямого» расстояния между смежными вершинами v_i и v_j и суммарного расстояния при проходе через дополнительную вершину. Затем – более длинные пути и т.д.

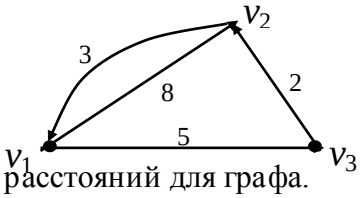
Обозначим через $d_{ij}^{(m)}$ длину кратчайшего пути из v_i в v_j с промежуточными вершинами во множестве $\{v_1,\ldots,v_m\}$. Алгоритм использует три правила:

1) $d_{ij}^{(0)} = a_{ij}$ – вес дуги, соединяющий вершины v_i и v_j (т.е. первоначально матрица D – это исходная матрица весов).

2) $d_{ij}^{(m+1)} = \min\left(d_{ij}^{(m)}, d_{i,m+1}^{(m)} + d_{m+1,j}^{(m)}\right)$.

3) Длина кратчайшего пути из вершины v_i в вершину v_j: $d(v_i, v_j) = d_{ij}^{(n)}$.

Алгоритм строит матрицу за n шагов, т.е. строится матрица $D^{(1)}$, …, $D^{(n)}=D$.



Найдем матрицу кратчайших расстояний для графа.

$$D^{(0)} = \begin{matrix} v_1 & v_2 & v_3 \end{matrix}\begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & \infty \\ \infty & 2 & 0 \end{pmatrix}\begin{matrix} v_1 \\ v_2 \\ v_3 \end{matrix}$$

Элементы матрицы $D^{(1)}$ находим по правилу $d_{ij}^{(1)} = \min\left(d_{ij}^{(0)}, d_{i1}^{(0)} + d_{1j}^{(0)}\right)$. Первая строка и первый столбец не меняются. Получаем $D^{(1)} = \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ \infty & 2 & 0 \end{pmatrix}$. Элементы матрицы $D^{(2)}$

находим по правилу $d_{ij}^{(2)} = \min\left(d_{ij}^{(1)}, d_{i2}^{(1)} + d_{2j}^{(1)}\right)$. Без изменений второй столбец и вторая строка. $D^{(2)} = \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{pmatrix}$.

Элементы матрицы $D^{(3)}$ находим по правилу $d_{ij}^{(3)} = \min\left(d_{ij}^{(2)}, d_{i3}^{(2)} + d_{3j}^{(2)}\right)$.

$D^{(3)} = \begin{pmatrix} 0 & 7 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{pmatrix}$, $D = D^{(3)}$. Каждый из элементов (i,j) матрицы D равен наименьшему расстоянию между вершинами v_i и v_j.

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

2.1 Лабораторная работа № 1 (2 часа).

Тема: «Нахождение экстремальных путей в сети: алгоритм Дейкстры»

2.1.1 Цель работы: сформировать умения и навыки решения оптимизационных задач (о поиске кратчайших путей между двумя заданными вершинами) на графах и сетях.

2.1.2 Задачи работы: изучить алгоритм Дейкстры нахождения кратчайшего пути между двумя заданными вершинами в ориентированной сети.

2.1.3 Перечень приборов, материалов, используемых в лабораторной работе:

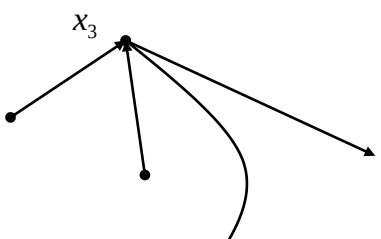
1. ПК.
2. Windows.
3. Open Office.

2.1.4 Описание (ход) работы: пример выполнения задания работы № 1.

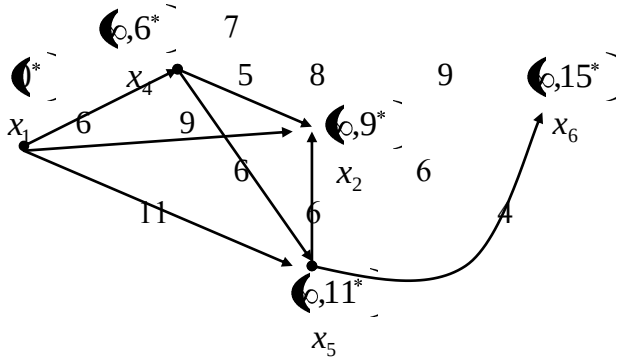
Задание. Задана весовая матрица сети G. Найти минимальный путь из вершины x_1 в вершину x_6 по алгоритму Дейкстры.

$$P = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 9 & \infty & 6 & 11 & \infty \\ \infty & - & 8 & \infty & \infty & \infty \\ \infty & \infty & - & \infty & 6 & 9 \\ \infty & 5 & 7 & - & 6 & \infty \\ \infty & 6 & \infty & \infty & - & 4 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix} \end{array}.$$

Решение. Изобразим теперь сам граф по данной матрице весов. $(6,13^*)$



Поскольку в данном графе есть цикл между вершинами x_2, x_3 и x_5, то верши-



ны графа нельзя упорядочить по алгоритму Фалкерсона. На рисунке графа временные и постоянные метки указаны над соответствующей вершиной. Итак, распишем подробно работу алгоритма Дейкстры по шагам.

Этап 1. Шаг1. Полагаем $d(x_1) = 0^*, \tilde{x} = x_1$, $d(x_2) = d(x_3) = d(x_4) = d(x_5) = d(x_6) = \infty$.

1-я итерация. Шаг 2. Множество вершин, непосредственно следующих за $\tilde{x} = x_1$ с временными метками $\tilde{S} = \{x_2, x_4, x_5\}$. Пересчитываем временные метки этих вершин $d(x_2) = \min\{\infty, 0^* + 9\} = 9$, $d(x_4) = \min\{\infty, 0^* + 6\} = 6$, $d(x_5) = \min\{\infty, 0^* + 11\} = 11$.

Шаг 3. Одна из временных меток превращается в постоянную $\min\{\infty, 6, 11, \infty\} = 6^* = d(x_4), \tilde{x} = x_4$.

Шаг 4. $\tilde{x} = x_4 \neq t = x_6$, происходит возвращение на второй шаг.

2-я итерация. Шаг 2. $\tilde{S} = \{x_2, x_3, x_5\}, d(x_2) = \min\{9, 6^* + 5\} = 9$, $d(x_3) = \min\{\infty, 6^* + 7\} = 13$, $d(x_5) = \min\{11, 6^* + 6\} = 11$.

Шаг 3. $\min\{d(x_2), d(x_3), d(x_5), d(x_6)\} = \min\{9, 13, 11, \infty\} = 9^* = d(x_2), \tilde{x} = x_2$.

Шаг 4. $x_2 \neq x_6$, возвращение на второй шаг.

3-я итерация. Шаг 2. $\tilde{S} = \{x_3\}, d(x_3) = \min\{13, 9^* + 8\} = 13$.

Шаг 3. $\min\{d(x_3), d(x_5), d(x_6)\} = \min\{13, 11, \infty\} = 11^* = d(x_5), \tilde{x} = x_5$.

Шаг 4. $x_5 \neq x_6$, возвращение на второй шаг.

4-я итерация. Шаг 2. $\tilde{S} = \{x_6\}, d(x_6) = \min\{\infty, 11^* + 4\} = 15$.

Шаг 3. . $\min\{d(x_3), d(x_6)\} = \min\{13, 15\} = 13^* = d(x_3), \tilde{x} = x_3$.

Шаг 4. $x_3 \neq x_6$, возвращение на второй шаг.

5-я итерация. Шаг 2. $\tilde{S} = \{x_6\}, d(x_6) = \min\{15, 13^* + 9\} = 15$.

Шаг 3. $\min\{d(x_6)\} = \min\{15\} = 15^*, \tilde{x} = x_6$.

Шаг 4. $x_6 = t = x_6$, конец первого этапа.

Этап 2. Шаг5. Составим множество вершин, непосредственно предшествующих $\tilde{x} = x_6$ с постоянными метками $\tilde{S} = \{x_3, x_5\}$. Проверим для этих двух вершин выполнение равенства (4.7.3).

$d(\tilde{x}) = 15 = 11^* + 4 = d(x_5) + \omega(x_5, x_6), d(\tilde{x}) = 15 \neq 13^* + 9 = d(x_3) + \omega(x_3, x_6)$. Включаем дугу (x_5, x_6) в кратчайший путь. $\tilde{x} = x_5$.

Шаг 6. $\tilde{x} \neq s = x_1$, возвращение на пятый шаг.

2-я итерация. Шаг 5. $\tilde{S} = \{x_1, x_4\}$.

$d(\tilde{x}) = 11 = 0^* + 11 = d(x_1) + \omega(x_1, x_5), d(\tilde{x}) = 11 \neq 6^* + 6 = d(x_4) + \omega(x_4, x_5)$. Включаем дугу (x_1, x_5) в кратчайший путь. $\tilde{x} = x_1$.

Шаг 6. $\tilde{x} = s = x_1$, завершение второго этапа.

Итак, кратчайший путь от вершины x_1 до вершины x_6 построен. Его длина (вес) равна 15, сам путь образует следующая последовательность дуг $\mu = (x_1, x_5), (x_5, x_6)$.

Задания лабораторной работы № 1

Задания. Задана весовая матрица сети G. Найти минимальный путь из вершины $s = x_1$ в вершину $t = x_6$ по алгоритму Дейкстры.

1)
$$P = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 5 & 10 & 13 & \infty & \infty \\ \infty & - & 8 & 9 & 13 & \infty \\ \infty & \infty & - & 5 & 3 & 6 \\ \infty & \infty & \infty & - & 8 & 10 \\ \infty & \infty & \infty & \infty & - & 9 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix} \end{array}.$$

2)
$$P = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 9 & \infty & 6 & 11 & \infty \\ \infty & - & 8 & \infty & \infty & \infty \\ \infty & \infty & - & \infty & 6 & 9 \\ \infty & 5 & 7 & - & 6 & \infty \\ \infty & 6 & \infty & \infty & - & 4 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix} \end{array}.$$

3)
$$P = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 9 & \infty & 6 & 11 & \infty \\ \infty & - & 8 & \infty & \infty & \infty \\ \infty & \infty & - & \infty & 6 & 9 \\ \infty & 5 & 7 & - & 6 & \infty \\ \infty & 6 & \infty & \infty & - & 4 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix} \end{array}.$$

4)
$$P = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 9 & \infty & 6 & 11 & \infty \\ \infty & - & 8 & \infty & \infty & \infty \\ \infty & \infty & - & \infty & 6 & 9 \\ \infty & 5 & 7 & - & 6 & \infty \\ \infty & 6 & \infty & \infty & - & 4 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix} \end{array}.$$

2.2 Лабораторная работа № 2 (2 часа).

Тема: «Нахождение экстремальных путей в сети с отрицательными весами: Алгоритм Беллмана – Мура»

2.2.1 Цель работы: сформировать умения и навыки решения оптимизационных задач (о поиске кратчайших путей между двумя заданными вершинами) на графах и сетях.

2.2.2 Задачи работы: изучить алгоритм Беллмана - Мура нахождения кратчайшего пути между двумя заданными вершинами в ориентированной сети.

2.2.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. ПК.
2. Windows.
3. Open Office.

2.2.4 Описание (ход) работы: пример выполнения задания работы № 2

Задания

Задана весовая матрица сети G.

а) построить по этой матрице сеть,

б) найти минимальный путь из вершины $s = x_1$ в вершину $t = x_6$ или $t = x_7$ по алгоритму Беллмана-Мура,

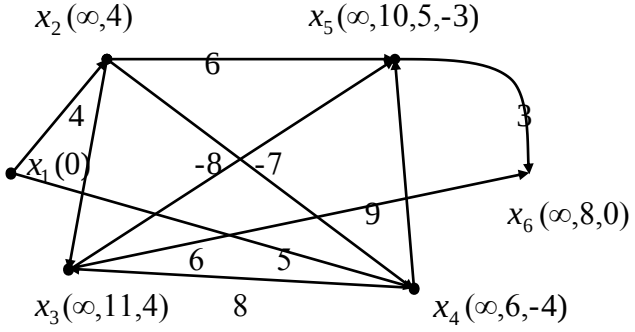
в) найти вес минимального пути.

$$1.\begin{pmatrix} - & 8 & 7 & 11 & \infty & \infty \\ \infty & - & -10 & 7 & \infty & \infty \\ \infty & \infty & - & \infty & 6 & \infty \\ \infty & \infty & 5 & - & \infty & 8 \\ \infty & \infty & \infty & -6 & - & 7 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix}. \qquad 2.\begin{pmatrix} - & 7 & \infty & -8 & \infty & \infty \\ \infty & - & 13 & -9 & 10 & \infty \\ \infty & \infty & - & 3 & -4 & -2 \\ \infty & \infty & \infty & - & 9 & \infty \\ \infty & \infty & \infty & \infty & - & 8 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix}.$$

Пример решения типовой задачи. Пусть граф G задан весовой матрицей W.

$$W = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 4 & \infty & 6 & \infty & \infty \\ \infty & - & 7 & -8 & 6 & \infty \\ \infty & \infty & - & \infty & -7 & 5 \\ \infty & \infty & 8 & - & 9 & \infty \\ \infty & \infty & \infty & \infty & - & 3 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix} \end{array}.$$

Изобразим эту сеть на рисунке и рассчитаем кратчайший путь от узла x_1 до узла x_6.



Этап 1. Шаг 1. $\tilde{x} = x_1$, $d(x_1) = 0$, $d(x_j) = \infty$, $j = \overline{2,6}$, $Q = \{x_1\}$.

Шаг 2. 2а). $\tilde{x} = x_1$, $Q = Q / \{x_1\} \neq \varnothing$.

2б). Пусть $\tilde{S}$ - множество вершин непосредственно достижимых из вершины $\tilde{x}$. $\tilde{S} = \{x_2, x_4\}$, $d(x_2) = \min\{\infty, 0+4\} = 4$.

2бб). $4<\infty$? Да.

2бв). $Q = \{x_2\}$, x_2 надо было поставить в конец очереди, но Q было пусто, поэтому конец очереди совпал с началом.

2б). $d(x_4) = \min\{6, 0+6\} = 6$.

2бб). $6 < \infty$? Да.

2бв). $Q = \{x_2, x_4\}$.

Шаг 3. $Q = \varnothing$? Нет. Переход на начало второго шага.

Вторая итерация. Шаг 2. 2а). $\tilde{x} = x_2$, $Q = Q / \{x_2\} = \{x_4\}$, $\tilde{S} = \{x_3, x_4, x_5\}$.

2б). $d(x_3) = \min\{\infty, 4+7\} = 11$.

2бб). $11<\infty$? Да.

2бв). $Q = \{x_4, x_3\}$.

2б). $d(x_4) = \min\{6, 4-8\} = -4$.

2бб). $-4<6$? Да.

2бв). $Q = \{x_4, x_3\}$. Вершину x_4 надо поставить в начало очереди, но она уже стоит там.

2б). $d(x_5) = \min\{\infty, 4+6\} = 10$.

2бб). $10<\infty$? Да.

2бв). $Q = \{x_4, x_3, x_5\}$.

Шаг 3. $Q = \varnothing$? Нет, переход на третью итерацию второго шага.

Третья итерация. Шаг 2. 2а). $\tilde{x} = x_4$, $Q = Q / \{x_4\} = \{x_3, x_5\}$, $\tilde{S} = \{x_3, x_5\}$.

2б). $d(x_3) = \min\{11, -4 + 8\} = 4$.

2бб). 4<11? Да.

2бв). $Q = \{x_3, x_5\}$. Вершину x_3 надо поставить в начало очереди, но она уже там.

2б). $d(x_5) = \min\{10, -4 + 9\} = 5$.

2бб). 5<10? Да.

2бв). $Q = \{x_5, x_3\}$. Вершину x_5 передвигаем из конца очереди в начало.

Шаг 3. $Q = \varnothing$? Нет, переход на четвертую итерацию второго шага.

Четвертая итерация. Шаг 2. 2а). $\tilde{x} = x_5$, $Q = Q / \{x_5\} = \{x_3\}$, $\tilde{S} = \{x_6\}$.

2б). $d(x_6) = \min\{\infty, 5 + 3\} = 8$.

2бб). 8<∞? Да.

2бв). $Q = \{x_3, x_6\}$.

Шаг 3. $Q = \varnothing$? Нет.

Пятая итерация. Шаг 2. 2а). $\tilde{x} = x_3$, $Q = Q / \{x_3\} = \{x_6\}$, $\tilde{S} = \{x_6, x_6\}$.

2б). $d(x_5) = \min\{5, 4 - 7\} = -3$.

2бб). -3<5? Да.

2бв). $Q = \{x_6, x_6\}$.

2б). $d(x_6) = \min\{8, 4 + 5\} = 8$.

2бб). 9<8? Нет.

Шаг 3. $Q = \varnothing$? Нет.

Шестая итерация. Шаг 2. 2а). $\tilde{x} = x_5$, $Q = Q / \{x_5\} = \{x_6\}$, $\tilde{S} = \{x_6\}$.

2б). $d(x_6) = \min\{8, -3 + 3\} = 0$.

2бб). 0<8? Да.

2бв). $Q = \{x_6\}$. Q содержало только вершину x_6 и она встала в начало очереди.

Шаг 3. $Q = \varnothing$? Нет.

Седьмая итерация. Шаг 2. 2а). $\tilde{x} = x_6$, $Q = Q / \{x_6\} = \varnothing$. $\tilde{S} = \varnothing$.

Шаг 3. $Q = \varnothing$. Конец первого этапа. Найдены минимальные расстояния до всех вершин от первой вершины. Эти расстояния таковы: $d(x_2) = 4$, $d(x_3) = 4$, $d(x_4) = -4$, $d(x_5) = -3$, $d(x_6) = 0$.

Второй этап. Шаг 4. Полагаем $\tilde{x} = x_6$. Пусть $\tilde{S}$ - множество вершин, непосредственно предшествующих $\tilde{x}$. $\tilde{S} = \{x_5\}$.

Первая итерация. $d(\tilde{x}) = d(x_6) = 0 \neq 4 + 5 = d(x_3) + \omega(x_3, x_6)$, $d(\tilde{x}) = d(x_6) = 0 = -3 + 3 = d(x_5) + \omega(x_5, x_6)$. Включаем дугу (x_5, x_6) в кратчайший путь. $\tilde{x} = x_5$. Возвращаемся на четвертый шаг.

Вторая итерация. $\tilde{x} = s$? Нет.

$\tilde{S} = \{x_2, x_3, x_4\}$. $d(\tilde{x}) = d(x_5) = -3 = 4 - 7 = d(x_3) + \omega(x_3, x_5)$,

$d(\tilde{x}) = d(x_5) = -3 \neq 4 + 6 = d(x_2) + \omega(x_2, x_5)$,

$d(\tilde{x}) = d(x_5) = -3 \neq -4 + 9 = d(x_4) + \omega(x_4, x_5)$. Включаем дугу (x_3, x_5) в кратчайший путь. $\tilde{x} = x_3$. Возвращаемся на четвертый шаг.

Третья итерация. $\tilde{x} = s$? Нет. $\tilde{S} = \{x_4\}$.

$d(\tilde{x}) = d(x_3) = 4 \neq 4 + 7 = d(x_2) + \omega(x_2, x_3)$,

$d(\tilde{x}) = d(x_3) = 4 = -4 + 8 = d(x_4) + \omega(x_3, x_4)$. Включаем дугу (x_3, x_4) в кратчайший путь. $\tilde{x} = x_4$. Возвращаемся на четвертый шаг.

Четвертая итерация. $\tilde{x} = s$? Нет. $\tilde{S} = \{x_2\}$.

$d(\tilde{x}) = d(x_4) = -4 \neq 0 + 6 = d(x_1) + \omega(x_1, x_4)$,

$d(\tilde{x}) = d(x_4) = -4 = 4 - 8 = d(x_2) + \omega(x_2, x_4)$. Включаем дугу (x_2, x_4) в кратчайший путь. $\tilde{x} = x_2$. Возвращаемся на четвертый шаг.

Пятая итерация. $\tilde{x} = s$? Нет. $\tilde{S} = \{x_1\}$.

$d(\tilde{x}) = d(x_2) = 4 = 0 + 4 = d(x_1) + \omega(x_1, x_2)$. Включаем дугу (x_1, x_2) в кратчайший путь. $\tilde{x} = x_1$. Возвращаемся на четвертый шаг.

Шестая итерация. $\tilde{x} = s$? Да. Задача закончена. Искомый кратчайший путь от вершины x_1 до вершины x_6 имеет нулевой вес и состоит из следующих дуг $(x_1, x_2) - (x_2, x_4) - (x_4, x_3) - (x_3, x_5) - (x_5, x_6)$.

2.5 Лабораторная работа № 3 (2 часа).

Тема: «Построение остовного дерева графа (сети): задача об остове минимального веса. Алгоритмы Краскала и Прима»

2.3.1 Цель работы: сформировать умения и навыки использования алгоритмов Краскала и Прима.

2.3.2 Задачи работы: изучить элементы алгоритмов Краскала и Прима, задачу об остове экстремального веса.

2.3.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. ПК.
2. Windows.
3. Open Office.

2.3.4 Описание (ход) работы: пример выполнения задания работы № 5.

Задания

Для графа(сети), заданного матрицей весов,

а)построить по этой матрице сеть(исходный граф),

б) построить остов наименьшего веса,

в) найти его вес.

г) изобразить остовный граф:

1) $\begin{pmatrix} - & 10 & \infty & 5 & \infty & \infty & 14 \\ 10 & - & 6 & 2 & 4 & 8 & \infty \\ \infty & 6 & - & 3 & 1 & 1 & \infty \\ 5 & 2 & 3 & - & 6 & \infty & 3 \\ \infty & 4 & 1 & 6 & - & 5 & \infty \\ \infty & 8 & 1 & \infty & 5 & - & 2 \\ 14 & \infty & \infty & 3 & \infty & 2 & - \end{pmatrix}$, 2) $\begin{pmatrix} - & 7 & 15 & 12 & \infty & 10 & \infty \\ 7 & - & 13 & 9 & \infty & \infty & 8 \\ 15 & 13 & - & 7 & 15 & 7 & \infty \\ 12 & 9 & 7 & - & 9 & \infty & 11 \\ \infty & \infty & 15 & 9 & - & 10 & \infty \\ 10 & \infty & 7 & \infty & 10 & - & 12 \\ \infty & 8 & \infty & 11 & \infty & 12 & - \end{pmatrix}$.

Пример выполнения типового задания. Для графа, заданного матрицей весов,
а)построить по этой матрице сеть(исходный граф),
б) построить остов наименьшего веса,
в) найти его вес.

$$W = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 5 & 10 & 14 & \infty & \infty \\ 5 & - & 5 & 6 & \infty & \infty \\ 10 & 5 & - & 7 & 8 & 9 \\ 14 & 6 & 7 & - & 4 & \infty \\ \infty & \infty & 8 & 4 & - & 12 \\ \infty & \infty & 9 & \infty & 12 & - \end{pmatrix} \end{array}.$$

Шаг1. $S' = \{x_1\}, S'' = \{x_2, x_3, x_4, x_5, x_6\}, U' = \varnothing.$

Первая итерация. Шаг 2.

$d(S', S'') = \omega(x_1, x_2) = 5,\ S' = \{x_1, x_2\}, S'' = \{x_3, x_4, x_5, x_6\},$

$U' = \{(x_1, x_2)\}.$

Шаг 3. $S' \neq S$, переход на начало второго шага.

Вторая итерация. Шаг 2.

$d(S', S'') = \omega(x_2, x_3) = 5,\ S' = \{x_1, x_2, x_3\}, S'' = \{x_4, x_5, x_6\},$

$U' = \{(x_1, x_2), (x_2, x_3)\}.$

Шаг 3. $S' \neq S$, переход на начало второго шага.

Третья итерация. Шаг 2.

$d(S', S'') = \omega(x_2, x_4) = 6,\ S' = \{x_1, x_2, x_3, x_4\}, S'' = \{x_5, x_6\},$

$U' = \{(x_1, x_2), (x_2, x_3), (x_2, x_4)\}.$

Шаг 3. $S' \neq S$, переход на начало второго шага.

Четвертая итерация. Шаг 2. $d(S', S'') = \omega(x_4, x_5) = 4,\ S' = \{x_1, x_2, x_3, x_4, x_5\}, S'' = \{x_6\},$

$U' = \{(x_1, x_2), (x_2, x_3), (x_2, x_4), (x_4, x_5)\}.$

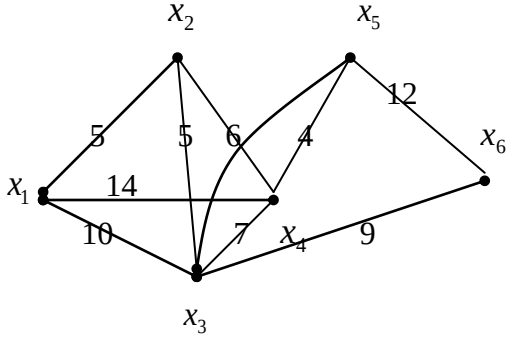
Шаг 3. $S' \neq S$, переход на начало второго шага.

Пятая итерация. Шаг 2. $d(S', S'') = \omega(x_3, x_6) = 9,\ S' = \{x_1, x_2, x_3, x_4, x_5, x_6\}, S'' = \varnothing,$

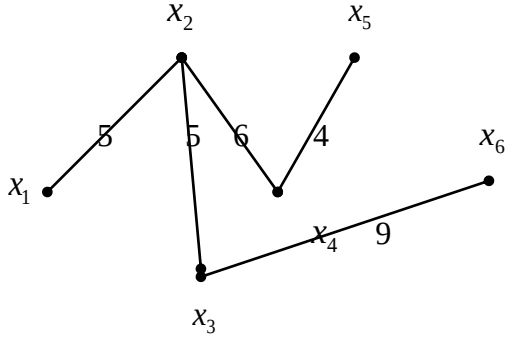
$U' = \{(x_1, x_2), (x_2, x_3), (x_2, x_4), (x_4, x_5), (x_3, x_6)\}.$

Шаг 3. $S' = S$. Итак, получен остовный граф. $G' = (S', U')$ изображен на рисунке справа, его вес $\omega(G') = 5+5+6+4+9 = 29.$

Исходный граф



Остовный граф



Краткая теоретическая справка

Деревья являются простейшим классом графов. Для них выполняются многие свойства, которые не всегда выполняются для обычных графов. Кроме того, деревья широко применяются в программировании при различного рода обработке данных, в частности, в алгоритмах сортировки, кодирования и т.п. Подробно алгоритмы работы с деревьями будут рассматриваться позднее в других курсах, а сейчас только краткое знакомство.

Дерево – это связный граф без циклов. Несколько деревьев (или несвязный граф без циклов) составляют *лес*. Таким образом, дерево является компонентой связности.

Пусть $G = (S, U)$ и $|S| = n$, $|U| = m$. Тогда справедлива эквивалентность следующих утверждений:

1). G - дерево;

2). G - связный граф и $m = n - 1$;

3). G - ациклический граф и $m = n - 1$;

4). любые две несовпадающие вершины графа соединяет единственная простая цепь;

5). G - ациклический граф, обладающий тем свойством, что если какую- либо пару его несмежных вершин соединить ребром, то полученный граф будет содержать ровно один цикл.

2.4 Лабораторная работа № 4 (2 часа).

Тема: «Потоки в сетях, задача о максимальном потоке и минимальном разрезе. Теорема Форда - Фалкерсона»

2.4.1 Цель работы: сформировать умения и навыки применения алгоритма решения задач о максимальном потоке и минимальном разрезе.

2.4.2 Задачи работы: изучить алгоритм решения задач о максимальном потоке и минимальном разрезе.

2.4.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. ПК.
2. Windows.
3. Open Office.

2.4.4 Описание (ход) работы: пример выполнения задания работы № 3

Задания

Пропускные способности дуг заданы матрицей. С помощью алгоритма Форда-Фалкерсона построить максимальный поток от s к t и указать минимальный разрез, отделяющий s от t.

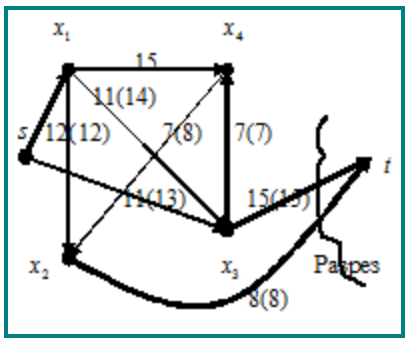
1). $\begin{pmatrix} - & 18 & 16 & - & - & 9 & - \\ - & - & 8 & 11 & 7 & - & 13 \\ - & - & - & - & 13 & - & 19 \\ - & - & 10 & - & - & 15 & - \\ - & - & - & 17 & - & 28 & - \\ - & - & - & - & - & - & 14 \\ - & - & - & - & - & - & - \end{pmatrix}$, 2) $\begin{pmatrix} - & 9 & - & 11 & - & 17 & - \\ - & - & 6 & - & 8 & - & 12 \\ - & - & - & - & - & - & 7 \\ - & 5 & - & - & - & 5 & 4 \\ - & - & - & - & - & 7 & - \\ - & - & - & - & - & - & 9 \\ - & - & - & - & - & - & - \end{pmatrix}$.

Пример решения типовой здачи. Пропускные способности дуг заданы следующей матрицей. Построить максимальный поток от s к t и указать минимальный разрез, отделяющий s от t.

$$W = \begin{array}{c} \\ s \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ t \end{array} \begin{array}{c} \begin{array}{cccccc} s & x_1 & x_2 & x_3 & x_4 & t \end{array} \\ \begin{pmatrix} - & 12 & - & 13 & - & - \\ - & - & 11 & 14 & 15 & - \\ - & - & - & - & - & 8 \\ - & - & - & - & 7 & 15 \\ - & - & 8 & - & - & - \\ - & - & - & - & - & - \end{pmatrix} \end{array}.$$

Этап 1. Путь $s \xrightarrow{12} x_1 \xrightarrow{14} x_3 \xrightarrow{15} t.$

$\delta = \min(12,14,15) = 12.$ Увеличим по этому пути поток до 12 единиц, ребро (s, x_1) становится насыщенным. Поставим величину потока на дугах (x_1, x_3) и (x_3, t).



$\delta = \min(3, 15-12) = 3.$ Поток можно увеличить на три единицы. Дуга (x_3, t) станет насыщенной. Путь $s \xrightarrow{3(3)} x_3 \xrightarrow{7} x_4 \xrightarrow{8} x_2 \xrightarrow{8} t.$ Можно увеличить поток на семь единиц;

Дуга (x_3, x_4) станет насыщенной, потоки примут вид

$$s \xrightarrow{10(3)} x_3 \xrightarrow{7(7)} x_4 \xrightarrow{7(8)} x_2 \xrightarrow{7(8)} t.$$

Больше путей нет. Конец первого этапа.

Этап 2. Рассмотрим теперь маршруты, содержащие противоположные дуги. Маршрут $s \xrightarrow{10(3)} x_3 \xleftarrow{12(4)} x_1 \xrightarrow{11} x_2 \xrightarrow{7(8)} t.$ Поток можно увеличить на единицу на дуге (x_2, t). Тогда потоки по дугам этого маршрута станут такими $s \xrightarrow{11(3)} x_3 \xleftarrow{11(4)} x_1 \xrightarrow{1(1)} x_2 \xrightarrow{8(8)} t.$ Дуга (x_2, t) стала насыщенной. **Больше маршрутов нет. Поток максимален. Делаем разрез вокруг t по насыщенным дугам и получаем его величину 15+8=23 единицы.**

2.4 Лабораторная работа № 5 (2 часа).

Тема: «Компьютерные технологии реализации алгоритмов Дейкстры, Краскала.»

2.4.1 Цель работы: сформировать умения и навыки применения алгоритма решения задач о кратчайшем пути.

2.4.2 Задачи работы: изучить алгоритмы решения задач о кратчайшем пути.

2.4.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. ПК.
2. Windows.
3. Open Office.

2.4.4 Описание (ход) работы: пример выполнения задания работы № 5

Задания

1. Задачи о поиске кратчайших путей

Решение задачи о поиске кратчайшего пути с использованием надстройки MS EXCEL «Поиск решения»

Для получения математической модели задачи введем булевы переменные x_{ij}, которые интерпретируются следующим образом: $x_{ij}=1$, если дуга (i, j) входит в маршрут; $x_{ij}=0$, если дуга (i, j) не входит в маршрут. Тогда математическая модель может иметь, например, такой вид, как на рис. 48 [12].

Ограничение (2) требует, чтобы искомый путь начинался в вершине s. Ограничение (3) требует, чтобы искомый путь заканчивался в вершине t. Ограничение (4) требует, чтобы искомый путь был связным, то есть проходил через вершины графа G. Ограничение (5) требует, чтобы все переменные модели были булевыми.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \to \min_{x\in\Delta_\beta}, \qquad (1)$$

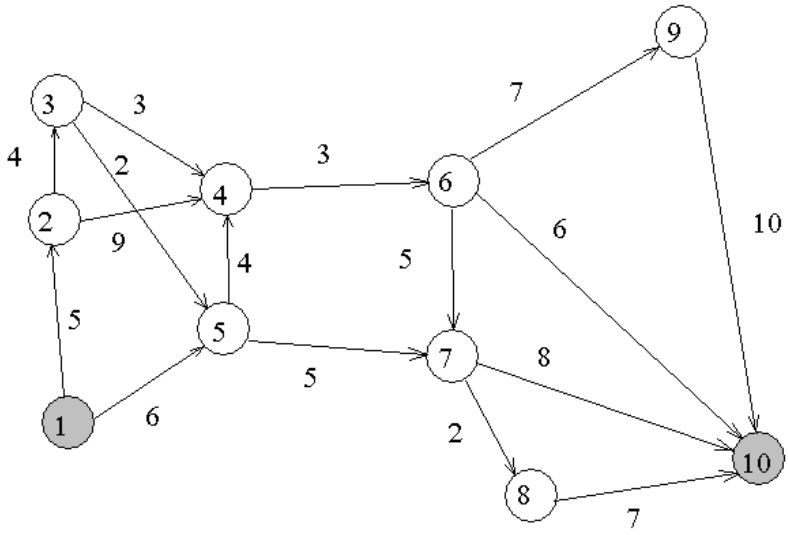
$$\begin{cases} \sum_{j=1}^{n} x_{sj} - \sum_{i=1}^{n} x_{is} = 1; & (2)\\ \sum_{j=1}^{n} x_{tj} - \sum_{i=1}^{n} x_{it} = -1; & (3)\\ \sum_{i=1}^{n} x_{ij} - \sum_{i=1}^{n} x_{ji} = 0 \ (\forall i \in \{1,2,\ldots,n\}, i \neq s, i \neq t); & (4)\\ x_{ij} \in \{0,1\} \ (\forall i, j \in \{1,2,\ldots,n\}); & (5) \end{cases}$$

Рисунок 1

Задача 5.3. Используя приведенный на рис. 49 граф, найти кратчайший путь между вершинами 1 и 10 с использованием надстройки «Поиск решения».

Решение. *В заданном графе 10 вершин и 16 дуг. Следовательно, переменными математической модели этой индивидуальной задачи о построении кратчайшего пути являются 16 переменных:*

$$x_{12}, x_{15}, x_{23}, x_{24}, x_{34}, x_{35}, x_{46}, x_{54}, x_{57}, x_{69}, x_{6,10}, x_{67}, x_{78}, x_{7,10},$$
$$x_{8,10}, x_{9,10}.$$



Рисунок

Математическая постановка такой индивидуальной задачи имеет следующий вид:

$$5x_{12}+6x_{15}+4x_{23}+9x_{24}+3x_{34}+2x_{35}+3x_{46}+4x_{54}+5x_{57}+7x_{69}+$$

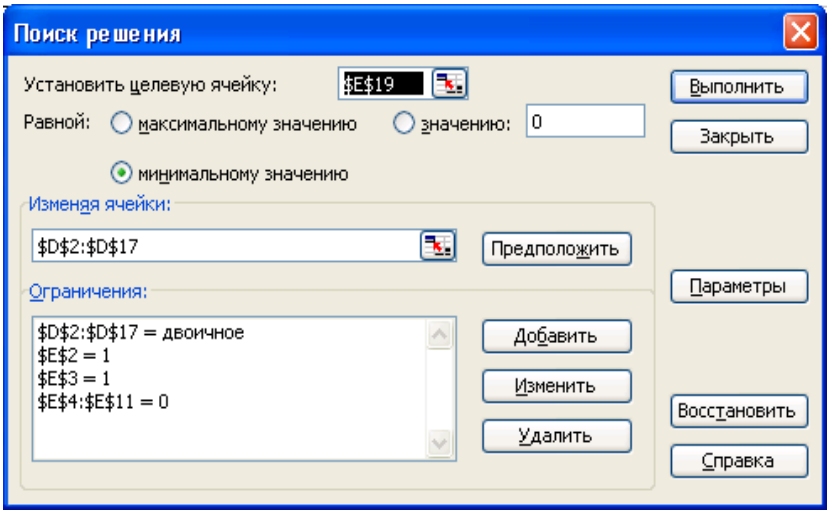
$$+6x_{6,10}+5x_{67}+2x_{78}+8x_{7,10}+7x_{8,10}+10x_{9,10}\to\min_{x\in\Delta_\beta},$$

где множество ограничений выглядит так, как на рис. 50.

Воспользуемся надстройкой «Поиск решения», входящей в состав MS EXCEL. Расположим исходные данные на рабочем листе, например, так, как на рис. 51. В ячейках показаны формулы, связывающие переменные модели. Целевая функция находится в ячейке E19. На рис. 52 приведено окно диалога надстройки перед запуском на выполнение. На рис. 53 показан результат работы надстройки, а на рис. 54 – путь из вершины 1 в вершину 10 минимальной длины. Это значение совпадает с решением, полученным в соответствии с алгоритмом Дейкстры.

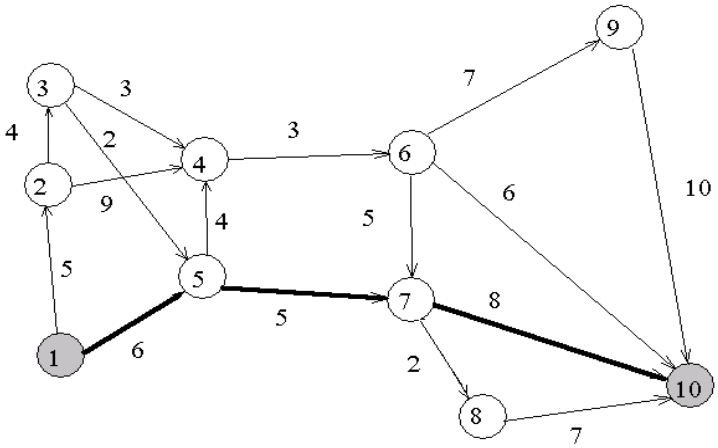
$$\begin{cases} x_{12}+x_{15}=1; \\ x_{6,10}+x_{7,10}+x_{8,10}+x_{9,10}=1; \\ x_{12}-x_{23}-x_{24}=0; \\ x_{23}-x_{34}-x_{35}=0; \\ x_{34}+x_{24}-x_{46}=0; \\ x_{35}+x_{15}-x_{54}-x_{57}=0; \\ x_{46}-x_{67}-x_{69}-x_{6,10}=0; \\ x_{57}+x_{67}-x_{78}-x_{7,19}=0; \\ x_{78}-x_{8,10}=0; \\ x_{69}-x_{9,10}=0; \\ x_{12}, x_{15}, x_{23}, x_{24}, x_{34}, x_{35}, x_{46}, x_{54}, x_{57}, x_{69}, x_{6,10}, x_{67}, \\ x_{78}, x_{7,10}, x_{8,10}, x_{9,10}\in\{0;1\}. \end{cases}$$

	A	B	C	D	E
1	начальная вершина ребра	конечная вершина ребра	вес ребра	Переменные	Ограничения
2	1	2	5	0	=СУММ(D2:D3)
3	1	5	6	0	=СУММ(D4:D7)
4	6	10	6	0	=D2-D8-D9
5	7	10	8	0	=D8-D10-D11
6	8	10	7	0	=D10+D9+D13-D12
7	9	10	10	0	=D11+D3-D13-D14
8	2	3	4	0	=D12-D16-D4-D15
9	2	4	9	0	=D14+D15-D17-D5
10	3	4	3	0	=D17-D6
11	3	5	2	0	=D16-D7
12	4	6	3	0	
13	5	4	4	0	
14	5	7	5	0	
15	6	7	5	0	
16	6	9	7	0	
17	7	8	2	0	
18					
19					=СУММПРОИЗВ(D2:D17;C2:C17)



	A	B	C	D	E
1	начальная вершина ребра	конечная вершина ребра	вес ребра	Переменные	Ограничения
2	1	2	5	0	1
3	1	5	6	1	1
4	6	10	6	0	0
5	7	10	8	1	0
6	8	10	7	0	0
7	9	10	10	0	0
8	2	3	4	0	0
9	2	4	9	0	0
10	3	4	3	0	0
11	3	5	2	0	0
12	4	6	3	0	
13	5	4	4	0	
14	5	7	5	1	
15	6	7	5	0	
16	6	9	7	0	
17	7	8	2	0	
18					
19					19
25	19				

Рисунок



2.6 Лабораторная работа № 6 (2 часа).

Тема: «Алгоритмы обхода и поиска в графе: поиск в глубину и в ширину. Эйлеровы циклы в графах»

2.6.1 Цель работы: сформировать умения и навыки применения алгоритмов обхода и поиска в графе: поиска в глубину и в ширину

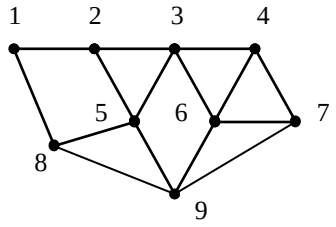
2.6.2 Задачи работы: изучить элементы алгоритмов обхода и поиска в графе: поиска в глубину и в ширину

2.6.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. ПК. 2. Windows. 3. Open Office.

2.6.4 Описание (ход) работы: пример выполнения задания работы № 6.

Задание. *Выполнить (обход) поиск в глубину и в ширину в графе*



Обходы графов. *Обход графа – это некоторое систематическое перечисление его вершин (ребер). Среди всех обходов наиболее известны поиск в глубину и в ширину. Алгоритмы такого поиска лежат в основе многих алгоритмов на графах*.

При поиске используется некоторая структура данных T, в которую помещаются вершины графа. Для обозначения пройденных вершин заводят дополнительный массив пометок этих вершин.

Поиск основывается на следующих действиях.

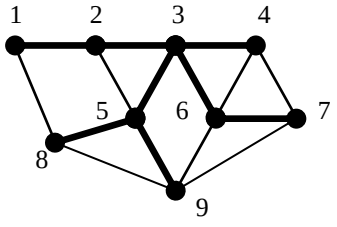
1.Сначала все вершины считаются неотмеченными.

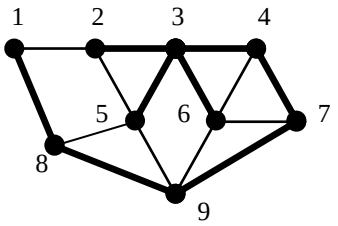
2.Выбирается любая вершина (нач. поиска), заносится в структуру данных T и помечается.

3.Следующие действия выполняются в цикле до тех пор, пока структура T не станет пустой: из структуры данных T выбирается вершина *u*; она выдается в качестве очередной пройденной вершины; перебираются все вершины из Г(u), и все те, которые не помечены, тоже заносятся в структуру T и помечаются.

Если T – это стек (LIFO), то обход называется поиском *в глубину* (т.е. первым делом из структуры T извлекается вершина, попавшая туда последней). Если T – это очередь (FIFO), то обход называется поиском *в ширину*. При поиске в глубину находятся более длинные пути.

*Если граф G связен и конечен, то поиск в ширину или поиск в глубину обойдет****все вершины графа по одному разу.***





Пример на алгоритмы поиска в глубину и в ширину.Рассмотрим алгоритмы поиска в глубину и в ширину на примере. В качестве стартовой возьмем вершину с номером 3. Тогда поиск <u>в ширину</u> даст последовательность вершин: 3→T. T={3}⇒ **3**: {2,5,6,4}→T; T={2,5,6,4}⇒ **2**: {1}→T; T={5,6,4,1}⇒ **5**: {8,9}→T; T={6,4,1,8,9}⇒ **6**: {7}→T; T={4,1,8,9,7}. Окружения всех этих вершин уже отмечены ⇒ они будут выданы по порядку. Итак, выполнен обход всех вершин графа в следующем порядке:

3,2,5,6,4,1,8,9,7. При поиске в глубину начало такое же: 3→T. T={3} ⇒ **3**: {2,5,6,4}→T; T={2,5,6,4}⇒ **4**: {7}→T; T={2,5,6,7}⇒ **7**: {9}→T; T={2,5,6,9} ⇒ **9**: {8}→T; T={2,5,6,8}⇒ **8**: {1}→T; T={2,5,6,1}. Оставшиеся вершины выдаются по порядку. В итоге последовательность вершин: **3,4,7,9,8,1,6,5,2.**

Любой из рассмотренных обходов позволяет построить остовное дерево исходного графа с корнем в исходной вершине (связный подграф без циклов).

Эйлеровы графы

Цикл в графе называется эйлеровым*, если он содержит все ребра графа. Связный граф, в котором существует эйлеров цикл, называется* эйлеровым графом. Эйлеровой цепью *(или путем) является цепь (путь), которая включает все ребра (дуги) графа по одному разу.* Собственная эйлерова цепь – *это эйлерова цепь, которая не является эйлеровым циклом.*

Эйлеров граф можно нарисовать, не отрывая карандаша от бумаги.

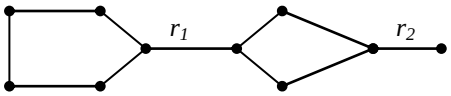
Теорема Эйлера. *Связный граф является эйлеровым тогда и только тогда, когда степени всех его вершин четны.*

Вспомним задачу о кенигсбергских мостах. Она сводится к вопросу – является ли граф, представляющий эту задачу, эйлеровым? Если вершины будут обозначать участки суши, а ребра – мосты, то получим граф следующего вида:
Очевидно, что этот граф не эйлеров, т.к. все его вершины имеют нечетные степени; поэтому требуемый цикл не существует. Если отменить требование возврата в ту же точку, то вопрос о пути по всем мостам сведется к вопросу о существовании собственной эйлеровой цепи.

Граф имеет собственную эйлерову цепь (путь) ⇔ когда он связный и ровно две его вершины имеют нечетную степень.

Вершины нечетной степени являются началом и концом эйлеровой цепи.
А теперь разберемся, как найти эйлеров цикл. Сначала еще определение.

Мостом (или *перешейком*) называется такое ребро графа *G*, удаление которого увеличивает число связных компонент. Если ребро *r* принадлежит некоторому циклу *C*, то оно не может быть мостом.



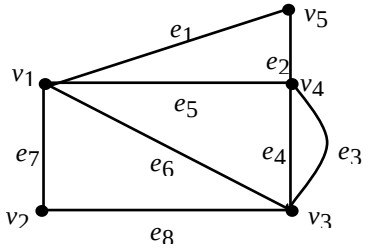
В данном графе ребра r_1 и r_2 являются мостами.

Для нахождения эйлеровой цепи в связном графе, который имеет вершины только четной степени, используют *алгоритм Флери*:

1) Движение начинается из произвольной вершины графа; идем по ребрам, включая эти ребра в эйлерову цепь и удаляя их из графа.

2) В очередной вершине выбираем путь по перешейку только в том случае, если нет пути по циклу.

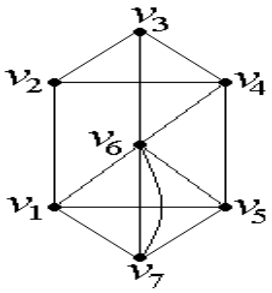
3) Если в графе остаются ребра, которые нельзя использовать для продолжения имеющегося пути, то следует начать строить простой замкнутый цикл из уже пройденной вершины и инцидентного ей ребра, если последнее ранее не использовалось.



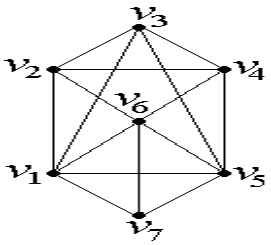
Пусть требуется построить эйлеров цикл для графа, изображенного на рисунке. Начать построение эйлерового цикла можно с любого ребра графа. Начиная с e_1, получим цикл v_1, e_1, v_5, e_2, v_4 ,e_3, v_3 ,e_4 , v_4 ,e_5 ,v_1,e_6 ,v_3, e_8, v_2, e_7 ,v_1. В данном случае сразу получили эйлерову цепь.

Задания

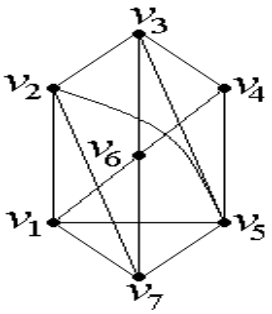
Найти Эйлерову цепь в неориентированном графе.



а)



б)



в)

Пример выполнения задания. Найти Эйлерову цепь в неориентированном графе G, изображенном на рисунке.

Решение. Прежде, чем приступать к нахождению Эйлеровой цепи, необходимо проверить степени вершин графа G − согласно утверждению 2, для существования Эйлеровой цепи, необходимо и достаточно, чтобы в графе G ровно 2 вершины нечетной степени.

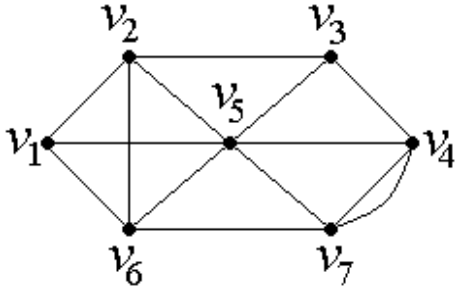


Рис. 1

В рассматриваемом графе нечетные степени имеют вершины v_3 и v_1 (степень этих вершин равна 3). Соединяя эти вершины фиктивным ребром так, как показано на рис. 2, получаем граф G':

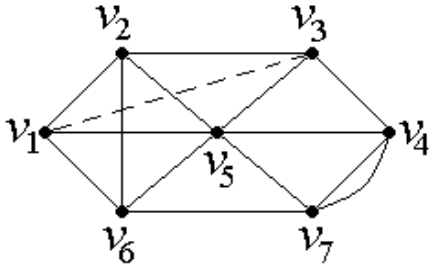


Рис. 2.

Поскольку в конечном итоге будет получена цепь, то очевидно, что началом и концом этой цепи будут вершины с нечетными степенями. Поэтому, следуя описанному выше алгоритму, будем циклы μ_i так, чтобы хотя бы один из них начинался или кончался на вершинах v_3 или v_1.

Пусть цикл μ_1 составят ребра, проходящие через следующие вершины: $v_3\ v_4\ v_7\ v_6\ v_1\ v_2\ v_3$. Согласно алгоритму, удаляем из G' все ребра, задействованные в цикле μ_1. Теперь граф G' будет таким, как показано на рис. 3.

Составляем следующий цикл μ_2: $v_4\ v_5\ v_6\ v_2\ v_5\ v_7\ v_4$. Граф G' после удаления ребер, составляющих цикл μ_2, изображен на рис. 4.

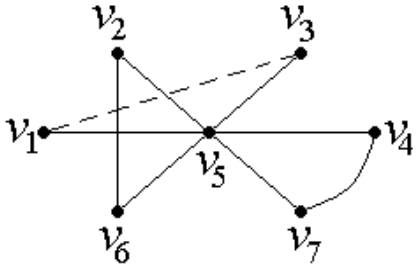


Рис.3

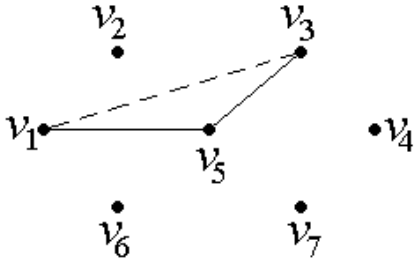


Рис. 4

Очевидно, что последний цикл μ_3 будет состоять из $v_3\ v_5\ v_1|v_3$, где последнее ребро, соединяющее вершины v_1 и v_3 – фиктивно. После удаления ребер, составляющих цикл μ_3, в графе G' не останется ни одного ребра.

Теперь по общим вершинам склеиваем полученные циклы. Поскольку μ_1 и μ_2 имеют общую вершину v_4, то, объединяя их, получим следующий цикл: $v_3\ v_4\ v_5\ v_6\ v_2\ v_5\ v_7\ v_4\ v_7\ v_6\ v_1\ v_2\ v_3$. Теперь склеим получившийся цикл с циклом μ_3: $v_3\ v_4\ v_5\ v_6\ v_2\ v_5\ v_7\ v_4\ v_7\ v_6\ v_1\ v_2\ v_3\ v_5\ v_1|v_3$. Удаляя фиктивное ребро, получаем искомую Эйлерову цепь: $v_3\ v_4\ v_5\ v_6\ v_2\ v_5\ v_7\ v_4\ v_7\ v_6\ v_1\ v_2\ v_3\ v_5\ v_1$.

Алгоритм выделения эйлерова цикла в связном мультиграфе с четными степенями вершин

1) Выделим из G цикл μ_1. (так как степени вершин четны, то висячие вершины отсутствуют). Положим l=1, G'=G.

2) Удаляем из G' ребра, принадлежащие выделенному циклу μ_1. Полученный псевдограф снова обозначаем как G'. Если в G' отсутствуют ребра, то переходим к шагу 4. Если ребра есть, то выделяем из G' цикл μ_{l+1} и переходим к шагу 3.

3) Присваиваем l:=l+1 и переходим к шагу 2.

4) По построению выделенные циклы содержат все ребра по одному разу. Если l:=1, то искомый Эйлеров цикл найден (конец работы алгоритма). В противном случае находим

циклы, содержащие хотя бы по одной общей вершине (в силу связности графа это всегда можно сделать). Склеиваем эти циклы. Повторяем эти операции, пока не останется один цикл, который является искомым.

3.6.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили понятия об алгоритмах обхода и поиска в графе: поиск в глубину и в ширину. Эйлеровых циклах в графах;
- приобрели умения и навыки выполнения алгоритмов обхода и поиска в графе.

2.7 Лабораторная работа № 7 (2 часа).

Тема: «Поиск расстояния между всеми парами вершин. Алгоритм Уоршалла-Флойда»

2.7.1 Цель работы: сформировать умения и навыки использования алгоритма Уоршалла-Флойда в прикладных моделх

2.7.2 Задачи работы: изучить элементы алгоритма Уоршалла-Флойда

2.7.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. ПК.
2. Windows.
3. Open Office.

2.7.4 Описание (ход) работы: пример выполнения задания работы № 7.

Задание и решение типовой задачи.

Наиболее широко известны два алгоритма поиска кратчайших путей. Алгоритм Дейкстры находит кратчайшее расстояние от одной фиксированной вершины до другой и указывает сам путь, длина которого равна этому расстоянию. Алгоритм Флойда-Уоршалла позволяет найти кратчайшие расстояния между всеми парами вершин графа.

С помощью алгоритма Флойда-Уоршалла найти кратчайшие расстояния между всеми парами вершин графа.

Идея: Строится специальная матрица D, элементы которой – кратчайшие пути между всевозможными парами вершин графа G. При определении кратчайшего пути выбирается минимум из «прямого» расстояния между смежными вершинами v_i и v_j и суммарного расстояния при проходе через дополнительную вершину. Затем – более длинные пути и т.д.

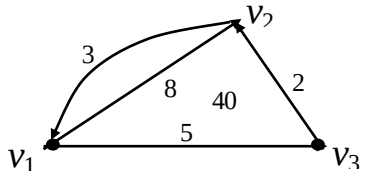
Обозначим через $d_{ij}^{(m)}$ длину кратчайшего пути из v_i в v_j с промежуточными вершинами во множестве $\{v_1,\ldots,v_m\}$. Алгоритм использует три правила:

1) $d_{ij}^{(0)} = a_{ij}$ – вес дуги, соединяющий вершины v_i и v_j (т.е. первоначально матрица D – это исходная матрица весов).

2) $d_{ij}^{(m+1)} = \min\left(d_{ij}^{(m)}, d_{i,m+1}^{(m)} + d_{m+1,j}^{(m)}\right)$.

3) Длина кратчайшего пути из вершины v_i в вершину v_j: $d(v_i, v_j) = d_{ij}^{(n)}$.

Алгоритм строит матрицу за n шагов, т.е. строится матрица $D^{(1)}$, …, $D^{(n)}$=D.



Решение типоой задачи.

Найдем матрицу кратчайших расстояний для графа v_1 v_2 v_3

$$D^{(0)} = \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & \infty \\ \infty & 2 & 0 \end{pmatrix} \begin{matrix} v_1 \\ v_2 \\ v_3 \end{matrix}$$

Элементы матрицы D(1) находим по правилу $d_{ij}^{(1)} = \min\left(d_{ij}^{(0)}, d_{i1}^{(0)} + d_{1j}^{(0)}\right)$. Первая строка и первый столбец не меняются. Получаем $D^{(1)} = \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ \infty & 2 & 0 \end{pmatrix}$. Элементы матрицы D(2) находим по правилу $d_{ij}^{(2)} = \min\left(d_{ij}^{(1)}, d_{i2}^{(1)} + d_{2j}^{(1)}\right)$. Без изменений второй столбец и вторая строка. $D^{(2)} = \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{pmatrix}$.

Элементы матрицы D(3) находим по правилу $d_{ij}^{(3)} = \min\left(d_{ij}^{(2)}, d_{i3}^{(2)} + d_{3j}^{(2)}\right)$.

$D^{(3)} = \begin{pmatrix} 0 & 7 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{pmatrix}$, $D = D^{(3)}$. Каждый из элементов (i,j) матрицы D равен наименьшему расстоянию между вершинами v_i и v_j.

2.8 Лабораторная работа № 8 (2 часа).

Тема: «Графы и задачи линейного программирования и компьютерные технологии их решения»

2.8.1 Цель работы: сформировать представление об использовании методов линейного программирования и компьютерных технологий при решения задач оптимизации графов.

2.8.2 Задачи работы: изучить элементы методов линейного программирования и компьютерных технологий при решения задач оптимизации графов.

.

2.8.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. ПК.
2. Windows.
3. Open Office.

2.8.4 Описание (ход) работы: пример выполнения задания работы № 8

1. Основные понятия и методы линейного программирования.

2. Основные задачи линейного программирования.

3. Компьютерные технологии решения задачи линейного программирования на графах.

Цель работы – Познакомить с компьютерными технологиями решения задач оптимизации (линейного программирования) в Microsoft Excel.

Теоретические сведения: Вуколов Э.А. Основы статистического анализа…(см. Электронные книги). Глава 9, стр. 300-326.

Задача. *Предприятие производит и продаёт продукцию двух видов: «1 Продукт» и «2 Продукт». Для производства продукции используются ресурсы двух категорий: A и B. Расходы ресурсов A и B на производство единицы продукции каждого вида, запасы ресурсов и цены продукции приведены в таблице 1.*

Таблица 1

Ресурсы	*Расход ресурсов на ед. продукции*		*Запасы ресурсов*
	1 Продукт	*2 Продукт*	
A	*1*	*2*	*3*
B	*3*	*1*	*3*
Количество продукции	x_1	x_2	
Цены	*2(ден. ед.)*	*1(ден. ед.)*	

Выяснить, какое количество продукции каждого вида надо производить предприятию (составить план производства), чтобы получить максимум прибыли.

Задание. 1. *Составить математическую модель задачи.*

2. *Решить задачу в Excel.*

Решение. 1. Составить математическую модель задачи. Для составления математической модели задачи прежде всего ***введём переменные (неизвестные) задачи***: x_1 - количество продукции 1-го вида, а x_2 - количество продукции 2-го вида, производимые предприятием.

Ограниченность запасов ресурсов приводит ***к ограничениям на*** x_1 ***и*** x_2***:*** ограничения на расход ресурса A $x_1 + 2 \cdot x_2 \le 3$,

ограничения на расход ресурса B $3 \cdot x_1 + x_2 \le 3$.

Кроме того, $x_1, x_2 \ge 0$.

Качество решения задачи определяется с помощью ***целевой функции задачи*** $Z(x_1, x_2)$ - функции, определяющей доход предприятия от продажи продукции: $Z = 2 \cdot x_1 + x_2$.

Задача об определении плана производства продукции свелась к следующей математической задаче: *найти вектор* (x_1, x_2) *(план производства), координаты которого удовлетворяют системе ограничений*

$$\begin{cases} x_1 + 2 \cdot x_2 \le 3 \\ 3 \cdot x_1 + x_2 \le 3 \end{cases}$$

и условиям неотрицательности $x_1, x_2 \ge 0$,

который доставляет максимум целевой функции $Z = 2 \cdot x_1 + x_2$.

Эту математическую задачу принято записывать в виде

$$Z = 2 \cdot x_1 + x_2 \to \max \quad (1)$$

$$\begin{cases} x_1 + 2 \cdot x_2 \le 3 \\ 3 \cdot x_1 + x_2 \le 3 \end{cases} \quad (2)$$

$$x_1, x_2 \ge 0. \quad (3)$$

и называть ***математической моделью*** данной производственной задачи.

Подобные задачи называются ***задачами линейного программирования***. Они изучаются в разделе математики, называемом ***математическим программированием***. Так как переменные x_1 и x_2 входят в систему ограничений (2) и целевую функцию Z (1) линейно, то эту задачу математического программирования называют ***задачей линейного программирования.***

Множество точек декартовой плоскости (x_1, x_2), координаты которых удовлетворяют системе ограничений (2) и условиям неотрицательности (3), называется областью допустимых решений задачи линейного программирования(областью допустимых планов). В данной задаче она представляет собой выпуклый четырёхугольник. Значения x_1^* и x_2^* из области допустимых планов, при которых Z принимает наибольшее значение в этой области, называются ***оптимальными(оптимальный план),*** а соответствующее наибольшее значение $Z^* = 2 \cdot x_1^* + x_2^*$ является **оптимальным значением прибыли**. Таким образом, задача о распределении ресурсов является задачей оптимизации и её математической моделью служит задача линейного программирования, заключающаяся в поиске оптимального плана и оптимального значения целевой функции.

Задачей оптимизации может быть поиск наименьшего значения.

2. Решение задачи в Excel.

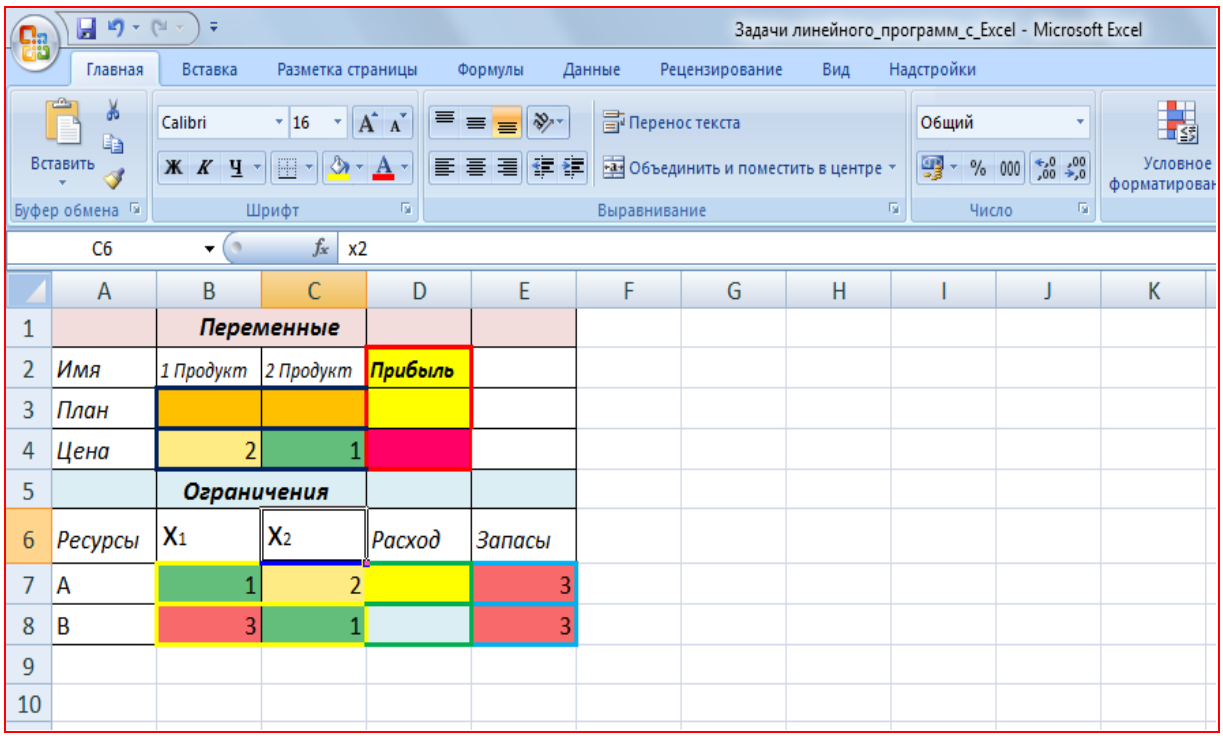
2.1. Ввод данных и формул в таблицу Excel. Открыть Книгу ***Excel,*** Лист1.

-Объединим ячейки B1 и C1. Для этого выделить ячейки, нажать правую кнопку мыши. В появившемся окне вызвать «Формат ячеек», затем «Выравнивание» и поставить галочку против опции «объединение ячеек», нажать ОК. В объединённые ячейки впишем заголовок «Переменные».

-В ячейку A2 вписать «Имя», в A3- «План», в ячейку A4 «Цена», в B2- «1 Продукт», в C2- «2 Продукт», в D2 « Прибыль».

-В ячейки B4 и C4 заносятся значения цен на продукцию.

-Для переменных x_1 и x_2 отводятся ячейки B3 и C3. Это изменяемые(рабочие) ячейки, В них исходные данные не заносятся и в результате решения задачи в эти ячейки будут вписаны оптимальные значения. Таблица данных будет иметь вид

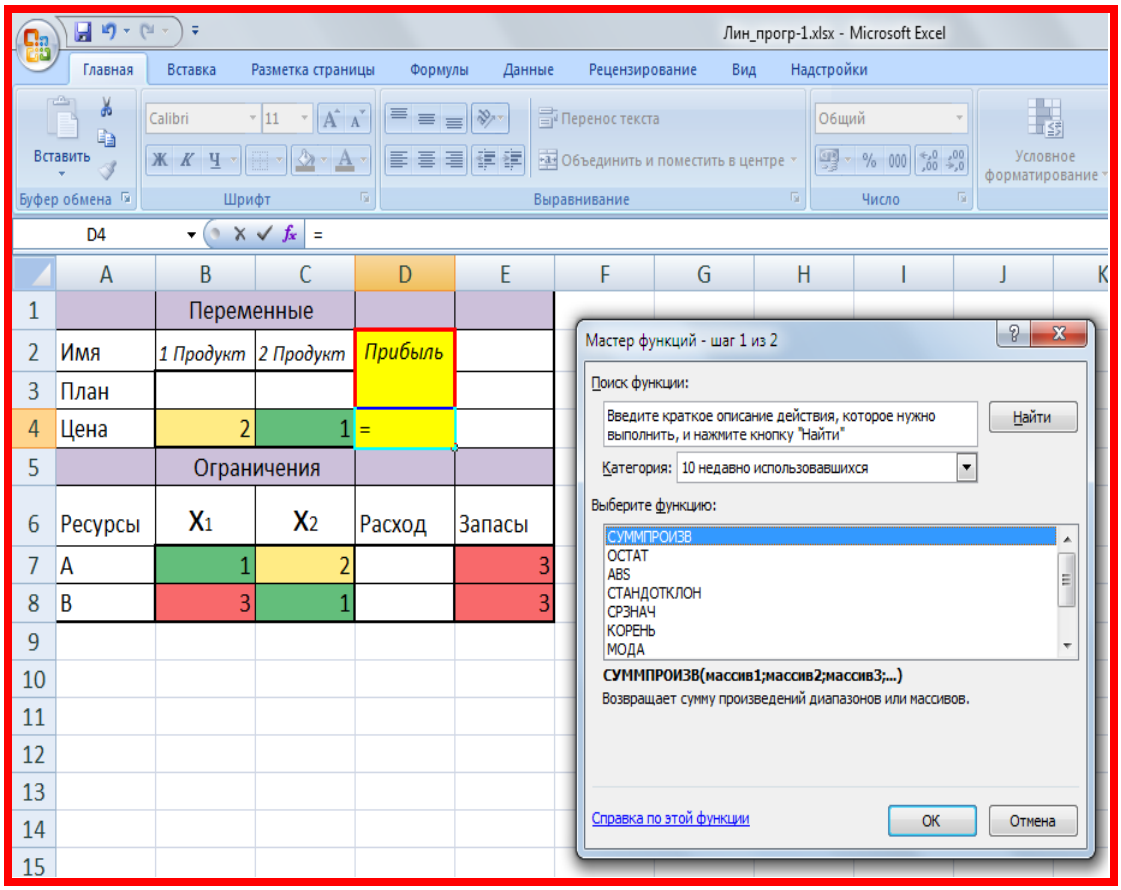


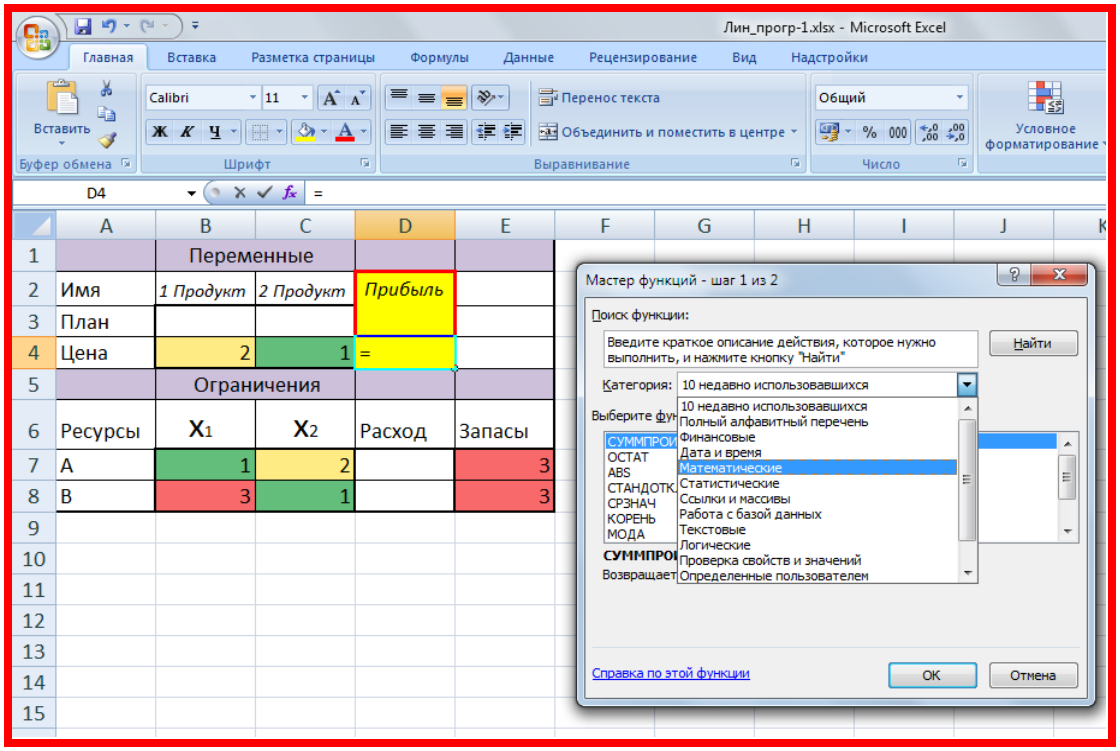
	A	B	C	D	E
1		**Переменные**			
2	*Имя*	*1 Продукт*	*2 Продукт*	***Прибыль***	
3	*План*				
4	*Цена*	2	1		
5		***Ограничения***			
6	*Ресурсы*	X_1	X_2	*Расход*	*Запасы*
7	A	1	2		3
8	B	3	1		3
9					
10					

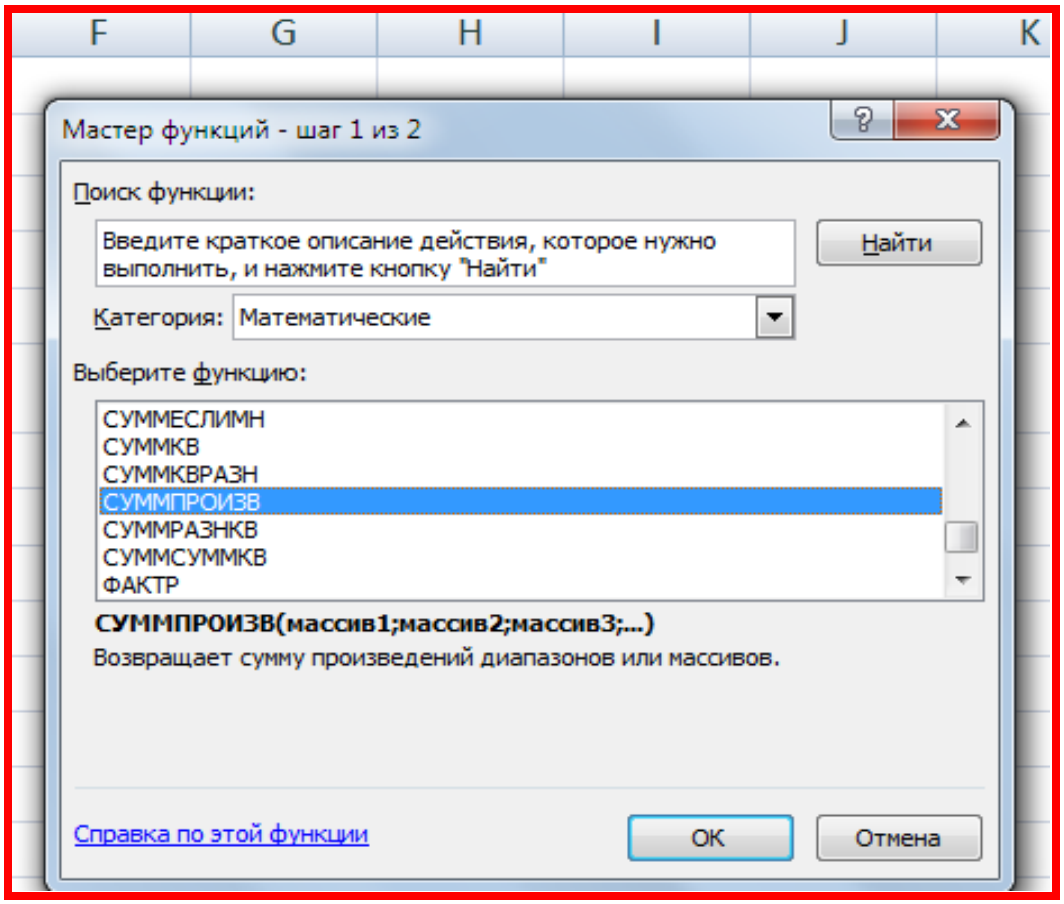
-В ячейке D4 после окончания решения задачи будет указана оптимальное значение прибыли(целевая ячейка). С этой целью в ячейку D4 вводится формула для вычисления значений целевой функции $Z = 2 \cdot x_1 + x_2$. Для этого надо выполнить следующие операции:

1) курсор в D4, выделить эту ячейку,

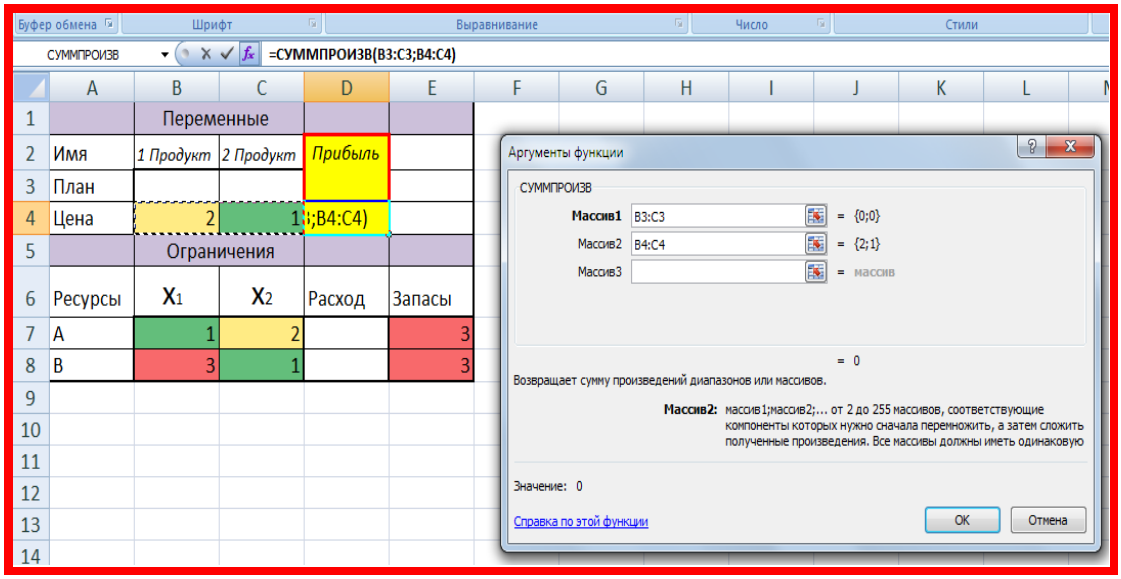
2) щёлкнув по кнопке f_x вызвать Мастера функций, в открывшемся окне в категории «10 недавно использовавшихся» выбрать «Математические», а затем «СУММПРОИЗВ», ОК.

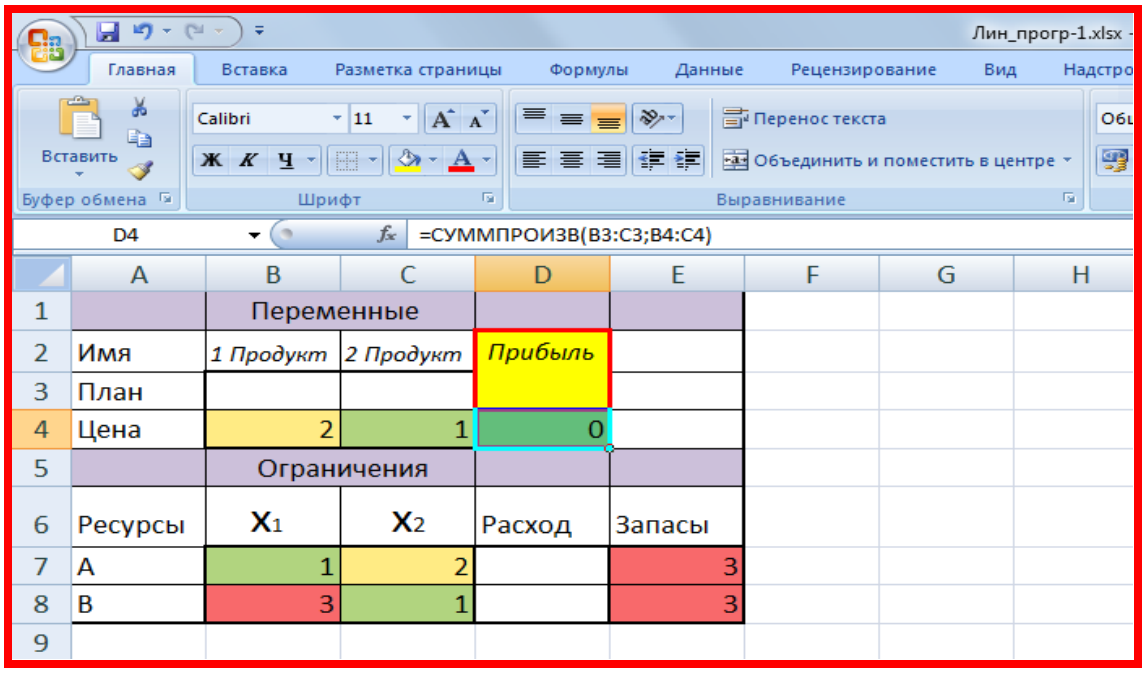

Лин_прогр-1.xlsx - Microsoft Excel
Главная
Вставка
Разметка страницы
Формулы
Данные
Рецензирование
Вид
Надстройки
Вставить
Буфер обмена
Calibri
11
Шрифт
Перенос текста
Объединить и поместить в центре
Выравнивание
Общий
Число
Условное форматирование
D4
=
Переменные
Имя
1 Продукт
2 Продукт
Прибыль
План
Цена
2
1
=
Ограничения
Ресурсы
X1
X2
Расход
Запасы
A
1
2
3
B
3
1
3
Мастер функций - шаг 1 из 2
Поиск функции:
Введите краткое описание действия, которое нужно выполнить, и нажмите кнопку "Найти"
Найти
Категория:
10 недавно использовавшихся
Выберите функцию:
СУММПРОИЗВ
ОСТАТ
ABS
СТАНДОТКЛОН
СРЗНАЧ
КОРЕНЬ
МОДА
СУММПРОИЗВ(массив1;массив2;массив3;...)
Возвращает сумму произведений диапазонов или массивов.
Справка по этой функции
OK
Отмена


Лин_прогр-1.xlsx - Microsoft Excel
Главная
Вставка
Разметка страницы
Формулы
Данные
Рецензирование
Вид
Надстройки
Вставить
Буфер обмена
Calibri
11
Шрифт
Перенос текста
Объединить и поместить в центре
Выравнивание
Общий
Число
Условное форматирование
D4
=
Переменные
Имя
1 Продукт
2 Продукт
Прибыль
План
Цена
2
1
=
Ограничения
Ресурсы
X1
X2
Расход
Запасы
A
1
2
3
B
3
1
3
Мастер функций - шаг 1 из 2
Поиск функции:
Введите краткое описание действия, которое нужно выполнить, и нажмите кнопку "Найти"
Найти
Категория:
10 недавно использовавшихся
Полный алфавитный перечень
Финансовые
Дата и время
Математические
Статистические
Ссылки и массивы
Работа с базой данных
Текстовые
Логические
Проверка свойств и значений
Определенные пользователем
Справка по этой функции
OK
Отмена



В появившемся окне «Аргументы функции» в поле «Массив 1» ввести адреса изменяемых ячеек B3:C3(протаскивая курсор мыши по ячейкам), в поле «Массив 2» вводятся адреса ячеек с ценами на продукцию B4:C4, «Массив 3» игнорируется. Нажать ОК. В ячейке D4 появится число 0.





	A	B	C	D	E	F	G	H
1		Переменные						
2	Имя	1 Продукт	2 Продукт	Прибыль				
3	План							
4	Цена	2	1	0				
5		Ограничения						
6	Ресурсы	X_1	X_2	Расход	Запасы			
7	A	1	2		3			
8	B	3	1		3			
9								

-Объединить ячейки B5 и C5 и вписать «Ограничения», в A6- «Ресурсы», в B6 и C6 x_1 и x_2, в D6 «Расход», в E6 «Запасы», A7 и A8 значки ресурсов,

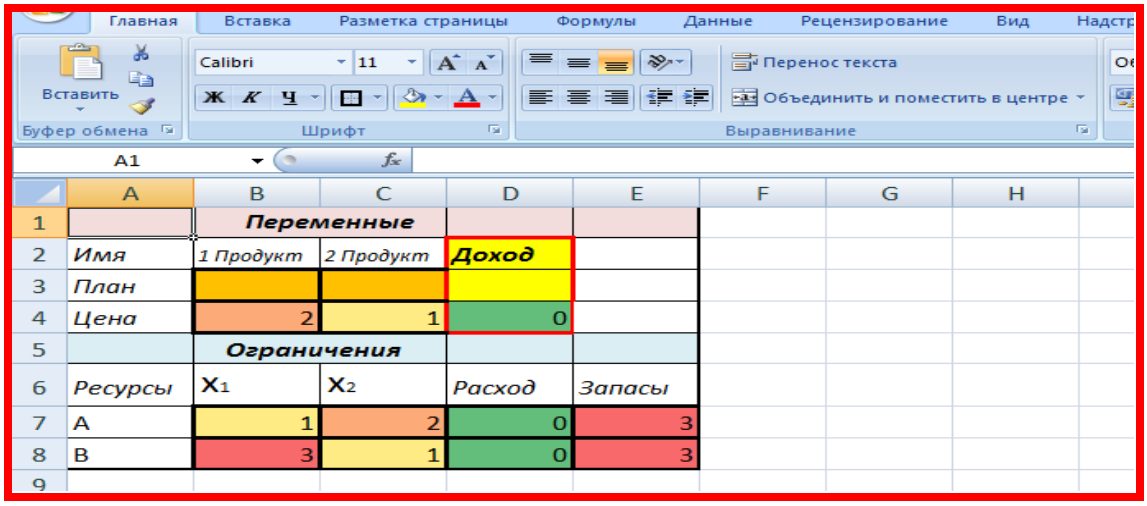
в поле B7:C8- нормы расхода ресурсов.

-В ячейку D7 вводится формула вычисления израсходованного ресурса A $x_1 + 2 \cdot x_2$, в ячейку D8- формула израсходованного ресурса B $3 \cdot x_1 + x_2$

(также, как и формула целевой функции).

- В ячейки E7 и E8 вносим размеры запасов ресурсов.

Данные и формулы введены. Интерфейс задачи будет иметь вид

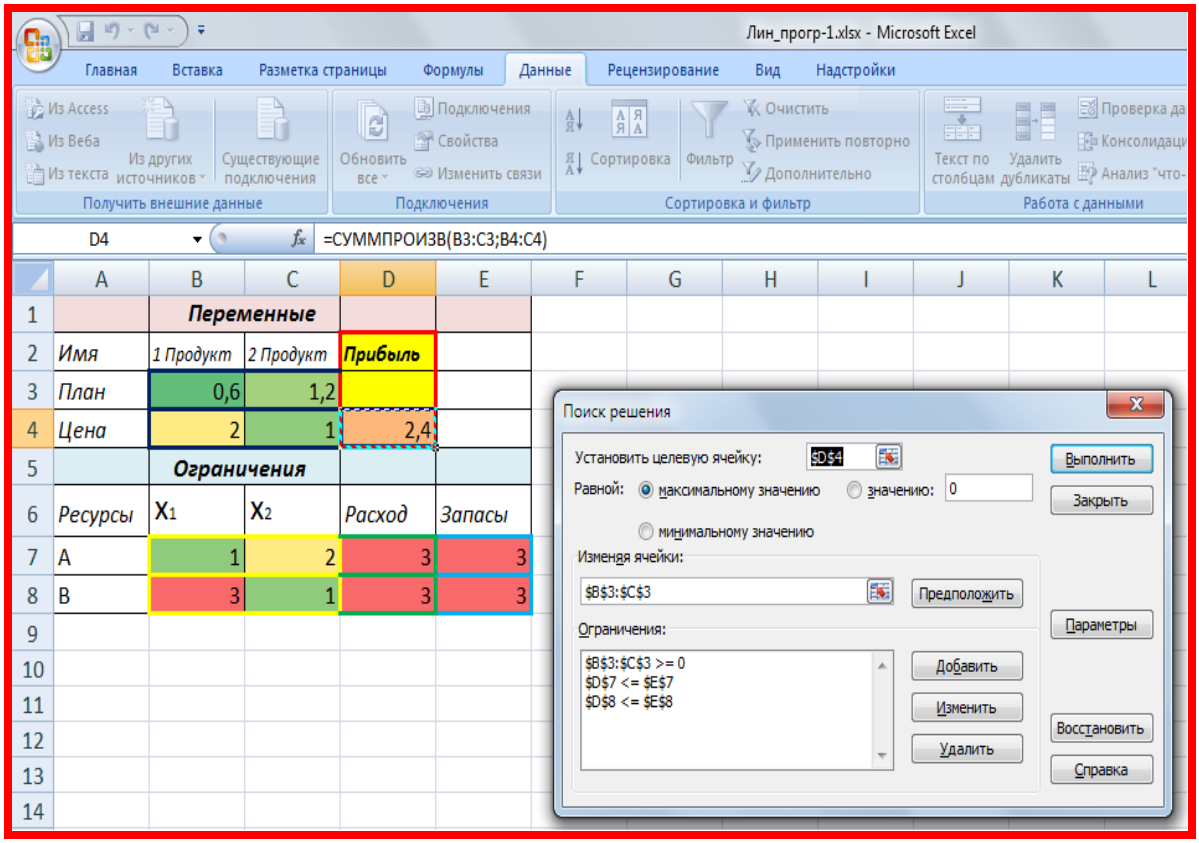


	A	B	C	D	E	F	G	H
1		Переменные						
2	Имя	1 Продукт	2 Продукт	Доход				
3	План							
4	Цена	2	1	0				
5		Ограничения						
6	Ресурсы	X_1	X_2	Расход	Запасы			
7	A	1	2	0	3			
8	B	3	1	0	3			
9								

2.2. Использование надстройки Excel «Поиск решения».

Надстройка Excel *«Поиск решения»* при первом использовании должна быть предварительно активирована. Открыв Excel, нажать кнопки «Office» → «Параметры Excel» → «Надстройки» → «Неактивные надстройки приложений» → выделить строку *«Поиск решения»* → «Управление: надстройки Excel» → «перейти» → ОК.

Щёлкнув на ленте кнопку «Данные», затем «Поиск решений» откроем окно «Поиск решений».



-В поле «Установить целевую ячейку» ввести адрес целевой ячейки D4, щёлкнув по ней курсором мыши.

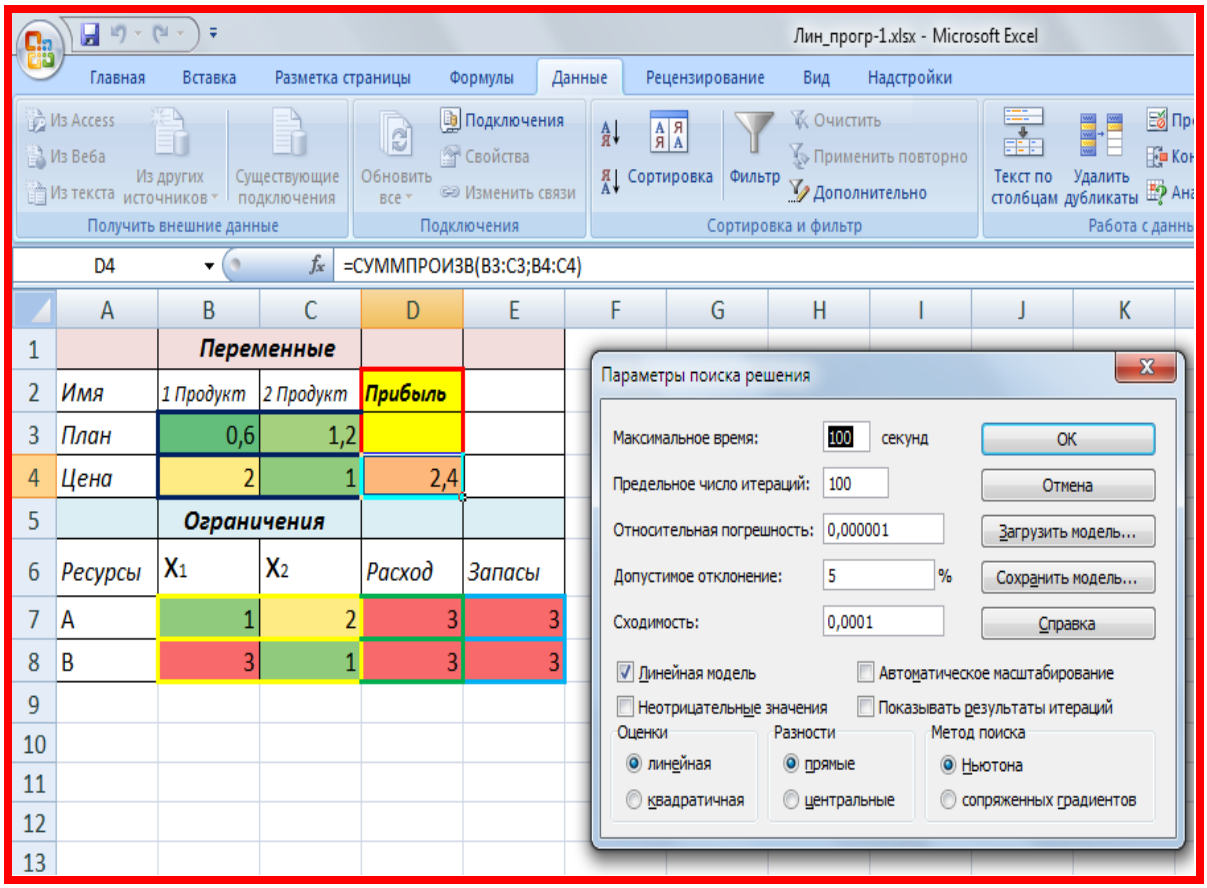
-Выбрать «равной максимальному значению».

-В поле «изменяя ячейки» указать адреса B3:C3.

-В поле «Ограничения» щёлкнуть «Добавить». После появления поля «Добавление ограничения» в поле «Ссылка на ячейку:» сделать ссылку на ячейку D7, выбрать знак $\leq$, в поле «Ограничение:» ввести адрес ячейки с запасом ресурса A- E7. Вновь выбрать «Добавить» провести ввод ограничения по ресурсу B, затем по ограничению $x_1, x_2 \geq 0$. После этого нажать ОК.

2.3. Настройка параметров решения задачи.

Выбрав в окне «Поиск решений» опцию «Параметры» в появившемся окне «Параметры поиска решения»установить флажок в поле «Линейная модель». При таком выборе при решении задачи будет использоваться симплекс-метод. Остальные значения можно оставить без изменения. Нажать ОК.



2.3. Завершение решения задачи и просмотр результатов.

В окне «Поиск решений» нажимаем кнопку «Выполнить». Появляется окно «Результаты поиска решения». Можно выбрать тип отчёта, сохранить найденное решение или восстановить исходные значения, ОК.

3. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПРОВЕДЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

3.1 Практическое занятие № 1 (2 часа).

Тема: «Определение графов, основные понятия теории графов. Виды графов. Способы задания графов. Матрицы смежности и инцидентности графа. Матрица Кирхгофа. Числовые характеристики графов»

3.1.1 Задание для работы:

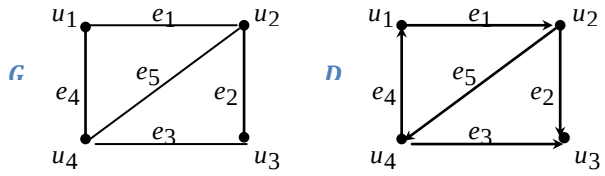
1. Определение графов, основные понятия теории графов. Виды графов. Способы задания графов. Матрицы смежности и инцидентности графа. Матрица Кирхгофа.
2. Числовые характеристики графов

3.1.2 Краткое описание проводимого занятия:

1. Определение графов, основные понятия теории графов. Виды графов. Способы задания графов. Матрицы смежности и инцидентности графа. Матрица Кирхгофа.
2. Числовые характеристики графов

1. Способы задания графов. Матричное представление графов.

1. Составить матрицы смежности графов.

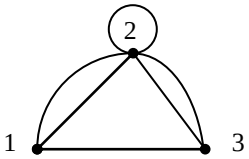


Решение. Матрицы смежности для заданных графа G и орграфа D

$$A(G)=\begin{pmatrix}0&1&0&1\\1&0&1&1\\0&1&0&1\\1&1&1&0\end{pmatrix}\quad A(D)=\begin{pmatrix}0&1&0&0\\0&0&1&1\\0&0&0&0\\1&0&1&0\end{pmatrix}$$

2. Найти матрицу смежности псевдографа.

$$A(P)=\begin{pmatrix}0&2&1\\2&2&2\\1&2&0\end{pmatrix}$$

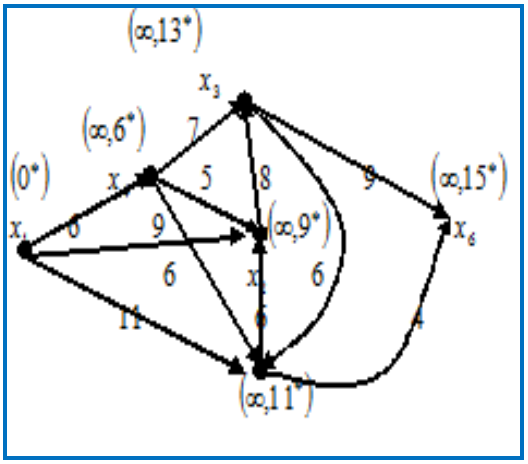


3. Найти матрицы инцидентности для заданных графа G и орграфа D

$$I(G)=\begin{pmatrix}1&0&0&1&0\\1&1&0&0&1\\0&1&1&0&0\\0&0&1&1&1\end{pmatrix}\quad I(D)=\begin{pmatrix}-1&0&0&1&0\\1&-1&0&0&-1\\0&1&1&0&0\\0&0&-1&-1&1\end{pmatrix}$$

4. Задана весовая матрица сети P. Построить по этой матрице сеть,
Изобразим теперь сам граф по данной матрице весов.

$$P=\begin{array}{c}\\x_1\\x_2\\x_3\\x_4\\x_5\\x_6\end{array}\begin{array}{c}\begin{array}{cccccc}x_1&x_2&x_3&x_4&x_5&x_6\end{array}\\\begin{pmatrix}-&9&\infty&6&11&\infty\\\infty&-&8&\infty&\infty&\infty\\\infty&\infty&-&\infty&6&9\\\infty&5&7&-&6&\infty\\\infty&6&\infty&\infty&-&4\\\infty&\infty&\infty&\infty&\infty&-\end{pmatrix}\end{array}$$

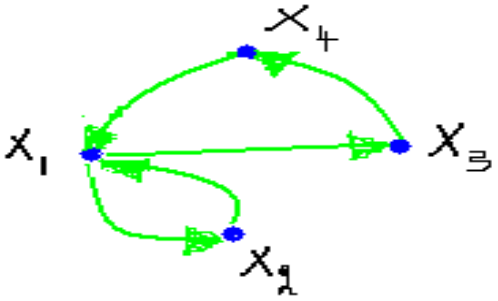


2. Числовые характеристики графов

1. Орграф G задан списком пар начальных и конечных вершин ориентированных рёбер: (x_1, x_2), (x_1, x_3), (x_2, x_1), (x_3, x_4), (x_4, x_1). Для графа G степень входа вершины in deg (x_4) равна-...

ОТВЕТ:1

2.



Степень выхода вершины x_1 равна-...

ОТВЕТ:2

3. В простом графе G , представленном парами смежных вершин G:(1,2), (1, 4), (2,3), (2,4), (2,5), (3,5), d(G) равно-...

ОТВЕТ:2

3.1.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили основные понятия теории графов, способы задания графов, матричное представление графов;
- приобрели умения и навыки применять основные понятия теории графов, способы задания графов, матричное представление графов при решении задач.

3.2 Практическое занятие № 2 (2 часа).

Тема: «Маршруты, циклы, связность. Свойства связных графов, Эйлеровы и гамильтоновы графы. Ориентированные графы и деревья. Сети»

3.2.1 Задание для работы:

1. Маршруты, циклы, связность. Свойства связных графов, Эйлеровы и гамильтоновы графы. 2. Ориентированные графы и деревья. Сети.

3.2.2 Краткое описание проводимого занятия:

1. Маршруты, циклы, связность. Свойства связных графов, Эйлеровы и гамильтоновы графы.
2. Ориентированные графы и деревья. Сети.

Маршруты, цепи, циклы. Маршрутом *от вершины* u *к вершине* v *или (*u,v*)*-маршрутом *в графе* G *называется всякая последовательность вида* $u=v_0,e_1,v_1,e_2,\ldots,e_n,v_n=v$, *в которой любые два соседних элемента инцидентны, т.е.* e_k *– ребро, соединяющее вершины* v_{k-1} *и* v_k, k = *1,2,…,*n.

Это определение подходит также для псевдо-, мульти- и орграфов. В случае орграфа v_{k-1} *– начало* ребра e_k , а v_k – его *конец*. При этом вершину *u* называют *началом маршрута*, а вершину *v* – его *концом*. В маршруте некоторые вершины и ребра могут совпадать. Если *u* = *v*, то маршрут *замкнут*, а иначе *открыт*. Для «обычного» графа маршрут можно задавать только последовательностью вершин $v_0,v_1,\ldots,v_n$ или ребер $e_1,e_2,\ldots,e_n$.

Маршрут называется *цепью*, если в нем нет совпадающих ребер, и *простой цепью* – если дополнительно нет совпадающих вершин, кроме, может быть, начала и конца цепи. Про цепь $u=v_0,v_1,\ldots,v_n=v$ говорят, что она *соединяет* вершины *u* и *v* и обозначают ⟨*u*,*v*⟩.

Очевидно, что если есть цепь, соединяющая вершины *u* и *v*, то есть и простая цепь, соединяющая эти вершины.

Замкнутая цепь называется *циклом*; замкнутая простая цепь – *простым циклом*. Число циклов в графе *G* обозначается *z(G)*. Граф без циклов называется *ациклическим*.

Для орграфов цепь называется *путем*, а цикл – *контуром*.

Число ребер в маршруте M (возможно, с повторениями) называется его *длиной*, обозначается |M|.

Расстоянием между вершинами u и *v* (обозначается $d(u,v)$) называется длина кратчайшей цепи $\langle u,v\rangle$, а сама кратчайшая цепь называется *геодезической*. Если не существует цепи, соединяющей вершины *u* и *v*, то по определению $d(u,v) = +\infty$.

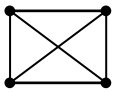
Диаметром графа *G* (обозначается *D(G)*) называется длина длиннейшей геодезической.

Максимальным удалением в графе *G от вершины v* называется $r(v) = max\ d(v,v'), \forall\ v' \in V$. Вершина *v* графа *G* является его *центром*, если максимальное удаление от нее до всех вершин принимает наименьшее значение.

Множество вершин, находящихся на одинаковом расстоянии *n* от вершины *v*, называется *ярусом* (обозначается $D(v,n)$): $D(v,n) = \{u \in V \mid d(v,u) = n\}$.

Граф, любая из вершин которого является его центром – максимальное удаление до всех вершин от любой =

Связность. *Если две вершины u и v в графе можно соединить цепью, то такие вершины связаны. Граф называется связным, если в нем связаны все вершины.*



Легко видеть, что отношение связности на множестве вершин является отношением эквивалентности. Данное отношение разбивает множество вершин графа на классы, объединяющие вершины, связанные друг с другом. Такие классы называются *компонентами связности*; число компонент связности обозначается *k(G)*.

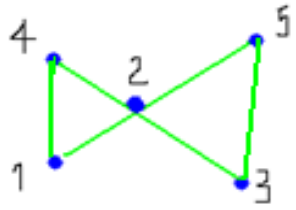
Граф *G* является связным тогда и только тогда, когда он имеет одну компоненту связности: $k(G) = 1$. Если $k(G) > 1$, то это *несвязный* граф. Граф, состоящий только из изолированных вершин (в котором k(G)=|V|, r(G)=0), называется *вполне несвязным*.

Вершина графа, удаление которой увеличивает число компонент связности, называется *разделяющей* или *точкой сочленения*.

Ориентированный граф *G(V,E)* является *слабо связным* (*слабым*), если симметричное замыкание множества *E* определяет связный граф (иными словами, если после замены всех дуг графа *G* ребрами полученный граф будет связным). Ориентированный граф является *сильно связным* (*сильным*), если для любой пары вершин $u,v \in V$ существует ориентированный путь из *u* в *v* (т.е. из любой вершины графа достижимы все его остальные вершины). Если для любой пары вершин по крайней мере одна достижима из другой, то такой граф является *односторонне связным*, или *односторонним*. Граф, состоящий из одной вершины, по определению считается сильно связным.

Множества вершин связных компонент образуют разбиение множества вершин графа.

1.



В графе, представленном на рисунке, e(1) равно-...
ОТВЕТ:2
2.В простом графе G , представленном парами смежных вершин G:(1,2), (1, 4), (2,3), (2,4), (2,5), (3,5), e(2) равно-...
ОТВЕТ:1
3.В простом графе G , представленном парами смежных вершин G:(1,2), (1, 4), (2,3), (2,4), (2,5), (3,5), d(G) равно-...
ОТВЕТ:2
4.В простом графе G , представленном парами смежных вершин G:(1,2), (1, 4), (2,3), (2,4), (2,5), (3,5), r(G) равно-...
ОТВЕТ:1
5.В простом графе G , представленном парами смежных вершин G:(1,2), (1, 4), (2,3), (2,4), (2,5), (3,5), центром является:
+а) 2
б) 1
в) 3,4
г) 2,3
д) 5
6.В простом графе G , представленном парами смежных вершин G:(1,2), (1, 4), (2,3), (2,4), (2,5), (3,5), диаметральной цепью является:
+а) 1-2-3
б) 1-4-2-3
в) 1-3
г) 1-4
д) 3-5
7.Количество периферийных вершин в простом графе G , представленном парами смежных вершин G:(1,2), (1, 4), (2,3), (2,4), (2,5), (3,5), равно-…
ОТВЕТ:4

3.2.3 Результаты и выводы: в результате проведенного занятия студенты:
- освоили основные понятия маршрута, цепи, цикла, связности, метрических характеристик;
- приобрели умения и навыки применять понятия маршрута, цепи, цикла, связности, метрических характеристик при описании графов.

3.3 Практическое занятие № 3 (2 часа).

Тема: «Нахождение экстремальных путей в сети: алгоритм Дейкстры и его прикладные аспекты. Компьютерные технологии реализации алгоритма Дейкстры»
1. Задачи глобального и локального анализа графов.
2.Нахождение кратчайших путей в сети: алгоритм Дейкстры и его прикладные аспекты»

3.3.1 Задание для работы:
1. Задачи глобального и локального анализа графов.
2.Нахождение кратчайших путей в сети: алгоритм Дейкстры и его прикладные аспекты

3.3.2 Краткое описание проводимого занятия:
1. Задачи глобального и локального анализа графов.

2.Нахождение кратчайших путей в сети: алгоритм Дейкстры и его прикладные аспекты

Задания

Задана весовая матрица сети G.

а) построить по этой матрице сеть,

б) найти минимальный путь из вершины $s = x_1$ в вершину $t = x_6$ или $t = x_7$ по алгоритму Дейкстры,

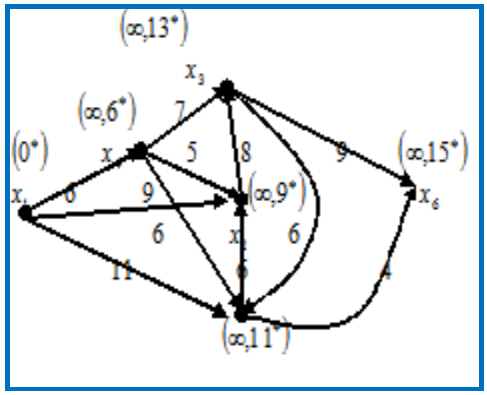
в) найти вес минимального пути.

1. $\begin{pmatrix} - & 5 & 10 & 13 & \infty & \infty \\ \infty & - & 8 & 9 & 13 & \infty \\ \infty & \infty & - & 5 & 3 & 6 \\ \infty & \infty & \infty & - & 8 & 10 \\ \infty & \infty & \infty & \infty & - & 9 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix}$. 2. $\begin{pmatrix} - & 11 & \infty & 14 & 15 & \infty \\ \infty & - & 13 & \infty & \infty & \infty \\ \infty & \infty & - & \infty & \infty & 13 \\ \infty & 7 & 11 & - & 9 & \infty \\ \infty & 11 & 10 & \infty & - & 14 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix}$.

Пример. Задана весовая матрица сети G. Найти минимальный путь из вершины x_1 в вершину x_6 по алгоритму Дейкстры.

Изобразим теперь сам граф по данной матрице весов.

$$P = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 9 & \infty & 6 & 11 & \infty \\ \infty & - & 8 & \infty & \infty & \infty \\ \infty & \infty & - & \infty & 6 & 9 \\ \infty & 5 & 7 & - & 6 & \infty \\ \infty & 6 & \infty & \infty & - & 4 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix} \end{array}.$$



Этап1. Шаг1. Полагаем $d(x_1) = 0^*, \tilde{x} = x_1, x_5$ $d(x_2) = d(x_3) = d(x_4) = d(x_5) = d(x_6) = \infty$.

1-я итерация. Шаг 2. Множество вершин, непосредственно следующих за $\tilde{x} = x_1$ с временными метками $\tilde{S} = \{x_2, x_4, x_5\}$. Пересчитываем временные метки этих вершин

$d(x_2) = \min\{\infty, 0^* + 9\} = 9,\ d(x_4) = \min\{\infty, 0^* + 6\} = 6,\ d(x_5) = \min\{\infty, 0^* + 11\} = 11.$

Шаг 3. Одна из временных меток превращается в постоянную $\min\{\infty, 6, 11, \infty\} = 6^* = d(x_4), \tilde{x} = x_4$.

Шаг 4. $\tilde{x} = x_4 \neq t = x_6$, происходит возвращение на второй шаг.

2-я итерация. Шаг 2. $\tilde{S} = \{x_2, x_3, x_5\}, d(x_2) = \min\{9, 6^* + 5\} = 9$, $d(x_3) = \min\{\infty, 6^* + 7\} = 13$, $d(x_5) = \min\{11, 6^* + 6\} = 11$.

Шаг 3. $\min\{d(x_2), d(x_3), d(x_5), d(x_6)\} = \min\{9, 13, 11, \infty\} = 9^* = d(x_2), \tilde{x} = x_2$.

Шаг 4. $x_2 \neq x_6$, возвращение на второй шаг.

3-я итерация. Шаг 2. $\tilde{S} = \{x_3\}, d(x_3) = \min\{13, 9^* + 8\} = 13$.

Шаг 3. . $\min\{d(x_3), d(x_5), d(x_6)\} = \min\{13, 11, \infty\} = 11^* = d(x_5), \tilde{x} = x_5$.

Шаг 4. $x_5 \neq x_6$, возвращение на второй шаг.

4-я итерация. Шаг 2. $\tilde{S} = \{x_6\}$, $d(x_6) = \min\{\infty, 11^* + 4\} = 15$.

Шаг 3. . $\min\{d(x_3), d(x_6)\} = \min\{13, 15\} = 13^* = d(x_3)$, $\tilde{x} = x_3$.

Шаг 4. $x_3 \neq x_6$, возвращение на второй шаг.

5-я итерация. Шаг 2. $\tilde{S} = \{x_6\}$, $d(x_6) = \min\{15, 13^* + 9\} = 15$.

Шаг 3. $\min\{d(x_6)\} = \min\{15\} = 15^*$, $\tilde{x} = x_6$.

Шаг 4. $x_6 = t = x_6$, конец первого этапа.

Этап 2. Шаг5. Составим множество вершин, непосредственно предшествующих $\tilde{x} = x_6$ с постоянными метками $\tilde{S} = \{x_3, x_5\}$. Проверим для этих двух вершин выполнение равенства (4.7.3).

$d(\tilde{x}) = 15 = 11^* + 4 = d(x_5) + \omega(x_5, x_6)$, $d(\tilde{x}) = 15 \neq 13^* + 9 = d(x_3) + \omega(x_3, x_6)$. Включаем дугу (x_5, x_6) в кратчайший путь. $\tilde{x} = x_5$.

Шаг 6. $\tilde{x} \neq s = x_1$, возвращение на пятый шаг.

2-я итерация. Шаг 5. $\tilde{S} = \{x_1, x_4\}$.

$d(\tilde{x}) = 11 = 0^* + 11 = d(x_1) + \omega(x_1, x_5)$, $d(\tilde{x}) = 11 \neq 6^* + 6 = d(x_4) + \omega(x_4, x_5)$. Включаем дугу (x_1, x_5) в кратчайший путь. $\tilde{x} = x_1$.

Шаг 6. $\tilde{x} = s = x_1$, завершение второго этапа.

Итак, кратчайший путь от вершины x_1 до вершины x_6 построен. Его длина (вес) равна 15, сам путь образует следующая последовательность дуг $\mu = (x_1, x_5) - (x_5, x_6)$.

3.3.3 Результаты и выводы: в результате проведенного занятия студенты:
- освоили понятия об алгоритме Дейкстры;
- приобрели умения и навыки работы с алгоритмом Дейкстры.

3.4 Практическое занятие № 4 (2 часа).

Тема: «Нахождение экстремальных путей в сети с отрицательными весами: Алгоритм Беллмана – Мура»

3.4.1 Задание для работы:

1. Нахождение кратчайших путей в сети с отрицательными весами.
2.Алгоритм Беллмана - Мура

3.4.2 Краткое описание проводимого занятия:

1. Нахождение кратчайших путей в сети с отрицательными весами.
2.Алгоритм Беллмана - Мура

Задана весовая матрица сети G.

а) построить по этой матрице сеть,

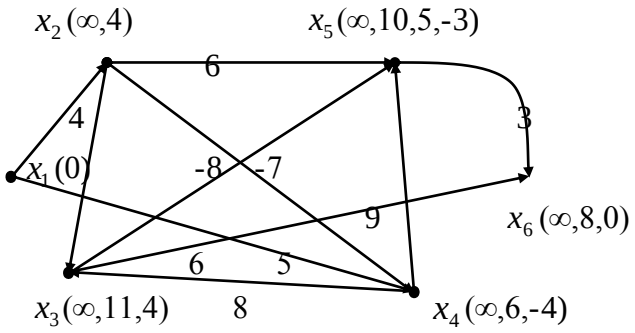
б) найти минимальный путь из вершины $s = x_1$ в вершину $t = x_6$ или $t = x_7$ по алгоритму Беллмана-Мура,

в) найти вес минимального пути.

Пусть граф G задан весовой матрицей W.

$$W = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 4 & \infty & 6 & \infty & \infty \\ \infty & - & 7 & -8 & 6 & \infty \\ \infty & \infty & - & \infty & -7 & 5 \\ \infty & \infty & 8 & - & 9 & \infty \\ \infty & \infty & \infty & \infty & - & 3 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix} \end{array}.$$

Изобразим эту сеть на рисунке и рассчитаем кратчайший путь от узла x_1 до узла x_6.



Этап 1. Шаг 1. $\tilde{x} = x_1$, $d(x_1) = 0$, $d(x_j) = \infty$, $j = \overline{2,6}$, $Q = \{x_1\}$.

Шаг 2. 2а). $\tilde{x} = x_1$, $Q = Q / \{x_1\} = \varnothing$.

2б). Пусть $\tilde{S}$ - множество вершин непосредственно достижимых из вершины $\tilde{x}$. $\tilde{S} = \{x_2, x_4\}$, $d(x_2) = \min\{\infty, 0+4\} = 4$.

2бб). 4<∞? Да.

2бв). $Q = \{x_2\}$, x_2 надо было поставить в конец очереди, но Q было пусто, поэтому конец очереди совпал с началом.

2б). $d(x_4) = \min\{6, 0+6\} = 6$.

2бб). $6 < \infty$? Да.

2бв). $Q = \{x_2, x_4\}$.

Шаг 3. $Q = \varnothing$? Нет. Переход на начало второго шага.

Вторая итерация. Шаг 2. 2а). $\tilde{x} = x_2$, $Q = Q / \{x_2\} = \{x_4\}$, $\tilde{S} = \{x_3, x_4, x_5\}$.

2б). $d(x_3) = \min\{\infty, 4+7\} = 11$.

2бб). 11<∞? Да.

2бв). $Q = \{x_4, x_3\}$.

2б). $d(x_4) = \min\{6, 4-8\} = -4$.

2бб). -4<6? Да.

2бв). $Q = \{x_4, x_3\}$. Вершину x_4 надо поставить в начало очереди, но она уже стоит там.

2б). $d(x_5) = \min\{\infty, 4+6\} = 10$.

2бб). 10<∞? Да.

2бв). $Q = \{x_4, x_3, x_5\}$.

Шаг 3. $Q = \varnothing$? Нет, переход на третью итерацию второго шага.

Третья итерация. Шаг 2. 2а). $\tilde{x} = x_4$, $Q = Q / \{x_4\} = \{x_3, x_5\}$, $\tilde{S} = \{x_3, x_5\}$.

2б). $d(x_3) = \min\{11, -4+8\} = 4$.

2бб). 4<11? Да.

2бв). $Q = \{x_6, x_5\}$. Вершину x_3 надо поставить в начало очереди, но она уже там.

2б). $d(x_5) = \min\{10, -4+9\} = 5$.

2бб). 5<10? Да.

2бв). $Q = \{x_6, x_3\}$. Вершину x_5 передвигаем из конца очереди в начало.

Шаг 3. $Q = \varnothing$? Нет, переход на четвертую итерацию второго шага.

Четвертая итерация. Шаг 2. 2а). $\tilde{x} = x_5$, $Q = Q / \{\tilde{x}\} = \{x_6\}$, $\tilde{S} = \{x_6\}$.

2б). $d(x_6) = \min\{\infty, 5+3\} = 8$.

2бб). 8<∞? Да.

2бв). $Q = \{x_6, x_6\}$.

Шаг 3. $Q = \varnothing$? Нет.

Пятая итерация. Шаг 2. 2а). $\tilde{x} = x_3$, $Q = Q / \{\tilde{x}\} = \{x_6\}$, $\tilde{S} = \{x_6, x_6\}$.

2б). $d(x_5) = \min\{5, 4-7\} = -3$.

2бб). -3<5? Да.

2бв). $Q = \{x_6, x_6\}$.

2б). $d(x_6) = \min\{8, 4+5\} = 8$.

2бб). 9<8? Нет.

Шаг 3. $Q = \varnothing$? Нет.

Шестая итерация. Шаг 2. 2а). $\tilde{x} = x_5$, $Q = Q / \{\tilde{x}\} = \{x_6\}$, $\tilde{S} = \{x_6\}$.

2б). $d(x_6) = \min\{8, -3+3\} = 0$.

2бб). 0<8? Да.

2бв). $Q = \{x_6\}$. Q содержало только вершину x_6 и она встала в начало очереди.

Шаг 3. $Q = \varnothing$? Нет.

Седьмая итерация. Шаг 2. 2а). $\tilde{x} = x_6$, $Q = Q / \{\tilde{x}\} = \varnothing$. $\tilde{S} = \varnothing$.

Шаг 3. $Q = \varnothing$. Конец первого этапа. Найдены минимальные расстояния до всех вершин от первой вершины. Эти расстояния таковы: $d(x_2) = 4$, $d(x_3) = 4$, $d(x_4) = -4$, $d(x_5) = -3$, $d(x_6) = 0$.

Второй этап. Шаг 4. Полагаем $\tilde{x} = x_6$. Пусть $\tilde{S}$ - множество вершин, непосредственно предшествующих $\tilde{x}$. $\tilde{S} = \{x_3, x_5\}$.

Первая итерация. $d(\tilde{x}) - d(x_6) = 0 \neq 4+5 = d(x_3) + \omega(x_3, x_6)$, $d(\tilde{x}) - d(x_6) = 0 = -3+3 = d(x_5) + \omega(x_5, x_6)$. Включаем дугу (x_5, x_6) в кратчайший путь. $\tilde{x} = x_5$. Возвращаемся на четвертый шаг.

Вторая итерация. $\tilde{x} = s$? Нет.

$\tilde{S} = \{x_2, x_3, x_4\}$. $d(\tilde{x}) - d(x_5) = -3 = 4-7 = d(x_3) + \omega(x_3, x_5)$,

$d(\tilde{x}) - d(x_5) = -3 \neq 4+6 = d(x_2) + \omega(x_2, x_5)$,

$d(\tilde{x}) - d(x_5) = -3 \neq -4+9 = d(x_4) + \omega(x_4, x_5)$. Включаем дугу (x_3, x_5) в кратчайший путь. $\tilde{x} = x_3$. Возвращаемся на четвертый шаг.

Третья итерация. $\tilde{x} = s$? Нет. $\tilde{S} = \{x_2, x_4\}$.

$d(\tilde{x}) - d(x_3) = 4 \neq 4+7 = d(x_2) + \omega(x_2, x_3)$,

$d(x_3) = d(x_4) = 4 = -4 + 8 = d(x_4) + \omega(x_3, x_4)$. Включаем дугу (x_3, x_4) в кратчайший путь. $\tilde{x} = x_4$. Возвращаемся на четвертый шаг.

Четвертая итерация. $\tilde{x} = s$? Нет. $\tilde{S} = \{x_4, x_2\}$.

$d(x_4) = -4 \neq 0 + 6 = d(x_1) + \omega(x_1, x_4)$,

$d(x_4) = -4 = 4 - 8 = d(x_2) + \omega(x_2, x_4)$. Включаем дугу (x_2, x_4) в кратчайший путь. $\tilde{x} = x_2$. Возвращаемся на четвертый шаг.

Пятая итерация. $\tilde{x} = s$? Нет. $\tilde{S} = \{x_1\}$.

$d(x_2) = 4 = 0 + 4 = d(x_1) + \omega(x_1, x_2)$. Включаем дугу (x_1, x_2) в кратчайший путь. $\tilde{x} = x_1$. Возвращаемся на четвертый шаг.

Шестая итерация. $\tilde{x} = s$? Да. Задача закончена. Искомый кратчайший путь от вершины x_1 до вершины x_6 имеет нулевой вес и состоит из следующих дуг $(x_1, x_2) - (x_2, x_4) - (x_4, x_3) - (x_3, x_5) - (x_5, x_6)$.

3.4.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили понятия об алгоритме Беллмана-Мура
- приобрели умения и навыки работы с алгоритмом Беллмана-Мура.

3.3 Практическое занятие № 5 (2 часа).

Тема: «Потоки в сетях, задача о максимальном потоке и минимальном разрезе. Теорема Форда - Фалкерсона. Компьютерные технологии реализации алгоритма Форда-Фалкерсона»

3.5.1 Задание для работы:

1. Потоки в сетях, задача о максимальном потоке и минимальном разрезе.
2. Теорема Форда - Фалкерсона

3.5.2 Краткое описание проводимого занятия:

1. Потоки в сетях, задача о максимальном потоке и минимальном разрезе.
2. Теорема Форда - Фалкерсона

Задания

Пропускные способности дуг заданы матрицей. С помощью алгоритма Форда-Фалкерсона построить максимальный поток от s к t и указать минимальный разрез, отделяющий s от t.

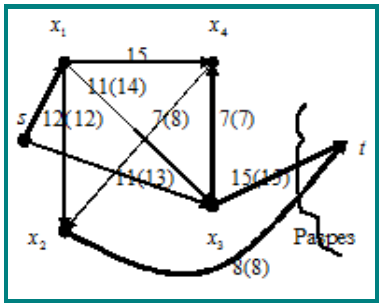
1). $\begin{pmatrix} - & 18 & 16 & - & - & 9 & - \\ - & - & 8 & 11 & 7 & - & 13 \\ - & - & - & - & 13 & - & 19 \\ - & - & 10 & - & - & 15 & - \\ - & - & - & 17 & - & 28 & - \\ - & - & - & - & - & - & 14 \\ - & - & - & - & - & - & - \end{pmatrix}$, 2) $\begin{pmatrix} - & 9 & - & 11 & - & 17 & - \\ - & - & 6 & - & 8 & - & 12 \\ - & - & - & - & - & - & 7 \\ - & 5 & - & - & - & 5 & 4 \\ - & - & - & - & - & 7 & - \\ - & - & - & - & - & - & 9 \\ - & - & - & - & - & - & - \end{pmatrix}$.

Пример . Пропускные способности дуг заданы следующей матрицей. Построить максимальный поток от s к t и указать минимальный разрез, отделяющий s от t.

$$W = \begin{array}{c} \\ s \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ t \end{array} \begin{array}{c} \begin{array}{cccccc} s & x_1 & x_2 & x_3 & x_4 & t \end{array} \\ \begin{pmatrix} - & 12 & - & 13 & - & - \\ - & - & 11 & 14 & 15 & - \\ - & - & - & - & - & 8 \\ - & - & - & - & 7 & 15 \\ - & - & 8 & - & - & - \\ - & - & - & - & - & - \end{pmatrix} \end{array}.$$

Этап 1. Путь $s \xrightarrow{12} x_1 \xrightarrow{14} x_3 \xrightarrow{15} t.$

$\delta = \min(12, 14, 15) = 12.$ Увеличим по этому пути поток до 12 единиц, ребро (s, x_1) становится насыщенным. Поставим величину потока на дугах (x_1, x_3) и (x_3, t).



$\delta = \min(3, 15-12) = 3.$ Поток можно увеличить на три единицы. Дуга (x_3, t) станет насыщенной. Путь $s \xrightarrow{3(3)} x_3 \xrightarrow{7} x_4 \xrightarrow{8} x_2 \xrightarrow{8} t.$ Можно увеличить поток на семь единиц;

Дуга (x_3, x_4) станет насыщенной, потоки примут вид

$$s \xrightarrow{10(3)} x_3 \xrightarrow{7(7)} x_4 \xrightarrow{7(8)} x_2 \xrightarrow{7(8)} t.$$

Больше путей нет. Конец первого этапа.

Этап 2. Рассмотрим теперь маршруты, содержащие противоположные дуги. Маршрут $s \xrightarrow{10(3)} x_3 \xleftarrow{12(4)} x_1 \xrightarrow{11} x_2 \xrightarrow{7(8)} t.$ Поток можно увеличить на единицу на дуге (x_2, t). Тогда потоки по дугам этого маршрута станут такими $s \xrightarrow{11(3)} x_3 \xleftarrow{11(4)} x_1 \xrightarrow{1(1)} x_2 \xrightarrow{8(8)} t.$ Дуга (x_2, t) стала насыщенной.

Больше маршрутов нет. Поток максимален. Делаем разрез вокруг t по насыщенным дугам и получаем его величину 15+8=23 единицы.

2. Другие вопросы практического занятия рассмотреть в форме докладов студентов.

3.3.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили понятия об алгоритме поиска максимального потока.
- приобрели умения и навыки работы с алгоритмом поиска максимального потока.

3.6 Практическое занятие № 6 (2 часа).

Тема: «Элементы сетевого планирования: критические пути, работы, резервы»

3.6.1 Задание для работы:

1.Элементы сетевого планирования.
2. Критические пути, работы, резервы.

3.6.2 Краткое описание проводимого занятия:

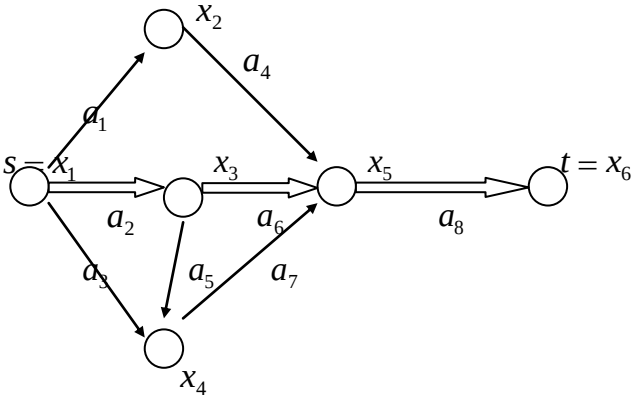
1.Элементы сетевого планирования.

2. Критические пути, работы, резервы.

Пример построения сети по таблице последовательности работ.

Последовательность работ		
Исходная Работа	Опирается на работу	Продолжительность работ
a_1	-	3
a_2	-	6
a_3	-	4
a_4	a_1	5
a_5	a_2	1
a_6	a_2	9
a_7	a_3, a_5	6
a_8	a_4, a_6, a_7	8

Работы a_1, a_2 и a_3 не имеют предшествующих, поэтому реализация проекта начинается с этих работ, и изобразятся они дугами, выходящими из одной вершины – события x_1. Работе a_4 предшествует работа a_1, поэтому дуга a_4 на сети изображена вслед за дугой a_1.То же самое с дугами a_5 и a_6. Далее надо изобразить дуги a_7 и a_8. Работа a_7 опирается на работы a_3 и a_5. Итоговая работа a_8 опирается на a_4, a_6 и a_7. На рисунках сети не рекомендуется во избежании путаницы изображать одновременно выполняемые работы параллельными дугами. Однако можно вводить дополнительные события и фиктивные работы (нулевой продолжительности), которые изображаются штриховыми линиями. Если бы, к примеру, работа a_5 опиралась бы еще на a_1, то между событиями x_2 и x_3 пришлось бы ввести штриховую дугу.



Имея сеть работ некоторого проекта можно посчитать время выполнения всего проекта и различных его частей, состоящих из разного набора работ. Для этого введем еще несколько определений. Определим сначала минимальное время, за которое можно выполнить все работы комплекса. Для этого найдем продолжительность $t(\mu_i)$ всех полных путей μ_i. В нашем случае таких путей четыре: $\mu_1: 1-2-5-6$; $\mu_2: 1-3-5-6$; $\mu_3: 1-4-5-6$; $\mu_4: 1-3-4-5-6$. Их продолжительности $t(\mu_1)=16$, $t(\mu_2)=23$, $t(\mu_3)=18$, $t(\mu_4)=21$. Наиболее продолжителен второй путь. Такой путь называют критическим. Этот путь определяет минимальное время выполнения всех работ комплекса. Ми-

нимальное время называют критическим сроком и обозначают $t_{кр.}$. Итак, в рассматриваемом примере $t_{кр.} = 23.$

Все работы и события, лежащие на критическом пути, называют критическими, все остальные работы и события – некритическими. Задержка любой критической работы вызывает задержку выполнения всего комплекса. Следовательно, чтобы уменьшить время выполнения комплекса работ, надо сократить сроки критических работ. Некритические работы допускают некоторое запаздывание их выполнения без нарушения критического срока. Это запаздывание измеряется резервом времени событий и работ.

Свершением события называется момент, к которому заканчиваются все входящие в него работы и может быть начата любая выходящая работа. Некоторые события можно совершать в разные моменты, то есть варьировать свершение этих событий. Например, событие x_2 может свершиться через три дня (по окончании работы a_1), но может наступить и позже на срок до семи дней, поскольку на пути μ_1, где лежит это событие, есть резерв времени $t_{кр.} - t(\mu_1) = 23 - 16 = 7$ дней. Поэтому для событий различают ранний и поздний сроки свершения.

Ранним сроком $t_p(x_j)$ свершения события x_j называется самый ранний момент времени, к которому завершатся все работы, предшествующие этому событию. Ранние сроки для всех событий могут быть рассчитаны по формуле

$$t_p(x_j) = \max_{(x_i, x_j) \in U_j^+} (t_p(x_i) + t(x_i, x_j)),$$

где U_j^+ - множество работ, входящих в x_j событие, $t_p(x_i)$ - ранний срок свершения начального события работы (x_i, x_j), $t(x_i, x_j)$ - продолжительность работы (x_i, x_j).

Поздним сроком $t_n(x_i)$ свершения события x_i называется самый поздний момент времени, после которого остается ровно столько времени, сколько необходимо для завершения всех работ, следующих за этим событием.

В нашем случае $t_n(x_6) = 23$. Чтобы не нарушался критический срок, событие x_5 должно произойти в крайнем случае на восемь дней раньше, поэтому $t_n(x_5) = 23 - 8 = 15$. Аналогично, $t_n(x_2) = 15 - 5 = 10$. Таким образом, поздние сроки событий рассчитываются по формуле

$$t_n(x_i) = \min_{(x_i, x_j) \in U_i^-} (t_n(x_j) - t(x_i, x_j)),$$

где U_i^- - множество работ, выходящих из x_i события, $t_n(x_j)$ - поздний срок свершения конечного события работы (x_i, x_j).

Разности между поздним и ранним сроками свершения события x_i составляет

резерв времени $R(x_i)$ этого события $R(x_i) = t_n(x_i) - t_p(x_i)$.

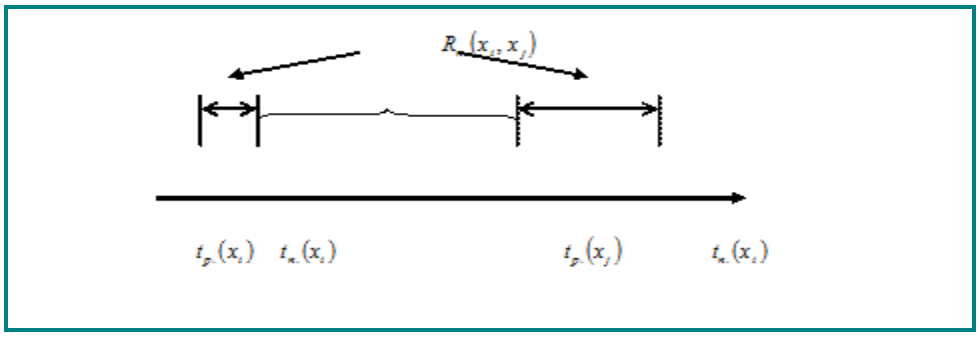
Резерв показывает, на какой предельно допустимый срок может задержаться свершение события x_i без изменения срока наступления итогового события t. У критических событий ранние и поздние сроки совершения совпадают, ибо резерв времени у них равен нулю. Зная сроки свершения событий, можно найти ранние и поздние сроки начала и окончания работы (x_i, x_j). Очевидно, что

$$\begin{cases} t_{р.н.}(x_i, x_j) = t_{р.}(x_i), & t_{р.о.}(x_i, x_j) = t_{р.}(x_i) + t(x_i, x_j), \\ t_{п.о.}(x_i, x_j) = t_{п.}(x_j), & t_{п.н.}(x_i, x_j) = t_{п.}(x_j) - t(x_i, x_j). \end{cases}$$

Для работ определяются два резерва времени. Полный резерв времени работы – это максимальное количество времени, на которое можно задержать начало работы или увеличить ее продолжительность, не нарушая критический срок

$$R_{п.}(x_i, x_j) = t_{п.}(x_j) - t_{р.}(x_i) - t(x_i, x_j).$$

Формулу (5.1.5) можно проиллюстрировать следующим рисунком.



Отдельные работы, помимо полного резерва, имеют свободный резерв времени, составляющий часть полного резерва, остающуюся после исключения резерва времени $R(x_j)$ конечного события x_j данной работы

$$R_{с.}(x_i, x_j) = t_{р.}(x_j) - t_{р.}(x_i) - t(x_i, x_j).$$

Свободный резерв времени – это запас времени, на который можно отсрочить начало работы или увеличить ее продолжительность при условии, что она начнется в свой ранний срок и при этом ранние сроки начала последующих работ не изменятся. Понятно, что все резервы критических работ равны нулю.

3.6.3 Результаты и выводы: в результате проведенного занятия студенты:
- освоили элементы сетевого планирования: критические пути, работы, резервы.

3.7 Практическое занятие № 7 (2 часа).

Тема: «Построение остовного дерева (леса) графа: алгоритмы Краскала и Прима»

3.7.1 Задание для работы:

1. Построение остовного дерева (леса) графа.
2. алгоритмы Краскала и Прима..

3.7.2 Краткое описание проводимого занятия:

1. Построение остовного дерева (леса) графа.
2. алгоритмы Краскала и Прима; задача об остове экстремального веса.

Пример выполнения типового задания. Для графа, заданного матрицей весов,

а)построить по этой матрице сеть(исходный граф),

б) построить остов наименьшего веса,

в) найти его вес.

Поскольку матрица симметрическая, то граф дан неориентированный.

$$W = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 5 & 10 & 14 & \infty & \infty \\ 5 & - & 5 & 6 & \infty & \infty \\ 10 & 5 & - & 7 & 8 & 9 \\ 14 & 6 & 7 & - & 4 & \infty \\ \infty & \infty & 8 & 4 & - & 12 \\ \infty & \infty & 9 & \infty & 12 & - \end{pmatrix} \end{array}.$$

Шаг1. $S' = \{x_1\}$, $S'' = \{x_2, x_3, x_4, x_5, x_6\}$, $U' = \varnothing$.

Первая итерация. Шаг 2.

$d(S', S'') = \omega(x_1, x_2) = 5$, $S' = \{x_1, x_2\}$, $S'' = \{x_3, x_4, x_5, x_6\}$,

$U' = \{(x_1, x_2)\}$.

Шаг 3. $S' \neq S$, переход на начало второго шага.

Вторая итерация. Шаг 2.

$d(S', S'') = \omega(x_2, x_3) = 5$, $S' = \{x_1, x_2, x_3\}$, $S'' = \{x_4, x_5, x_6\}$,

$U' = \{(x_1, x_2), (x_2, x_3)\}$.

Шаг 3. $S' \neq S$, переход на начало второго шага.

Третья итерация. Шаг 2.

$d(S', S'') = \omega(x_2, x_4) = 6$, $S' = \{x_1, x_2, x_3, x_4\}$, $S'' = \{x_5, x_6\}$,

$U' = \{(x_1, x_2), (x_2, x_3), (x_2, x_4)\}$.

Шаг 3. $S' \neq S$, переход на начало второго шага.

Четвертая итерация. Шаг 2. $d(S', S'') = \omega(x_4, x_5) = 4$, $S' = \{x_1, x_2, x_3, x_4, x_5\}$, $S'' = \{x_6\}$,

$U' = \{(x_1, x_2), (x_2, x_3), (x_2, x_4), (x_4, x_5)\}$.

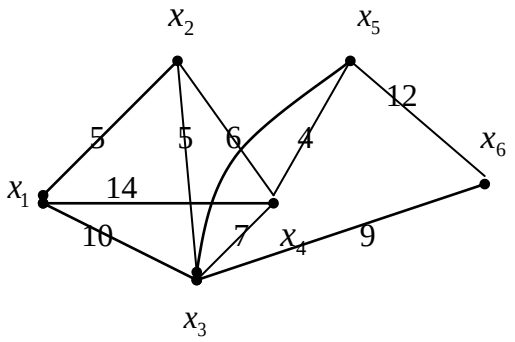
Шаг 3. $S' \neq S$, переход на начало второго шага.

Пятая итерация. Шаг 2. $d(S', S'') = \omega(x_3, x_6) = 9$, $S' = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $S'' = \varnothing$,

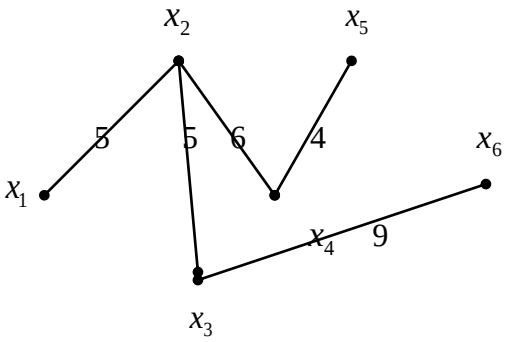
$U' = \{(x_1, x_2), (x_2, x_3), (x_2, x_4), (x_4, x_5), (x_3, x_6)\}$.

Шаг 3. $S' = S$. Итак, получен остовный граф. $G' = (S', U')$ изображен на рисунке справа, его вес $\omega(G') = 5+5+6+4+9 = 29$.

Исходный граф



Остовный граф



Краткая теоретическая справка

Деревья являются простейшим классом графов. Для них выполняются многие свойства, которые не всегда выполняются для обычных графов. Кроме того, деревья широко применяются в программировании при различного рода обработке данных, в частности, в алгоритмах сортировки, кодирования и т.п. Подробно алгоритмы работы с деревьями будут рассматриваться позднее в других курсах, а сейчас только краткое знакомство.

Дерево – это связный граф без циклов. Несколько деревьев (или несвязный граф без циклов) составляют *лес*. Таким образом, дерево является компонентой связности.

Пусть $G = (S, U)$ и $|S| = n$, $|U| = m$. Тогда справедлива эквивалентность следующих утверждений:

1). G - дерево;

2). G - связный граф и $m = n - 1$;

3). G - ациклический граф и $m = n - 1$;

4). любые две несовпадающие вершины графа соединяет единственная простая цепь;

5). G - ациклический граф, обладающий тем свойством, что если какую- либо пару его несмежных вершин соединить ребром, то полученный граф будет содержать ровно один цикл.

3.7.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили алгоритмы Краскала и Прима; задачу об остове экстремального веса;
- приобрели умения и навыки применения алгоритмов Краскала и Прима, решения задачи об остове экстремального веса.

3.8 Практическое занятие № 8 (2 часа).

Тема: «Алгоритмы обхода и поиска в графе: поиск в глубину и в ширину. Эйлеровы циклы в графах. Поиск расстояния между всеми парами вершин. Алгоритм Уоршалла - Флойда»

3.8.1 Задание для работы:

1. Обходы графов.
2. Эйлеровы циклы в графах.
3. Поиск расстояния между всеми парами вершин. Алгоритм Уоршалла -Флойда

3.8.2 Краткое описание проводимого занятия:

1. Обходы графов.
2. Эйлеровы циклы в графах.
3. Поиск расстояния между всеми парами вершин. Алгоритм Уоршалла -Флойда

Обходы графов. Пример на алгоритмы поиска в глубину и в ширину.

Обходы графов. *Обход графа – это некоторое систематическое перечисление его вершин (ребер). Среди всех обходов наиболее известны поиск в глубину и в ширину. Алгоритмы такого поиска лежат в основе многих алгоритмов на графах.*

При поиске используется некоторая структура данных Т, в которую помещаются вершины графа. Для обозначения пройденных вершин заводят дополнительный массив пометок этих вершин.

Поиск основывается на следующих действиях.

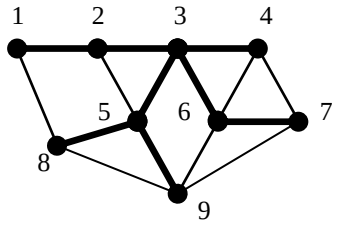
1.Сначала все вершины считаются неотмеченными.

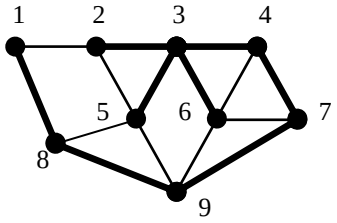
2.Выбирается любая вершина (нач. поиска), заносится в структуру данных Т и помечается.

3.Следующие действия выполняются в цикле до тех пор, пока структура Т не станет пустой: из структуры данных Т выбирается вершина *u*; она выдается в качестве очередной пройденной вершины; перебираются все вершины из Г(u), и все те, которые не помечены, тоже заносятся в структуру Т и помечаются.

Если T – это стек (LIFO), то обход называется поиском *в глубину* (т.е. первым делом из структуры T извлекается вершина, попавшая туда последней). Если T – это очередь (FIFO), то обход называется поиском *в ширину*. При поиске в глубину находятся более длинные пути.

Если граф G связен и конечен, то поиск в ширину или поиск в глубину обойдет ***все вершины графа по одному разу.***





Пример на алгоритмы поиска в глубину и в ширину.Рассмотрим алгоритмы поиска в глубину и в ширину на примере. В качестве стартовой возьмем вершину с номером 3. Тогда поиск в ширину даст последовательность вершин: 3→T. T={3}⇒ **3**: {2,5,6,4}→T; T={2,5,6,4}⇒ **2**: {1}→T; T={5,6,4,1}⇒ **5**: {8,9}→T; T={6,4,1,8,9}⇒ **6**: {7}→T; T={4,1,8,9,7}. Окружения всех этих вершин уже отмечены ⇒ они будут выданы по порядку. Итак, выполнен обход всех вершин графа в следующем порядке: **3,2,5,6,4,1,8,9,7**. При поиске в глубину начало такое же: 3→T. T={3} ⇒ **3**: {2,5,6,4}→T; T={2,5,6,4}⇒ **4**: {7}→T; T={2,5,6,7}⇒ **7**: {9}→T; T={2,5,6,9} ⇒ **9**: {8}→T; T={2,5,6,8}⇒ **8**: {1}→T; T={2,5,6,1}. Оставшиеся вершины выдаются по порядку. В итоге последовательность вершин: **3,4,7,9,8,1,6,5,2.**

Любой из рассмотренных обходов позволяет построить остовное дерево исходного графа с корнем в исходной вершине (связный подграф без циклов).

Эйлеровы графы

Цикл в графе называется эйлеровым*, если он содержит все ребра графа. Связный граф, в котором существует эйлеров цикл, называется* эйлеровым графом. Эйлеровой цепью *(или путем) является цепь (путь), которая включает все ребра (дуги) графа по одному разу.* Собственная эйлерова цепь – *это эйлерова цепь, которая не является эйлеровым циклом.*

Эйлеров граф можно нарисовать, не отрывая карандаша от бумаги.

Теорема Эйлера. *Связный граф является эйлеровым тогда и только тогда, когда степени всех его вершин четны.*

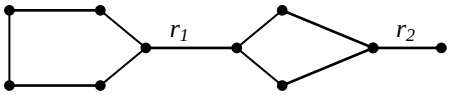
Вспомним задачу о кенигсбергских мостах. Она сводится к вопросу – является ли граф, представляющий эту задачу, эйлеровым? Если вершины будут обозначать участки суши, а ребра – мосты, то получим граф следующего вида:
Очевидно, что этот граф не эйлеров, т.к. все его вершины имеют нечетные степени; поэтому требуемый цикл не существует. Если отменить требование возврата в ту же точку, то вопрос о пути по всем мостам сведется к вопросу о существовании собственной эйлеровой цепи.

Граф имеет собственную эйлерову цепь (путь) ⇔ когда он связный и ровно две его вершины имеют нечетную степень.

Вершины нечетной степени являются началом и концом эйлеровой цепи.

А теперь разберемся, как найти эйлеров цикл. Сначала еще определение.

Мостом (или *перешейком*) называется такое ребро графа G, удаление которого увеличивает число связных компонент. Если ребро r принадлежит некоторому циклу C, то оно не может быть мостом.



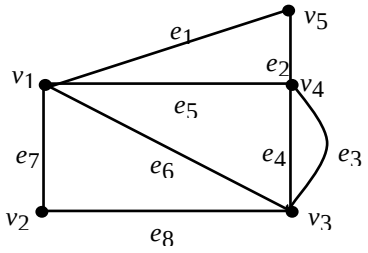
В данном графе ребра r_1 и r_2 являются мостами.

Для нахождения эйлеровой цепи в связном графе, который имеет вершины только четной степени, используют *алгоритм Флери*:

1) Движение начинается из произвольной вершины графа; идем по ребрам, включая эти ребра в эйлерову цепь и удаляя их из графа.

2) В очередной вершине выбираем путь по перешейку только в том случае, если нет пути по циклу.

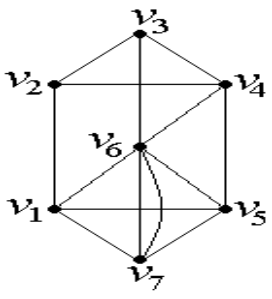
3) Если в графе остаются ребра, которые нельзя использовать для продолжения имеющегося пути, то следует начать строить простой замкнутый цикл из уже пройденной вершины и инцидентного ей ребра, если последнее ранее не использовалось.



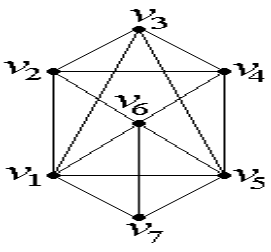
Пусть требуется построить эйлеров цикл для графа, изображенного на рисунке. Начать построение эйлерового цикла можно с любого ребра графа. Начиная с e_1, получим цикл v_1, e_1, v_5, e_2, v_4 ,e_3, v_3 ,e_4 , v_4 ,e_5 ,v_1,e_6 ,v_3, e_8, v_2, e_7 ,v_1. В данном случае сразу получили эйлерову цепь.

Задания

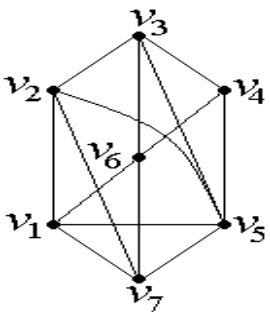
Найти Эйлерову цепь в неориентированном графе.



а)



б)



в)

Пример выполнения задания. Найти Эйлерову цепь в неориентированном графе G, изображенном на рисунке.

Решение. Прежде, чем приступать к нахождению Эйлеровой цепи, необходимо проверить степени вершин графа G − согласно утверждению 2, для существования Эйлеровой цепи, необходимо и достаточно, чтобы в графе G ровно 2 вершины нечетной степени.

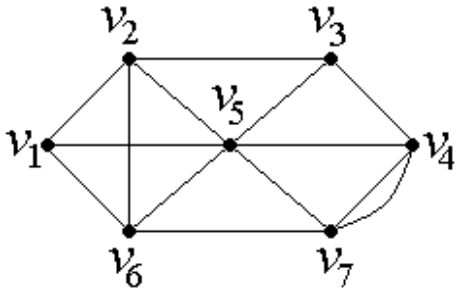


Рис. 1

В рассматриваемом графе нечетные степени имеют вершины v_3 и v_1 (степень этих вершин равна 3). Соединяя эти вершины фиктивным ребром так, как показано на рис. 2, получаем граф G':

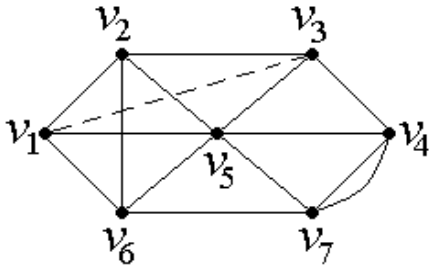


Рис. 2.

Поскольку в конечном итоге будет получена цепь, то очевидно, что началом и концом этой цепи будут вершины с нечетными степенями. Поэтому, следуя описанному выше алгоритму, будем циклы μ_i так, чтобы хотя бы один из них начинался или кончался на вершинах v_3 или v_1.

Пусть цикл μ_1 составят ребра, проходящие через следующие вершины: $v_3\ v_4\ v_7\ v_6\ v_1\ v_2\ v_3$. Согласно алгоритму, удаляем из G' все ребра, задействованные в цикле μ_1. Теперь граф G' будет таким, как показано на рис. 3.

Составляем следующий цикл μ_2: $v_4\ v_5\ v_6\ v_2\ v_5\ v_7\ v_4$. Граф G' после удаления ребер, составляющих цикл μ_2, изображен на рис. 4.

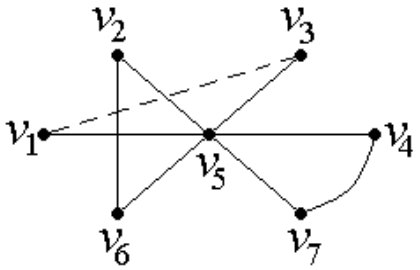


Рис.3

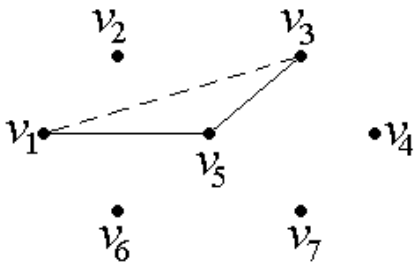


Рис. 4

Очевидно, что последний цикл μ_3 будет состоять из $v_3\ v_5\ v_1|v_3$, где последнее ребро, соединяющее вершины v_1 и v_3 – фиктивно. После удаления ребер, составляющих цикл μ_3, в графе G' не останется ни одного ребра.

Теперь по общим вершинам склеиваем полученные циклы. Поскольку μ_1 и μ_2 имеют общую вершину v_4, то, объединяя их, получим следующий цикл: $v_3\ v_4\ v_5\ v_6\ v_2\ v_5\ v_7\ v_4\ v_7\ v_6\ v_1\ v_2\ v_3$. Теперь склеим получившийся цикл с циклом μ_3: $v_3\ v_4\ v_5\ v_6\ v_2\ v_5\ v_7\ v_4\ v_7\ v_6\ v_1\ v_2\ v_3\ v_5\ v_1|v_3$. Удаляя фиктивное ребро, получаем искомую Эйлерову цепь: $v_3\ v_4\ v_5\ v_6\ v_2\ v_5\ v_7\ v_4\ v_7\ v_6\ v_1\ v_2\ v_3\ v_5\ v_1$.

Алгоритм выделения эйлерова цикла в связном мультиграфе с четными степенями вершин

5) Выделим из G цикл μ_1. (так как степени вершин четны, то висячие вершины отсутствуют). Положим l=1, G'=G.

6) Удаляем из G' ребра, принадлежащие выделенному циклу μ_1. Полученный псевдограф снова обозначаем как G'. Если в G' отсутствуют ребра, то переходим к шагу 4. Если ребра есть, то выделяем из G' цикл μ_{l+1} и переходим к шагу 3.

7) Присваиваем l:=l+1 и переходим к шагу 2.

8) По построению выделенные циклы содержат все ребра по одному разу. Если l:=1, то искомый Эйлеров цикл найден (конец работы алгоритма). В противном случае находим циклы, содержащие хотя бы по одной общей вершине (в силу связности графа это всегда можно сделать). Склеиваем эти циклы. Повторяем эти операции, пока не останется один цикл, который является искомым.

3.8.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили понятия об алгоритмах обхода и поиска в графе: поиск в глубину и в ширину. Эйлеровых циклах в графах;
- приобрели умения и навыки выполнения алгоритмов обхода и поиска в графе.

Тема: «Поиск расстояния между всеми парами вершин. Алгоритм Уоршалла -Флойда**»**

Кратчайшие пути. *Задача поиска кратчайшего пути (наиболее дешевого? короткого?) «от пункта А до пункта В» имеет массу практических приложений и различные алгоритмы решения. Математическая постановка задачи имеет следующий вид.*

Рассматривается взвешенный граф (орграф) $G(V,E)$, ребрам (дугам) которого сопоставлены веса, обозначающие длину (или стоимость) пути из одного конца ребра в другой. Если из вершины v_i нет ребра (дуги) в вершину v_j, то вес ребра (v_i,v_j) считается равным ∞. Для ребер, являющихся петлями (диагональ матрицы смежности), их веса считаются равными 0. Все компоненты матрицы – веса ребер, соединяющих соответствующие вершины. Требуется определить кратчайший путь из одной вершины в другую.

Наиболее широко известны два алгоритма поиска кратчайших путей. Алгоритм Дейкстры находит кратчайшее расстояние от одной фиксированной вершины до другой и указывает сам путь, длина которого равна этому расстоянию. Алгоритм Флойда-Уоршалла позволяет найти кратчайшие расстояния между всеми парами вершин графа.

Алгоритм Флойда-Уоршалла

находит кратчайшие расстояния между всеми парами вершин графа

Идея: Строится специальная матрица *D*, элементы которой – кратчайшие пути между всевозможными парами вершин графа *G*. При определении кратчайшего пути выбирается минимум из «прямого» расстояния между смежными вершинами v_i и v_j и суммарного расстояния при проходе через дополнительную вершину. Затем – более длинные пути и т.д.

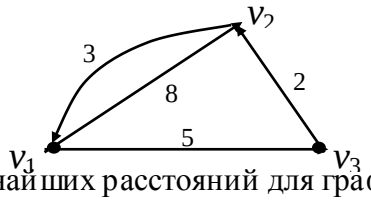
Обозначим через $d_{ij}^{(m)}$ длину кратчайшего пути из v_i в v_j с промежуточными вершинами во множестве $\{v_1,...,v_m\}$. Алгоритм использует три правила:

1) $d_{ij}^{(0)} = a_{ij}$ – вес дуги, соединяющий вершины v_i и v_j (т.е. первоначально матрица D – это исходная матрица весов).

2) $d_{ij}^{(m+1)} = \min\left(d_{ij}^{(m)}, d_{i,m+1}^{(m)} + d_{m+1,j}^{(m)}\right)$.

3) Длина кратчайшего пути из вершины v_i в вершину v_j: $d(v_i, v_j) = d_{ij}^{(n)}$.

Алгоритм строит матрицу за *n* шагов, т.е. строится матрица $D^{(1)}$, …, $D^{(n)}$=D.



Найдем матрицу кратчайших расстояний для графа v_1 v_2 v_3

$$D^{(0)} = \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & \infty \\ \infty & 2 & 0 \end{pmatrix} \begin{matrix} v_1 \\ v_2 \\ v_3 \end{matrix}$$

Элементы матрицы $D^{(1)}$ находим по правилу $d_{ij}^{(1)} = \min\left(d_{ij}^{(0)}, d_{i1}^{(0)} + d_{1j}^{(0)}\right)$. Первая строка и первый столбец не меняются. Получаем $D^{(1)} = \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ \infty & 2 & 0 \end{pmatrix}$. Элементы матрицы $D^{(2)}$ находим по правилу $d_{ij}^{(2)} = \min\left(d_{ij}^{(1)}, d_{i2}^{(1)} + d_{2j}^{(1)}\right)$. Без изменений второй столбец и вторая строка. $D^{(2)} = \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{pmatrix}$.

Элементы матрицы $D^{(3)}$ находим по правилу $d_{ij}^{(3)} = \min\left(d_{ij}^{(2)}, d_{i3}^{(2)} + d_{3j}^{(2)}\right)$.

$D^{(3)} = \begin{pmatrix} 0 & 7 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{pmatrix}$, $D = D^{(3)}$. Каждый из элементов (i,j) матрицы D равен наименьшему расстоянию между вершинами v_i и v_j.

3.8.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили понятия об алгоритме Уоршалла - Флойда;
- приобрели умения и навыки применения алгоритма Уоршалла – Флойда.

4. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПРОВЕДЕНИЮ СЕМИНАРСКИХ ЗАНЯТИЙ

Семинарские занятия не предусмотрены РУП.