ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

СТАВРОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ

КУРС ЛЕКЦИЙ

Б1.Б.1 Специальные главы математики

Шифр и наименование дисциплины в соответствии с учебным планом

09.04.02 Информационные системы и технологии

код и наименование направления подготовки/ специальности

Территориальные информационные системы

наименование профиля/специализации/магистерской программы

Ставрополь, 2017

Л Е К Ц И Я

по учебной дисциплине:
«Специальные главы математики»

Для магистров направления:
09.04.02 Информационные системы и технологии

Тема 1. Теория множеств

Лекция 1. . Основы теории множеств. Множества и подмножества. Операции над множествами. Упорядоченные множества. Отношения на множествах. Соответствие и функции. Мультимножества

Лекция 1. Основы теории множеств. Множества и подмножества. Операции над множествами. Упорядоченные множества. Отношения на множествах. Соответствие и функции. Мультимножества

Цель лекции

Дать систематизированные основы научных знаний по указанной теме занятия. Организовать целенаправленную познавательную деятельность студентов по овладению программным материалом по данной теме. Дать общее представление о роли познавательных процессов в жизни человека. Познакомить с основными понятиями по теме лекции.

Сформировать познавательный интерес к содержанию учебного материала помочь выработать у студентов стремление к самостоятельной работе и всестороннему овладению специальностью, развивать интерес к учебной дисциплине, содействовать активизации мышления студентов.

Учебные вопросы:

Основы теории множеств. Множества и подмножества. Операции над множествами. Упорядоченные множества. Отношения на множествах. Соответствие и функции. Мультимножества

1.1 Элементы и множества

Понятие множества принадлежит к числу фундаментальных неопределяемых понятий математики. Можно сказать, что *множество* — это любая определенная совокупность объектов. Объекты, из которых составлено множество, называются его *элементами.* Элементы множества различны и отличны друг от друга.

Примеры. Множество *S* страниц в данной книге. Множество N *натуральных* чисел 1, 2, 3, ...Множество P *простых* чисел 2, 3, 5, 7, 11, ...

Если объект *x* является элементом множества M, то говорят, что *x принадлежит M.* Обозначение: $x \in M$. В противном случае говорят, что *x не принадлежит M.* Обозначение: $x \notin M$.

Множества, как объекты, могут быть элементами других множеств. Множество, элементами которого являются множества, обычно называется *классом* или *семейством*.

Обычно в конкретных рассуждениях элементы всех множеств берутся из некоторого одного, достаточно широкого множества U (своего для каждого случая), которое называется *универсальным множеством*, или *универсумом*.

Из соображений формального удобства вводят еще так называемое "пустое множество", а именно, множество, не содержащее ни одного элемента. Его обозначают Ø, иногда символом 0 (совпадение с обозначением числа нуль не ведет к путанице, так как смысл символа каждый раз ясен).

Обозначим множество n первых натуральных чисел через $J_n = \{1,2, ..., n\}$. Множество X называется *конечным*, если оно эквивалентно J_n при некоторомn. Число n называется *количеством*, или числом, элементов множества X. Для конечного множества X через |X| обозначим число элементов этого множества, т.е. |X| = n. Пустое множество считается конечным с числом элементов равным нулю, т. е. |Ø| =0. Множества, не являющиеся конечными, называются *бесконечными*.

Если каждый элемент множества A входит во множество B, то A называется подмножеством B, а B называется надмножеством A. ПишутA⊆ B, B⊇A (A входит в B или A содержится в B, B содержит A). Очевидно, что если A⊆B, и B⊇A, то A = B. Пустое множество по определению считается подмножеством любого множества.

Если каждый элемент множества A входит в B, но множество B содержит хотя бы один элемент, не входящий в A, т. е. если A⊆B, иA≠B, то A называется *собственным подмножеством* B, а B - *собственным надмножеством* A. В этом случае пишут A⊂B, B⊃A. Например, запись A≠0 и A⊃0 означают одно и то же, а именно, что множество A не пусто.

Заметим еще, что надо различать элемент a и множество {a}, содержащее a в качестве единственного элемента. Такое различие диктуется не только тем, что элемент и множество играют неодинаковую роль (отношение не симметрично), но и необходимостью избежать противоречия. Так, пусть A = {a, b} содержит два элемента. Рассмотрим множество {A}, содержащее своим единственным элементом множество A. Тогда A содержит два элемента, в то время как {A} - лишь один элемент, и потому отождествление этих двух множеств невозможно. Поэтому рекомендуется применять запись $a \in A$, и не пользоваться записью $a \subset A$.

1.2 Способы задания множеств

Существуют два основных способа задания множеств:

1. Перечислительный способ (перечисление элементов)
2. Высказывательный способ (описание свойств элемента).

Перечислительный способ состоит в составлении полного списка элементов множества, заключенного в фигурные скобки и применяется только для конечных множеств с небольшим числом элементов. Множество записывается в следующей форме:

$X = \{X_1, X_2, \ldots, X_n\}$.

Примеры. A={Меркурий, Венера, Земля, Марс, Юпитер, Сатурн, Уран, Нептун, Плутон} - это множество планет солнечной системы, B={ФИТиУ, ФКСИС, ФРЭ, ФТК, ИЭФ, ВФ} - это множество факультетов университета.

Высказывательный способ состоит в задании такого свойства, наличие которого у элементов определенного множества является истиной. Описание свойства элементов обычно задается так: пусть

P(x) — утверждение, заключающееся в том, что элемент x обладает свойством P. Тогда запись

$X = \{x \in M \mid P(x)\}$

означает, что рассматриваемое множество X состоит из элементов некоторого множества M, обладающих свойством P.

Пример. Запись $A=\{x \in R \mid (x^2+2x-15=0)\}$ означает, что множество A состоит из действительных корней уравнения: $x^2+2x-15=0$. Это же множество можно также задать перечислительным способом: A={-5,3}.

Конечное множество может быть задано обоими способами, а бесконечные – лишь высказывательным способом. Пустые множества относятся к конечным.

Если рассматривать теорию множеств без ограничений на способы задания множеств, то такая теория называется *наивной теорией множеств*.

Еще при жизни Г. Кантора, создателя наивной теории множеств, были обнаружены многочисленные парадоксы в этой теории.

Приведем один из известных парадоксов Б. Рассела.

Пусть M — множество всех множеств. Тогда очевидно, что M ϵ M. Тем самым, существуют множества, содержащие себя как свой элемент.

Рассмотрим некоторое множество X, которое не содержит себя как свой элемент.

Пусть Y — множество всех таких множеств X, т. е. множество всех множеств, не содержащих себя как свой элемент.

Зададимся вопросом: каково множество Y? Содержит оно себя как элемент или нет? Возможны два случая: 1) содержит; 2) не содержит.

В первом случае YϵY. Тогда по определению множества Y имеем Y∉Y. Получили противоречие.

Во втором случае Y∉Y. Тогда по определению множества Y имеем YϵY. Получили также противоречие.

Иногда этот парадокс Рассела облекают в бытовую форму. Тогда появляется следующая парадоксальная ситуация. В полку имеется полковой

Брадобрей, который руководствуется следующим приказом. Брить бороды только у тех людей, которые сами себя не бреют. Спрашивается, может ли Брадобрей брить себе бороду? Получается так, что если он не бреет себе бороду, то по приказу он должен себя брить. Как только Брадобрей начинает брить себе бороду, то по приказу он не должен себя брить. Парадокс.

Избежать парадоксов удается только в рамках аксиоматической теории множеств, т. е. теории, которая ограничивает способы задания множеств специальной аксиоматикой.

Операции над множествами

2.1 Сравнение множеств

Если из элементов двух множеств можно составить пары таким образом, чтобы каждому элементу первого множества соответствовал определенный элемент второго множества, а каждому элементу второго множества соответствовал один и только один элемент первого множества, то говорят, что между такими двумя множествами установлено взаимно однозначное соответствие.

Два множества *равны,* если они являются подмножествами друг друга:

$A = B = A \subseteq B \ \& \ B \subseteq A.$

Мощность множества M обозначается как |M|. Для конечных множеств мощность - это число элементов. Например, |0| = 0, но |{0}| = 1. Если |A| = |B|, то множества A и B называются *равномощными.*

Если множества A и B содержат одни и те же элементы и мощности их равны, то эти множества считаются равными. Это обозначается так: A=B. Неравные множества состоят из различных элементов, обозначается так: A≠B.

Равенство множеств обладает следующими свойствами:

• A = B - рефлексивность;

• если A = B, то B = A - симметричность;

• если A = B и B = C, то A = C - транзитивность.

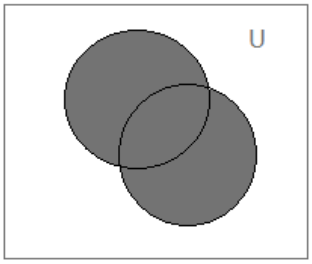
Отметим, что множества, которые содержат себя в качестве одного из своих элементов, называются экстраординарными. Остальные множества, не относящиеся к ним, называются ординарными.

2.2 Операции над множествами

Объединением, двух множеств X и Y называется множество, обозначаемое XUY и состоящее из элементов, принадлежащих хотя бы одному из множеств X или Y:

$XUY = \{x \mid x \in X \text{ или } x \in Y\}$.

Поясним определение объединения множеств с помощью диаграммы Эйлера-Венна:



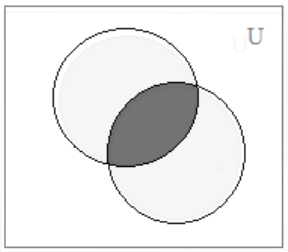
XUY

Пример. Рассмотрим два множества X = {1,3,5} и Y = {3,5,9}. Их объединением XUY будет множество {1,3,5,9}.

Пересечением, множеств X и Y называется множество, обозначаемое X∩Y состоящее из элементов, принадлежащих каждому из множеств X и Y:

$X \cap Y = \{x \mid x \in X \text{ и } x \in Y\}$.

Поясним определение пересечения множеств с помощью диаграммы Эйлера-Венна:



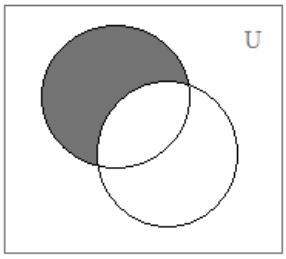
X∩Y

Пример. Рассмотрим два множества X = {1,3,5} и Y= {3,5,9}. Тогда пересечением этих множеств будет X ∩Y = {3,5}.

Разностью множеств X и Y называется множество, обозначаемое X\Y и состоящее из всех элементов X, не принадлежащих Y:

$X \backslash Y = \{x \mid x \in X \text{ и } x \notin Y\}$.

Поясним определение разности множеств с помощью диаграммы Эйлера-Венна:



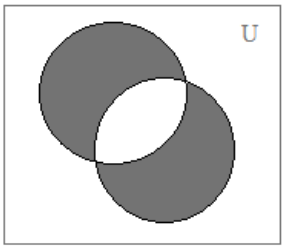
X\Y

Пример. Рассмотрим два множества X = {1,3,5}, Y= {3,8,9}. Разностью этих множеств будет множество X\Y= {1,5}.

Симметричной разностью множеств X и Y называется множество

$X \Delta Y = (X \backslash Y) U (Y \backslash X)$:

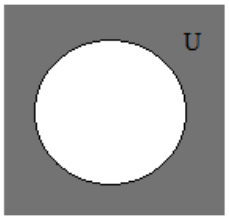
$X \Delta Y = \{x \mid (x \in X \text{ и } x \notin Y) \text{ и } (x \notin X \text{ и } x \in Y)\}$.



XΔY

Дополнением к множеству Хотносительно универсального множества U называется множество X’ = U\X :

$X'=\{x \mid x \notin X\}$



X’

Разбиением множества Y называется набор его *попарно непересекающихся* подмножеств $X_{ά}$, ά∈ A, где A – некоторое множество индексов, такой, что $Y = UX_{ά}$, ά∈ A.

Приоритет выполнения операций.

Сначала выполняются операции дополнения, затем пересечения, объединения и разности, которые имеют одинаковый приоритет. Последовательность выполнения операций может быть изменена скобками. Если в выражении есть знаки пересечения и объединения и нет скобок, то сначала выполняется операция пересечения, а потом – операция объединения (аналог сложению и умножению в арифметике).

2.3 Свойства операций над множествами

Пусть задан универсум U. Тогда∀*A, B, C* ⊂U выполняются следующие свойства:

1. *идемпотентность:*

A∪A=A,

A∩A = A;

2. коммутативность:

A∪B = B∪A,

A∩ B = B∩ A;

3.*ассоциативность:*

A∪ (B ∪ C) = (A∪B) ∪C,

A ∩(B ∩C) = (A∩B) ∩ C;

4.*дистрибутивность:*

A∪ (B∩ C) = (A∪B) ∩ (A∪C),

A ∩ (B ∪ C) = (A ∩ B)∪(A∩C);

5.*поглощение:*

(A∩B)∪ A = A,

(A∪B) ∩A = A;

6.*свойства нуля:*

A∪Ø = A,

A∩Ø= Ø;

7.*свойства единицы:*

A∪U = U,

A∩U = A;

8.*законыдеМоргана:*

(A ∩ B)’ = A’∪B’,

(A∪ B)’= A’ ∩ B’;

9. *свойства дополнения:*

A ∪A’ = U,

A ∩ A’ = Ø;

10. *выражение для разности:*

A\B =A∩ B’.

2.4 Примеры доказательств тождеств с множествами

Пример 1. Доказать или опровергнуть справедливость тождества$(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$.

Доказательство. Докажем, используя метод взаимного включения. Пусть$(A \cup B) \cap C = E$, а$(A \cap C) \cup (B \cap C) = F$.Тогда необходимо доказать или опровергнуть следующее:

$E \subseteq F \ \& \ F \subseteq E$.

1. Докажем необходимость: $E \subseteq F$.

$\forall a \in E \Rightarrow a \in (A \cup B) \cap C \Rightarrow a \in (A \cup B) \& \ a \in C \Rightarrow (a \in A \vee a \in B) \& \ a \in C \Rightarrow a \in (A \cap C) \vee a \in (B \cap C) \Rightarrow a \in (A \cap C) \cup (B \cap C) \Rightarrow a \in F$.

2. Докажем достаточность:$F \subseteq E$

$\forall a \in F \Rightarrow a \in (A \cap C) \cup (B \cap C) \Rightarrow a \in (A \cap C) \vee a \in (B \cap C) \Rightarrow (a \in A \ \& \ a \in C) \vee (a \in B \& \ a \in C) \Rightarrow a \in (A \cup B) \& \ a \in C \Rightarrow a \in (A \cup B) \cap C \Rightarrow a \in E$.

3. Следовательно, $E = F$, т.е. исходное тождество справедливо.

Пример 2. Доказать или опровергнуть справедливость тождества $A \setminus ((A \cap B) \cup (A \setminus B)) = \varnothing$.

Доказательство. Докажем методом от противного: предположим, что это выражение не равно пустому множеству.

$a \in A \setminus ((A \cap B) \cup (A \setminus B)) \Rightarrow a \in A \ \& \ a \notin ((A \cap B) \cup (A \setminus B)) \Rightarrow a \in A \ \& \ (a \notin (A \cap B) \& \ a \notin (A \setminus B)) \Rightarrow a \in A \ \& \ (a \notin A \ \& \ a \notin B) \ \& (a \notin A \vee a \in B)$

Получаем противоречие: элементa одновременно принадлежит и не принадлежит множеству A. Значит, первоначальное предположение неверно и исходное тождество справедливо, т.е. равно $\varnothing$.

Пример 3. Доказать, что$A \subseteq B \Rightarrow B' \subseteq A'$.

Доказательство. Пусть A и B – подмножества некоторого универсума U, $A \subseteq B$

$\forall x \in U, \quad x \in A \Rightarrow x \in B$

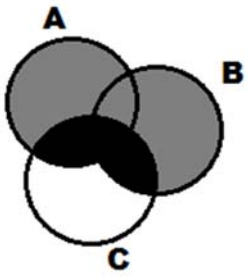
$\forall x \in U, \quad x \notin A \Rightarrow x \notin B$

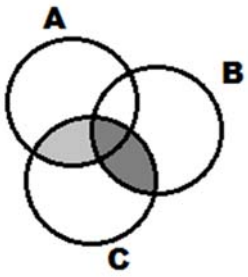
$\forall x \in U, \quad x \in B' \Rightarrow x \in A'$

Значит $B' \subseteq A'$.

Пример 4.Доказать$(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$.

Доказательство. Докажем, используя геометрический метод. Построим диаграммы Эйлера-Венна для множеств$(A \cup B) \cap C$ и $(A \cap C) \cup (B \cap C)$:





На первой диаграмме множество $(A \cup B) \cap C$выделено черной штриховкой, на второй множество $(A \cap C)$ – светлой, множество $(B \cap C)$ – серой, а множество$(A \cap C) \cup (B \cap C)$является их объединением. Сравнивая эти два рисунка, можно сделать вывод, что эти множества равны, следовательно, тождество доказано.

2.5 Булеан

Множество всех подмножеств множества M называется *булеаном* и обозначается 2^M:

$2^M = \{A | A \subset M\}$

Теорема.Для конечного множества M $|2^M| = 2^{|M|}$

Доказательство:

Индукция по |M|. База: если |M| = 0, то M= Ø и $2^{Ø} = \{Ø\}$. Следовательно,

$|2^{Ø}| = |\{Ø\}| = 1 = 2^0 = 2^{|Ø|}$.

Индукционный переход: пусть $\forall M\ |M| < k \rightarrow |2^M| = 2^{|M|}$. Рассмотрим $M = \{a_1, \ldots, a_k\}$,

|M| = k. Положим

$M_1 = \{X \subset 2^M \mid a_k \in X\}$ и $M_2 = \{X \subset 2^M \mid a_k \notin X\}$.

Имеем $2^M = M_1 \cup M_2$ и $Mi \cap M_2 = Ø$. По индукционному предположению $|M_1| = 2^{k-1}$,

$|M_2| = 2^{k-1}$. Следовательно, $|2^M| = |M_1| + |M_2| = 2^{k-1} + 2^{k-1} = 2 \times 2^{k-1} = 2^k = 2^M$.

Пример. Пусть X = {1,2,3}. Тогда множество всех подмножеств X будет

$2^X = \{0, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, X\}$.

Упорядоченные множества

3.1 Кортеж

Пусть *A* и *B* — произвольные множества. *Упорядоченная пара* на множествах *A* и *B*, обозначаемая записью <*a,b*>, определяется не только самими элементами *a*∈*A* и *b*∈*B*, но и порядком, в котором они записаны. И в этом состоит ее существенное отличие от неупорядоченной пары. Если *A* = *B*, то говорят об упорядоченной паре на множестве *A*.

Две упорядоченные пары <*a,b*>и <*c, d*>на множествах *A* и *B* называют *равными*, если *a* = *c* и *b* = *d*.

Упорядоченную пару <*a,b*>не следует связывать с множеством *{a, b},* так как упорядоченная пара характеризуется не только составом, но и порядком элементов в ней. Более того, определение этого объекта вообще не позволяет рассматривать его как множество. Но упорядоченную пару можно определить и как множество, полагая, что упорядоченная пара <*a,b*> есть неупорядоченная пара *{{a}, {a, b}}*, включающая в себя одноэлементное множество *{a}* и неупорядоченную пару *{a, b}*. При *a* = *b* получаем <*a,a*>= *{{a}}*. Такое определение не изменит сути понятия, но тогда следует не определять явно равенство упорядоченных пар, а доказывать теорему о равенстве упорядоченных пар как определенного вида множеств.

Обобщением понятия упорядоченной пары является упорядоченный n-набор, или *кортеж*. В отличие от конечного множества $\{a_1,...,a_n\}$ кортеж $<a_1, ..., a_n>$ на множествах $A_1, ..., A_n$ характеризуется не только входящими в него элементами $a_1 \in A_1, ..., a_n \in A_n$, но и порядком, в котором они перечисляются.

Два кортежа $\alpha=<a_1, ..., a_n>$ и $\beta=<b_1, ..., b_n>$ на множествах $A_1, ..., A_n$ равны, если $a_i=b_i$, $i=\overline{1,n}$.

Число n называется *длиной* кортежа (или размерностью кортежа), а элемент a_i — i-й *проекцией* (компонентой) кортежа. Для двух кортежей одинаковой размерности их компоненты с одинаковыми номерами называют *одноименными компонентами*. Определение равенства кортежей можно переформулировать так: два кортежа одинаковой размерности равны тогда и только тогда, когда их одноименные компоненты совпадают. В отличие от множества, кортеж может иметь повторяющиеся элементы, но все эти элементы различны. Компоненты кортежа могут обозначать любые понятия, объекты, в том числе элементы множества или кортежа.

Простейшим примером кортежа является арифметический вектор.

Кортеж, который не содержит компонентов в своем составе, называется *пустым* кортежем и обозначается $\alpha=<>$. Длина этого кортежа равна нулю.

Для любых кортежей α, β, γ справедливы утверждения:

- Если α=β, то β=α
- Если α=β и β = γ, то α= γ

3.2 Операция проекции

Операция проектирования унарна. Она применима не к двум множествам, а к одному. Кроме этого, операция проектирования применима только к множеству кортежей одинаковой длины. Проекция множеств определяется через проекцию кортежей.

Определим понятие *проекции кортежей*.

Пусть задан кортеж $\alpha = <a_1, a_2, \ldots, a_s>$длины s.

1) Пусть $1 \le i \le s$. Тогда проекцией кортежа αна i-тую ось называется i-тая компонента кортежа α.

2) Пусть задано произвольное число q, такое, что $2 \le q \le$s. И пусть задано число осей $1 \le i_1 \le i_2 \le \ldots \le i_q \le$s. Тогда проекцией кортежа αна оси с номерами $i_1, i_2, \ldots, i_q$ называется кортеж $<a_{i1}, a_{i2}, \ldots, a_{iq}>$, который обозначается следующим образом: $np_{i1, i2, \ldots, iq}\alpha = < a_{i1}, a_{i2}, \ldots, a_{iq}>$.

3) Проекцией кортежа a на пустое множество осей называется пустой кортеж. Аналогично проекцией пустого кортежа на пустое множество осей называется пустой кортеж.

Пример. Пусть задан кортеж $\alpha = < 12, 15, 6, 8 >$, $np_{i1}\ \alpha = <12>$, $np_{i2}\ \alpha = <15>$, $np_{i3}\ \alpha = <6>$, $np_{i4}\ \alpha = <8>$, $np_{i1,i2}\ \alpha = <12,15>$, $np_{i1,i4}\ \alpha = <12,8>$, $np_{i5,i8}\ \alpha = <>$.

Определим понятие *проекции множества*. Как отмечено это понятие будет определено только для случая, когда проектируемое множество состоит из кортежей, причем все кортежи имеют одинаковую длину.

Проекция множества M — это множество проекций кортежей из M.

Пусть задано множество кортежей M длины s, s> 0.

1) Пусть $1 \le$i$\le$s, тогда проекцией множества M на i-тую ось называется множество проекций кортежей из M на i-тую ось и обозначается: $пр_i$M.

2) Пусть задано произвольное число q, такое, что $2 \le q \le$s, и задано число осей $1 \le i_1 \le i_2 \le \ldots \le i_q \le$s. Тогда проекцией множества M на оси с номерами $i_1, i_2, \ldots, i_q$ называется множество проекций кортежей из M на оси с номерами $i_1, i_2, \ldots, i_q$.

3) Проекцией множества M на пустое множество осей называется множество проекций кортежей из M на пустое множество: $пр_{\varnothing}$M.

Пример. Пусть *M = { < 1, 2, 3, 4, 5 >, < 2, 1, 3, 5, 5 >, < 3, 3, 3, 3, 3><3, 2, 3, 4, 3><a, b, a, 1,a>}.*Тогда *np_2M = { 2, 1,3, 2, b }, $np_{2,4}M$ = { <2,4>, <1, 5>, <3, 3>, <2, 4>, <b, 1> }, $np_{67}M = Ø$.*

Пусть M — произвольное множество, длина которого s, s≥2. Тогда множество M^s состоит из кортежей длины s и значит, его можно проектировать. Операция проектирования множества основана на описанных правилах построения проекций кортежей и множеств. Для любого натурального числа 1 ≤*i*≤sпроекция $np_iM^s = M$.

Согласно определению операции проектирования, можно сказать, что для произвольного кортежа <x, y> истинны следующие высказывания:

$<x, y> \in A \Rightarrow x \in np_1A \ \& \ y \in np_2A,$

$x \notin np_1A \vee y \notin np_2A \Rightarrow <x,y> \notin A.$

Приведем основные свойства операции проектирования:

Пусть A⊆X×Y, B⊆X×Y, тогда для любых x∈X и y∈Y (∀x∈X &y∈Y) *истинно*:

- $\begin{cases} np_1(X \times Y) = X \\ np_2(X \times Y) = Y \end{cases}$
- $np_1X^2 = np_2X^2 = X$
- $\begin{cases} np_1(A \cup B) = np_1A \cup np_1B \\ np_2(A \cup B) = np_2A \cup np_2B \end{cases}$
- $\begin{cases} np_1(A \cap B) \subseteq np_1A \cap np_1B \\ np_2(A \cap B) \subseteq np_2A \cap np_2B \end{cases}$
- $\begin{cases} np_1A = np_2A^{-1} \\ np_2A = np_1A^{-1} \end{cases}$
- $\begin{cases} x \in np_1A \Rightarrow (\exists y \in Y)(< x, y > \in A) \\ y \in np_2A \Rightarrow (\exists x \in X)(< x, y > \in A) \end{cases}$
- $x \in np_1A \& \ y \in np_2A \Rightarrow (\exists w \in Y)(\exists z \in X)(< x, w > \in A \& < z, y > \in A)$

В то же время в общем случае ложными являются следующие высказывания:

- $\begin{cases} x \in np_1 A \Rightarrow < x, y > \in A \\ y \in np_2 A \Rightarrow < x, y > \in A \end{cases}$
- $x \in np_1 A \& \, y \in np_2 A \rightarrow < x, y > \in A$
- $\begin{cases} np_1 A = np_1 B \Rightarrow A = B \\ np_2 A = np_2 B \Rightarrow A = B \end{cases}$
- $np_1 A = np_1 B \& np_2 A = np_2 B \Rightarrow A = B$

Некоторые из перечисленных высказываний следуют из определения прямого произведения. Для доказательства других свойств необходимо использовать методы доказательств тождеств с множествами.

Рассмотрим операции над кортежами: инверсия и композиция.

1) *Инверсия*.

Инверсия кортежа определяется следующим образом. Пара *<c, d>* называется *инверсией* пары *<a,b>*,если *c=b&d=a*. Инверсия пары α обозначается α^{-1}

Пример.*α = <a, b>,*тогда $\alpha^{-1} = <b, a>$. $(\alpha^{-1})^{-1} = \alpha$, $((\alpha^{-1})^{-1})^{-1} = \alpha^{-1}$. Тогда $\alpha^{-n} = \alpha$ и $\alpha^{-(n-1)} = \alpha^{-1}$, при четном *n*.

2) *Композиция*.

Кортеж *α= <x, y>*называется композицией двух кортежей *β = <x, z>*и *γ = <z, y>*и записывается *α = β·γ*. Операция композиции справедлива, когда вторая компонента кортежа *β*совпадает с первой компонентой кортежа *γ*. Здесь как бы происходит "склеивание" двух кортежей по компоненте *z*.

3.3 Декартово произведение множеств

Пусть $X_1, X_2, ..., X_n$ — множества.

Прямым (декартовым) произведением множеств X_i, i = 1,2, ..., n:

$X_1 \times X_2 \ldots \times X_n$

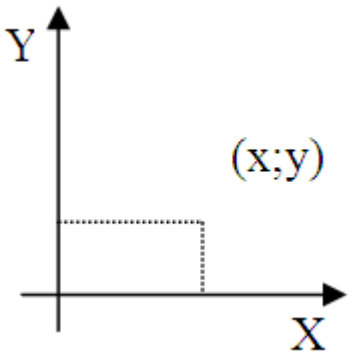
называется множество всех упорядоченных наборов (x_1,x_2, ..., x_n), где $x_i \in X_i$, i = 1,2, ..., n.

Из определения декартова произведения следует, что $A \times B = \varnothing$, если $A = \varnothing$ или $B = \varnothing$:

$A \times B = \varnothing \Rightarrow A = \varnothing \vee B = \varnothing$.

По аналогии можно утверждать, что прямое произведение нескольких множеств равно пустому множеству тогда и только тогда, когда хотя бы одно из этих множеств пусто.

Пример 1. Пусть X = R, Y = R — множества точек двух числовых осей. Тогда декартовым произведением $X \times Y = R^2$ является множество точек плоскости (см.рис.). Каждой точке плоскости соответствует пара точек (проекций) на числовых осях.



Пример 2. Пусть заданы множества $A=\{1,2\}, B=\{3, 4, 5\}$, тогда $A \times B=\{<1, 3>,<1, 4>,<1, 5>,<2, 3>,<2, 4>,<2, 5>\}$

Декартово произведение двух множеств обладает следующими свойствами:

- $A \times B \neq B \times A$–некоммутативность
- $A \times (B \times C) = (A \times B) \times C = A \times B \times C$ – ассоциативность
- $A \times (B \cup C) = (A \times B) \cup (A \times C)$ – дистрибутивность по объединению
- $A \times (B \cap C) = (A \times B) \cap (A \times C)$ – дистрибутивность по пересечению
- $A \times (B \setminus C) = (A \times B) \setminus (B \times C)$– дистрибутивность по разности

- $(A \times B) \cap (C \times D) = (A \cap C) \times (B \cap D)$

3.4 Графики

График — это множество пар, т.е. множество, каждый элемент которого является парой или кортежем длины 2. Множество P называется графиком, если каждый элемент его пара.

Пример. Множество $P = \{<a, b>, <a, 1>, <c, d>\}$является графиком.

Если M — произвольное множество, то M^2, а также любое множество $C \subseteq M^2$ является графиком. В частности графиком является множество D^2 действительных чисел. Пусть заданы множества A и B, тогда $A \times B$, $C \subseteq A \times B$ являются графиками.

Понятие графика является обобщенным. В принципе оно происходит от понятия графика функции.

Областью определения графика P называется множество np_1P(проекция на первую ось (ось абсцисс) данного графика).

Областью значения графика называется множество проекций на вторую ось (ось ординат) (np_2P).

Легко видеть, что если P — график, тогда если $P = \emptyset$, то $np_1P = \emptyset$ & $np_2P = \emptyset$.

Рассмотрим основные операции над графиками:

1. *Инверсия* (определяется через инверсию кортежа)

Инверсией графика P называют множество инверсий пар из P.

Пример. $P = \{<c, d>, <a, b>\}$, $P^{-1} = \{<d, c>, <b, a>\}$.

График Q называется инверсией графика P, если $\exists \alpha \in Q$ тогда и только тогда, когда $\alpha^{-1} \in P$,где α - произвольный кортеж.

В теоретико-множественном виде запишем:

$\alpha^{-1} \in P \rightarrow \alpha \in P^{-1}$

$\alpha \in P \rightarrow \alpha^{-1} \in P^{-1}$

График P называется *симметричным*, если он наряду с любой своей парой содержит ее инверсию. Например, $P = \{<a, b>, <b, a>\}$

Пусть M — произвольное множество. Тогда считают ΔM — множество всех пар вида $<x, x>$, где x присутствует во всем множестве M. Таким образом, если $M = \{a, b\}$,то $\Delta M = \{<a, a>, <b, b>\}$— является симметричным графиком и называется *диагональю*.

2. Композиция

График R называется композицией двух графиков P и Q, а также $<x, y> \in R$, тогда и только тогда, когда $\exists z$такое, что $<x, z> \in P \ \& <z, y> \in Q$.

Переход от графиков P и Q к их композиции $(P \cdot Q)$ называется *компонированием* графиков P и Q.

Пример. Пусть $P = \{<a, a>, <a, c>\}$, а $Q = \{<a, b>, <b, c>\}$, тогда $P \cdot Q = \{<a,b>\}$.

Композиция графика P и Ø равна Ø, то есть $P \cdot Ø = Ø \cdot P = Ø$.

Если M — произвольное множество и $P \subseteq M^2$, тогда$P \cdot \Delta M = \Delta M \cdot P = P$.

Если операцию композиции графиков сопоставить с умножением чисел, то роль нуля будет играть пустое множество, а роль единицы диагональ (Δ).

Пусть <x,z>- произвольная пара из A·B. Тогда для нее справедливо высказывание:

$<x, z> \in A \cdot B \Rightarrow (\exists y \in (Y \cap W))(<x, y> \in A \& <y, z> \in B)$.

Если некоторая пара <x, z>не принадлежит A· B, то истинно высказывание:

$<x, z> \notin A \cdot B \Rightarrow (\exists y \in (Y \cap W))(<x, y> \notin A \& <y, z> \notin B)$.

В операции композиции элемент y называется *компонирующим* элементом для пар <x, y>∈ A и <y, z> ∈ B. Если множество компонирующих элементов пусто, то и результат композиции является пустым множеством:

$A \cdot B = \emptyset \Rightarrow np_2A \cap np_1B = \emptyset \Rightarrow A \cdot \emptyset = \emptyset \cdot A = \emptyset.$

Свойства операции композиции:

- $A \cdot B \neq B \cdot A$ – некоммутативность
- $A \cdot (B \cdot C) = (A \cdot B) \cdot C$ – ассоциативность
- $A \cdot (B \cup C) = (A \cdot B) \cup (A \cdot C)$ – дистрибутивность по объединению
- $A \cdot (B \cap C) = (A \cdot B) \cap (A \cdot C)$ – дистрибутивность по пересечению
- $(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$

Некоторые тождества следуют из определения композиции, остальные тождества доказываются уже известными методами.

Пример. Пусть P, Q, R – графики. Необходимо доказать следующее тождество: $(P \cdot Q) \cdot R = P \cdot (Q \cdot R)$

Доказательство.

1. Необходимость: $<a, b> \in (P \cdot Q) \cdot R \Rightarrow <a, z> \in (P \cdot Q)$ & $<z, b> \in R \Rightarrow <a, x> \in P$ & $<x, z> \in Q$ & $<z, b> \in R \Rightarrow <a, x> \in P$ & $<x, b> \in (Q \cdot R) \Rightarrow <a, b> \in (P \cdot (Q \cdot R)) \Rightarrow$ первая часть доказана
2. Достаточность: $<a, b> \in (P \cdot (Q \cdot R)) \Rightarrow <a, x> \in P$ & $<x, b> \in (Q \cdot R) \Rightarrow <a, x> \in P$ & ($<x, d> \in Q$ & $<d, b> \in R$) $\Rightarrow <a, x> \in P$ & $<x, d> \in Q$ & $<d, b> \in R \Rightarrow <a, d> \in (P \cdot Q)$ & $<d, b> \in R \Rightarrow <a, b> \in ((P \cdot Q) \cdot R) \Rightarrow$ вторая часть доказана.
3. Значит, исходное тождество справедливо.

Основные свойства графиков:

- График P называется *функциональным*, если в нем нет пар с одинаковыми первыми и разными вторыми компонентами. Например, P ={<b, a>, <c, a>, <d, a>}.

- График P называется *инъективным*, если в нем нет пар с различными первыми и одинаковыми вторыми компонентами. Например, P ={<a,b>, < a, c>, <a, d>}.

Композиция функциональных графиков есть функциональный график, т.е. композиция сохраняет функциональность. Композиция инъективных графиков инъективна.

Итак, говорят, что график P функционален тогда и только тогда, когда P-1 инъективен. График P инъективен тогда и только тогда, когда P-1 функционален.

Отношения на множествах

4.1 Понятие отношения

Отношение – это связь между любыми объектами в природе. На формальном языке *отношение* – это пара множеств, причем упорядоченное, первая компонента которой является подмножеством квадрата второй компоненты.

Бинарным отношением на множестве A называется пара Ф = (A,G), где A —область задания отношения, G —график отношения, причём $G \in A^2$.

Если $(x,y) \in G$, то будем писать $x \varphi y$ и говорить, что x и y *вступают в отношение* φ. Если x и y не вступают в отношение φ, будем писать $(x \varphi y)'$.

Диагональю множества A^2 называется график $\Delta_A = \{(x,x) | x \in A\}$.

Множество $D_R = \{x : (\exists y) xRy\}$ называется *областью определения* бинарного отношения R. *Областью значений* бинарного отношения R называется множество $I_R = \{y : (\exists x) xRy\}$.

Каждое бинарное отношение R есть подмножество прямого (декартова) произведения некоторых множеств X и У, таких, что $D_R \subseteq X$ и $I_R \subseteq Y$.

Пример. Рассмотрим множество {(1,2); (2,4); (3,3); (2,1)}. Это бинарное отношение R для X = {1,2,3}; Y = {1,2,3,4}. Область определения такого отношения D_R есть {1,2,3}X, а область значений I_R — множество {2,4,3,1} Y.

Обратным отношением для отношения R называется отношение R^{-1}, такое, что $R^{-1} = \{(x,y) : (y,x) \in R\}$

Множество упорядоченных *n*-к, т. е. $R \subseteq X_1 \times X_2 \times \ldots \times X_n$, называется n-*местным отношением* φ для $X_1, X_2, \ldots, X_n$.

Реляционная таблица

X_1	X_2	...	X_n
x_1	x_2	...	x_n
x'_1	x'_2	...	x'_n
...	...	...	...

Многоместные отношения удобно задавать с помощью реляционных таблиц. Такое задание соответствует

перечислению множества *n*-к отношения φ. Реляционные таблицы широко используют в компьютерной практике в реляционных базах данных. При этом имена множеств X_i называют *атрибутами* (свойствами), а элементы $x_i \in X_i$ называют *доменами* (значениями) атрибутов. Заметим, что реляционные таблицы широко используются в повседневной практике. Всевозможные производственные, финансовые, научные и другие отчеты часто имеют форму реляционных таблиц.

4.2 Свойства отношений

Свойства отношений:

1. Рефлексивность: если $\forall a \in A a \varphi a$;

2. Антирефлексивность: если $\forall a \in A a' \varphi a$;

3. Симметричность: если $\forall a,b \in A a \varphi b \rightarrow b \varphi a$;

4. Антисимметричность: если $\forall a,b \in A a \varphi b \& b v a \rightarrow a=b$;

5. Транзитивность: если $\forall a,b,c \in A a \varphi b \& b \varphi c \rightarrow a \varphi c$;

6. Полнота, или линейность: если $\forall a,b \in A a \neq b a \varphi b$ или $b \varphi a$.

Отношение φ=<Ф,М> называется *пустым*, если график Ф является пустым множеством. Т.е. φ=<Ø,М>. Другими словами имеется область задания отношения, на котором не задан график отношения.

Отношение φ=<Ф,М> называется *отношением равенства*, если Ф=ΔМ. В теоретико-множественном плане можно записать, (∀a,b∈M)(aφb→a=b). Например задано φ=<Ф,М>, М={1,2,3}, Ф={<1,1>,<2,2>,<3,3>}. Данное отношение является отношением *равенства*.

Отношение φ=<Ф,М> называется *отношением неравенства*, если $Ф=M^2 \backslash \Delta M$, т.е. (∀a,b∈M) (aφb→a≠b). Например задано φ=<Ф,М>, М={1,2,3}, Ф={<1,2>,<2,3>,<3,1>}. Данное отношение является отношением *неравенства*. Отношения «5 > 3» и «3 < 10»также являются примерами отношения неравенства.

Отношение называется отношением *частичного порядка*, если оно рефлексивно, антисимметрично и транзитивно.

Отношение называется отношением *линейного порядка*, если оно является отношением частичного порядка и линейно.

Отношение называется отношением *строгого порядка*, если оно антирефлексивно, антисимметрично и транзитивно.

Отношение называется отношением *строгого линейного порядка*, если оно — линейное отношение строгого порядка.

Отношение называется отношением *эквивалентности*, если оно рефлексивно, симметрично и транзитивно.

Классом эквивалентности, порождённым элементом x, называется множество всех элементов из A, вступающих с x в отношение эквивалентности.

Фактор-множеством множества A по отношению эквивалентности φ называется множество всех различных классов эквивалентности, которое обозначается A/φ.

Мощность фактор-множества A/φ называется индексом разбиения, порождённого отношением φ.

4.3 Операции над отношениями

На отношения переносятся основные операции над множествами, но они могут выполняться только на одной и той же области задания.

Объединением отношений φ_1 и φ_2 на множестве M называется отношение φ_3:

$\varphi_1 \cup \varphi_2 = \varphi_3$, $\varphi_1 = <\Phi_1, M>$, $\varphi_2 = <\Phi_2, M>$.

$\varphi_3 = <\Phi_1 \cup \Phi_2, M>$,

$<a, b> \in \Phi_1 \cup \Phi_2 \rightarrow <a, b> \in \Phi_1 \vee <a, b> \in \Phi_2 \,\&\, a, b \in M.$

Например, пусть имеем два отношения: $\varphi_1 = <\Phi_1, M>$, $\varphi_2 = <\Phi_2, M>$, $M = \{2, 3, 4\}$, $\Phi_1 = \{<2,1>, <2,2>, <2,4>\}$, $\Phi_2 = \{<2,1>, <2,3>, <4,4>\}$

Тогда объединение этих отношений $\varphi_3 = <\Phi_3, M>$, $\Phi_3 = \Phi_1 \cup \Phi_2 = \{<2,1>, <2,2>, <2,4>, <2,3>, <4,4>\}$.

Отметим, что для операции объединения над отношениями справедлива следующая запись:

$x(\varphi_1 \cup \varphi_2)y \rightarrow x\ \varphi_1 y \vee x\ \varphi_2 y$

Пересечением отношений φ_1 и φ_2 на множестве M называется отношение φ_3:

$\varphi_1 \cap \varphi_2 = \varphi_3$, $\varphi_1 = <\Phi_1, M>$, $\varphi_2 = <\Phi_2, M>$.

$\varphi_3 = <\Phi_1 \cap \Phi_2, M>$,

$<a, b> \in \Phi_1 \cap \Phi_2 \rightarrow <a, b> \in \Phi_1 \& <a, b> \in \Phi_2 \& a, b \in M$.

Например, пусть имеем два отношения: $\varphi_1 = <\Phi_1, M>$, $\varphi_2 = <\Phi_2, M>$, $M = \{1, 2\}$, $\varphi_1 = <\{<1,1>, <1,2>\}, \{1, 2\}>$, $\varphi_2 = <\{<1,2>, <2,2>\}, \{1, 2\}>$

Тогда пересечение этих отношений $\varphi_3 = <\Phi_3, M> = <\{<1,2>\}, \{1, 2\}>$.

Отметим, что для операции пересечения над отношениями справедлива следующая запись:

$x(\varphi_1 \cap \varphi_2)y \rightarrow x\ \varphi_1 y \& x\ \varphi_2 y$

Операции объединения и пересечения также, как и для множеств применимы для любого числа отношений.

Отношение φ_3 называется *разностью* отношений φ_1 и φ_2, если

$\varphi_1 = <\Phi_1, M>$, $\varphi_2 = <\Phi_2, M>$.

$\varphi_3 = \varphi_1 \setminus \varphi_2 = <\Phi 1 \setminus \Phi 2, M>$,

$<a, b> \in \Phi_1 \setminus \Phi_2 \rightarrow <a, b> \in \Phi_1 \& <a, b> \notin \Phi_2 \& a, b \in M$.

Например, пусть имеем два отношения: $\varphi_1 = <\Phi_1, M>$, $\varphi_2 = <\Phi_2, M>$, $M = \{1, 2, 3\}$, $\Phi_1 = \{<2,2>, <1,2>, <3,3>\}$, $\Phi_2 = \{<1,1>, <2,2>, <3,3>\}$

Тогда разность этих отношений $\varphi_3 = <\Phi_3, M> = <\{<1,2>\},\{1, 2, 3\}>$.

Отметим, что для операции разности над отношениями справедлива следующая запись:

$x(\varphi_1 \backslash \varphi_2)y \rightarrow x\ \varphi_1 y \& x\ \varphi_2{}' y$

Над отношениями выполняются также операции *инверсии* и *композиции*.

Если $\varphi = <\Phi, M>$, то *инверсия* $\varphi^{-1} = < \Phi^{-1}, M >$.

Для того, чтобы найти *инверсию* отношения, необходимо проинвертировать элементы его графика на множестве М. Отметим, что для операции инверсии над отношениями справедлива следующая запись

$x\ \varphi^{-1} y \rightarrow y \varphi\ x.$

Например, для отношения $\varphi=<\Phi, M>$, $M=\{1,2,3\}$, $\Phi=\{<1, 1>, <1, 2>,<1,3>\}$, инверсия $\varphi^{-1}= < \Phi^{-1},M>$ и $\Phi^{-1}= \{<1, 1>,<2, 1>, <3, 1>)$.

Композицией двух отношении является новое отношение, у которого компонируют графики отношений.

$\varphi_1 = <\Phi_1, M>, \varphi_2 = <\Phi_2, M>.$

$\varphi_1 \cdot \varphi_2 = <\Phi_1 \cdot \Phi_2, M>$

Например, пусть имеем два отношения: $\varphi_1 = <\Phi_1, M>$, $\varphi_2 = <\Phi_2, M>$, $M = \{1, 2, 3\}$, $\Phi_1= \{<1, 1>, <1,2>, <1, 3>,<3,3>\}$, $\Phi_2 = \{<1,1>,<2,3>, <3,1>,<3,2>\}$

Тогда композиция графиков этих отношений равна $\Phi_3 = \Phi_1 \cdot \Phi_2 = \{<1,1>, <1, 3>,<1,2>, <3,2>, <3,1>\}$.

Отметим, что все операции над отношениями могут выполняться только на одной и той же области задания, и в результате выполнения операций снова получается отношение с той же самой областью задания

Введем операцию, меняющую область задания отношений.

Пусть φ =<Ф, М>, и∃А⊆М, тогда *сужением отношения* φ на множестве А называется новое отношение

$\varphi_1 = <\Phi \cap A^2, A>$

Например, φ =<Ф, М>, М={ 1,2,3}, Ф = {<1, 1>,<1, 2>, <1, 3>}, А = {1,2}. Тогда φ_1 = <{<1,1>,<1,2>},{1,2}>.

4.4 Отношение эквивалентности

Отношение эквивалентности является формализацией такой ситуации, когда говорят о сходстве двух элементов множества.

Бинарное отношение R называется отношением *эквивалентности*, если оно рефлексивно, симметрично и транзитивно. Отношение эквивалентности xRy часто обозначается: x ~ y.

Пример 1. Отношение «одного роста» есть отношение эквивалентности на множестве X людей. Рефлексивность. Каждый человек такого же роста, как он сам. Симметричность. Сидоров одного роста с Петровым тогда и только тогда, когда Петров одного роста с Сидоровым. Транзитивность. Если Сидоров одного роста с Петровым, а Петров одного роста с Ивановым, то Сидоров одного роста с Ивановым.

Пример 2. Отношение обычного равенства на множестве целых чисел есть отношение эквивалентности.

Пример 3. Отношение x < y на множестве действительных чисел не есть отношение эквивалентности, так как оно не рефлексивно, не симметрично, а лишь транзитивно.

Если для бинарного отношения потребовать только выполнения свойств рефлексивности и симметричности, а транзитивности не требовать, то получим другой тип отношения. Оно называется отношением *толерантности* и является формализацией случая, когда два элемента множества не сходны, а только почти сходны (похожи).

Подмножество $[x] = \{y \in X: y \sim x\}$ называется *классом эквивалентности*, содержащим x (это множество всех элементов X, эквивалентных данному элементу x). Любой элемент $y \in [x]$ называется *представителем* этого класса.

Пример. Рассмотрим отношение принадлежности к одной студенческой группе. Классом эквивалентности является все множество студентов одной группы.

Теорема 1. Пусть R — отношение эквивалентности на множестве X. Тогда:

1) для $\forall\, x \in X$ имеем $x \in [x]$;

2) если $x, y \in X$ и xRy, то $[x] = [y]$.

Другими словами, класс эквивалентности порождается любым своим элементом.

Доказательство. Воспользуемся рефлексивностью отношения эквивалентности R, т. е. xRx. Следовательно, по определению, $x \in [x]$.

Пусть $z \in [y]$. Тогда yRz, и в силу транзитивности отношения эквивалентности имеем: из xRy и yRz справедливо xRz, т. е. $z \in [x]$. Отсюда $[y] \subseteq [x]$.

Аналогично в силу симметричности R получаем $[x] \subseteq [y]$. Следовательно, $[x] = [y]$.

Теорема 2. Всякое отношение эквивалентности R определяет разбиение множества X на классы эквивалентности относительно этого отношения R.

Теорема 3. Пусть задано разбиение множества X на попарно непересекающиеся подмножества. Тогда эти подмножества будут классами эквивалентности по некоторому отношению эквивалентности на X.

4.5 Отношение порядка

Антисимметричное транзитивное отношение называется отношением *порядка.* Отношение порядка может быть рефлексивным, и тогда оно называется отношением *нестрогого порядка.* Отношение порядка может быть антирефлексивным, и тогда оно называется отношением *строгого порядка.* Отношение порядка может быть полным (линейным), и тогда оно называется отношением *полного,* или *линейного порядка.* Отношение порядка может не обладать свойством полноты (линейности), и тогда оно называется отношением *частичного порядка.* Обычно отношение строгого порядка (полного или частичного) обозначается знаком <, а отношение нестрогого порядка — знаком ≤. Отношение порядка в общем случае обозначается знаком≺. Бинарное отношение, которое только рефлексивно и транзитивно, называется *отношением предпорядка*.

Пример 1.Отношение < на множестве чисел является отношением строгого полного порядка. Отношение ≤ на множестве чисел является отношением нестрогого полного порядка.

Пример 2.Рассмотрим множество {1,2,3} и отношение R = {(1,1), (1,2), (1,3), (2,2), (3,1), (3,2), (3,3)}. Это отношение рефлексивно, так как здесь присутствуют элементы (1,1), (2,2), (3,3). Это отношение транзитивно, так как из присутствия элементов (x, y),

(y, z) следует присутствие элемента (x, z). В самом деле, у нас есть (1,3), (3,2), и есть (1,2). Также имеются элементы (3,2), (2,2) и есть элемент (3,2). Аналогично и для элементов (3,1), (1,2) есть (3,2), для (1,2), (2,2) — элемент (1,2). Следовательно, R есть отношение предпорядка.

Пример 3.Теперь рассмотрим множество $X_1 \times X_2$ и попробуем задать на этом множестве отношение порядка, т. е. введем сравнение между парами элементов из X_1 и 1 X_2 . При этом пусть $<X_1,R_1>$ и $<X_2, R_2>$ — упорядоченные множества.

Это можно сделать, например, так. Определим отношение П условием: $(a_1,a_2)\Pi(b_1,b_2) \leftrightarrow a_1R_1b_1 , a_2R_2b_2$. Получится отношение порядка на $X_1 \times X_2$.

Такое отношение рефлексивно, так как $a_1R_1a_1$,$a_2R_2a_2$ и, следовательно, $(a_1,a_2)\Pi(a_1,a_2)$.

Далее П антисимметрично, так как $(a_1,a_2)\Pi(b_1,b_2)$, (a_1,a_2) П (a_1,a_2) следует $(a_1,a_2) = (b_1,b_2)$. В самом деле, из $a_1R_1b_1$, $b_1R_1a_1$ получаем $a_1= b_1$, а из $a_2R_2b_2$, $b_2R_2a_2$ следует $a_2 = b_2$.

Это отношение также и транзитивно. Пусть $(a_1,a_2)\Pi(b_1,b_2)$, $(b_1,b_2)\Pi(c_1,c_2)$.

Отсюда $(a_1,a_2)\Pi(c_1,c_2)$, так как $a_1R_1b_1$, $b_1R_1c_1$ влечет $a_1R_1c_1$, а $a_2R_2b_2$, $b_2R_2c_2 – a_2R_2c_2$.

Такое отношение порядка называется *отношением Парето*.

Множество, на котором определено отношение частичного порядка, называется *частично упорядоченным.* Множество, на котором определено отношение полного порядка, называется *вполне упорядоченным.*

Пример. Множество чисел упорядочено линейно, а булеан упорядочен частично.

Элемент x множества M с отношением порядка $\prec$ называется *минимальным,* если не существует меньших элементов: $\neg\exists y\in M\ y \prec x\ \&\ y\neq x$.

Элемент $a\in M$ называется *максимальным* в упорядоченном множестве M, если из $a\preccurlyeq x$ следует $x = a$. Всякий наибольший элемент является максимальным. Обратное неверно.

Пример. Пустое множество Ø является минимальным элементом булеана по включению.

Теорема. Во всяком конечном непустом частично упорядоченном множестве существует минимальный элемент.

Доказательство.

От противного. Пусть $\neg(\exists x \in M\ \neg\exists y \in M\ y \prec x)$.

Тогда$\forall x\in M\exists y\in M y\prec x \Longrightarrow \exists(u_i)_{i=1}^{\infty}\forall i u_{i+1}<u_i \& u_{i+1}\neq u_i$.

Поскольку $|M| < \infty$, имеем $\forall i,j i<j \& u_i = u_j$.

Но по транзитивности $u_i \succ u_{i+1} \succ \ldots \succ u_j \Longrightarrow u_{i+1} \succ u_j = u_i$.

Таким образом, $u_{i+1} \prec u_i \& u_{i+1} \succ u_i \Longrightarrow u_{i+1} = u_i$ – противоречие.

Вполне упорядоченное конечное множество содержит один минимальный элемент, а в произвольном конечном частично упорядоченном множестве их может быть несколько.

Теорема. Всякий частичный порядок на конечном множестве может быть дополнен до линейного.

Соответствия и функции

5.1 Основные понятия соответствия

Соответствием между множествами X и Y будем называть тройку объектов:

Г = (X, Y,G), где X — область отправления соответствия, Y — область прибытия соответствия, G — график соответствия, причём $G \subseteq X \times Y$.

Существует три способа задания соответствия:

- Теоретический
- Матричный
- Графический

Теоретический способ заключается в задании графика соответствия и множеств X и Y. Для графика соответствия справедливо: $G \subseteq X \times Y \Rightarrow G = X \times Y \vee G \subset X \times Y$.

При задании *матричным* способом соответствие представляется в виде матрицы $R_Г$, размером $n \times m$, где строки представляют элементы множества X, столбцы – элементы множества Y, а элемент матрицы r_{ij} принимает значения:

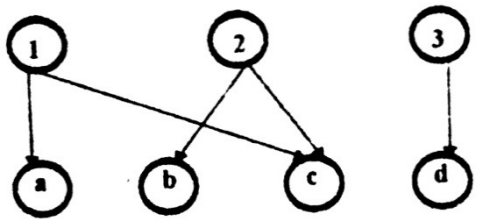
$r_{ij}=1$ – если существует кортеж $<x_i,y_j> \in G$;

$r_{ij}=0$, в противном случае.

Таким образом, соответствие можно представить в виде следующей матрицы:

$$R_{\Gamma}=\begin{vmatrix}1 & 0 & 1 & 0\\ 0 & 1 & 1 & 0\\ 0 & 0 & 0 & 1\end{vmatrix}$$

Соответствие, заданное в *графическом* виде, представляет собой граф, вершинами которого являются элементы, принадлежащие множествам X и Y соответствия Γ = (X, Y,G),а кортежи вида $<x_i,y_j>\in G$ представляются на графике соответствия в виде стрелок, направленных от x_i,к y_j:



Областью определения соответствия будем называть $пр_1$ G.

Областью значений соответствия будем называть $пр_2$ G.

Соответствие называется *всюду определённым*, если $пр_1$ G = X.

Соответствие называется *сюръективным*, если $пр_2$ G = Y.

Соответствие будем называть *функциональным*, или функцией, если его график не содержит пар с одинаковыми первыми и различными вторыми координатами.

Соответствие называется *инъективным*, если его график не содержит пар с одинаковыми вторыми и различными первыми координатами.

Соответствие называется *отображениемX в Y*, если оно всюду определено и функционально.

Соответствие называется *отображениемX на Y*, если оно всюду определено, функционально и сюръективно.

Соответствие называется *взаимно однозначным*, если оно функционально и инъективно.

Соответствие называется *биекцией*, если оно всюду определено, сюръективно, функционально и инъективно.

Образом множества A при данном соответствии называется множество $\Gamma(B) = \{y|(x,y)\in G \text{ и } x\in A\}$.

Прообразом множества B при данном соответствии называется множество $\Gamma^{-1}(B) = \{x|(x,y)\in G \text{ и } y\in B\}$.

Множества называются *равномощными*, если между ними можно установить биекцию.

Множество называется *счётным*, если оно равномощно множеству натуральных чисел.

Множество называется *континуальным*, если оно равномощно множеству действительных чисел отрезка [0,1].

5.2 Операции над соответствиями

Поскольку соответствия можно считать множествами, то все операции над множествами (пересечение, объединение, разность, дополнение и т.д.) можно применить и к соответствиям. Заметим, что, говоря о дополнении соответствия из A в B, мы имеем в виду дополнение до *универсального соответствия* из A в B, т.е. до *декартова произведения* A×B. Естественно, что и равенство соответствий можно трактовать как равенство множеств.

Объединением соответствий $\Gamma_1 = <X, Y, F>$ и $\Gamma_2 = <W, Z, P>$ называют соответствие $\Gamma_1 \cup \Gamma_2 = <X\cup W, Y\cup Z, F\cup P>$.

Пересечением соответствий $\Gamma_1 = <X, Y, F>$ и $\Gamma_2 = <W, Z, P>$ называют соответствие $\Gamma_1 \cap \Gamma_2 = <X\cap W, Y\cap Z, F\cap P>$.

Разностью соответствий $\Gamma_1 = <X, Y, F>$ и $\Gamma_2 = <W, Z, P>$ называют соответствие $\Gamma_1 \setminus \Gamma_2 = <X\setminus W, Y\setminus Z, F\setminus P>$

Инверсией соответствия $\Gamma = <X, Y, F>$ является соответствие Γ^{-1}, такое, что множество Y является областью отправления соответствия Γ^{-1};

множество X является областью прибытия соответствия Γ^{-1}, а график соответствия F^{-1} является инверсией графика F соответствия Г.

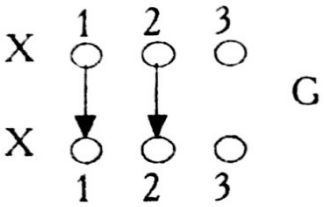
Композицией (произведением) соответствий $\Gamma_1 = <X, Y, F>$ и $\Gamma_2 = <W, Z, P>$ называют соответствие $\Gamma_1 \cdot \Gamma_2 = <X, Z, F \cdot P>$. Поясним построение композиции двух соответствий. Областью отправления является область отправления Γ_1, областью прибытия – область прибытияΓ_2, а графиком – композиция графиков F и P.

В случае, если $Y \cap W = \emptyset$, то результатом композиции соответствий будет соответствие с *пустым графиком.*

Соответствие Ω называется инверсией соответствия Г, если область отправления Г равна области прибытия Ω и график Г является инверсией графика Ω.

Четная инверсия оставляет соответствие самим собой, а нечетная – инвертирует. То есть $(\Gamma^{-1})^{-1} = \Gamma$, а $((\Gamma^{-1})^{-1})^{-1} = \Gamma^{-1}$. Соответствие $\Gamma^{-1} = \Gamma$ тогда и только тогда, когда график соответствия симметричен $G=G^{-1}$, а область отправления соответствия совпадает с областью прибытия.

Пример. $\Gamma = <G, X, X>$, $X = \{1,2, 3\}$, $G = \{< 1,1>< 2, 2 >\}$. Графическое представление этого соответствия:



Для соответствия так же, как для отношений и множеств справедлива операция композиции. *Композиция соответствий* определяется через композицию их графиков. Композиция соответствий не является пустой, если существует хотя бы один элемент $y \in Y \& y \in Z$. Пусть заданы соответствия $\Gamma_1 = <G, X, Y>$ и $\Gamma_2 = <H, Z, U>$. Тогда $\Gamma_1 \cdot \Gamma_2 = <G \cdot H. X, U>$- определяет композицию двух соответствий.

Например, пусть заданы множества X = {a, b}, Y = {c, d} Z = {d, e} U = <k, l>. Для получения непустого результата композиции соответствий множество Z должно частично или полностью совпадать с множеством Y

Для любых трех соответствий существует следующее правило композиции:

$(\Gamma_1 \cdot \Gamma_2) \cdot \Gamma_3 = \Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_3)$

Докажем это тождество.

1. Необходимость: $\langle a, b\rangle \in (\Gamma_1 \cdot \Gamma_2) \cdot \Gamma_3 \rightarrow \langle a, x\rangle \in \Gamma_1 \cdot \Gamma_2 \,\&\, \langle x, b\rangle \in \Gamma_3 \rightarrow \langle a, x_1\rangle \in \Gamma_1 \,\&\, \langle x_1, x\rangle \in \Gamma_2 \,\&\, \langle x, b\rangle \in \Gamma_3 \rightarrow \langle a, x_1\rangle \in \Gamma_1 \,\&\, \langle x_1, b\rangle \in \Gamma_2 \cdot \Gamma_3 \rightarrow \langle a, b\rangle \in \Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_3)$.

2. Достаточность: $\langle a, b\rangle \in \Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_3) \rightarrow \langle a, x\rangle \in \Gamma_1 \,\&\, \langle x, b\rangle \in \Gamma_2 \cdot \Gamma_3 \rightarrow \langle a, x\rangle \in \Gamma_1 \,\&\, \langle x, z\rangle \in \Gamma_2 \,\&\, \langle z, b\rangle \in \Gamma_3 \rightarrow \langle a, z\rangle \in \Gamma_1 \cdot \Gamma_2 \,\&\, \langle z, b\rangle \in \Gamma_3 \rightarrow \langle a, b\rangle \in (\Gamma_1 \cdot \Gamma_2) \cdot \Gamma_3$.

3. Следовательно, тождество справедливо.

5.3 Свойства соответствий

Соответствие Γ называется *функциональным*, если его трафик G функционален. График G называется *функциональным*, если в нем нет пар с одинаковыми первыми и разными вторыми элементами. Другими словами, из элементов области отправления может выходить не более одной стрелки.

Следовательно, соответствие $\Gamma = \langle G, X, Y\rangle$ функционально тогда, когда истинно

$(\forall x \in X, \forall y_1, y_2 \in Y)\ [\langle x, y_1\rangle \in G \,\&\, \langle x, y_2\rangle \in G \Rightarrow y_1 = y_2]$.

Соответствие Γ называется *инъективным*, если его график инъективен. График G называется *инъективным*, если в нем нет пар с разными первыми и одинаковыми вторыми элементами.

Отметим, что в частном случае инъективные и функциональные, графики могут совпадать.

Соответствие инъективно, когда справедливо:

$(\forall x_1, x_2 \in X, \forall y \in Y)\ [<x_1,y> \in G \ \& <x_2, y> \in G \Rightarrow x_1 = x_2]$.

Соответствие Г = <G, X, Y> называется *всюду определенным*, если его область определения совпадает с его областью отправления.

Пример. Г = <G,X,Y> = < {<1,2>, <3,2>, <4,5>}; {1, 3, 4}; {2, 5}>. Здесь область отправления соответствия X = {1, 3, 4}совпадает с областью определения.

Для всюду определенного соответствия справедливо выражение:

$пр_1 G = X$.

Аналогично можно записать:

$(\forall x)(\exists y)[<x,y> \in G]$.

Соответствие Г = <G, X, Y>называется *сюръективным*, если его область значений совпадает с его областью прибытия.

Пример. Г =<G, X, Y> = <{<l, b>, <2, a>}; {1, 2, 3}; {a, b}>. Здесь область прибытия соответствия X ={a, b} совпадает с областью значений.

Для сюръективного соответствия справедливо выражение:

$пр_2 G = Y$.

Аналогично можно записать:

$(\forall y)(\exists x)[<x,y> \in G]$.

Соответствие Г = <G, X, Y> называется *биективным* соответствием или *биекцией*, или *взаимооднозначным соответствием*, если оно функционально, инъективно, всюду определено и сюрьективно.

Частным случаем соответствия является понятие *отображения*. Всюду определенное соответствие называется *отображением* X в Y и записывается

$G: X \rightarrow Y$.

5.4 Отображения множеств

Пусть X и Y — два непустых множества.

Отображением $f: X \to Y$ (множества X во множество Y) называется тройка (X, Y, f). Здесь X, Y — два непустых множества, f – правило, сопоставляющее каждому элементу $x \in X$ однозначно определенный элемент $y = f(x) \in Y$. Множество X называется *областью определения* отображения, элемент $x \in X$ – *аргументом* отображения f, элемент $f(x) \in Y$ –*образом* элемента x при отображении f. При этом пишут $x \to f(x)$. Часто, в случае когда множества X, Y –числовые, отображение называют *функцией*. Если только множество Y– числовое, то отображение называют **функционалом**.

Если $A \subseteq X$, то $f(A) = \{f(x) : x \in A\}$ называется *образом* подмножества A при отображении f. *Прообразом* подмножества $B \subseteq Y$ называется множество $\{x \in X : f(x) \in B\}$,которое будем обозначать $f^{-1}B$. В частности, для $B = \{y\}$ любой элемент из множества $f^{-1}B(\{y\})$ называется *прообразому*.

Пример. Пусть $X = Y = \{1,2,3\}$. Отображение $f: X \to Y$ задано следующим образом:

$f(1)=1; f(2)=1; f(3)=2.$

Тогда $f(X) = \{1,2\}$. У элемента $1 \in Y$ два прообраза — 1 и 2; у элемента $2 \in У$ один прообраз — 3; у элемента $3 \in Y$ прообразов нет.

Отображение $f: X \to Y$ называется *сюръективным* (или *отображением «на»*), если $f(X) = Y$, т. е. для каждого элемента из Y есть прообраз.

Отображение $f: X \to Y$ называется *инъективным* (или *отображением «в»*), если из $f(x) = f(x^1)$ следует, что $x = x^1$, т. е. для каждого элемента Y имеется не более одного прообраза.

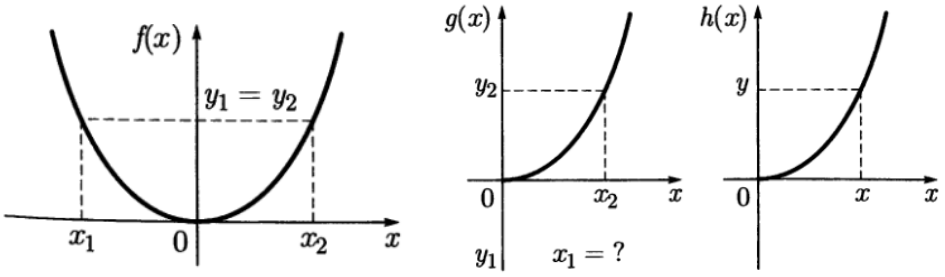
Отображение $f: X \to Y$ называется *биективным* (или *взаимно-однозначным*), если это отображение одновременно и сюръективно, и инъективно, т. е. это отображение «на» и каждый элемент множества Y имеет ровно один прообраз. (Одно и то же правило соответствия может быть

сюръективным, инъективным или биективным отображением в зависимости от исходных множеств X и У.)

Пример. Обозначим через $R^+ = \{x \in R: x \geq 0\}$. Рассмотрим следующие три отображения:

$f: R \to R^+$; $g: R^+ \to R$; $h: R^+ \to R^+$;

Эти отображения зададим одной формулой: $f(x) = x^2$; $g(x) = x^2$; $h(x) = x^2$. Они различны, так как различны исходные множества. При этом *f* является сюръективным, но не инъективным; *g* — инъективно, но не сюръективно; *h* — биективно.



Отображения вида $f: X \to X$ называются *преобразованиями* множества X.

Тождественным преобразованием данного множества *X* называется преобразование e_x такое, что $e_x(x) = x, \forall x \in X$.

Пусть $f: X \to Y$ и $g: Y \to Z$ – некоторые отображения. *Суперпозицией* этих отображений называется отображение $gf: X \to Z$, определяемое следующим образом:

$(gf)(x) = g(f(x)), x \in X$.

Заметим, что суперпозиция определена не для любых пар отображений. Однако суперпозиция двух преобразований одного и того же множества определена всегда.

Пусть $f: X \to Y$ и $g: Y \to X$

Отображение *g* называется *обратным* к отображению *f* (а отображение *f* обратным к *g*), если $fg = e_y; gf = e_x$.

Если обратное отображение существует, то оно единственно. В самом деле, пусть $f: X \to Y$ — некоторое отображение множества X во множество Y и отображения $g: Y \to X$ и $h: Y \to X$ — отображения, обратные к f.

Тогда

$(g(fh))(y) = (ge_y)(y) = g(y)$

и $((gf)h)(y) = (e_x h)(y) = h(y)$.

Имеем $g(fh) = (gf)h$. Отсюда получаем $g(y) = h(y), \forall y \epsilon Y$, т.е. отображения g и hсовпадают.

Обратное отображение обозначается f^{-1}. Оно существует не всегда. Необходимое и достаточное условие существования обратного отображения дает следующая теорема.

Теорема. Отображение f имеет обратное тогда и только тогда, когда оно биективно.

Доказательство. Пусть $f: X \to Y$.

1. Необходимость: Итак, пусть существует обратное отображение $f^{-1} = g: Y \to X$.

Рассмотрим любой $y \epsilon Y$ и $x = g(y)$. Тогда $f(x) = f(g(y)) = y$ и x – прообраз y при отображении f. Таким образом, любой $y \epsilon Y$ имеет прообраз x, т. е. f сюръективно.

Далее, если $x, x^1 \epsilon X$, причем $f(x) = f(x^1)$, то $g(f(x)) = g(f(x^1))$. Следовательно, т. е.

$e_x(x)=e_x(x^1)$,

$x = x^1$ и f инъективно. Отсюда f биективно, и необходимость доказана.

2. Достаточность: Пусть f биективно.

Определим отображение $g: Y \to X$ следующим образом. Положим $g(y) = x$, если $f(x) = y$. В силу биективности f отображение g определено на всем Y, и $g = f^{-1}$.

5.5 Функция

Понятие «функции» является одним из основополагающих в математике. В данном случае подразумеваются прежде всего функции, отображающие одно конечное множество объектов в другое конечное множество. Мы избегаем использования термина «отображение» и предпочитаем слово «функция» в расчете на постоянное сопоставление читателем математического понятия функции с понятием функции в языках программирования.

Пусть f — отношение из A в B, такое что

$\forall a\ (a,b) \in f \& (a,c) \in f \Longrightarrow b = c.$

Такое свойство отношения называется *однозначностью,* или *функциональностью,* а само отношение называется *функцией из A в B* и обозначается следующим образом: $f:A\rightarrow$B. Если f: $A\rightarrow B,$ то обычно используется *префиксная* форма записи:

$b = f(a) = (a,b) \in f.$

Если b = f(a), то a называют аргументом, а b — значением функции.

Пусть $f:A\rightarrow B$, тогда

область определения функции: $f_A = \{a \in A \mid \exists b \in B\ b = f(a)\};$

область значений функции: $f_B = \{b \in B \mid \exists a \in A\, b = f(a)\}.$

Если $f_A = A$, то функция называется *тотальной,* а если $f_A \neq A$ — *частичной. Сужением* функции $f:A\rightarrow B$ на множество M ⊂ A называется функция $f|_M$, определяемая следующим образом:

$f|_M = \{(a, b) \mid (a, b) \in f \& a \in M\}$

Для тотальной функции $f = f|_{fA}.$

Функция f: $A_1 \times \ldots \times A_n \rightarrow B$ называется функцией *n аргументов,* или *n-местной* функцией.

Пусть $f:A \to B$. Тогда функция f называется:

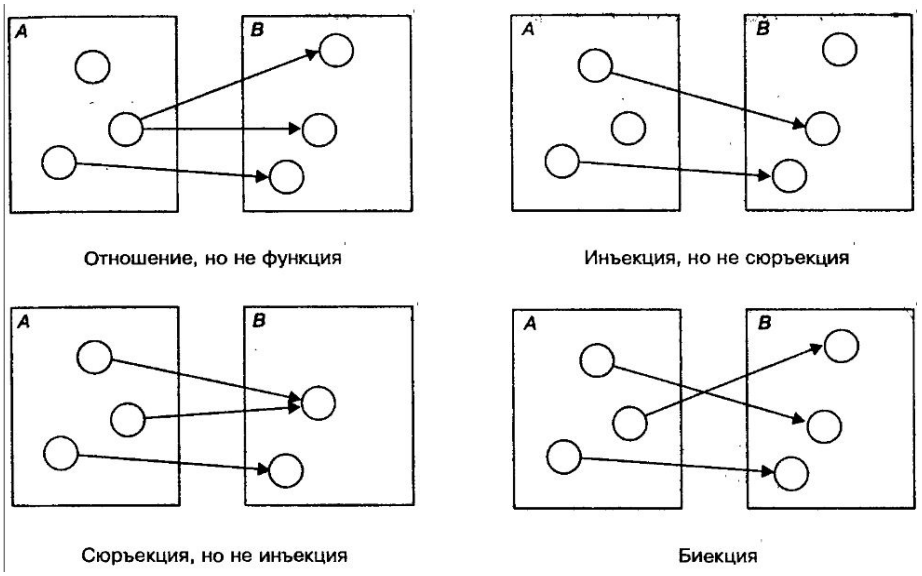
инъективной, если $b = f(a_1) \,\&\, b = f(a_2) \Rightarrow a_1 = a_2$;

сюръективной, если $\forall b \in B \exists a \in A\ b = f(a)$;

биективной, если она инъективная и сюръективная.

Биективную функцию также называют *взаимнооднозначной.*

Следующий рисунок иллюстрирует понятия отношения, функции, инъекции, сюръекции и биекции.



Теорема. Если $f:A \to B$ — тотальная биекция (f_A = A), то отношение $f^{-1} \subset B \times A$ (обратная функция) является биекцией.

Доказательство.

Поскольку f — биекция, имеем $(b_1 = f(a) \,\&\, b_2 = f(a) \Rightarrow b_1 = b_2)\ \&(b = f(a_1) \,\&\, b = f(a_2) \Rightarrow a_1 = a_2)\ \&(\forall b \in B \exists a \in A\ b = f(a))$.

Покажем, что f^{-1} — функция.

$f^{-1} = \{(b,a) \mid a \in A\ \&b \in B \& b = f(a)\}$.

Пусть $a_1 = f^{-1}(b)\ \&a_2 = f^{-1}(b)$. Тогда $b = f(a_1)\ \&b = f(a_2) \Rightarrow a_1 = a_2$.

Покажем, что f^{-1}— инъекция. Пусть $a_1 = f^{-1}(b_1)$& $a_2 = f^{-1}(b_2)$. Тогда $b_1 = f(a)$&$b_2 = f(a) \Longrightarrow b_1 = b_2$.

Покажем от противного, что f^{-1}— сюръекция.

Пусть $\exists a \epsilon A \neg \exists b \epsilon B\ a = f^{-1}(b)$. Тогда $\exists a \epsilon A \forall b \epsilon B\ a \neq f^{-1}(b)$. Обозначим этот элемент a_0. Имеем $\forall b a_0 \neq f^{-1}(b) \implies \forall b b \neq f(a_0) \Longrightarrow a_0 \notin f_A \subset A \rightarrow a_0 \notin A$.

Пусть $f: A \rightarrow B$ и пусть $A_1 \subset A$, $B_1 \subset B$. Тогда множество

$F(A_1) = \{b \epsilon B \mid \exists a \epsilon A_1 b = f(a)\}$

называется *образом* множества A_1, а множество

$F^{-1}(B_1) = \{a \epsilon A \mid \exists b \epsilon B_1 b = f(a)\}$

прообразом множества B_1. Заметим, что Fявляется отношением из множества 2^{fA}в множество 2^{fB}:

$F = \{(A_1, B_1) \mid A_1 \subset A \& B_1 \subset B \ \& \ B_1 = F(A_1)\}$.

Теорема. Если $f: A \rightarrow B$ функция, то$F: 2^{fA} \rightarrow 2^{fB}$и $F^{-1}: 2^{fB} \rightarrow 2^{fA}$– тоже функции.

F называется *индуцированной* функцией, а F^{-1}— *переходом к прообразам.*

Принцип Дирихле. Пусть $f: A \rightarrow B$– функция, причем X и Y – конечные множества. Если |X|>то по крайней мере одно значение fвстретится более одного раза. Неформально, принцип Дирихле можно например записать следующим образом:

Если X – множество белок, аY – множество клеток, и |X| = 12, а |Y| =11, то 12 белок нельзя посадить в 11 клеток так, чтобы в каждой клетке находилась одна белка.

Мультимножества

6.1 Понятие мультимножества

Мы рассмотрели конечные множества, в которых отсутствуют повторяющиеся элементы. В кортежах возможны повторяющиеся элементы, но при этом значение каждого элемента определяется его местоположением. В задачах искусственного интеллекта начинают использоваться объекты с повторяющимися элементами. Л.Б. Петровский такие элементы назвал мультимножествами и разработал основы их теории.

Мультимножество - это множество с повторяющимися элементами, где один и тот же элемент может присутствовать многократно, особенностью мультимножества является понятие кратности вхождения элемента. Элементы мультимножеств будем обозначать строчными буквами с подстрочным индексом М: a_M, b_M, ...; а мультимножества – прописными буквами с подстрочным индексом М: A_M, B_M, ...

Примером мультимножеств могут служить, например, следующие совокупности элементов *a, b, c, d, e, f, g, h*:

A_M = {a, b, a, d, e, c, a, b, h, h}, B_M = {d, d, e, b, b, d, e, e, h}, C_M = {a, a, d, a, c, a, a, e, c, c, g, g, g}.

Порядок следования элементов в мультимножестве считается несущественным. Тогда приведенные мультимножества A, B, C можно переписать следующим образом:

A_M = {3a, 2b, c, d, e, 2h}

Отметим, что отсутствующие элементы не указываются в записи мультимножества.

Формальное определение мультимножества, данное А.Б Петровским:

Мультимножеством A_M, определенном на множестве A={x_1, x_2, …}, вес элементы x_i, которого различны, называется совокупность групп одинаковых элементов

$A_M=\{k_1x_1, k_2x_2, \ldots\}, x_i \in A.$

Группу одинаковых элементов k_ix_i, называют *компонентой мультимножества*, элементы x_i, входящие в компоненту k_ix_i, – *экземплярами элементов мультимножества.* Функция k_i принимающая числовые значения, определяет число вхождений элемента $x_i \in A$в мультимножество A_M. Ее также называют *функцией кратности* или *функцией числа экземпляров мультимножестваA*$_M$.

Говорят, что элемент x_i принадлежит мультимножеству A_M (обозначается $x_i \in A_M$) и в мультимножестве A_M имеется ровно kэкземпляров элемента x_i, тогда и только тогда, когда кратность элемента x_i равна $k_ix_i > 0$. Когда кратность элемента x_i равна нулю $k_ix_i = 0$, тогда говорят, что элемент x_i не содержится в мультимножестве A_M (обозначается $x_i \notin A_M$). Тем самым принадлежность элемента x_i мультимножеству A_M определяется значением функции кратности.

Если все мультимножества семейства $\Theta(A_M) = \{A1_M, A2_M, \ldots\}$ образуются из элементов одного и того же множества $G = \{x_1, x_2, \ldots\}$, то множество G называется *порождающим множеством* или *доменом* для семейства $\Theta(A_M)$. В качестве порождающего множества G может выступать любое непустое (конечное или бесконечное) множество.

Основными характеристиками мультимножества являются мощность и размерность. *Мощность* мультимножества A_Mопределяется как общее число экземпляров всех его элементов

$|A_M| = \text{card}A_M$,

а *размерность* мультимножества *A*– как общее число различных элементов

$/A_M/ = \dim A_M$.

Размерность мультимножества не превосходит его мощности и мощности домена $/A_M/ \leq |A_M|, /A_M/ \leq |G|$. Мощность мультимножества $|A_M|$ в

общем случае не связана с мощностью домена $|G|$. Конечные мультимножества, имеющие мощность m и состоящие из m элементов (считая повторения), называют *m-кардинальными мультимножествами* или *m-мультимножествами*, а имеющие размерность nи состоящие из n компонент - *n-мерными мультимножествами.*

Высотой или *пиковым значением* мультимножества A_M называется максимальное значение его функции кратности k_i, а элемент x_{A*}, для которого функция кратности k_A максимальна, - *пиком* или пиковым элементом мультимножества A_M.

Мультимножество удобно изображать графически в виде ступенчатой гистограммы, по оси абсцисс которой расположены элементы основного множества A или домена G, а по оси ординат отложены значения $k_i(x_i)$ функции кратности, показывающие количество экземпляров элемента x_i в мультимножестве A_M. Таким образом, каждый столбец гистограммы соответствует определенной компоненте мультимножества A_M. Ширина гистограммы равна размерности $/A_M/$ мультимножества, а высота гистограммы есть высота мультимножества A_M. Мощность мультимножества $|A_M|$ будет численно равна площади фигуры, ограниченной гистограммой.

Для мультимножеств справедливы теоретико-множественные понятия, введенные для множеств.

Рассмотрим возможные способы сопоставления мультимножеств, обусловленные особенностями их различных характеристик. Мультимножества A_M и B_M называются равными ($A_M = B_M$), если $k_i(x_i) = k_j(x_j)$ для всех элементов x_i, $x_j \in G$, $k_i(x_i) \in A_M, k_j(x_j) \in B_M$. В противном случае эти мультимножества неравны. Для равных мультимножеств имеем $|A| = |B|$, $/A/ = /B/$.

Мультимножества A и B называют:

- *равномощными*, если $|A|=|B|$.

- *равноразмерными*, если /A/=/B/.
- *равными,* если они равномощны и равноразмерны.

Говорят, что мультимножество B_Mсодержится или включено в мультимножество A_M($A_M \subseteq B_M$), если $k_j x_j \leq k_i x_i$,для каждого элемента x_i, $x_j \in$G, $k_i x_i \in A_M$, $k_j x_j \in B_M$. Мультимножество B_M называемся тогда *подмультимножеством* мультимножества A_M, а мультимножество A_M – *надмультимножеством* мультимножества B_M.

Включение мультимножества обладает свойствами рефлексивности ($A_M \subseteq A_M$) и транзитивности ($A_M \subseteq B_M$, $B_M \subseteq C_M \rightarrow A_M \subseteq C_M$), а значит, является отношением предпорядка.

6.2 Операции над мультимножествами

Над мультимножествами определены следующие основные операции: объединение, пересечение, арифметическое сложение, арифметическое вычитание, дополнение, симметрическая разность, умножение на число, арифметическое умножение и возведение в арифметическую степень, прямое произведение и возведение в прямую степень.

Объединением мультимножеств A_M и B_M называется мультимножество, состоящее из всех элементов, которые присутствуют хотя бы в одном из мультимножеств, и кратность каждого элемента равна максимальной кратности соответствующих элементов в объединяемых мультимножествах:

$C_M = A_M \cup B_M = \{\max(k_i x_i, k_j x_j)\}$, $k_i x_i \in A_M$, $k_j x_j \in B_M$.

Другими словами, производится попарное сравнивание каждого экземпляра мультимножеств и в каждой паре выбирается экземпляр с наибольшим значением функции кратности.

Пересечением мультимножеств A_M и B_M называется мультимножество, состоящее из всех элементов, которые одновременно присутствуют в каждом из мультимножеств, и кратность каждого элемента равна минимальной кратности соответствующих элементов в пересекаемых мультимножествах:

$C_M = A_M \cap B_M = \{\min(k_i x_i, k_j x_j)\}$, $k_i x_i \in A_M$, $k_j x_j \in B_M$.

Другими словами, производится попарное сравнивание каждого экземпляра мультимножеств и в каждой паре выбирается экземпляр с наименьшим значением функции кратности.

Арифметической суммой мультимножеств A_M и B_M называется мультимножество, состоящее из всех элементов, которые присутствуют хотя бы в одном из мультимножеств, и кратность каждого элемента равна сумме кратностей соответствующих элементов в складываемых мультимножествах:

$C_M = A_M + B_M = \{kx \mid kx = k_i x_i + k_j x_j\}$, $k_i x_i \in A_M$, $k_j x_j \in B_M$.

Операции объединение, пересечение и арифметическое сложение можно выполнять для произвольного числа мультимножеств.

Арифметической разностью мультимножеств A_M и B_M называется мультимножество, состоящее из тех элементов мультимножества A_M, кратность которых превышает кратность соответствующих элементов в мультимножестве B_M. Кратность каждого элемента результирующего множества равна разности кратностей соответствующих элементов в вычитаемых мультимножествах:

$C_M = A_M - B_M = \{kx \mid kx = k_i x_i - k_j x_j$, если $k_i > k_j$; 0, в противном случае$\}$, $k_i x_i \in A_M$, $k_j x_j \in B_M$.

$A_M \subseteq B_M \leftrightarrow A_M - B_M = \emptyset$

Симметрической разностью мультимножеств A_M и B_M называется мультимножество, состоящее из тех элементов мультимножества A_Mи B_M, кратности которых различны. Кратность каждого элемента результирующего множества равна модулю разности кратностей соответствующих элементов в вычитаемых мультимножествах:

$C_M = A_M \backslash B_M = \{kx \mid kx = |k_i x_i - k_j x_j|\}$, $k_i x_i \in A_M$, $k_j x_j \in B_M$.

Арифметическая и симметрическая разности мультимножеств применима только к двум мультимножествам.

Дополнением мультимножества A_M до универсума U называется мультимножество, состоящее из тех элементов, кратность которых равна разности кратностей соответствующих элементов в универсуме U и дополняемом мультимножестве A_M. Под *универсумом* в данном случае понимается некоторое мультимножество U, такое, что все остальные мультимножества являются подмультимножествами данного множества U.

$A_M' = U - A_M = \{k_A, x \mid k_U x - k_A x, \forall x \in U\}$

Из определений пустого мультимножества и дополнения мультимножества следует, что пустое мультимножество ∅ и универсум U взаимно дополняют друг друга: $\emptyset' = U$, $U' = \emptyset$.

Арифметической произведением мультимножеств A_M и B_M называется мультимножество, состоящее из элементов, которые одновременно присутствуют в каждом из мультимножеств, и их кратность равна произведению кратностей соответствующих элементов в перемножаемых мультимножествах:

$C_M = A_M \cdot B_M = \{kx \mid kx = |k_i x_i \cdot k_j x_j|\}$, $k_i x_i \in A_M$, $k_j x_j \in B_M$.

Прямым произведением мультимножеств A_M и B_M называется мультимножество, состоящее из всех упорядоченных пар элементов $\langle x_i, x_j \rangle$,

таких, что первый элемент каждой пары является элементом первого сомножителя $x_i \in A_M$,второй элемент пары - элементом второго сомножителя $x_j \in B_M$и кратность каждой пары $\langle x_i, x_j \rangle$ равна произведению кратностей элементов x_i и x_jв перемножаемых мультимножествах:

$C_M = A_M \times B_M = \{k_{A \times B} \langle x_i, x_j \rangle \mid k_{A \times B} = k_i x_i \cdot k_j x_j\}$, $x_i \in A_M$, $x_j \in B_M$.

По аналогии с множествами, для мультимножеств также можно сформулировать некоторые правила выполнения операций:

$(A \cup B)' = A' \cap B'$;

$(A \cap B)' = A' \cup B'$;

$(A+B)' = A' - B = B' - A$;

$(A - B)' = A' + B$;

$A' - B' = B - A$;

$A + B = (A \cup B) + (A \cap B)$;

$A \backslash B = (A \cup B) - (A \cap B) = A' \backslash B'$;

$(A - B) \cap (B - A) = \emptyset$;

$A \cup B = (A + B) - (A \cap B) = (A \cap B) + A \backslash B = A + (B - A) = B + (A - B)$;

$A \cap B = (A + B) - (A \cup B) = (A \cup B) - A \backslash B = A - (A - B) = B - (B - A)$;

$A - B = (A \cup B) - B = A - (A \cap B) = (A \cup B) \backslash B = A \backslash (A \cap B)$;

$A \backslash B = (A \cup B) - (A \cap B) = (A - B) + (B - A) = (A - B) \cup (B - A) = (A + B) - 2 \cdot (A \cap B)$;

$A + B = (A \cup B) + (A \cap B) = A \backslash B + 2 \cdot (A \cap B)$.

Л Е К Ц И Я

по учебной дисциплине:
«Специальные главы математики»

Для магистров направления:
09.04.02 Информационные системы и технологии

Тема 2. Сетевое планирование и управление

Лекция 2. Сетевые модели топологических процессов. Классификация сетевых моделей. Этапы построения сетевой модели. Способы построения сетевого графика. Критический путь. Методы определения критического пути. Резервы, содержащиеся в некритических работах. Формализованное представление сетевого графика. Оптимизация сетевого графика. Форсирование критических работ. Перераспределение резервов. Высвобождение средств за счет пролонгирования работ.

Лекция 2. Сетевые модели топологических процессов. Классификация сетевых моделей. Этапы построения сетевой модели. Способы построения сетевого графика. Критический путь. Методы определения критического пути. Резервы, содержащиеся в некритических работах. Формализованное представление сетевого графика. Оптимизация сетевого графика. Форсирование критических работ. Перераспределение резервов. Высвобождение средств за счет пролонгирования работ.

Цель лекции: Изучить основные сетевые модели. Ознакомиться с экономической и геометрической интерпретацией задач. Изучить основные понятия и принципы применения сетевых графиков.

План лекции:

1. Сетевые модели топологических процессов.
2. Классификация сетевых моделей.
3. Этапы построения сетевой модели.
4. Способы построения сетевого графика.
5. Критический путь.
6. Методы определения критического пути.
7. Резервы, содержащиеся в некритических работах.
8. Формализованное представление сетевого графика.
9. Оптимизация сетевого графика.
10. Форсирование критических работ.
11. Перераспределение резервов.
12. Высвобождение средств за счет пролонгирования работ.

Методика проведения лекции – лекция –визуализация

Данный вид лекции является результатом нового использования дидактического принципа наглядности. Психологические и педагогические исследования показывают, что наглядность не только способствует более успешному восприятию и запоминанию учебного материала, но и позволяет активизировать умственную деятельность, глубже проникать в сущность изучаемых явлений, показывает ее связь с творческими процессами принятия решений, подтверждает регулирующую роль образа в деятельности человека.

Лекция - визуализация учит студентов преобразовывать устную и письменную информацию в визуальную форму, что формирует у них

профессиональное мышление за счет систематизации и выделения наиболее значимых, существенных элементов содержания обучения. Этот процесс визуализации является свертыванием мыслительных содержаний, включая разные виды информации, в наглядный образ; будучи воспринят, этот образ, может быть развернут и может служить опорой для мыслительных и практических действий.

При подготовке и проведении лекции-визуализации преподавателю следует обратить внимание на следующие особенности реализации рассматриваемой формы проведения занятия. По содержанию визуализованная лекция представляет собой устную информацию, преобразованную в визуальную форму. Видеоряд, будучи воспринятым и осознанным, сможет служить опорой адекватных мыслей и практических действий. Преподаватель должен выполнить такие демонстрационные материалы, такие формы наглядности, которые не только дополняют словесную информацию, но сами выступают носителями содержательной информации.

Подготовка такой лекции состоит в реконструировании, перекодировании содержания лекции или ее части в визуальную форму для предъявления студентам через технические средства обучения.

Чтение такой лекции сводится к сводному, развернутому комментированию подготовленных визуальных материалов.

Визуализированные материалы:

1. **Графы**
2. **Этапы построения сетевой модели.**
3. **Способы построения сетевого графика.**
4. **Методы определения критического пути.**
5. **Оптимизация сетевого графика.**

Основные понятия теории графов

1.1 Определения и примеры

Граф G=(V, E) состоит из двух множеств: конечного множества элементов, называемых вершинами, и конечного множества элементов, называемых ребрами. Каждое ребро определяется парой вершин. Если ребра графа определяются упорядоченными парами вершин, то G называется направленным или *ориентированным графом*. В противном случае G называется ненаправленным или *неориентированным графом.* Для обозначения вершин графа будем использовать символы v_1, v_2, v_3,…, а для обозначения ребер - e_1, e_2, e_3, ... Вершины v_i и v_j, определяющие ребро e_i, называются концевыми вершинами ребра e_i. В этом случае ребро e_i обозначается как $e_i=(v_i, v_j)$. Заметим, что в множестве E допускается более чем одно ребро с одинаковыми концевыми вершинами. Все ребра с одинаковыми концевыми вершинами называются параллельными. Кроме того, концевые вершины ребра не обязательно различны. Если $e_i= (v_i, v_i)$, то ребро e_i называется петлей. Граф называется простым, если он не содержит петель и параллельных ребер. Граф G является графом порядка n, если множество его вершин состоит из n элементов.

Граф, не имеющий ребер, называется *пустым*. Граф, не имеющий вершин (и, следовательно, ребер), называется *нуль-графом*.

Графически граф может быть представлен диаграммой, в которой вершина изображена точкой или кружком, а ребро — отрезком линии, соединяющим точки или кружки, соответствующие концевым вершинам ребра. Например, если $V=\{v_1\ v_2, v_3, v_4, v_5, v_6\}$ и $E= \{e_1, e_2, e_3, e_4, e_5\}$, такие, что $e_1=(v_1, v_2)$, $e_2=(v_1, v_4)$, $e_3= (v_5, v_6)$, $e_4= (v_1, v_2)$, $e_5=(v_5,v_5)$, тогда граф G=(V, E) представляется так, как изображено на рис. 1. В этом графе e_1 и e_4 - параллельные ребра, e_5 - петля. Говорят, что ребро *инцидентно* своим концевым вершинам. Две вершины *смежны*, если они являются концевыми

вершинами некоторого ребра. Если два ребра имеют общую концевую вершину, они называются *смежными*.

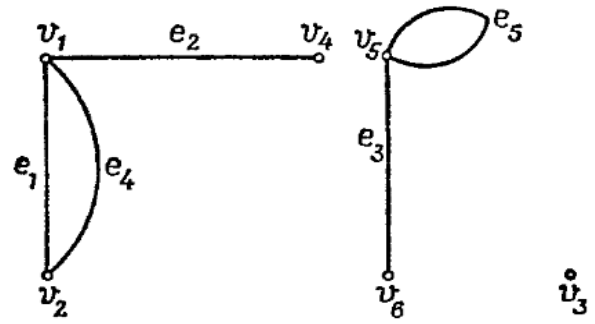


Рисунок 1

Например, в графе на рис. 1 ребро e_1 инцидентно вершинам v_1 и v_2; v_1 и v_4 являются смежными вершинами, а e_1 и e_2 - смежными ребрами.

Число инцидентных вершине v_i ребер называется степенью вершины и обозначается $d(v_i)$. Иногда степень вершины называется также ее валентностью. Вершина степени 1 называется *висячей вершиной*. Единственное ребро, инцидентное висячей вершине, называется висячим. Вершина степени 0 называется *изолированной*. По определению петля при вершине v_i добавляет 2 в степень соответствующей вершины. Величины $\delta(G)$ и $\Delta(G)$ обозначают минимальную и максимальную степени вершины в G соответственно.

В графе G на рис. 1 $d(v_1) = 3$, $d(v_2) = 2$, $d(v_3) = 0$, $d(v_4) = 1$, $d(v_5) = 3$, $d(v_6) = 1$.

Заметим, что v_3- изолированная вершина, v_4 и v_6- висячие вершины, e_2 — висячее ребро. Легко проверить, что сумма степеней вершин в данном графе G равна 10, тогда как число ребер равно 5. Таким образом, сумма степеней вершин графа G равна удвоенному числу ребер графа G и, следовательно, является четным числом. Более того, можно показать, что число вершин графа G нечетной степени также четно. Эти результаты свойственны не только графу на рис. 1.

1.2 Способы задания графов

Первое и на наш взгляд самое простое задание графа - это представление его с помощью картинки в соответствии с геометрическим определением графа. При этом, в соответствии с договоренностью выше, вершинам конкретного представления графа будут приписаны номера.

Так на рис.2 даны два представления одного и того же графа.

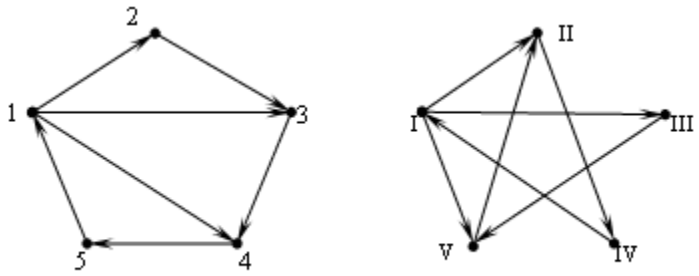


Рисунок 2

Другое задание графа - списком. Можно считать, что в соответствии с теоретико-множественным определением графа все элементы множества $R \subseteq V \times V$, входящего в определение, т.е. упорядоченные пары, упорядочены сначала по первым элементам пар, а затем по вторым, в соответствии с нумерацией вершин (нумерацией элементов множества V). Тогда два представления графа с рис.2 будут заданы двумя списками (рис. 3):

1	2, 3, 4
2	3
3	4
4	5
5	1

I	II, III, V
II	IV
III	V
IV	I
V	II

Рисунок 3

В первом столбце - первые элементы пар, затем по строкам, списком через запятую, идут вторые элементы.

Третье задание графа - матрицами. Ниже выписаны две матрицы - A и B, задающие два представления графа с рис. 2:

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Рисунок 4

Большинство задач удобно решать при использовании матричного задания графов. Квадратная таблица R = $||r_{i,j}||n_x(n)$ называется матрицей смежности, если ее элементы образуются по правилу:

$$r_{i,j} = \begin{cases} 1, \text{если вершина xi смежна с вершиной xj} \\ 0, \text{в противном случае} \end{cases}$$

Представления графа в соответствии с различными определениями будем называть различными видами представлений. Между различными видами представлений графа существует взаимнооднозначное соответствие. Действительно, поскольку речь идет о представлении графа, то множество вершин можно считать пронумерованным. Тогда дуге (ребро будем рассматривать как пару противоположно направленных дуг), идущей из вершины i в вершину j будет соответствовать упорядоченная пара (i, j) или, что то же самое, в списке вершины i будет присутствовать вершина j, а в матрице A = (a_{ij}), представляющей граф, элемента$_{ij}$ = 1. Отсутствию дуги, идущей из вершины i в вершину j, будет соответствовать отсутствие вершины j в списке вершины i, а a_{ij} = 0.

В силу указанного выше взаимнооднозначного соответствия между различными видами представлений мы и можем воспользоваться различными определениями одного и того же понятия - граф. При этом при изучении различных свойств графа мы стараемся каждый раз пользоваться тем языком, который наиболее удобен для описания выбранного свойства.

Вероятно, не следует выбирать один из языков в качестве единственного языка теории графов. Иногда для описания того или иного свойства, атрибута графа, требуется конкретный язык. Так если мы говорим о плоском (планарном) графе, то нам по необходимости приходится использовать геометрический язык теории графов. Если же мы говорим о "спектре" графа, то мы формулируем это понятие на матричном языке.

Теория графов замечательна тем, что трудные задачи (проблемы) в ней появляются и легко формулируются сразу же после формулировки основных понятий теории графов. Так после формулировки понятия представления (представителя) графа сразу же появляется задача - если нам даны два представления, то это представления одного и того же графа или разных? Т.е. мы сразу же вышли на так называемую проблему изоморфизма графов.

2.1 Типы графов

Граф G = (V, E) называют *полным*, если для любой пары вершин v_i и v_j в V существует ребро (v_i, v_j) в неориентированном графе $\overline{G}$=(V,$\overline{E}$) т. е. для каждой пары вершин графа G должна существовать, по крайней мере, одна дуга, соединяющая их (рис. 5,а).

Граф G = (V, E) называется *симметрическим*, если в множестве дуг E для любой дуги (v_i, v_j) существует также противоположно ориентированная дуга (v_j, v_i) (рис. 5,б).

Антисимметрическим называется такой граф, для которого справедливо следующее условие: если дуга (vi, vj)∈ A, то во множестве A нет противоположно ориентированной дуги, т. е. (v_j, v_i)∉ A (рис. 5,в). Очевидно, что в антисимметрическом графе нет петель.

В качестве примера можно рассмотреть граф, являющийся моделью некоторой группы людей: вершины графа интерпретируют людей, а дуги – их взаимоотношения. Так, если в графе дуга, нарисованная от вершины v_i к вершине v_j, означает, что v_i является другом или родственником v_j, тогда

данный граф должен быть симметрическим. Если дуга, направленная от v_i к v_j, означает, что вершина v_j подчинена вершине v_i, то такой граф должен быть антисимметрическим.

Комбинируя определения *полного* и *симметрического* графов и полного и *антисимметрического* графов, получили следующие определения:

- граф G =(V, E), в котором любая пара вершин (v_i, v_j) соединена двунаправленными дугами, называется *полным симметрическим* (рис. 5,г);
- граф G =(V, E), имеющий для каждой пары вершин (v_i, v_j) только одну дугу, называется полным *антисимметрическим* или *турниром*.

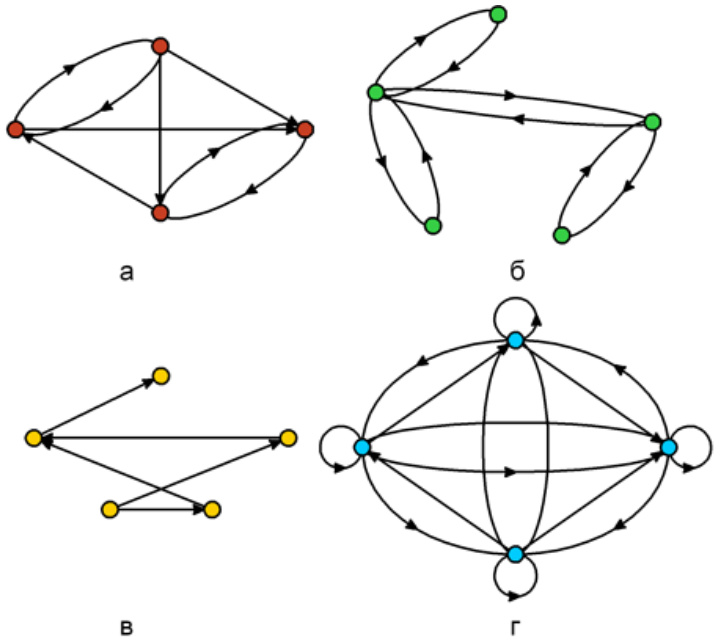


Рисунок 5

Связный граф, не имеющий циклов, либо граф, в котором каждая пара вершин соединена одной и только одной простой цепью, называется *деревом* (рис 6). Граф типа “дерево”: а – *неориентированное дерево*, б – *ориентированное дерево*.

Ориентированное дерево представляет собой ориентированный граф без циклов, в котором полустепень захода каждой вершины, за исключением одной (например, вершины v_1), равна 1, а полустепень захода вершины v_1 (называют корнем этого дерева) равна 0 (рис 6,б).

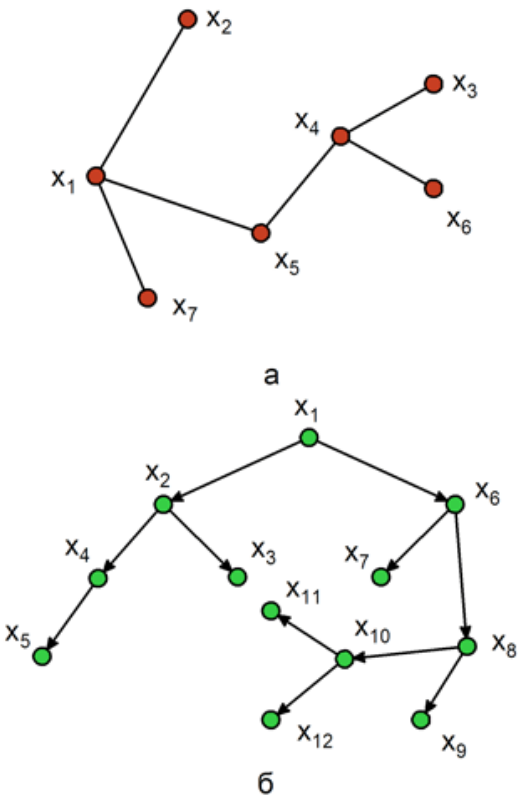


Рисунок 6

Граф G =(V, E), который может быть изображен на плоскости или сфере без пересечений называется *планарным* (рис 7).

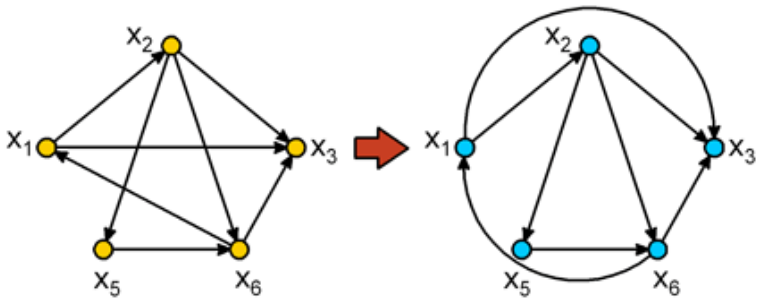


Рисунок 7

На рис. 8 показаны *непланарные* графы. Эти два графа играют важную роль в теории планарных графов и известны как графы Куратовского.

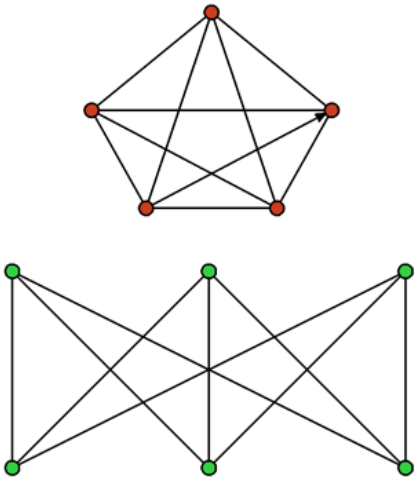


Рисунок 8

Неориентированный граф G = (V, E)называют *двудольным*, если множество его вершин V может быть разбито на такие два подмножества V^a и V^b, что каждое ребро имеет один конец в V^a , а другой в V^b (рис. 9,а).

Ориентированный граф G называется *двудольным*, если его *неориентированный двойник – двудольный граф (рис. 9,б,в).*

Двудольный граф G=($V^a \cup V^b$, E) называют полным, если для любых двух вершин $v_i \in V^a$ и $v_j \in V^b$ существует ребро (v_i,v_j) в G=(V,E) (рис. 9,г).

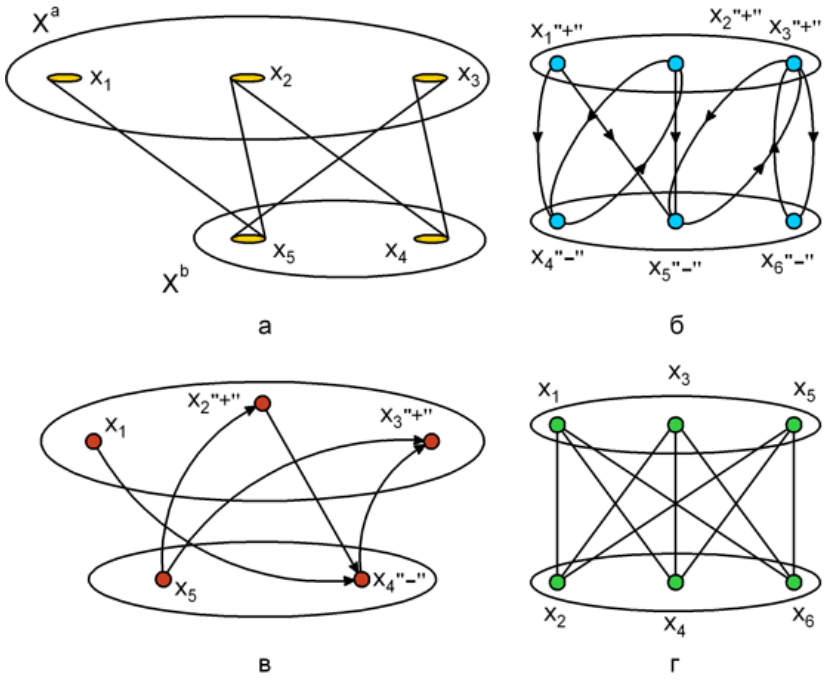


Рисунок 9

2.2 Подграфы

Граф G'(V, E') называется подграфом графа G(V, E) (обозначается G' ⊂G), если V' ⊂V и/или E' ⊂ E.

Если V' = V, то G' называется *остовным подграфом* G.

Если V' ⊂V& E' ⊂E&(V' ≠ V∨ E' ≠ E) то граф G' называется *собственным подграфом* графа G.

Подграф G'(V',E') называется правильным подграфом графа G(V,E), если G'содержит все возможные ребра G:

∀u, v∈V' (u,v) ∈E⟹ (u,v) ∈ E'.

Правильный подграф G'(V', E') графа G(V, E) определяется подмножеством вершин V'.

Виды подграфов (рис 10): а – исходный граф; б – подграфы; в – остовные подграфы; г – порожденные подграфы

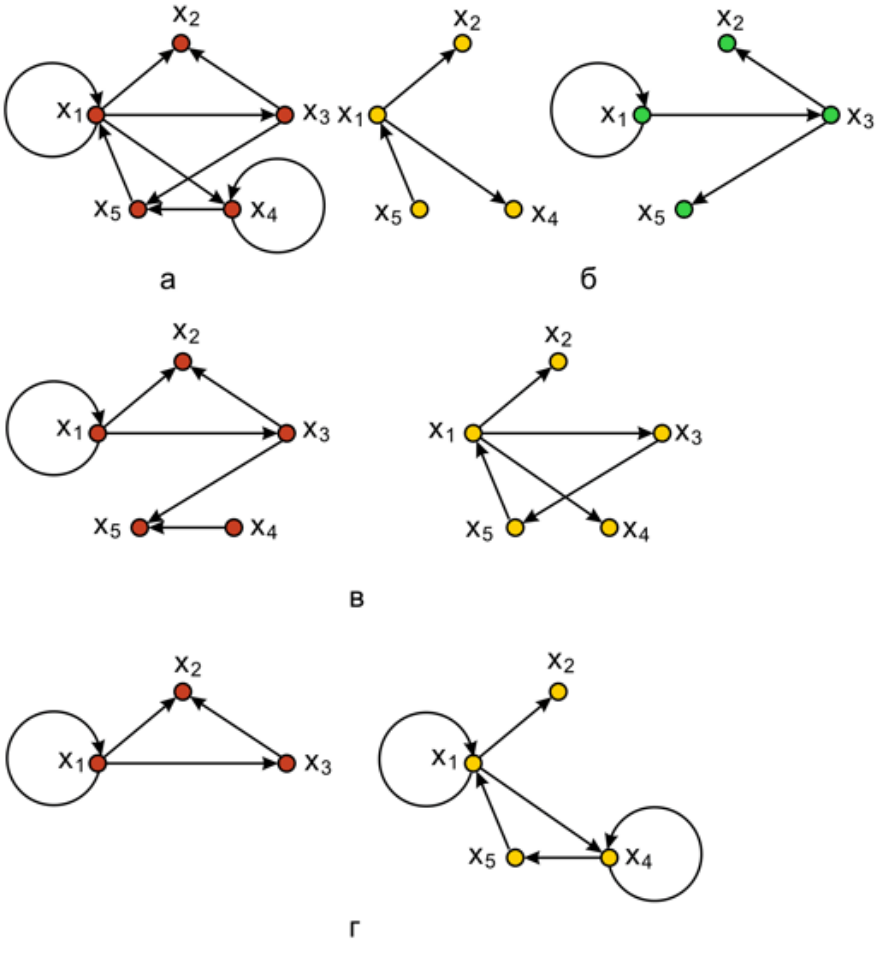


Рисунок 10

Остовным подграфом $G_p = (V, E_p)$ графа G называется граф, для которого $E_p \subset A$. Таким образом, остовный подграф имеет то же самое множество вершин, что и исходный граф G, но множество дуг подграфа G_p является подмножеством множества дуг исходного графа. Примеры остовных подграфов приведены на рис. 10,в. Для графа, имеющего m дуг, можно построить k остовных подграфов

$$k=C^1_m+C^2_m+...+C^{m-1}_m=2^m-1$$

Порожденным подграфом $G_s =(V_s, \Gamma_s)$ называется граф, для которого $V_s \subset V$ и для каждой вершины $v_i \in V_s$ прямое отображение $\Gamma_s(v_i) = \Gamma(v_i) \cap V_s$. Таким образом, порожденный подграф состоит из подмножества вершин V_s множества вершин исходного графа и всех таких дуг графа G, у которого конечные и начальные вершины принадлежат подмножеству V_s . Примеры порожденных подграфов приведены на рис. 10,г.

2.3 Сильно связные графы и компоненты графа

Кроме классификации типов графов данной в п. 2.2 графы могут быть классифицированы по связности: сильно связные, односторонне связные, слабо связные и несвязные.

Орграф называется *сильно связным*, или *сильным*, если для двух любых различных его вершин v_i и v_j существует, по крайней мере, один путь, соединяющий эти вершины. Это определение означает также, что любые две вершины сильно связного графа взаимодостижимы. Пример данного графа показан на рис. 11,а.

Виды графов по связности (рис. 11): а – сильно связный граф; б – односторонне связный граф; в – слабо связный граф; г – несвязный граф

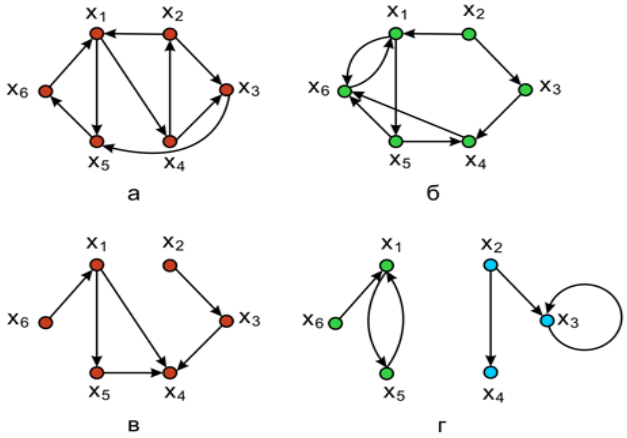


Рисунок 11

Орграф называется *односторонне связным*, или *односторонним*, если для любых двух различных его вершин v_i и v_j существует, по крайней мере, один путь из v_i в v_j или из v_j в v_i или оба пути существуют одновременно. Граф на рис. 11,б не является сильным, так как в нем нет пути из x_1 в x_3 , но является односторонне связным.

Орграф называется *слабо связным*, или *слабым*, если для любых двух различных вершин графа существует по крайней мере один маршрут, соединяющий их. Граф, изображенный на рис. 11,в, не является ни сильным, ни односторонним, поскольку в нем не существует путей от x_2 к x_5 и от x_5 к x_2 . Он слабо связный.

Орграф называется *несвязным*, если для некоторой пары вершин орграфа не существует маршрута, соединяющего их (рис. 11,г).

По признаку связности могут быть классифицированы и подграфы, но сначала введем понятие максимального подграфа. Пусть дано некоторое свойство P, которым могут обладать графы.

Максимальным подграфом графа G относительно свойства P называется *порожденный* подграф G_{sm} , обладающий этим свойством и такой, что не существует другого порожденного графа G_s , у которого $V_s \supset V_{sm}$ и который так же обладает свойством P. Так, например, если в качестве свойства P взята сильная связанность, то максимальным сильным подграфом

графа G является сильный подграф, который не содержится ни в каком другом сильном подграфе. Такой подграф называется сильной компонентой графа. Аналогично, односторонняя компонента представляет собой односторонний максимальный подграф, а слабая компонента – максимальный слабый подграф.

Например, в графе, приведенном на рис. 11,б, подграф, состоящий из вершин$\{x_1, x_4, x_5, x_6\}$, является сильной компонентой графа. С другой стороны подграфы, включающие вершины $\{x_1, x_6\}$ и $\{x_1, x_5, x_6\}$, не являются сильными компонентами (хотя и являются сильными подграфами), поскольку они содержатся в графе, состоящем из вершин $\{x_1, x_4, x_5, x_6\}$ и, следовательно, не максимальные. В графе, показанном на рис. 11,в, подграф не содержит вершины $\{x_1, x_4, x_5, x_6\}$, является односторонней компонентой.

В графе, приведенном на рис. 11,г, оба подграфа, включающие вершины $\{x_1, x_5, x_6\}$ и $\{x_2, x_3, x_4\}$ являются слабыми компонентами, и у этого графа только две компоненты.

Из определений сразу же следует, что односторонние компоненты графа могут иметь общие вершины. Сильная компонента должна содержаться по крайней мере в одной односторонней компоненте, а односторонняя компонента содержится в некоторой слабой компоненте данного графа.

2.4 Маршруты, цепи, пути и циклы

Маршрут в графе G = (V, E) представляет собой конечную чередующуюся последовательность вершин и ребер $v_0, e_1, v_2, e_2, \dots, v_{k-1}, e_k, v_k$, начинающуюся и кончающуюся на вершинах, причем V_{i-t} и V_i являются концевыми вершинами ребра e_1, $1 \le i \le k$. С другой стороны, маршрут можно рассматривать как конечную последовательность таких вершин $v_0, v_1, v_2, \dots, v_k$, что (v_{i-1}, v_i), $1 \le i \le k$ - ребро графа G. Такой маршрут обычно называется $v_0 - v_k$ -маршрутом, а v_0 и v_k - концевыми или терминальными вершинами маршрута. Все другие вершины маршрута называются внутренними.

Заметим, что ребра и вершины в маршруте могут появляться более одного раза.

Маршрут называется *открытым*, если его концевые вершины различны, в противном случае он называется замкнутым. В графе на рис. 12 последовательность v_1, e_1, v_2, e_2, v_3, e_8, v_6, e_9, v_5, e_7, v_3, e_{11}, v_6 является *открытым маршрутом*, а последовательность v_1, e_1, v_2, e_2, v_3, e_7, v_5, e_3, v_2, e_4, v_4, e_5, v_1-*замкнутым*.

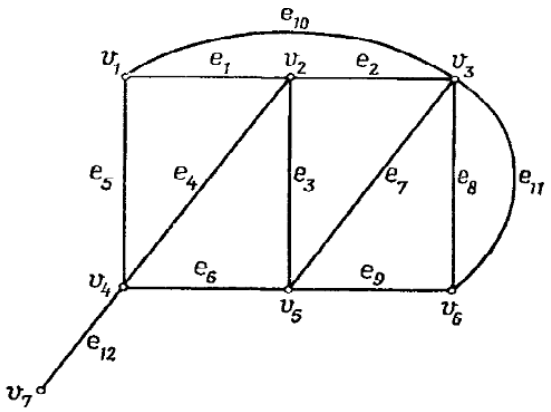


Рисунок 12

Маршрут называется *цепью*, если все его ребра различны. Цепь называется открытой, если ее концевые вершины различны, в противном случае она называется замкнутой. На рис. 12 цепь v_1, e_1, v_2, e_2, v_3, e_8, v_6, e_{11}, v_3 - *открытая*, а v_1, e_1, v_2, e_2, v_3, e_7, v_5, e_3, v_2, e_4, v_4, e_5, v_1 - *замкнутая*.

Открытая цепь называется *путем*, если все ее вершины различны. *Замкнутая цепь* называется *циклом*, если различны все ее вершины, за исключением концевых. Например, на рис. 12 последовательность v_1, e_1, v_2, e_2, v_3 является путем, а последовательность v_1, e_1, v_2, e_3, v_5, e_6, v_4, e_5, v_1 - циклом.

Ребро графа G называется циклическим, если в графе G существует цикл, содержащий ребро. В противном случае ребро называется нециклическим. На рис. 12 все ребра, за исключением e_{12}, циклические.

Число ребер в пути называется *длиной пути*. Аналогично определяется *длина цикла*.

Необходимо указать следующие свойства путей и циклов.

1. Степень каждой не концевой вершины пути равна 2, концевые вершины имеют степень, равную 1.

2. Каждая вершина цикла имеет степень 2 или другую четную степень. Обращение этого утверждения, а именно то, что ребра подграфа, в котором каждая вершина имеет четную степень, образуют цикл,— неверно.

3. Число вершин в пути на единицу больше числа ребер, тогда как в цикле число ребер равно числу вершин.

2.5 Связность и компоненты графа

Важным понятием в теории графов является *связность*. Две вершины v_i и v_j называются связанными в графе G, если в нем существует путь v_i - v_j. Вершина связана сама с собой.

Граф G называется связным, если в нем существует путь между каждой парой вершин. Например, граф, представленный на рис. 12, связный.

Рассмотрим несвязный граф G = (V, E). Тогда множество вершин V графа G можно разбить на такие подмножества V_1, V_2, … ,V_p, что вершинно-порожденные подграфы $<V_i>$, i= 1, 2, . . p, связны, и никакая вершина подмножества V_i не связана ни с какой вершиной подмножества V_j, $j \neq i$. Подграфы $<V_i>$, i=l, 2, ..., p, называются компонентами графа G. Легко видеть, что компонентой графа G является максимально связный подграф графа G, т. е. компонента графа G не является собственным подграфом любого другого связного подграфа графа G.

Например, граф G на рис. 13 не связен. Его четыре компоненты G_1, G_2, G_3, G_4 имеют множества вершин $\{v_1, v_2, v_3\}$, $\{v_4, v_5\}$, $\{v_6, v_7, v_8\}$, $\{v_9\}$ соответственно.

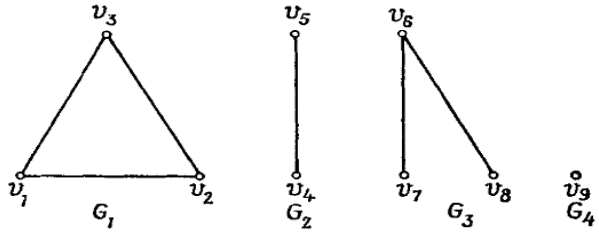


Рисунок 13

Отметим, что изолированную вершину также следует рассматривать как компоненту, поскольку по определению вершина связана сама с собой. Кроме того, следует отметить, что если граф G связен, то он имеет только одну компоненту, которая является графом G.

Теперь рассмотрим некоторые свойства связных графов.

Теорема. В связном графе любые два пути максимальной длины имеютобщую вершину.

Теорема. Если граф G= (V, E) связен, то граф G'= (V, E - e), получающийся после удаления циклического ребра e, тоже связен.

2.6 Операции над графами

Введем несколько операций над графами. Первые три операции, включающие два графа, *бинарные*, а остальные четыре - *унарные*, т. е. определены на одном графе. Рассмотрим графы $G_1=(V_1, E_1)$ и $G_2=(V_2, E_2)$.

Объединение графов G_1 и G_2, обозначаемое как $G_1 \cup G_2$, представляет собой такой граф $G_3= (V_1 \cup V_2, E_1 \cup E_2)$, что множество его вершин является объединением V_1 и V_2, а множество ребер - объединением E_1 и E_2. Например, графы G_1 и G_2 и их объединение представлены на рис. 14, а, б и 15,в.

Пересечение графов G_1 и G_2, обозначаемое как $G_1 \cap G_2$, представляет собой граф $G_3 = \{V_1 \cap V_2, E_1 \cap E_2)$. Таким образом, множество вершин G_3 состоит только из вершин, присутствующих одновременно в графах G_1 и G_2, а множество ребер G_3 состоит только из ребер, присутствующих одновременно в G_1 и G_2. Пересечение графов G_1 и G_2 (рис. 14, а, б) показано на рис. 15, г. Кольцевая сумма двух графов G_1 и G_2, обозначаемая как $G_1 \oplus$

G_2, представляет собой граф G_3, порожденный на множестве ребер $E_1 \oplus E_2$. Другими словами, граф G_3 не имеет изолированных вершин и состоит только из ребер, присутствующих либо в G_1, либо в G_2, но не в обоих графах одновременно. Кольцевая сумма графов (рис. 14, а, б) показана на рис. 16, д.

Легко убедиться в том, что три рассмотренные *операции коммутативны*, т.е. $G_1 \cup G_2 = G_2 \cup G_1$, $G_1 \cap G_2 = G_2 \cap G_1$, $G_1 \oplus G_2 = G_2 \oplus G_1$.

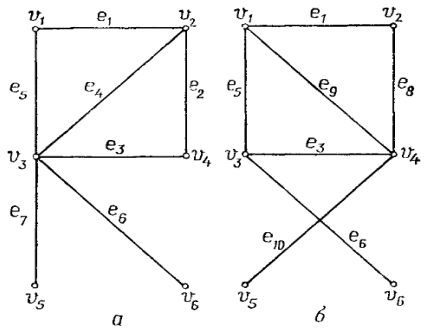


Рисунок 14 (а,б)

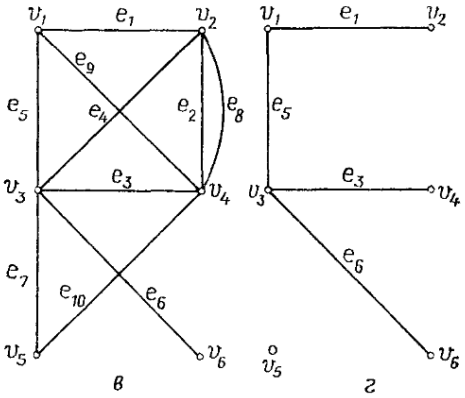


Рисунок 15 (в,г)

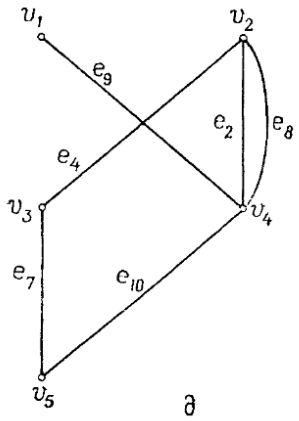


Рисунок 16 д

Заметим также, что эти операции бинарные, т. е. определены по отношению к двум графам. Очевидно, определение этих операций можно расширить на большее число графов.

Теперь рассмотрим *унарные операции* на графе.

Удаление вершины. Если v_i - вершина графа G = (V, E), то G - v_i - порожденный подграф графа G на множестве вершин V - v_i, т. е. G - v_i является графом, получившимся после удаления из графа G вершины v_i и всех ребер, инцидентных этой вершине.

Удаление ребра. Если e_i - ребро графа G = (V, E), то G - e_i - подграф графа G, получающийся после удаления из G ребра e_i. Заметим, что концевые вершины ребра e_i не удаляются из G. Удаление из графа множества вершин или ребер определяется как последовательное удаление отдельных вершин или ребер.

Если G_1 = (V', E') - подграф графа G = (V, E), то через G - G_1 будем обозначать граф G' = (V, E - E'). Таким образом, G - G_1 - дополнение подграфа G_1 в G. Удаление вершины показано на рис. 17 (а – исходный граф, б – вершина удалена).

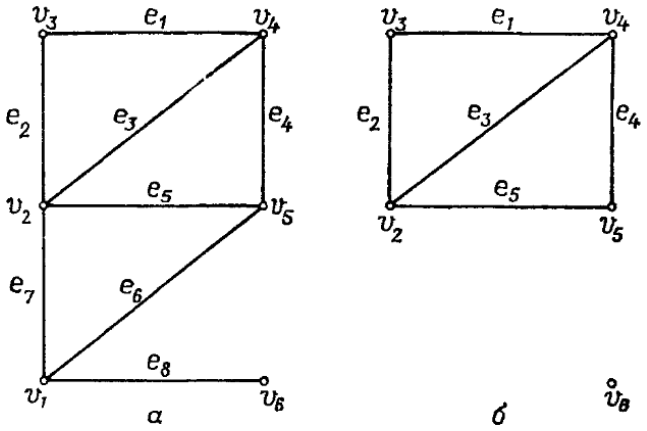


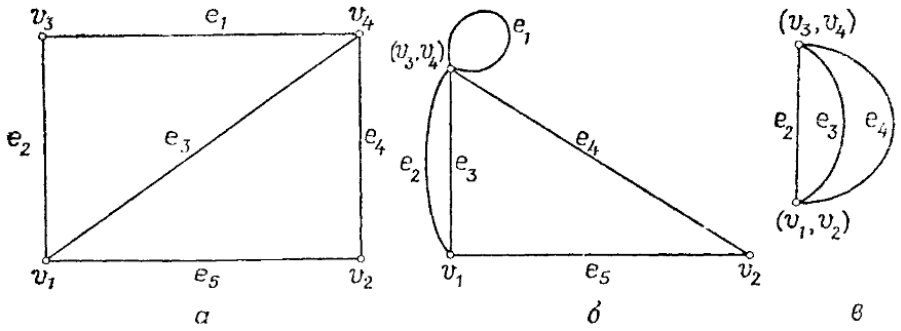
Рисунок 17

Замыкание или отождествление. Говорят, что пара вершин v_i и v_j в графе G замыкается (или отождествляется), если они заменяются такой новой вершиной, что все ребра в графе G, инцидентные v_i и v_j, становятся инцидентными новой вершине.

Например, результат замыкания вершин v_3 и v_4 в графе рис. 18, а представлен на рис. 18, б.

Стягивание. Под стягиванием мы подразумеваем операцию удаления ребра e и отождествление его концевых вершин. Граф G является стягиваемым графом к графу H, если H можно получить из G последовательностью стягиваний.

Граф, изображенный на рис. 18, в, получен стягиванием ребер e_1 и e_5 в графе G (рис. 18, а).



а б в

Рисунок 18

2.7 Матрица смежности и инцидентности

Пусть G = (V, E) — ориентированный граф без параллельных дуг, в котором $V=\{v_1,v_2,...,v_n\}$. *Матрицей смежности* $M=[m_{ij}]$ графа G называется матрица порядка n×n, элементы которой m_{ij} определяются следующим образом:

$$mij = \begin{cases} 1, \text{если } (vi, vj) \in E \\ 0 \text{ в противном случае} \end{cases}$$

Например, граф, изображенный на рис. 19, имеет следующую матрицу смежности:

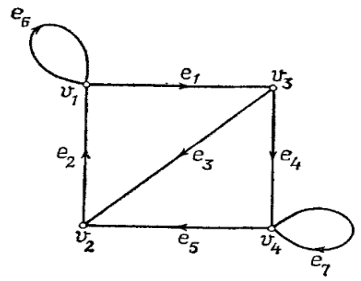


Рисунок 19

$v_1 \quad v_2 \quad v_3 \quad v_4$

$$M = \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

В случае неориентированного графа $m_{ij}=1$ тогда и только тогда, когда существует ребро, соединяющее вершины v_i и v_j. Перейдем к изучению результатов, связанных с матрицей смежности.

Матрица инциденций. Рассмотрим граф G без петель на n вершинах и m ребрах. *Матрица инциденций* $A_c = [a_{ij}]$ графа G имеет n строк (по одной на каждую вершину) и m столбцов (по одному на каждую дугу). Элемент a_{ij} матрицы A_c определяется следующим образом:

Если граф G ориентированный $a_{ij}=$ $\begin{cases} 1, \text{если i - я дуга инцидентна j - й вершине и исходит из нее} \\ -1, \text{если i - я дуга инцидентна j - й вершине и заходит в нее} \\ 0, \text{если i - я дуга не инцидентна j - й} \end{cases}$

Если граф G ориентированный $a_{ij}=$ $\begin{cases} 1, \text{если j - е ребро не инцидентно i - й вершине} \\ 0, \text{в противном случае} \end{cases}$

Строки матрицы A_c называют векторами инциденций графа G. На рис. 20, а и б представлены два графа со своими матрицами инциденций.

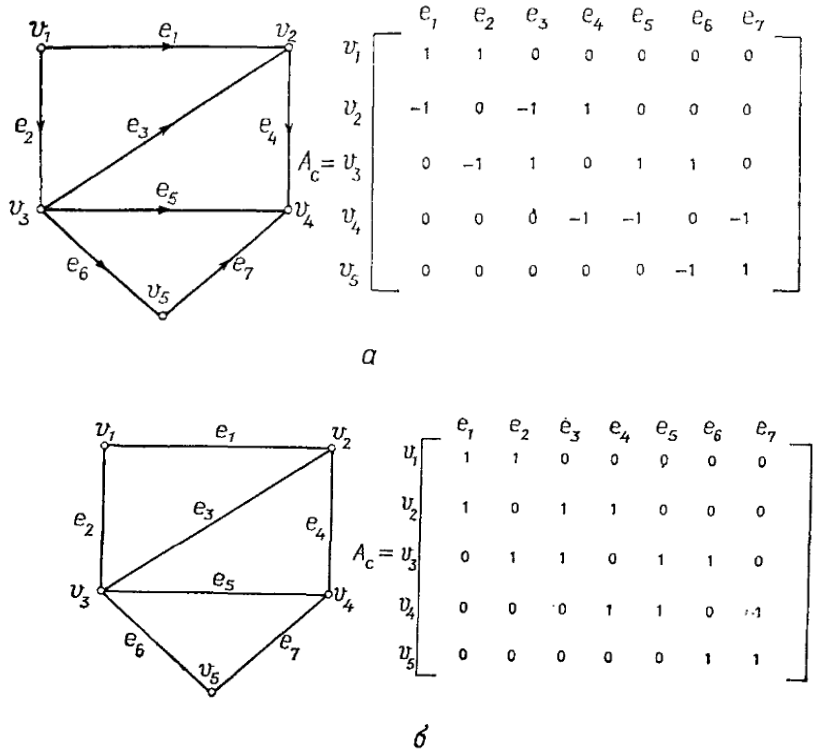


Рисунок 20

Из определения очевидно, что всякий столбец матрицы A_c содержит точно два ненулевых элемента: +1 и -1. Поэтому любую строку этой матрицы можно определить по остальным n-1 строкам. Таким образом, произвольные n-1 строк матрицы A_c содержат всю информацию об этой матрице. Другими словами, строки матрицы A_c линейно - зависимы.

Подматрицу А матрицы Ас на n-1 строке называют *усеченной* матрицей инциденций. Если вершина соответствует строке матрицы Ас, отсутствующей в подматрице А, то говорят, что А — матрица инциденций, усеченная по строке— соответствует данной вершине.

Орграфы

3.1 Определения и примеры

Орграфом D называется пара (V(D), A(D)), где V(D) - непустое конечное множество элементов, называемых вершинами, и A(D) - конечное семейство упорядоченных пар элементов из V(D), называемых дугами; V(D) и A(D) называются соответственно множеством вершин и семейством дуг

орграфа D. Так, на рис. 21 представлен орграф, дугами которого являются (u, v), (v, v), (v, w), (v, w), (w, v), (w, u) и (z, w); порядок вершин на дуге указан стрелкой.

Граф, полученный из орграфа D «удалением стрелок» (т.е. заменой каждой дуги вида (v, w) на соответствующее ребро {v,w}), называется *основанием орграфа* D (рис. 22).

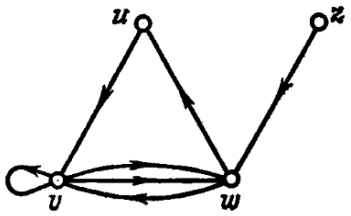


Рисунок 21

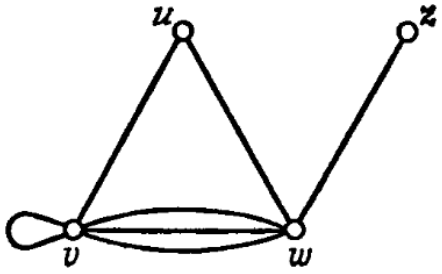


Рисунок 22

Две *вершины* v и w орграфа D называются *смежными*, если в A(D) существует дуга вида (v, w) или (w, v); при этом вершины v и w называются *инцидентными* любой такой дуге (а дуга — инцидентной соответствующим вершинам).

Два орграфа называются *изоморфными*, если существует изоморфизм между их основаниями, сохраняющий порядок вершин на каждой дуге. *Матрицей смежности орграфа* G множеством вершин $\{v_1,\ldots, v_n\}$ является матрица $A = (a_{ij})$, в которой a_{ij} равно числу дуг вида (v_i, v_j) в семействе A(D). Матрица, показанная на рис.23, является матрицей смежности для орграфа, изображенного на рис. 21. Простой орграф определяется очевидным образом.

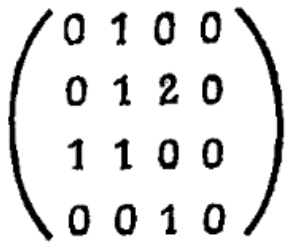


Рисунок 23

Ориентированный маршрут в орграфе D представляет собой конечную последовательность дуг вида (v_0, v_j, (v_l, v_2), v_m). Можно записывать эту последовательность в виде v_0 ->v_1 ->... ->v_m и говорить об ориентированном маршруте из v_0 в v_m. Аналогичным образом можно определить *ориентированные цепи*, ориентированные простые цепи и *ориентированные циклы*, или *орцепи*, *простые орцепи* и *орциклы*.

Заметим, что хотя *орцепь* не может содержать данную дугу (v, w) более одного раза, она может содержать одновременно (v, w) и (w, v); например, на рис. 21 z ->w ->v ->w ->u является *орцепью*. Теперь мы в состоянии определить связность. Точнее, мы определим здесь два наиболее естественных и полезных типа связности орграфов, которые возникают в соответствии с тем, хотим мы или нет принимать во внимание ориентацию дуг.

Говорят, что орграф D *связен* (или *слабо связен*), если он не может быть представлен в виде объединения двух различных орграфов (определенного обычным образом); это эквивалентно тому, что связно основание орграфа D. Предположим дополнительно, что для любых двух вершин v и w орграфа D существует простая орцепь из v в w; тогда D называется сильно связным (этот термин настолько устоялся, что мы использовали его вместо более естественного «орсвязный»). Ясно, что любой сильно связный граф связен, но обратное неверно: на рис. 21 изображен связный орграф, не являющийся сильно связным.

Различие между *связным* и *сильно связным* орграфом станет яснее, если мы рассмотрим план города, по всем улицам которого допускается только одностороннее движение. Тогда связность соответствующего орграфа означает, что мы можем проехать из любой части города в любую другую, не обращая внимания на правила одностороннего движения; если же этот орграф сильно связен, то мы можем проехать из любой части города в любую другую, следуя всегда «правильным путем» вдоль улиц с односторонним движением. Важно, чтобы система с односторонним движением была сильно связной, и естественно возникает вопрос: при каких условиях карту улиц можно превратить в систему с односторонним движением таким способом, чтобы можно было проехать из любой части города в любую другую? Если, к примеру, город состоит из двух частей, связанных одним мостом, то мы никогда не сможем сделать все его улицы односторонними, поскольку какое бы направление мы ни приписали мосту, одна часть города будет отрезана. (Заметим, что сюда включается и тот случай, когда в городе имеется тупик.) С другой стороны, если мостов нет, то всегда найдется подходящая односторонняя система.

Для удобства будем называть граф G ориентируемым, если каждое его ребро (рассматриваемое как пара вершин) может быть упорядочено таким образом, что полученный в результате орграф будет сильно связным. Этот процесс упорядочения ребер будем называть «заданием ориентации графа» или «приписыванием направлений ребрам». Если, например, G - граф, изображенный на рис. 24, то его можно ориентировать и получить сильно связный орграф, изображенный на рис. 25.

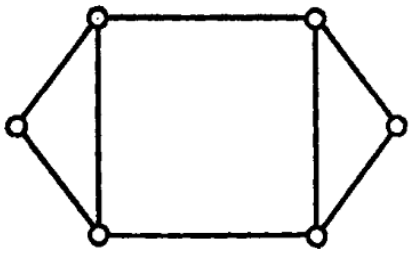


Рисунок 24

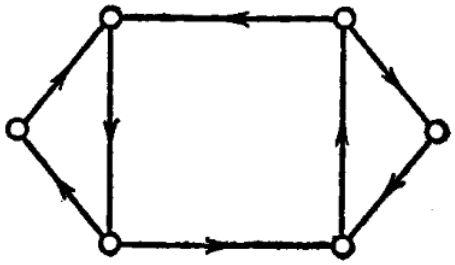


Рисунок 25

Теорема. Пусть G — связный граф; он ориентируем тогда и только тогда, когда каждое его ребро содержится по крайней мере в одном цикле.

3.2 Орграфы и матрицы

Матрицей смежностей A(D) орграфа D называется (p×p)-матрица $||a_{ij}||$, у которой a_{ij}= 1, если V_iV_j- дуга орграфа D, и a_{ij}=0 в противном случае. Матрица смежностей которого имеет вид (рис. 26):

	v_1	v_2	v_3	v_4	v_5	Сумма по строке
v_1	0	0	0	0	0	0
v_2	1	0	1	1	0	3
$A(D)=v_3$	1	0	0	0	0	1
v_4	0	0	1	0	0	1
v_5	0	0	0	0	0	0
Сумма по столбцу	2	0	2	1	0	

Рисунок 26

Легко проверить, что суммы элементов по строкам матрицы A(D) равны полустепеням исхода вершин орграфа D, а суммы элементов по столбцам - полустепеням захода.

Как и в случае графов, степени матрицы смежностей A орграфа дают полную информацию о числе маршрутов, идущих из одной вершины в другую. Теорема. (i, j)-й элемент a_{ij}^n матрицы A" равен числу маршрутов длины n, идущих из вершины v_i в вершину v_j.

Упомянем здесь вкратце еще о трех матрицах, связанных с орграфом D_s - о *матрице достижимостей*, *матрице расстояний* и *матрице обходов*. В матрице достижимостей R элемент r_{ij} равен 1,если вершина v_i достижима из

v_j и равен 0 в противном случае. В матрице расстояний (i, j)-й элемент равен расстоянию из вершины v_i в вершину v_j; если же из v_i в v_j нет путей, то соответствующий элемент полагаем равным бесконечности. В матрице обходов (i, j)-й элемент равен длине наиболее длинного пути из v_i в v_j, а если таких путей нет, то опять-таки полагаем этот элемент равным бесконечности. Для орграфа D, показанного на рис. 27.

Матрица достижимостей

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

Матрица расстояний

$$\begin{Vmatrix} 0 & \infty & \infty & \infty & \infty \\ 1 & 0 & 1 & 1 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ 2 & \infty & 1 & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{Vmatrix}$$

Матрица обходов

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

Рисунок 27

Следствие. Элементы матриц достижимостей и расстояний связаны со степенями матрицы A следующими соотношениями:

1) $r_{ii} = 1$ и $d_{ii} = 0$ для всех i;

2) $r_{ij} = 1$ тогда и только тогда, когда $a_{ij}^n > 0$ для некоторого n;

3) $d(v_i, v_j)$ равно наименьшему из чисел n, для которых $a_{ijn} > 0$

Эффективных методов для нахождения элементов матрицы обходов не существует. Эта проблема тесно связана с некоторыми другими давно поставленными алгоритмическими проблемами теории графов, такими, как нахождение остовных циклов и контуров, а также решение задачи о коммивояжере.

Поэлементное произведение B×C матриц $B=\|b_{ij}\|$ и $C=\|c_{ij}\|$ имеет своим (i, j)-м элементом $b_{ij}c_{ij}$. Матрицу достижимостей орграфа можно использовать для нахождения его сильных компонент.

3.3 Ориентированные эйлеровы графы

Ориентированной эйлеровой цепью ориентированного графа G называется замкнутая ориентированная цепь, содержащая все дуги G.

Открытой ориентированной эйлеровой цепью называется открытая ориентированная цепь, содержащая все дуги графа G.

Ориентированный граф, обладающий ориентированной эйлеровой цепью, называется *ориентированным эйлеровым графом*(рис. 28).

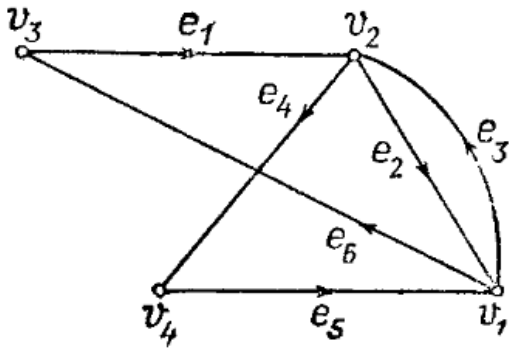


Рисунок 28

Ориентированным эйлеровым графом является граф, изображенный на рис. 28, поскольку дуги e_1 e_2, e_3, e_4, e_5, e_6 образуют в графе G ориентированную эйлерову цепь.

Теорема. Для связного ориентированного графа G следующие утверждения равносильны:

1) G - ориентированный эйлеров граф;

2) для любой вершины v графа G справедливо равенство d - (v) = d+(v);

3) G - объединение нескольких реберно-непересекающихся контуров.

Рассмотрим, например, ориентированный эйлеров граф G на рис. 28. Легко проверить, что он обладает свойством, сформулированным в п. 2 теоремы, и является также объединением реберно-непересекающихся контуров $\{e_2, e_3)$ и $\{e_1, e_4, e_5, e_6\}$.

Легко доказать и следующую теорему:

Теорема. Связный ориентированный граф содержит открытую ориентированную эйлерову цепь тогда и только тогда, когда выполняются условия:

1) в графе G имеются такие две вершины v1 и v2, что d+ (v1)=d- (v1)+1 и d- (v2) = d+(v2)+1;

2) для любой вершины v, отличной от v1 и v2, справедливо равенство d- (v) - d+ (v).

Например, условиям этой теоремы удовлетворяет граф на рис. 29. Открытой ориентированной эйлеровой цепью графа G является последовательность e_1, e_2, e_3, e_4, e_5, e_6.

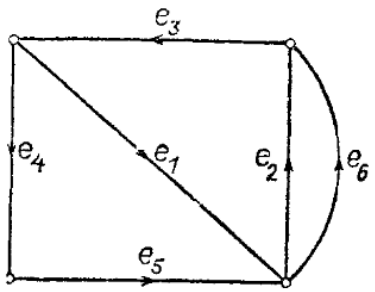


Рисунок 29

Эйлеров контур в орграфе D — это замкнутый остовный маршрут, в котором каждая дуга орграфа D встречается по одному разу. Орграф называется эйлеровым, если в нем есть эйлеров контур.

Для орграфа, показанного на рис. 30; в этом орграфе 14 эйлеровых контуров. Два из них такие: v_1, v_2, v_3, v_4, v_2, v_1, v_3, v_1, v_4, v_1 и v_1, v_2, v_1, v_4, v_2, v_3, v_4, v_1, v_3, v_1.

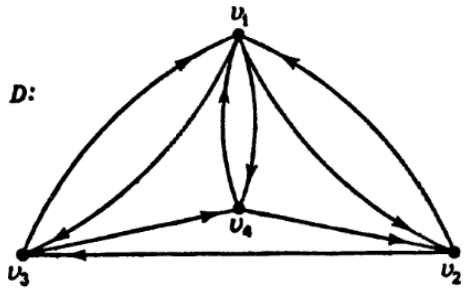


Рисунок 30

4.1 Ориентированные ациклические графы

В этом разделе изучаются свойства важного класса ориентированных графов, а именно *ациклических*. Как мы знаем, ориентированный граф —

ациклический, если он не содержит контуров. Очевидно, простейшим примером ациклического ориентированного графа является ориентированное дерево.

Основной результат, который мы получим в этом разделе, заключается в том, что вершины ациклического ориентированного графа G на n вершинах можно пометить таким образом целыми числами из множества {1, 2, ... , n}, что если в графе G имеется дуга (i, j), то i <j.

Определенное нами упорядочение вершин называется топологической сортировкой. Топологически отсортированы, например, вершины ациклического ориентированного графа на рисунке 31.

Справедливость основного результата этого раздела определяется следующей теоремой:

Теорема 1. В ациклическом ориентированном графе имеются по крайней мере одна вершина с нулевой полустепенью захода и одна вершина с нулевой полустепенью исхода.

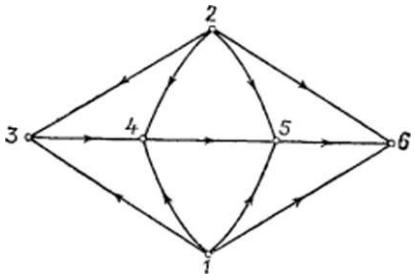


Рисунок 31

4.2 Деревья

Важным частным случаем ориентированнных ациклических графов являются деревья. Класс деревьев занимает в теории графов особое положение. С одной стороны, это достаточно просто устроенные графы, и многие задачи, весьма сложные в общей ситуации, для деревьев решаются легко. С другой стороны, деревья часто встречаются в областях, на первый взгляд не имеющих отношения к теории графов.

Деревом называется связный граф, не содержащий циклов. Любой граф без циклов называется ациклическим (или лесом). Таким образом, компонентами леса являются деревья. На рисунке 32 изображены все деревья шестого порядка.

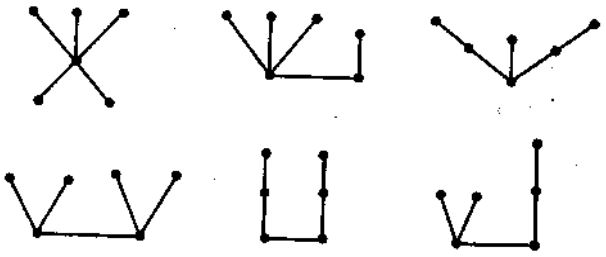


Рисунок 32

В следующей теореме перечислены некоторые простые свойства деревьев.

Теорема 2. Пусть граф T имеет n вершин. Тогда следующие утверждения эквивалентны: (i) T является деревом; (ii) T не содержит циклов и имеет n — I ребер; (iii) T связен и имеет n — 1 ребер; (iv) T связен и каждое его ребро является мостом; (v) любые две вершины графа T соединены ровно одной простой цепью; (vi) T не содержит циклов, но, добавляя к нему любое новое ребро, мы получаем ровно один цикл.

Следствие. Пусть G — лес с n вершинами и k компонентами; тогда G имеет n — k ребер.

Известно, что в связном графе G удаление одного ребра, принадлежащего некоторому выбранному циклу, не нарушает связности оставшегося графа. Применим эту процедуру к одному из оставшихся циклов, и так до тех пор, пока не останется ни одного цикла. В результате получим дерево, связывающее все вершины графа G; оно называется *остовным деревом* (или остовом, каркасом) графа G. Пример графа и одного из его остовных деревьев дан на рисунке 33.

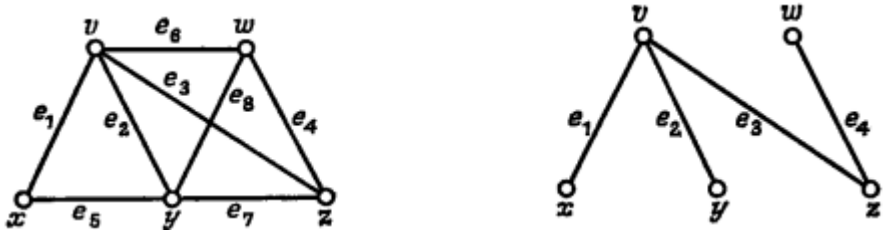


Рисунок 33

1.1 Классификация сетевых моделей и элементы сетевых графиков.

Модель - любой упрощенный образ какого-либо объекта, должен удовлетворять требованиям:

- Отражать существующие связи.
- Быть наглядным.
- Изложен понятным языком.

Моделирование - процесс исследование на модели, представляет собой модель исследования. В качестве графической модели строительного производства служит:

- Линейные графики.
- Циклограммы.
- Табличные матрицы.
- Сетевые графики.

Линейные графики удается отобразить 1 значение, взаимосвязь и последующие работы, но при сложных зависимостях между работами графики мало эффективны.

Циклограммы наглядно изображают развитие строительно процесса во времени и пространстве. Наиболее эффективно при возведении однотипных зданий и сооружений. За единицу продукции принимаем участок/захватку. При возведении крупного строительного комплекса промышленности, которого отличает сложная связь работ наглядность циклограмм существенно снижается и пользоваться на практике не удобно.

При использовании матричной модели легко определить продолжительность работ, общую продолжительность строительства и простой фронтов/бригад и уровень совмещения работ.

Первым шагом в анализе любого проекта является составление списка входящих в него операций. Детали такого списка зависят от специфики конкретного проекта. Тем не менее, во всех случаях необходимо выделить непосредственно предшествующую операцию или операции. Непосредственно предшествующими называются операции, выполнение которых должно быть закончено прежде, чем может начаться данная операция. Например, при постройке дома крыша не может быть построена до того момента, пока не закончится возведение стен.

После того как составлен список, логическая последовательность выполнения операций может быть проиллюстрирована с помощью графа. Существуют различные типы графов, но наиболее широкое применение получили так называемые **вершинные и стрелочные графы.** Однако каждый из них имеет свои преимущества и недостатки, и выбор того или иного графа является вопросом личных предпочтений или же определяется целью создания и использования данного графа.

Сетевой граф позволяет лучше всего отобразить порядок и осуществление, обоснование принятых методов строительства. Сетевой график – документ позволяющий оперативно руководить строительством и перераспределять ресурсы в зависимости от фактического строительства. Обычно их применяют при строительстве сложных промышленных зданий, при большом количестве строительных организаций. Сетевой моделью называется ориентировочный график отражающий последовательность и организационно-технологические взаимосвязи, выполнение которых необходимо для достижения цели. Сетевая модель с рассчитанными временными и ресурсами параметрам называется сетевым графиком.

Сетевой график используют для решения задач перспективного

планирования. Определения продолжительности и сроков выполнения основных этапов и планирования капитальных вложений по периодам строительства объекта.

Классификация сетевых графиков:

По виду целей: Целевые и многоцелевые

По числу охвата объекта: частный и комплексный.

По характеру оценок параметров модели: детерминированные (заранее и полностью обусловленные данные), вероятностные (учитывая влияние случайных факторов).

Модель с целевым направлением временным, ресурсным, стоимостным.

1.2 Стрелочные графы

Всякий намеченный комплекс работ, необходимых для достижения некоторой цели, называют проектом. Проект (или комплекс работ) подразделяется на отдельные работы. Каждая отдельная работа, входящая в комплекс (проект), требует затрат времени. Некоторые работы могут выполняться только в определенном порядке. При выполнении комплекса работ всегда можно выделить ряд событий, то есть итогов какой-то деятельности, позволяющих приступить к выполнению следующих работ. Если каждому событию поставить в соответствие вершину графа, а каждой работе — ориентированное ребро, то получится некоторый граф. Он будет отражать последовательность выполнения отдельных работ и наступление событий в едином комплексе. Если над ребрами проставить время, необходимое для завершения соответствующей работы, то получится сеть. Изображение такой сети называют сетевым графиком.Сетевой график состоит из двух типов основных элементов: работ и событий. Работа представляет собой выполнение некоторого мероприятия

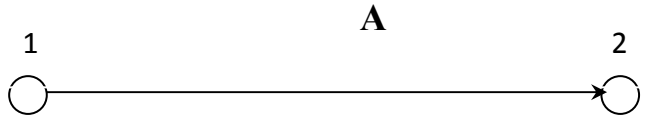
(например, погрузка боезапаса или переход корабля в пункт базирования). Этот элемент сетевого графика связан с затратой времен и расходом ресурсов. Поэтому работа всегда имеет начало и конец. Кроме того, каждая работа должна иметьопределение, раскрывающее ее содержание (например, уяснение боевой задачи, приготовление корабля к походу и т.д.).

На сетевом графике работа изображается стрелкой, над которой проставляется ее продолжительность или затрачиваемые ресурсы, или то и другое одновременно. Работа, отражающая только зависимость одного мероприятия от другого, называется фиктивной работой. Такая работа имеет нулевую продолжительность (или нулевой расход ресурсов) и обозначается пунктирной стрелкой.

Начальная и конечная точки работы, то есть начало и окончание некоторого мероприятия (например, окончание приготовления корабля к бою), называются событиями. Следовательно, событие, в отличие от работы, не является процессом и не сопровождается никакими затратами времени или ресурсов.

Событие, следующее непосредственно за данной работой, называется последующим событием по отношению к рассматриваемой работе. Событие, непосредственно предшествующее рассматриваемой работе, называется предшествующим.

Наименования "предшествующий" и "последующий" относятся также и к работам. Каждая входящая в данное событие работа считается предшествующей каждой выходящей работе, и наоборот, каждая выходящая работа считается последующей для каждой входящей.



Предшествующее событие (начало) *Операции* *Последующее событие (окончание)*

Рис. 1. Изображение операции на графе

Из определения отношения "предшествующий—последующий" вытекают свойства сетевого графика.

Во-первых, ни одно событие не может произойти до тех пор, пока не будут закончены все входящие в него работы. Во-вторых, ни одна работа, выходящая из данного события, не может начаться до тех пор, пока не произойдет данное событие. И, наконец, ни одна последующая работа не может начаться раньше, чем будут закончены все предшествующие ей.

Событие обозначается кружком с цифрой внутри, определяющей его номер.

Из всех событий, входящих в планируемый процесс, можно выделить два специфических — событие начала процесса, получившее название исходного события, которому присваивается нулевой номер, и событие конца процесса (завершающее событие), которому присваивается последний номер. Остальные события нумеруются так, чтобы номер предыдущего события был меньше номера последующего.

Операции обозначают буквой или словом, а события - числом. Поскольку любая операция характеризуется парой событий, ее можно также обозначать с помощью чисел, соответствующих этим событиям. Например, на рис. 1 операция А означает то же самое, что и операция (1, 2). Одному узлу может соответствовать (входить или выходить из него) несколько операций. Событие, изображаемое на графе с помощью узла, не считается свершившимся до тех пор, пока не окончены все входящие в него операции. Операция, выходящая из некоторого узла, не может начаться до тех пор, пока не будет достигнуто **начальное событие,** т.е. пока не будут завершены все операции, входящие в **узловое начальное событие.**

Если операция С не может быть начата до момента окончания работ А и В, логическую схему данной ситуации можно представить графически следующим образом (см. рис. 2).

Начальным событием для С является конечное событие для А и В. Существенно, что в стрелочном графе сохраняется логическая зависимость операций. Иногда, чтобы достичь этого, необходимо включить в граф одну или более **фиктивных логических операций.**

Фиктивная логическая стрелка вводится в граф, если необходимо отразить, что некоторое событие не может появиться раньше другого события, а с помощью обычных стрелок, соответствующих операциям, этого сделать нельзя. Функция фиктивной логической операции состоит в том, чтобы показать последовательность появления событий.

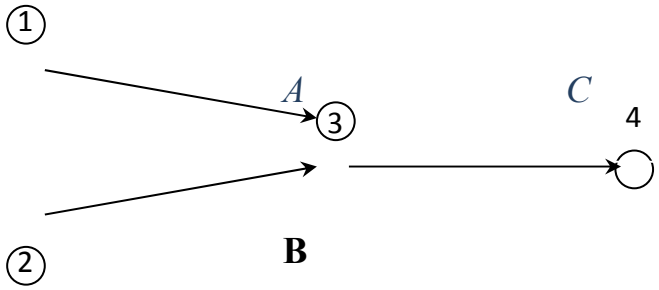


Рис. 2. Логические взаимосвязи в стрелочном графе

Фиктивным логическим операциям ставится в соответствие нулевая продолжительность выполнения, а изображаются они обычно пунктиром. Например, если работу С нельзя начать прежде, чем завершится операция А, а работу О нельзя начать до тех пор, пока не завершатся работы А и В, соответствующий стрелочный граф будет выглядеть следующим образом:

Кроме того, в стрелочных графах для избежания неоднозначности используются **фиктивные операции идентификации.** В некоторых пакетах прикладных программ, используемых в сетевом анализе, операции обозначаются не с помощью букв или слов, а числами, обозначающими соответствующие им события. Если же две или более операций выполняются

одновременно и имеют одни и те же начальное и конечное события, то компьютер не сможет отличить их друг от друга и не воспримет вводимую исходную информацию. Как показано на рис. 4, включение фиктивной операции идентификации позволяет решить данную проблему. На практике принято нумеровать события таким образом, чтобы номер конечного события был больше, чем номер начального события.

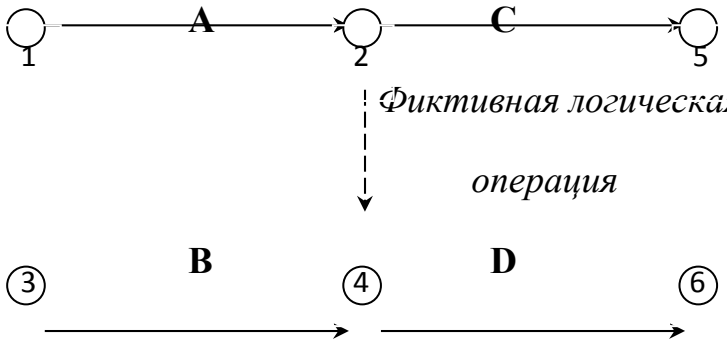
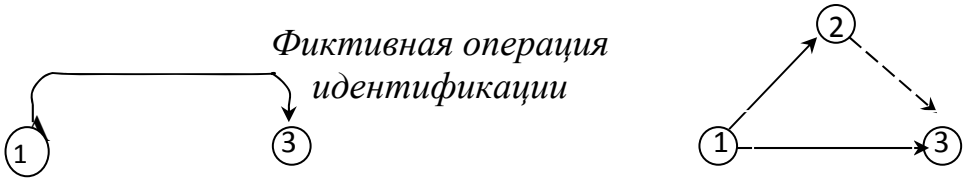


Рис. 3. Использование в стрелочном графе фиктивной логической операции

Первый шаг после составления списка операций, входящих в проект, состоит в том, чтобы создать таблицу операций, в которой отражаются все операции, а также операции, непосредственно им предшествующие.

В данный список не включаются фиктивные логические операции или операции идентификации. На основе полученного списка строится стрелочный сетевой граф, включающий действительные и фиктивные операции и отражающий установленные взаимосвязи между ними. После того, как закончено построение исходного графа, можно выявить и исключить из рассмотрения ненужные фиктивные операции. Затем для улучшения логической схемы исходный граф можно модифицировать и перекомпоновать.



Заменяется на

Рис. 4. Использование в стрелочном графе фиктивной операции идентификации

Ненужные фиктивные логические операции можно выявить с помощью простого практического правила. Если единственной операцией, выходящей из некоторого узла, является фиктивная логическая операция, то по всей вероятности без нее можно обойтись.

Пример 1. Компания "Эвриком" - это промышленная фирма, которая заключила контракт о производстве партии станков, предназначенных к использованию крупным предприятием обувной промышленности для массового производства обуви. Ниже перечислены операции, которые необходимо выполнить в процессе разработки и производства этих станков (табл. 1).

Нужно изобразить операции с помощью стрелочного графа.

Решение.

Сетевой граф должен начинаться с единственного начального события, которое показано на рис. 5 кружочком, и заканчиваться единственным конечным событием. Построение графа мы начали с первого события. С этого события начинаются все операции, которым не предшествуют никакие виды работ. Начинать построение полезно с примерного эскиза будущего графа:

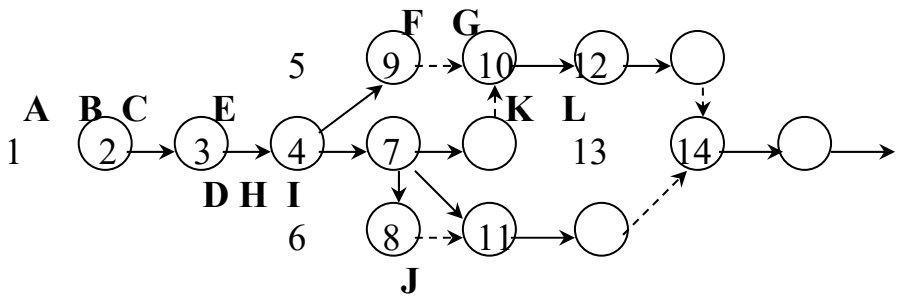


Рис. 5. Примерный эскиз графа для примера 1

Таблица 1. Таблица операций для задачи из примера 1

ОПЕРАЦИИ	Непосредственно предшествующая операция
A Составление сметы затрат	-
B Согласованные оценки	A
C Покупка собственного оборудования	B
D Подготовка конструкторских проектов	B
E Строительство основного цеха	D
F Монтаж оборудования	C,E
G Испытания оборудования	F
H Определение типа модели	D
I Проектирование внешнего корпуса	D
J Создание внешнего корпуса	H,I
K Конечная сборка	G,J
L Контрольная проверка	K

В соответствии с приведенной выше таблицей необходимо тщательно, переходя от одной операции к другой, проверить построенный в первом приближении граф.

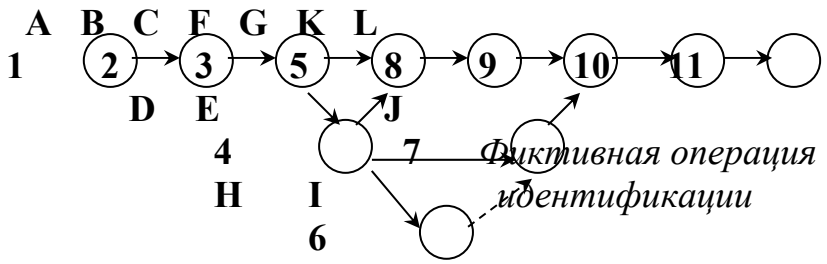


Рис.6. Новый чертеж стрелочного графа для примера 1

Пример 2. Компания "Эвриком" является участником другого проекта, детали которого приведены ниже. Изобразим данный проект при помощи стрелочного графа.

Решение

Построение начинаем с начального события, обозначенного кружком 1. Из таблицы следует, что существуют три операции - A, B и C, которым не предшествует ни одна из операций. Поэтому из начального события выходят

три стрелки. На первый взгляд таблица операций выглядит чрезвычайно простой, однако отразить присущую ей логику с помощью сетевого графа достаточно трудно, вследствие чего мы вынуждены использовать три фиктивные логические операции (см. рис. 7).

Таблица 2. Таблица операций для примера 2

Операция	Непосредственно Предшествующая операция	Операция	Непосредственно предшествующая операция
A	-	E	B,C
B	-	F	C
C	-	G	D,E
D	A,B	H	F,G

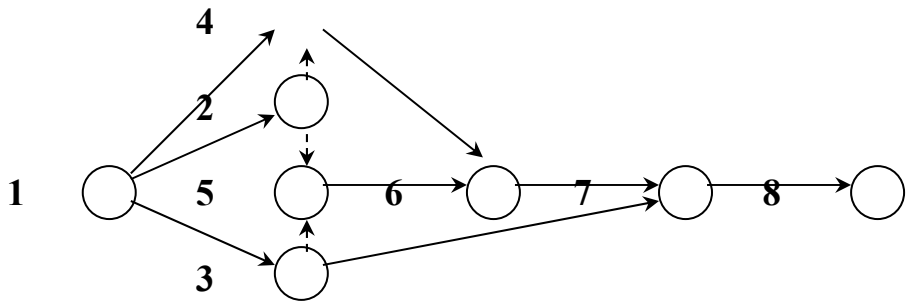


Рис. 7. Стрелочный граф для примера 2

1.2 Вершинные графы

В этом типе сетевых графов операции представлены узлами графа, а стрелками изображаются их взаимосвязи. В таких графах не возникает необходимости вводить фиктивные операции. Как и в предыдущем случае, течение времени следует изображать в направлении слева направо.

Пример 3. Обратившись к данным из примера **2,** модифицируем полученную в этом примере схему, поставив в соответствие операциям узлы графа

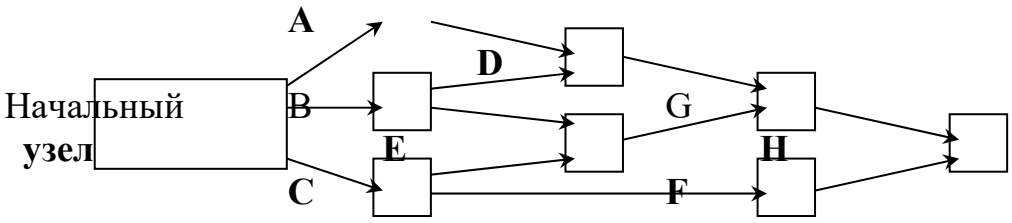


Рис. 8. Вершинный граф

Каждый из описанных типов графов имеет свои преимущества и недостатки. Обычно не имеет принципиального значения, какая из систем используется. Если в стрелочные графы приходится вводить достаточно большое число фиктивных операций, то гораздо более предпочтительным является выбор вершинного графа. Ниже приведено сравнение двух видов изображения операций и их основных особенностей (см. рис. 9).

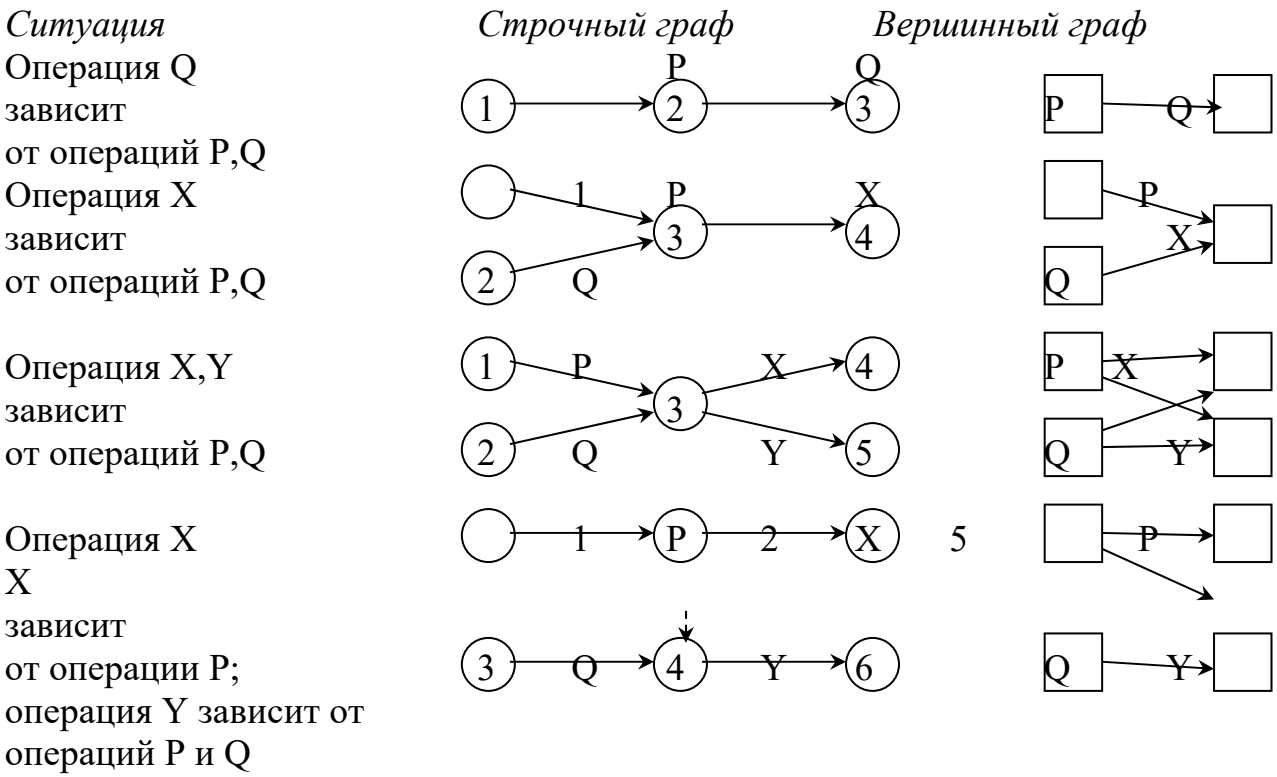


Рис. 9. Сравнение сетевых стрелочного и вершинного графов

1.3 Критический путь

После того как проведена идентификация операций, можно оценить их продолжительность. На основе продолжительности выполнения каждой операции и руководствуясь логической схемой, можно найти время выполнения проекта в целом. На данном этапе предполагается, что продолжительность выполнения каждой операции является фиксированной величиной, не испытывающей влияний неопределенности. В последнем разделе главы мы рассмотрим вопрос о том, какие поправки следует внести в этот анализ, чтобы учесть неопределенность времени выполнения операций. В каждом графе существует несколько возможных путей. Общее время, необходимое для того, чтобы пройти какой-либо путь, есть сумма времени выполнения всех операций, принадлежащих данному пути. Продолжительность выполнения всего проекта занимает наибольшее время. Более длительные операции называются **критическими.** Любая задержка срока начала или окончания выполнения этих работ повлечет за собой задержку срока выполнения проекта в целом. Критические операции образуют непрерывную цепь, проходящую через весь граф. Эта цепь критических операций называется **критическим путем. В** каждом графе найдется, по крайней мере, один критический путь.

Для того чтобы найти общую продолжительность выполнения проекта, нужно определить продолжительность критического пути. В большинстве графов идентифицировать все идущие сквозь граф пути, чтобы выявить среди них тот, который занимает наибольшее время, достаточно трудно. Существуют два возможных метода, позволяющих отследить движение времени в графе:

1. Определение для каждой операции наиболее ранних сроков начала и окончания ее выполнения.

2. Определение для каждого события наиболее раннего срока его наступления. Следует отметить, что второй метод может использоваться только в стрелочных графах.

1.4 Анализ критического пути с применением вершинных графов

Пример 4. В табл. 3 указана продолжительность выполнения каждой операции проекта, о котором шла речь в примерах **2 и 3** Определим общую продолжительность выполнения проекта. Вершинный граф, соответствующий данному проекту, был построен в примере **3.**

Таблица 3. Операции и их продолжительность для примера 4

Операция	Непосредственно Предшествующая Операция	Время, дней
A	-	8
B	-	10
C	-	6
D	A,B	8
T	B,C	9
F	C	14
G	D,E	14
H	F,G	6

Решение

Предположим, что каждая из исходных операций A, B и C начинается в нулевой момент времени. Это наиболее ранний срок начала этих E5 операций. Наиболее ранний срок, к которому их выполнение может быть завершено, определяется следующим образом:

Наиболее ранний срок окончания EP=ES+Продолжительность операции.

Обычно найденные значения этих сроков наносятся непосредственно на граф, однако, мы занесем их сначала в таблицу, чтобы продемонстрировать методику проведения расчетов.

Таблица 4. Расчет наиболее ранних сроков начала окончания операций для примера 4

Операция	Продолжи-тельность, дней	Наиболее ранний срок начала	Наиболее ранний срок окончания	Комментарии
A	8	0	0+8=8	
B	10	0	0+10=10	
C	6	0	0+6=6	
D	8	10	10+8=18	Нельзя начать, пока не завершены A и B
E	9	10	10+9=19	Нельзя начать, пока не завершены B и C
F	14	6	6+14=20	Нельзя начать, пока не завершена C
G	14	19	19+14=33	Нельзя начать, пока не завершены D и E
H	6	33	33+6=39	Нельзя начать, пока не завершены F и G

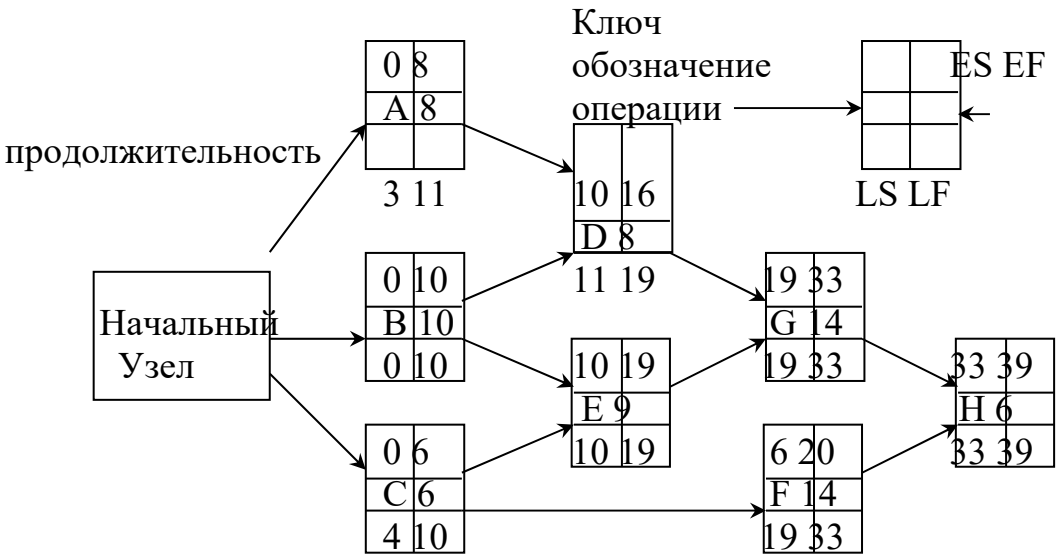


Рис. 10 Вершинный граф для примера 4

Наиболее ранние сроки начала и окончания операций занесены в вершинный граф, изображенный на рис. Нетрудно заметить, что операция H завершится на 39-й день, следовательно, это значение дает нам искомую продолжительность выполнения проекта в целом.

Таблица 5. Расчет наиболее поздних сроков начала и окончания операций для примера 4

Операция	Продолжительность, дней	Наиболее Поздний срок окончания	Наиболее Поздний Срок Начала	Комментарии
H	6	39	39-6=33	
G	14	33	33-14=19	G нужно завершить до наступления наиболее позднего срока начала H
F	14	33	33-14=19	F нужно завершить до наступления наиболее позднего срока начала H
E	9	19	19-9=10	E нужно завершить до наступления наиболее позднего срока начала G
D	8	19	19-8=11	D нужно завершить до наступления наиболее позднего срока начала G
C	6	10	10-6=4	C нужно завершить до наступления наиболее позднего срока начала E и F.
B	10	10	10-10=0	B нужно завершить до наступления наиболее позднего срока начала D и E. Нужно использовать наименьший из этих сроков, равным 10 дням.
A	8	11	11-8=3	A нужно завершить до наступления наиболее позднего срока начала D

На данном этапе мы еще не можем определить критические операции. Чтобы это осуществить, необходимо для каждой операции рассчитать два срока, ей соответствующие, а именно **наиболее поздний срок начала LS и наиболее поздний срок окончания** LF операции. В данном случае процедуру расчетов мы начнем с последней операции в графе и предположим, что наиболее поздний и наиболее ранний сроки ее окончания совпадают. Затем вычитанием из этой величины продолжительности

выполнения операций находим наиболее поздний срок ее начала. Ход выполнения расчетов показан в табл. 5.

Критической является операция, для которой справедливы следующие соотношения:

ES = LS и EF = LF,

т. е. операция, для которой не существует резерва времени между наиболее ранним сроком ее начала и наиболее поздним сроком ее окончания. Нетрудно, заметить, что в нашем примере критическими являются операции B, E, G и H. Путь в вершинном графе, соединяющий эти операции, называется критическим путем. В нашем примере критическим является путь B-E-G-H.

1.5 Анализ критического пути с применением стрелочных графов

Приведенная выше методика анализа аналогичным образом может использоваться. и для стрелочных графов. Значения сроков ES, EF, LS и LF записываются в графе вдоль стрелок, соответствующих операциям:

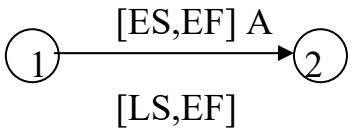


Рис. 11. Нанесение на стрелочный граф сроков, соответствующих операциям

Можно провести подобный анализ в терминах сроков наступления каждой события. Производится расчет наиболее раннего срока, к которому может завершиться каждое событие. Этот срок называется **наиболее ранним сроком события** (earliest event time - EET). Общая продолжительность выполнения проекта определяется EET конечного узла графа. EET исходного события равен нулю.

Для того чтобы выявить критические операции, необходимо, начиная с конца графа, вычислить **наиболее поздние сроки событий** (1atest event time -

LET), к которым события могут закончиться. События, для которых выполняются соотношения

LET начала- EETокончания + продолжительность = *O* или

EETначала - LETокончания + продолжительность = O,

являются критическими.

Пример 5. Применив EET и LET, повторим задачу из примера **4** при условии, что продолжительность выполнения фиктивных операций равна нулю. *Решение*

В первую очередь для каждого события вычислим значение наиболее раннего срока. Если некоторому событию соответствует более одной операции, появляется проблема выбора соответствующего значения. Поскольку событие считается незавершенным до тех пор, пока не будет завершено выполнение всех составляющих его операций, следует выбрать наибольшее из значений.

Таблица 6. Расчет значений EET для примера 5

Узел	EET, дней	Комментарии
1	0	Начальное событие
2	0+10=10	EET узла 1 + продолжительность операции B
3	0+6=6	EET узла 1 + продолжительность операции C
4	0+8=8 или 10+0=10*	EET узла 1 + продолжительность операции A. EET узла 2 + продолжительность фиктивной операции. Выбирается максимальный срок, т. е. 10 дней
5	10+0=10* или 6+0=6	EET узла 2 + продолжительность фиктивной операции. EET узла 3 + продолжительность фиктивной операции. Выбирается максимальный срок, т. е. 10 дней
6	10+8=18 или 10+9=19*	EET узла 4 + продолжительность операции D EET узла 5 + продолжительность операции E. Выбирается максимальный срок, т. е. 19 дней
7	19+14=33* или 6+14=20	EET узла б + продолжительность операции C EET узла 3 + продолжительность операции P. Выбирается максимальный срок, т. е. 33 дня
8	33+6=39	EET узла 7 + продолжительность операции H

*Выбранное значение EET

Полученные значения сроков наносятся на стрелочный граф, как это показано на рис. 12.

EET последнего события равно 39 дням, которые также определяют *общую* продолжительность выполнения проекта.

Чтобы определить критические операции, будем двигаться по графу начиная с конечного узла и вычисляя LET каждого события. Предположим, что для конечного события EET = LET. Если в некоторый узел входит более одной стрелки, то возникает проблема выбора значения LET. Так как событие должно завершиться к сроку, удовлетворяющему всем наиболее поздним срокам начала событий, которые выходят из данного узла для LET, следует выбрать наименьшее значение.

Найденные значения сроков наносятся на стрелочный граф, изображенный на рис. 12.

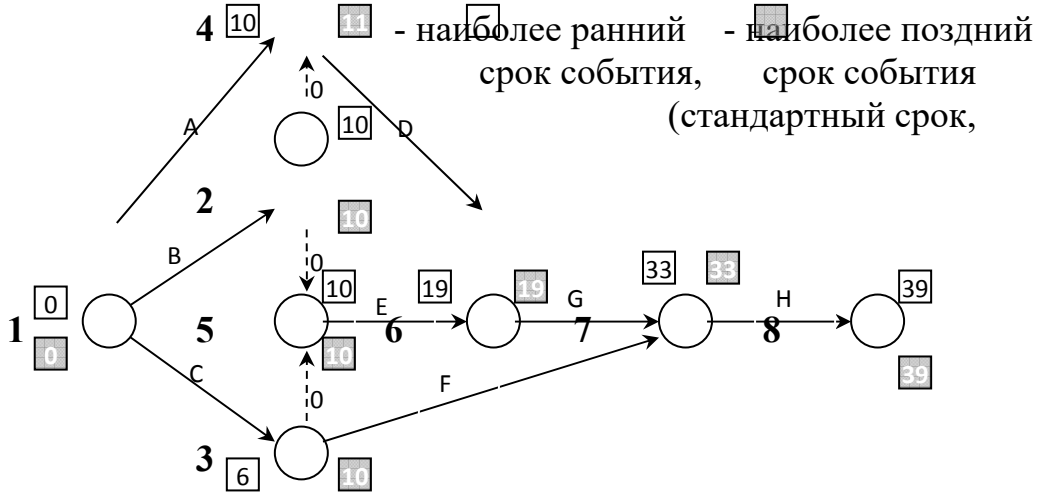


Рис. 12. Стрелочный граф для примера 5 с указанием EET и событий

Операция является критической, если для нее справедливы следующие соотношения:

EETначала = LETначала и EETокончания = LETокончания

LETокончания - EETначала - Продолжительность = 0.

Из рисунка 12 видно, что критическими, как и ранее, являются операции B, E, G и H. Любые замедления на критическом пути приведут к задержке срока выполнения, всего проекта. Между тем для некритических путей можно допустить некоторые задержки при выполнении составляющих их операций или пересмотреть график их выполнения. Запас времени, который существует в схеме проекта, называется **резервом времени.**

Различают несколько видов резерва времени, возникающих под влиянием различных воздействий, которые оказывает запас времени на схему выполнения проекта.

Общим резервом называется количество времени, на которое можно увеличить продолжительность операции в результате продления срока ее выполнения или пересмотра плана, не влияющего на продолжительность выполнения проекта в целом. **Свободным резервом** называется количество времени, на которое можно увеличить продолжительность операции в результате продления срока ее выполнения или пересмотра плана, не оказывающего воздействия на наиболее ранний срок выполнения любой последующей операции. Иногда используют третий вид, так называемый **независимый резерв времени.** Он не оказывает никакого влияния на предшествующие или последующие операции. Для любой операции

Общий резерв времени = LETокончания - EETначала - Продолжительность, также

Свободный резерв времени = EETокончания - EETначала – Продолжительность

Независимый резерв = EETокончания-LETначала – Продолжительность.

Иногда бывает полезно изобразить на графе имеющийся в наличии резерв времени, особенно если план выполнения операций необходимо

пересмотреть. В этом случае одним из возможных методов является **график Ганта.**

Таблица 7. Расчет значений EET для примера 5

Узел	LET,дней	Комментарий
8	39	Конечный узел LET = EET
7	39-6=33	LET узла 8 – продолжительность операции H
6	33-14=19	LET узла 7 – продолжительность операции G
5	19-9=10	LET узла б – продолжительность операции E
4	19-8=11	LET узла 6 – продолжительность операции D
3	10-0=0* или 33-14=19	LET узла 5 – продолжительность фиктивной операции или LET узла 7 – продолжительность операции F. Выбирается минимальный срок, т.е. 10 дней
2	10-0=10* или 11-0=11	LET узла 5 – продолжительность фиктивной операции или LET узла 4 - продолжительность фиктивной операции Выбирается минимальный срок, т.е. 10 дней
1	11-8=3 или 10-10=0* или 10-6=4	LET узла 4 – продолжительность операции A или LET узла 2 продолжительность операции B или LET узла 3 – продолжительность операции C Выбирается минимальный срок, т.е. О дней

*Выбранное значение LET.

Пример 6. По данным примера **5** для каждой операции найдем общий резерв времени.

Операции, общий резерв времени которых равен нулю, являются критическими. На рис. 13 построен график Ганта, и отмечены, возможно наиболее ранние сроки начала операций.

Таблица 8. Расчет резерва времени операций для примера 5 (дней)

операция	LET окончания	LET начала	продолжительность	Общий резерв времени
A	11	0	8	3
B	10	0	10	0
C	10	0	6	4
D	19	10	8	1
E	19	10	9	0

F G H	33 33 39	6 19 33	14 14 6	13 0 0

Стандартные сроки

Стандартные сроки

Операции

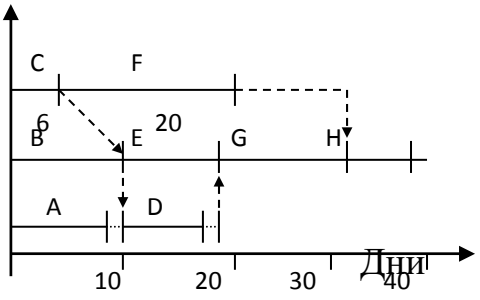


Рис. 13. График Ганта для примера 5

1.6 Стоимость проекта

Общая стоимость проекта зависит от стоимости выполнения каждой операция, а также от любых дополнительных переменных или постоянных расходов. Так как необходимо завершить все операции, независимо от того, являются они критическими или нет, общая стоимость выполнения операций представляет собой арифметическую сумму отдельных значений стоимости каждой операции.

Можно снижать продолжительность выполнения некоторых операций с помощью дополнительных ресурсов. Косвенным последствием такой меры является увеличение стоимости данных операций. Однако если операция критическая, то экономия времени ее выполнения может привести к общей экономии времени выполнения проекта в целом, а следовательно, и к снижению общей стоимости проекта.

Возможный наименьший срок, к которому можно завершить операцию, получил название **критического срока.** В некоторых случаях завершить операцию можно только либо к стандартному, либо к критическому сроку их выполнения, но не между ними. Иногда, напротив, существует возможность

постепенно уменьшать время выполнения операции до того момента, пока не будет достигнут критический срок ее выполнения. Рассмотрим, как действует уменьшение времени выполнения операций на календарный план, и стоимость выполнения проекта, принимая во внимание две различные цели:

1. Минимизацию общего времени выполнения проекта;
2. Минимизацию общей стоимости проекта.

Оптимизация сетевого графика

1.7 Минимизация общей продолжительности проекта с минимальными дополнительными расходами

Для этой цели необходимо обладать информацией о стоимости каждой операции, любом возможном уменьшении времени ее выполнения и о дополнительных издержках, связанных со снижением времени выполнения операции.

Пример 7. Обратимся к данным примера **2.** Ниже приводится дополнительная информация о стоимости операций и возможном уменьшении времени их выполнения.

Показатели критических значений отражают минимальное время, за которое можно выполнить операцию, и общую стоимость выполнения операции в течение этого времени. Необходимо сделать выбор между стандартными значениями времени и издержек и их критическими значениями. Практически невозможно получить экономию времени выполнения операции в один день при пропорциональном возрастании ее

стоимости. Помимо стоимости каждой операции необходимо учесть стоимость строительной площадки, составляющую 1000 руб. в день.

1. Каково минимальное время, в течение которого можно завершить проект?

2. Какова соответствующая минимальная дополнительная стоимость?

Решение

Минимальное время можно найти, рассчитав для всех, как критических, так и некритических операций, критическое время их выполнения. Ниже изображен стрелочный граф, построенный в примере **2.** На граф нанесены значения EET и LET, найденные на основе критических значений времени выполнения операций.

Нетрудно заметить, что EET узла 8 равно 28 дням, поэтому минимальное время выполнения проекта также составляет 28 дней. Критический путь остается неизменным: B-E-G-H.

Общую стоимость можно найти из следующего уравнения:

Общая стоимость = Критическая стоимость операций + 28 х

х Стоимость строительной площадки в день = 102 750 руб. +

+ 28 х 1000 руб. = 130750 руб.

Таблица 9. Значения стандартных и критических сроков и соответствующих издержек выполнения операций для примера 7

ОПЕРА-ЦИЯ	НЕПОСРЕДСТ-ВЕННО ПРЕДШЕ-СТВУЮЩИЕ ОПЕРАЦИИ	СТАНДАРТНОЕ ЗНАЧЕНИЕ		Критическое значение	
		ВРЕМЕ-НИ, ДНЕЙ	СТОИМО-СТИ, РУБ.	Времени, дней	Стоимости, руб.
A	-	8	7500	4	9000
B	-	10	8500	8	11000
C	-	6	6000	5	7000
D	A,B	8	13000	5	16000
E	B,C	9	14000	6	16500
F	C	14	14500	11	18000
G	D,E	14	13500	10	18750

H	F,G	6	5500	4	6500
ОБЩИЕ ИЗДЕРЖКИ ВЫПОЛНЕНИЯ ОПЕРАЦИЙ			82500		102750

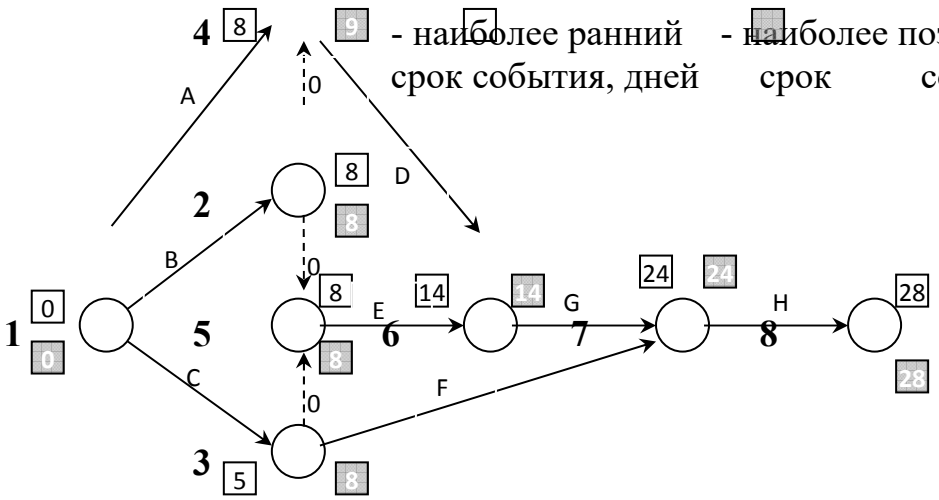


Рис. 14. Стрелочный граф для примера 7 с указанием критического времени

Между тем найденное значение стоимости выполнения проекта в указанное время не является минимальным, поскольку необходимости использовать критические значения для некритических операций нет. Некритическими являются операции A, C, D и F. Поэтому необходимо найти эффект от использования соответствующих этим операциям некритических значений показателей. В случае, если существует возможность восстановить их стандартную продолжительность, не увеличивая при этом общую продолжительность выполнения проекта, можно будет одновременно достичь и экономию стоимости в результате использования ее некритических значений.

Критические значения для операции A можно не использовать, поскольку увеличение продолжительности ее выполнения до 8 дней не

меняет EET узла 4, и, следовательно, не оказывает воздействия на выполнение остальных операций календарного плана. Использование некритических значений для операции A позволяет достичь экономии, составляющей 1500 руб.

Увеличение продолжительности операции C с 5 до 6 дней приведет к увеличению значения EET узла 3 до 6, однако не окажет воздействия на EET узлов 5 и 7. Вследствие этого продолжительность выполнения проекта останется неизменной. Использование некритических значений, соответствующих операции C, позволит получить экономию 1000 руб.

Если использовать некритические значения показателей операции D, то EET узла 6 возрастет до 16 дней. Узел 6 принадлежит критическому пути, поэтому для того, чтобы достичь минимального общего времени выполнения проекта, составляющего 28 дней, необходимо применять критические значения времени и стоимости операции D.

Использование некритических значений для операции F не изменит EET узла 7 и не приведет к увеличению продолжительности проекта в целом. Используя для F некритические значения, мы сможем достичь экономии, составляющей 3500 руб.

Минимальная стоимость выполнения проекта за 28 дней составит:

130750 - 1500 (A) - 1000 (C) - 3500 (P) = 124750 руб.

Стоимость выполнения проекта в стандартные сроки равна:

82500 (стоимость операций) + 39000 (стоимость строительной площадки)= = 121500 руб

Следовательно, дополнительная стоимость, связанная с завершением выполнения проекта на 11 дней раньше, будет равна:

124750 - 121500 = 3250 руб.

Пример 8. Обратимся к данным примера 1. В табл. 10 приводится дополнительная информация о стоимости операций и возможном сокращении времени их выполнения.

Переменные накладные расходы составляют 300 руб. в неделю в течение всего времени выполнения проекта.

1. Определить стандартные значения общего времени выполнения и общей стоимости проекта.

2. Найти минимальное время, за которое можно выполнить данный проект, и соответствующее ему минимальное значение стоимости.

Таблица 10. Стандартные и критические значения сроков выполнения и стоимости операций для примера 8

Операция	Стандартное значение		Возможное сокращение времени, недель	Критическое время, недель	Дополнительные издержки сокращения времени на неделю, руб.
	Времени, недели	Стоимость, руб.			
A	2	400	1	1	400
B	1	0	0	1	0
C	4	200	2	2	125
D	6	450	4	2	175
E	3	700	2	1	250
F	3	200	2	1	200
G	4	600	3	1	125
H	2	0	0	2	0
I	3	250	1	2	200
J	8	600	4	4	100
K	2	450	1	1	250
L	2	200	1	1	150
Стоимость операций 4050					

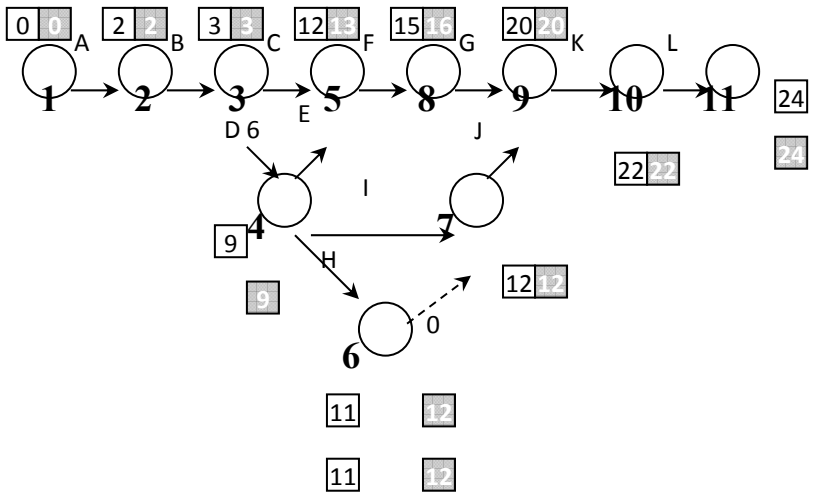


Рис. 15. Стрелочный граф для примера 8 с указанием стандартных сроков

□- наиболее ранний - наиболее поздний

срок события, срок события (стандартный срок, дней)

Решение

На рис. 15 воспроизведен стрелочный граф, построенный в примере **1.** Для каждой операции на графе указаны значения EET и LET. Стандартный срок выполнения проекта составляет 24 недели, а соответствующий ему критический путь имеет следующий вид:

A-B-D-1-J-K-L.

Общая стоимость проекта составляет:

4050 (стоимость операций) + 24x300 (переменные накладные расходы) = 11250руб.

Чтобы определить минимальное время, требующееся для выполнения проекта в целом, каждой операции поставим в соответствие минимальный срок ее завершения. На рис. 16 показаны значения этих сроков и итоговые значения EET и LET.

Минимальная продолжительность проекта составляет 12 недель. В данном случае критическими оказываются следующие пути:

A-B-D-1-J-K-L и *A-B-D-H-J-K-L.*

Проверим, можно ли, используя некритические значения для некоторых некритических операций, получить экономию денежных средств.

Таблица 11. Использование некритических значений показателей для некритических операций из примера 8

Опера-ция	Изменение продолжитель-ности	Эффект
E	Увеличение на 2 недели	EET узла 5 становится равным 7 неделям; EET узла 8 становится равным 8 неделям, не влияя при этом на EET узла 9, принадлежащего критическому пути; других воздействий нет. E выполняется в стандартный срок
F	Увеличение на1 неделю	EET узла 8 становится равным 9 неделям. Операции E, F и G становится критическими.
G	Увеличение невозможно	
C	Увеличение на 2 недели	На узел 5 не оказывается никакого воздействия. C выполняется в стандартный срок.

Некритическими являются операции C, E, F и G. Продолжительность операций в данном примере можно изменять по интервалам в одну неделю,

так как единицей измерения продолжительности является неделя. В первую очередь рассмотрим операции, которые, если использовать их некритические значения, могут принести наибольшую экономию денежных средств. Операции будем рассматривать в следующем порядке: E (250 руб.), F (200 руб.), и C (125 руб.) или G (125 руб.).

Минимальная стоимость выполнения проекта за 12 недель составила:

(4050 (стандартная стоимость операций) + 1 • 400 (A) + 4 • 175 (B) + 1 • 200 (F) + 3 • 125 (G) + 1 • 200 (I) + 4 • 100 (J) + 1 • 250 (K) + 1 • 150 (L) (предельные издержки) + 12 • 300 (переменные накладные расходы) = 4050 + 2675 + 3600 = 10325 руб.

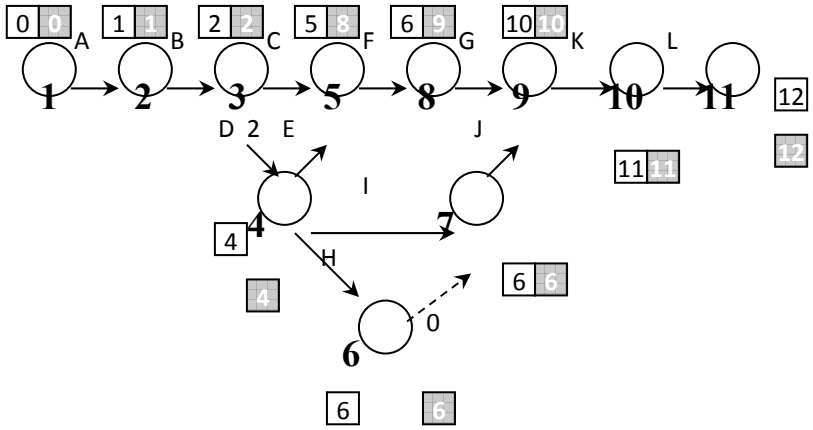


Рис. 16. Стрелочный граф для примера 8 с указанием критических сроков

□- наиболее ранний - наиболее поздний

срок события, срок события (стандартный срок, дней)

1.8 Выполнение проекта с минимальными издержками

Если выполнение проекта требует оплаты переменных накладных расходов, таких, например, как расходы, связанные с оборудованием строительной площадки, то может оказаться выгодным снижение продолжительности выполнения проекта. Поскольку сами эти сокращения влекут за собой определенные издержки, необходимо подвести баланс.

Экономия времени может быть достигнута только в том случае, если сократить продолжительность критических операций. Критические значения должны использоваться только по тем критическим операциям, по которым величина экономии накладных расходов превышает стоимость выполнения операции за критическое время.

Пример 9. Обратившись к данным примера **7,** определим минимальную стоимость проекта и соответствующее время его выполнения. Предполагается, что операции можно выполнять либо в стандартные, либо в критические сроки, но не в промежутке между ними.

Решение

Используя граф, построенный в примере **5** для стандартных сроков выполнения операций, перечислим все критические операции и соответствующие им показатели максимально возможной экономии времени и чистой экономии стоимости.

Таблица 12. Расчет минимальной стоимости проекта для примера 9

Операция	Число дней экономии для критического времени	Дополнительная стоимость критического времени, руб.	Экономия, руб.	Чистая экономия, руб.	Комментарии
B	2	2500	2•1000	-500	Критические значения не используются
Либо E	1*	2500	1•1000	-1500	Критические значения не используются
Либо E и D	3	5500	3•1000	-2500	Критические значения не используются
G	4	5250	4•1000	-1250	Критические значения не используются
H	2	1000	2•1000	+1000	Используются критические значения. Снижение продолжительности проекта с 39 до 37 дней

*Достичь экономии, равной 3 дням, нельзя, поскольку в этом случае путь A-D-G-H становится критическим. Поэтому общая продолжительность снижается только на один день. Если же использовать критические значения одновременно для E и O, достигается экономия времени, равная 3 дням. Однако соответствующая стоимость становится равной: 2500 руб. + 3000 руб., т.е. такая экономия времени не целесообразна.

Минимальная стоимость проекта равна: 121500 - 1000 = 120500 руб. Соответствующее время его выполнения составляет 37 дней.

1.9 Неопределённость времени выполнения операций

В приведенных выше методах анализа предполагалось, что время выполнении операций точно известно. Однако на практике сроки выполнения операций обычней являются довольно неопределенными. Управляющий производством может выдвинуть некоторые предположения о том, сколько времени потребуется для выполнения каждой работы, но не может предусмотреть возможные трудности или задержки выполнения. Неопределенность сроков выполнения операций означает, что общая продолжительность проекта также подвержена неопределенности.

Выбор метода, позволяющего учесть эту неопределенность, зависит от типа проекта и природы неопределенности. Если можно определить минимальную и максимальную продолжительности каждой операции, то их рассчитывают с помощью показателей ожидаемой (средней) продолжительности и ожидаемого времени выполнения проекта. Алгоритм, получивший наиболее широкое применение, называется **методом оценки и пересмотра проектов (Project Eva1uation and Reiew Technique -- PERT).** При вычислении ожидаемого времени выполнения проекта методом PERT используются показатели ожидаемого времени выполнения операций. Оставшаяся часть алгоритма аналогична описанным выше алгоритмам,

применяемым в случаях, когда время выполнения операций является фиксированной величиной.

Если время выполнения операций подвержено влиянию неопределенности, то большое значение приобретают некритические пути в графе, когда могут изменяться сроки выполнения всех операций. На практике может оказаться, что путь, который на основе ожидаемых значений сроков считался некритическим, становится критическим в соответствии с результатами метода определения критического пути.

В основу метода PERT положена предпосылка о проведении продолжительности операции. Предполагается, что время выполнения каждой отдельно взятой операции аппроксимируется р-распределением. Если это верно, то распределение времени выполнения проекта в целом является нормальным. Метод PERT может применяться при анализе конкретного проекта только в случае выполнения данной предпосылки. График р-распределения изображен на рис. 17. Возможное наименьшее время выполнения операции называют **оптимистическим сроком** (a), а возможное наибольшее время ее выполнения - **пессимистическим сроком** (b).

Пику распределения соответствует наиболее вероятное время выполнения операции (m). Необходимо произвести оценку каждого из этих трех сроков для всех операций, входящих в граф.

Исходя из этих трех значений можно найти ожидаемую продолжительность операции (t) и ее дисперсию. Ожидаемая продолжительность операции определяется следующим образом:

$$t = \frac{a + 4m + b}{6}$$

Соответствующая дисперсия ожидаемой продолжительности определяется по формуле:

$$\sigma_t^2 = \left(\frac{b - a}{6} \right)^2$$

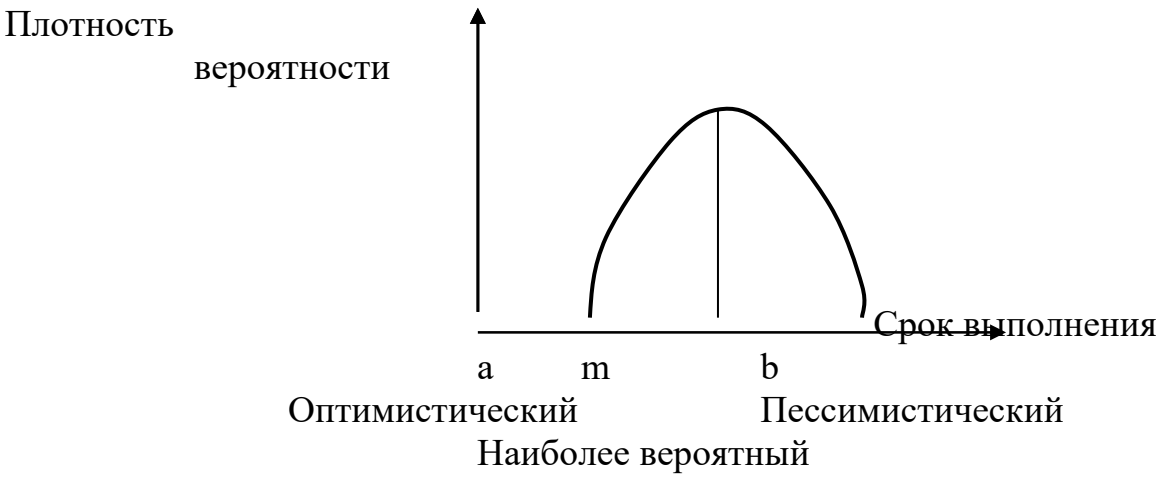


Рис. 17. Стандартное β-распределение для времени выполнения операций

Время выполнения проекта можно найти непосредственно из графа, используя для этого ожидаемые значения продолжительности операций. Предполагается, что время выполнения проекта в целом распределено по нормальному закону.

В предположении, что сроки выполнения операций не зависят друг от друга, среднее значение нормального распределения определяется как сумма математических ожиданий продолжительности критических операций, а дисперсия - как сумма их дисперсий. Полученное нормальное распределение можно использовать для оценки вероятности завершения проекта к заранее установленной дате.

Алгоритм метода PERT аналогичен анализу сетевого графа с фиксированными значениями продолжительности операций.

1.Составить список всех операций, входящих в проект, с указанием непосредственно предшествующих операций, а также оптимистического, наиболее вероятного и пессимистического сроков их выполнения.

2. Построить сетевой граф.

3.В предположении, что время выполнения любой операции аппроксимируется **р-распределением, оценить** для **каждой операции ожидаемое время ее выполнения и его дисперсию.**

4. Используя ожидаемые значения сроков выполнения операций, найти продолжительность проекта в целом.

5. Определить критические операции и критический путь.

6. С помощью значений дисперсии для критических операций оценить дисперсию ожидаемой продолжительности всего проекта.

Пример: Процесс создания и серийного производства нового вида продукта компаний "НЕВА" включает в себя следующие операции (см. табл. 13).

1. Определим ожидаемое число недель, необходимое для выполнения проекта.

Какие операции являются критическими?

2. Какова вероятность того, что выполнение проекта займет более 16 недель?

Таблица 13. Таблица операций и сроков их выполнения для примера 10

Опера-ция	Непосред-ственно, предшест-вующие операции	Сроки выполнения операций, недель		
		Оптимисти-ческий, a	Наиболее вероятный, m	Пессимисти-ческий,b
A	-	1,5	2	2,5
B	A	2	2,5	6
C	-	1	2	3
D	C	1,5	2	2,5
E	B,D	0,5	1	1,5
F	E	1	2	3
G	B,D	3	3,5	7
H	G	3	4	5
I	F,H	1,5	2	2,5

Решение

Ожидаемые сроки выполнения операций и соответствующие дисперсии приведены в таблице 14.

Ниже приведен сетевой граф с указанием ожидаемой продолжительности каждой операции (см. рис. 18).

Расчет ожидаемого срока выполнения проекта в целом производится обычным способом. Как показано на рис. 18, выполнение проекта предполагается осуществить за 15 недель. Критическими являются операции A, B, G, H и I. Приведем для сравнения другие возможные пути в графе:

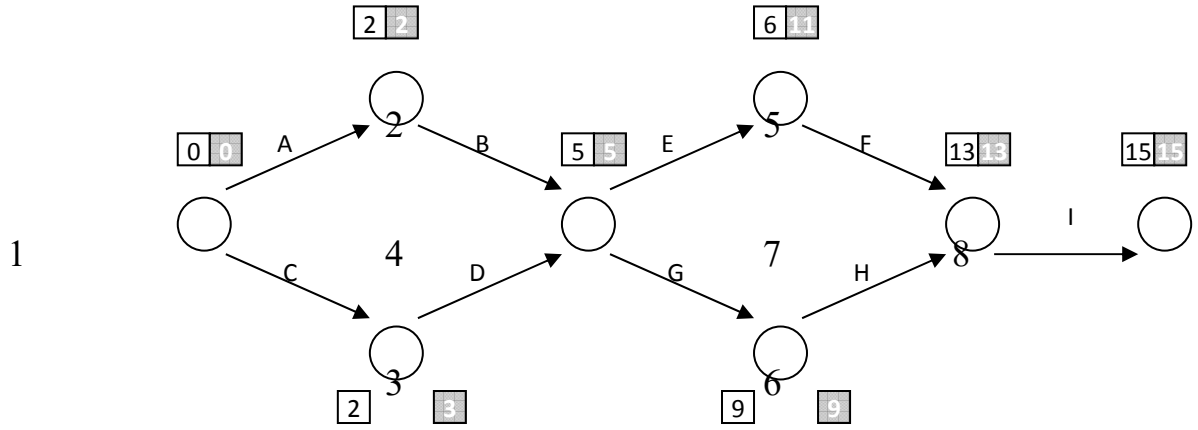


Рис. 18. Стрелочный граф с указанием ожидаемых сроков выполнения операций для примера 10

□- наиболее ранний - наиболее поздний

срок события, срок события (ожидаемый срок, дней)

A, B, E, F, I - занимает 10 недель,

C, D, E, F, I - занимает 9 недель,

C, D, G, H, I - занимает 14 недель.

Следует отметить, что путь - C,D,G,H, I - занимает время, которое меньше выполнения критического пути всего на одну неделю. Поэтому небольшие изменения времени выполнения некоторых операций могут привести к изменению критического пути.

Дисперсия ожидаемого времени выполнения всего проекта определяется как сумма дисперсий критических операций:

$\sigma^2=\sigma^2_A+\sigma^2_B+\sigma^2_G+\sigma^2_H+\sigma^2_I$

следовательно,

$\sigma^2= 1/36 + 16/36 + 6/36 +4/36 + 1/36 = 38/36 =1{,}11$ недель^2

Стандартное отклонение времени выполнения проекта составит:

$\sqrt{1,11}$=1,03 недель

Вероятность того, что выполнение проекта займет более 16 недель, можно найти следующим образом: Шестнадцать недель составляют z стандартных отклонений от среднего, где:

$$z = \frac{16-15}{1,03} = 0,97$$

По таблице стандартного нормального распределения находим:

P *(*z > 0,97) =0,166.

Следовательно, вероятность того, что выполнение проекта займет более 16 недель, равна 16,6%.

Таблица 14. Расчет ожидаемых сроков выполнения операций и их дисперсий по данным примера 10

Операция	Ожидаемый срок выполнения	Дисперсия, недель^2
A	$\frac{1,5+4\cdot 2+2,5}{6}=2$	$\left(\frac{2,5-1,5}{6}\right)^2=1/36$
B	$\frac{1,5+4\cdot 2,5+6}{6}=3$	$\left(\frac{6-2}{6}\right)^2=16/36$
C	$\frac{1+4\cdot 2+3}{6}=2$	$\left(\frac{3-1}{6}\right)^2=4/36$
D	2	=1/36
E	1	=1/36
F	2	=4/36
G	4	=16/36
H	4	=4/36
I	2	=1/36

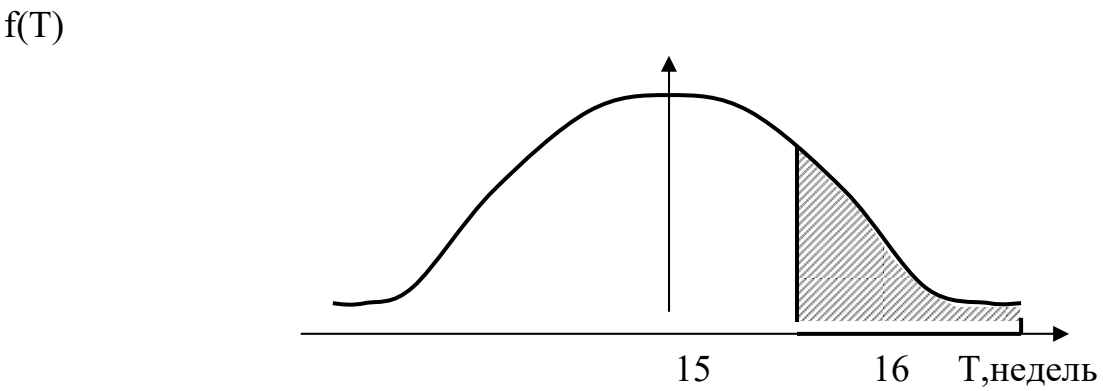


Рис. 19. Распределение времени выполнения проекта для примера 10

Распределение ресурсов

Сетевой граф отражает логическую последовательность выполнения операций, входящих в проект. Ни в одном из видов анализа, рассмотренных нами выше, не принимались во внимание какие-либо ограничения на обеспечение ресурсами. Исходный календарный план выполнения операций составлялся при условии, что все необходимые ресурсы имеются в достаточном количестве. Однако такая ситуация имеет место далеко не всегда, а если это так, то использование ресурсов в соответствии с потребностями, указанными в исходном календарном плане, может оказаться неэкономичным.

Метод составления календарного плана с учетом обеспечения ресурсами зависит от конкретных целей лиц, осуществляющих контроль за ходом выполнения проекта. Например, вопросом первостепенной важности может оказаться завершение проекта к определенному сроку безотносительно к затратам ресурсов - такие планы ограничены по времени. И наоборот, в условиях ограниченности в денежных средствах на выполнение проекта отводится определенное количество ресурсов, тогда как срок выполнения не принимается в расчет - такие планы ограничены по

ресурсам. В данном контексте к ресурсам можно отнести рабочую силу, оборудование, сырье, денежные средства, производственные площади и т.д.

Перед тем как приступить к выполнение проекта, управляющий производством должен четко сформулировать критерий, в соответствии с которым будет осуществляться распределение ресурсов. В качестве такого критерия можно выбрать:

1. Максимальное использование ресурсов. Оценить использование ресурсов можно через соответствующий коэффициент:

$$\text{Коэффициент использования} = \frac{\text{Общее количество используемых ресурсов}}{\text{Общее количество наличных ресурсов}}$$

2. Минимизацию максимальных потребностей в ресурсах.

3. Минимизацию максимальных изменений потребностей в ресурсах.

Кроме названных, существует множество других критериев. Существует также множество возможных методов решения проблемы распределения ресурсов, таких, как, например, эвристические методы, методы линейного и других видов математического программирования. Рассмотрим один из простейших алгоритмов, в котором используются графики ресурсов и "метод проб и ошибок".

1.11 Графики ресурсов

Если общая потребность в некотором ресурсе определяется на основе постоянных интервалов, например, за один день или за одну неделю, то можно построить **график ресурса.** Ресурсы, требуемые для осуществления каждой работы, складываются по всем работам, выполняемым одновременно, в предположении, что каждая работа начинается в наиболее

ранний срок ее выполнения. Необходимо построить отдельные графики по каждому виду ресурса. На рис. 20 схематично изображен график ресурса "рабочая сила". Как следует из приведенного графика, иногда потребности в рабочей силе превышают ее наличие, но в то же время общее число требуемых человеко-часов не превосходит их наличного количества.

Если потребность в ресурсе превысила его лимит, необходимо либо вложить в проект дополнительное количество ресурса, либо пересмотреть календарный план выполнения операций. Иногда в таких ситуациях необходимо задержать срок выполнения проекта. Несмотря на то, что некоторые операции проекта не имеют явной логически последовательной взаимосвязи, одновременное их выполнение часто оказывается невозможным вследствие ограничений на ресурсы. Это ограничение можно отразить на графике ресурса, если провести линию, соответствующую наличному количеству данного ресурса. Такой прием позволит не планировать выполнение определенных операций на один и тот же период.

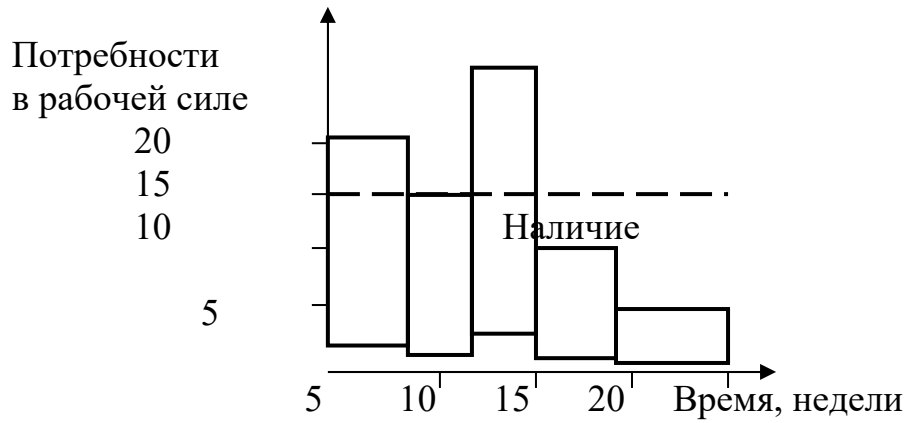


Рис. 20. График ресурса "рабочая сила"

Пример 11. Компания с ограниченной ответственностью "ТРАСТ" заключила контракт на проведение работ по асфальтированию стоянки автомобилей. Менеджер проекта установил, что данная работа состоит из восьми основных операций. Приведем детальное описание этих операций:

Таблица 15. Операции для примера 11, с указанием сроков выполнения и потребностей в рабочей силе

Операция	Предшествующие Операции	Время, дней	Число человек, требуемое для выполнения операции
A B C D E F G H	- - - A C B,E C F,G	3 6 7 8 4 3 10 3	1 1 2 2 1 2 2 1

Ввиду необходимости срочного выполнения работ на других участках, "ТРАСТ" может выделить только четырех человек для проведения работ на автомобильной стоянке. Определим, сколько времени займет проведение работ и как следует распределить рабочих. Предположим, что каждый из рабочих может выполнять любую операцию.

Решение

Предположив, что все операции начинаются в наиболее ранний срок, построим соответствующий график "рабочей силы". После этого можно составить календарный план выполнения операций, удовлетворяющий ограничению на количество работников. Сначала построим сетевой граф и определим критический путь.

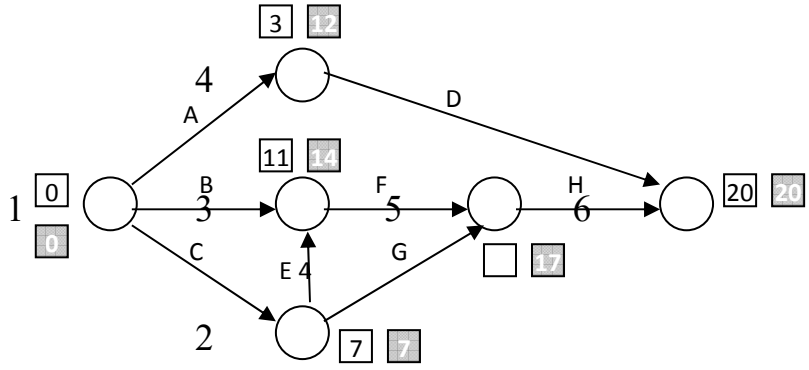


Рис. 21. Стрелочный граф для примера 11

- наиболее ранний - наиболее поздний

срок события, срок события (стандартные сроки, дней)

Время выполнения проекта в целом, если не принимать во внимание обеспечение ресурсами, составляет 20 дней. Критический путь выглядит следующим образом: C - G - H.

В предположении, что выполнение всех операций начинается в наиболее ранние сроки, посмотрим график Ганта и соответствующий график ресурса. График Ганта отражает распределение резерва времени на момент окончания каждой операции. С его помощью мы можем определить, какие операции выполняются одновременно и по каким операциям можно изменить календарный план их выполнения таким образом, чтобы эти изменения не привели к задержке выполнения проекта в целом.

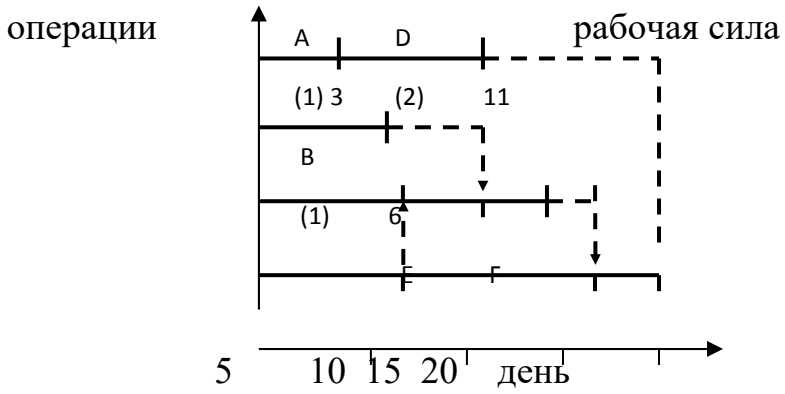


Рис. 22. График Гранта для примера 11

Из графика ресурса следует, что лимит, равный четырем рабочим, превышается, когда выполнение операции D попадает в промежуток между 3 и 11 днями осуществления проекта. Пересмотреть календарный план и полностью удовлетворить потребности в рабочей силе, соответствующие операции D, нельзя. Для выполнения критических операций C и D требуются два человека, поэтому операция D не может быть начата в течение 17 дней,

т.е. до тех пор, пока не закончится выполнение остальных некритических операций.

Если операцию D отложить на 12 дней, то в дни с 12 по 14 потребность в рабочих все еще будет превышать их наличие: в эти дни будут выполняться операции G(2 человека), F(2 человека) и (2 человека). В этом случае придется либо привлечь к работе: одного рабочего дополнительно на указанный период, либо отложить операцию D до момента, когда будет завершена операция F, т.е. до 14-го дня. При последнем варианте будет иметь место задержка в выполнении проекта, равная двум дням. Таким образом, его продолжительность возрастает с 20 до 22 дней.

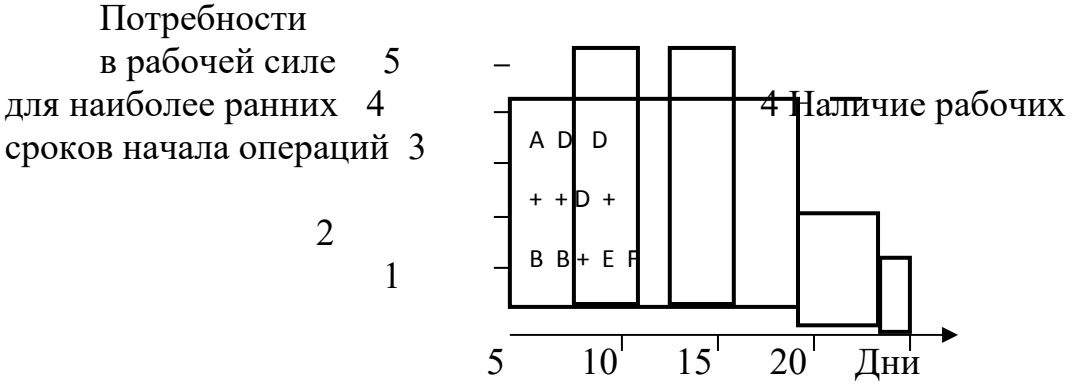


Рис. 23. График ресурса для примера 11, соответствующий наиболее ранним срокам начала выполнения операций.

Л Е К Ц И Я

по учебной дисциплине:
«Специальные главы математики»

Для магистров направления:
09.04.02 Информационные системы и технологии

Тема 3. Динамическое программирование
Лекция 3-4. Условное и безусловное управление. Рекуррентные соотношения. Уравнение Беллмана.

Лекция 3-4. Условное и безусловное управление. Рекуррентные соотношения. Уравнение Беллмана

Цель лекции: Изучить основные понятия динамического программирования. Ознакомиться с задачами управления и замены оборудования. Изучить основные понятия и принципы применения задач динамического программирования.

План лекции:

1. Условное и безусловное управление.
2. Рекуррентные соотношения.
3. Уравнение Беллмана

Методика проведения лекции – лекция –визуализация

Данный вид лекции является результатом нового использования дидактического принципа наглядности. Психологические и педагогические исследования показывают, что наглядность не только способствует более успешному восприятию и запоминанию учебного материала, но и позволяет активизировать умственную деятельность, глубже проникать в сущность изучаемых явлений, показывает ее связь с творческими процессами принятия решений, подтверждает регулирующую роль образа в деятельности человека.

Лекция - визуализация учит студентов преобразовывать устную и письменную информацию в визуальную форму, что формирует у них профессиональное мышление за счет систематизации и выделения наиболее значимых, существенных элементов содержания обучения. Этот процесс визуализации является свертыванием мыслительных содержаний, включая разные виды информации, в наглядный образ; будучи воспринят, этот образ, может быть развернут и может служить опорой для мыслительных и практических действий.

При подготовке и проведении лекции-визуализации преподавателю следует обратить внимание на следующие особенности реализации рассматриваемой формы проведения занятия. По содержанию визуализованная лекция представляет собой устную информацию,

преобразованную в визуальную форму. Видеоряд, будучи воспринятым и осознанным, сможет служить опорой адекватных мыслей и практических действий. Преподаватель должен выполнить такие демонстрационные материалы, такие формы наглядности, которые не только дополняют словесную информацию, но сами выступают носителями содержательной информации.

Подготовка такой лекции состоит в реконструировании, перекодировании содержания лекции или ее части в визуальную форму для предъявления студентам через технические средства обучения.

Чтение такой лекции сводится к сводному, развернутому комментированию подготовленных визуальных материалов.

Визуализированные материалы:

1. **Задача планирования рабочей силы**
2. **Задача замены оборудования.**
3. **Задача о загрузке.**

В начале 50-х годов прошлого века американский исследователь Р. Беллман привлек внимание к новому методу решения оптимизационных задач, в основе которого лежит сведение данной задачи к последовательности однотипных, притом более простых задач. В тех случаях, когда изучаемый процесс разворачивается во времени, такое расчленение на более простые задачи обусловлено хронологией процесса. Однако и в тех случаях, когда фактор времени вообще не присутствует, часто оказывается возможным расчленение задачи на ряд более простых, когда условия каждой следующей задачи зависят от результатов решения предыдущей. Грубо говоря, оптимизационную задачу с $N_1 \cdot N_2$ неизвестными сводят к

последовательности N_1 задач, решаемых по цепочке, причем в каждой из них участвует N_2 неизвестных. Метод, предложенный Р. Беллманом, получил название метода *динамического программирования*. Реализация различных задач оптимизации связана, как обычно, с ЭВМ. Тем более это относится к задаче динамического программирования, где решение задачи есть, как правило, трудоемкий процесс, связанный с большим объемом вычислений.

1.1 Задача динамического программирования

Большинство методов исследования операций связано в первую очередь с задачами вполне определенного содержания. Классический аппарат математики оказался малопригодным для решения многих задач оптимизации, включающих большое число переменных и/или ограничений в виде неравенств. Несомненна привлекательность идеи разбиения задачи большой размерности на подзадачи меньшей размерности, включающие всего по нескольких переменных, и последующего решения общей задачи по частям. Именно на этой идее основан метод динамического программирования.

Динамическое программирование (ДП) представляет собой математический метод, заслуга создания и развития которого принадлежит прежде всего Беллману. Метод можно использовать для решения весьма широкого круга задач, включая задачи распределения ресурсов, замены и управления запасами, задачи о загрузке. Характерным для динамического программирования является подход к решению задачи по этапам, с каждым из которых ассоциирована одна управляемая переменная. Набор рекуррентных вычислительных процедур, связывающих различные этапы, обеспечивает получение допустимого оптимального решения задачи в целом при достижении последнего этапа.

Происхождение названия динамическое программирование, вероятно, связано с использованием методов ДП в задачах принятия решений через фиксированные промежутки времени (например, в задачах управления запасами). Однако методы ДП успешно применяются также для решения задач, в которых фактор времени не учитывается. По этой причине более удачным представляется термин многоэтапное программирование, отражающий пошаговый характер процесса решения задачи.

Фундаментальным принципом, положенным в основу теории ДП, является принцип оптимальности. По существу, он определяет порядок поэтапного решения допускающей декомпозицию задачи (это более приемлемый путь, чем непосредственное решение задачи в исходной постановке) с помощью рекуррентных вычислительных процедур.

Динамическое программирование позволяет осуществлять оптимальное планирование управляемых процессов. Под «управляемыми» понимаются процессы, на ход которых мы можем в той или другой степени влиять.

Пусть предполагается к осуществлению некоторое мероприятие или серия мероприятий («операция»), преследующая определенную цель. Спрашивается: как нужно организовать (спланировать) операцию для того, чтобы она была наиболее эффективной? Для того, чтобы поставленная задача приобрела количественный, математический характер, необходимо ввести в рассмотрение некоторый численный критерий W, которым мы будем характеризовать качество, успешность, эффективность операции. Критерий эффективности в каждом конкретном случаи выбирается исходя из целевой направленности операции и задачи исследования (какой элемент управления оптимизируется и для чего).

Сформулируем общий принцип, лежащий в основе решения всех задач динамического программирования («принцип оптимальности»):

«Каково бы ни было состояние системы S перед очередным шагом, надо выбрать управление на этом шаге так, чтобы выигрыш на данном шаге плюс оптимальный выигрыш на всех последующих шагах был максимальным».

Динамическое программирование – это поэтапное планирование многошагового процесса, при котором на каждом этапе оптимизируется только один шаг. Управление на каждом шаге должно выбираться с учетом всех его последствий в будущем.

При постановке задач динамического программирования следует руководствоваться следующими принципами:

1. Выбрать параметры (фазовые координаты), характеризующие состояние S управляемой системы перед каждым шагом.

2. Расчленить операцию на этапы (шаги).

3. Выяснить набор шаговых управлений x_i для каждого шага и налагаемые на них ограничения.

4. Определить какой выигрыш приносит на i-ом шаге управление x_i, если перед этим система была в состоянии S, т.е. записать «функцию выигрыша»:

$$W_i = f_i(S, x_i).$$

5. Определить, как изменяется состояние S системы S под влиянием управление x_i на i-ом шаге: оно переходит в новое состояние

$$S' = \varphi_i(S, x_i). \quad (1.1)$$

6. Записать основное рекуррентное уравнение динамического программирования, выражающее условный оптимальный выигрыш $W_i(S)$ (начиная с i-го шага и до конца) через уже известную функцию $W_{i+1}(S)$:

$$W_i(S) = \max_{x_i}\{f_i(S, x_i) + W_{i+1}(\varphi_i(S, x_i))\}. \quad (1.2)$$

Этому выигрышу соответствует условное оптимальное управление на i-м шаге $x_i(S)$ (причем в уже известную функцию $W_{i+1}(S)$ надо вместо S подставить измененное состояние $S' = \varphi_i(S, x_i)$)

7. Произвести условную оптимизацию последнего (m-го) шага, задаваясь гаммой состояний S, из которых можно за один шаг дойти до конечного состояния, вычисляя для каждого из них условный оптимальный выигрыш по формуле $W_m(S) = \max_{x_{mi}}\{f_m(S, x_m)\}$

8. Произвести условную оптимизацию (m-1)-го, (m-2)-го и т.д. шагов по формуле (1.2), полагая в ней i=(m-1),(m-2),…, и для каждого из шагов указать условное оптимальное управление $x_i(S)$, при котором максимум достигается.

Заметим, что если состояние системы в начальный момент известно (а это обычно бывает так), то на первом шаге варьировать состояние системы не нужно - прямо находим оптимальный выигрыш для данного начального состояния S_0. Это и есть оптимальный выигрыш за всю операцию

$$W^* = W_1(S_0)$$

9. Произвести безусловную оптимизацию управления, «читая» соответствующие рекомендации на каждом шаге. Взять найденное оптимальное управление на первом шаге $x_1^* = x_1(S_0)$; изменить состояние системы по формуле (1.1); для вновь найденного состояния найти оптимальное управление на втором шаге x_2^* и т.д. до конца.

Данные этапы рассматривались для аддитивных задач, в которых выигрыш за всю операцию равен сумме выигрышей на отдельных шагах. Метод динамического программирования применим также и к задачам с так называемым «мультипликативным» критерием, имеющим вид произведения:

$$W = \prod_{i=1}^{m} w_i$$

(если только выигрыши w_i положительны). Эти задачи решаются точно так же, как задачи с аддитивным критерием, с той единственной разницей, что в основном уравнении (1.2) вместо знака «плюс» ставится знак «умножения»: $W_i(S) = \max_{x_i}\{f_i(S, x_i) * W_{i+1}(\varphi_i(S, x_i))\}$

1.2 Идеи метода динамического программирования

Мы отметили, что планируя многошаговый процесс, необходимо выбирать УВ на каждом шаге с учетом его будущих последствий на еще предстоящих шагах. Однако, из этого правила есть исключение. Среди всех шагов существует один, который может планироваться без "заглядыва-ния в будущее". Какой это шаг? Очевидно, последний — после него других шагов нет. Этот шаг, единственный из всех, можно

планировать так, чтобы он как таковой принес наибольшую выгоду. Спланировав оптимально этот последний шаг, можно к нему пристраивать предпоследний, к предпоследнему — предпредпоследний и т.д.

Поэтому процесс динамического программирования на 1-м этапе разворачивается от конца к началу, то есть раньше всех планируется последний,

N-й шаг. А как его спланировать, если мы не знаем, чем кончился предпоследний? Очевидно, нужно сделать все возможные предположения о том, чем кончился предпоследний, *(N* — 1)-й шаг, и для каждого из них найти такое управление, при котором выигрыш (доход) на последнем шаге был бы максимален. Решив эту задачу, мы найдем условно оптимальное управление (УОУ) на *N*-м шаге, т.е. управление, которое надо применить, если (*N* — 1)-й шаг закончился определенным образом.

Предположим, что эта процедура выполнена, то есть для каждого исхода

(N — 1)-го шага мы знаем УОУ на *N*-м шаге и соответствующий ему условно оптимальный выигрыш (УОВ). Теперь мы можем оптимизировать управление на предпоследнем, (*N* — 1)-м шаге. Сделаем все возможные предположения о том, чем кончился предпредпоследпий, то есть *(N* — 2)-й шаг, и для каждого из этих предположений найдем такое управление на *(N* — 1)-м шаге, чтобы выигрыш за последние два шага (из которых последний уже оптимизирован) был максимален. Далее оптимизируется управ чение на *(N* — 2)-м шаге, и т.д.

Одним словом, на каждом шаге ищется такое управление, которое обеспечивает оптимальное продолжение процесса относительно достиг-нутого в данный момент состояния. Этот принцип выбора управления , называется принципом оптимальности. Само управление, обеспечивающее оптимальное продолжение процесса относительно заданного состояния, называется УОУ на данном шаге.

Теперь предположим, что УОУ на каждом шаге нам известно: мы знаем, что делать дальше, в каком бы состоянии ни был процесс к началу каждого шага. Тогда мы можем найти уже не "условное", а дейсгвительно оптимальное управление на каждом шаге.

Действительно, пусть нам известно начальное состояние процесса. Теперь мы уже знаем, что делать на первом шаге: надо применить УОУ, найденное для первого шага и начального сосюяния. В результате этого управления после первого шага система перейдет в другое состояние; но для этого состояния мы знаем УОУ и г д. Таким образом, мы найдем оптимальное управление процессом, приводящее к максимально возможному выигрышу.

Таким образом, в процессе оптимизации управления методом динамического программирования многошаговый процесс "проходится" дважды:

— первый раз — от конца к началу, в результате чего находятся УОУ| на каждом шаге и оптимальный выигрыш (тоже условный) на всех шагах, начиная с данного и до конца процесса

— второй раз — от начала к концу, в результате чего находятся оптимальные управления на всех шагах процесса.

Можно сказать, что процедура построения оптимального управления методом динамического программирования распадается на две стадии: предварительную и окончательную. На предварительной стадии для каждого шага определяется УОУ, зависящее от состояния системы (достигнутого в результате предыдущих шагов), и условно оптимальный выигрыш на всех оставшихся шагах, начиная с данного, также зависящий от состояния. На окончательной стадии определяется (безусловное) оптимальное управление для каждого шага. Предварительная (условная) оптимизация производится по шагам в обратном порядке: от последнего шага к первому; окончательная (безусловная) оптимизация — также по шагам, но в естественном порядке: от первого шага к последнему. Из двух

стадий оптимизации несравненно более важной и трудоемкой является первая. После окончания первой стадии выполнение второй трудности не представляет: остается только "прочесть" рекомендации, уже заготовленные на первой стадии.

Динамическое программирование – это математический метод поиска оптимального управления, специально приспособленный к многошаговым процессам. Рассмотрим пример такого процесса.

Пусть планируется деятельность группы предприятий на *N* лет. Здесь шагом является один год. В начале 1-го года на развитие предприятий выделяются средства, которые должны быть как-то распределены между этими предприятиями. В процессе их функционирования выделенные средства частично расходуются. Каждое предприятие за год приносит некоторый доход, зависящий от вложенных средств. В начале года имеющиеся средства могут перераспределяться между предприятиями : каждому из них выделяется какая-то доля средств.

Ставится вопрос : как в начале каждого года распределять имеющиеся средства между предприятиями, чтобы суммарный доход от всех предприятий за *N* лет был максимальным?

Перед нами типичная задача динамического программирования, в которой рассматривается управляемый процесс – функционирование группы предприятий. Управление процессом состоит в распределении (и перераспределении) средств. Управляющим воздействием (УВ) является выделене каких-то средств каждому из предприятий в начале года.

УВ на каждом шаге должно выбираться с учетом всех его последствий в будущем. УВ должно быть дальновидным, с учетом перспективы. Нет смысла выбирать на рассматриваемом шаге наилучшее УВ, если в дальнейшем это помешает получить наилучшие результаты других шагов. УВ на каждом шаге надо выбирать “с заглядыванием в будущее”, иначе возможны серьезные ошибки.

Действительно, предположим, что в рассмотренной группе предприятий одни заняты выпуском предметов потребления, а другие производят для этого машины. Причем целью является получение за N лет максимального объема выпуска предметов потребления. Пусть планируются капиталовложения на первый год. Исходя их узких интересов данного шага (года), мы должны были бы все средства вложить в производство предметов потребления, пустить имеющиеся машины на полную мощность и добиться к концу года максимального объема продукции. Но правильным ли будет такое решение в целом? Очевидно, нет. Имея в виду будущее, необходимо выделить какую-то долю средств и на производство машин. При этом объем продукции за первый год, естественно, снизится, зато будут созданы условия, позволяющие увеличивать ее производство в последующие годы.

В формализме решения задач методом динамического программирования будут использоваться следующие обозначения:

N – число шагов.

$\bar{x}_k = (x_{1k}, x_{2k}, ..., x_{nk})$ – вектор,описывающий состояние системы на k-м шаге.

$\bar{x}_0$ – начальное состояние, т. е. состояние на 1-м шаге.

$\bar{x}_N$ – конечное состояние, т. е. состояние на последнем шаге.

X_k – область допустимых состояний на k-ом шаге.

$\bar{u} = (u_{1k}, u_{2k}, ..., u_{mk})$ – вектор УВ на k-ом шаге, обеспечивающий переход системы из состояния x_{k-1} в состояние x_k.

U_k – область допустимых УВ на k-ом шаге.

W_k – величина выигрыша, полученного в результате реализации k-го шага.

S – общий выигрыш за N шагов.

$\bar{u}^* = (\bar{u}_1^*, \bar{u}_2^*, ..., \bar{u}_N^*)$ – вектор оптимальной стратегии управления или ОУВ за N шагов.

$S_{k+1}(\bar{x}_k)$ – максимальный выигрыш, получаемый при переходе из любого состояния $\bar{x}_k$ в конечное состояние $\bar{x}_0$ при оптимальной стратегии управления начиная с (k+1)-го шага.

$S_1(\bar{x}_0)$ – максимальный выигрыш, получаемый за N шагов при переходе системы из начального состояния $\bar{x}_0$ в конечное $\bar{x}_N$ при реализации оптимальной стратегии управления $\bar{u}^*$. Очевидно, что S = $S_1(\bar{x}_0)$, если $\bar{x}_0$ –фиксировано.

Метод динамического программирования опирается на условие отсутствия последействия и условие аддитивности целевой функции.

Условие отсутствия последействия. Состояние $\bar{x}_k$, в которое перешла система за один k-й шаг, зависит от состояния $\bar{x}_{k-1}$ и выбранного УВ $\bar{u}_k$ и не зависит от того, каким образом система пришла в состояние $\bar{x}_{k-1}$, то есть

$$\bar{x}_k = \bar{f}_k(\bar{x}_{k-1}, \bar{u}_k).$$

Аналогично, величина выигрыша W_k зависит от состояния $\bar{x}_{k-1}$ и выбранного УВ $\bar{u}_k$, то есть

$$W_k = W_k(\bar{x}_{k-1}, \bar{u}_k).$$

Условие аддитивности целевой функции. Общий выигрыш за N шагов вычисляется по формуле

$$S = \sum_{k=1}^{N} W_k(\bar{x}_{k-1}, \bar{u}_k).$$

Определение. Оптимальной стратегией управления $\bar{u}^*$ называется совокупность УВ $\bar{u}_1^*, \bar{u}_2^*, ..., \bar{u}_N^*$, то есть $\bar{u}^* = (\bar{u}_1^*, \bar{u}_2^*, ..., \bar{u}_N^*)$, в результате реализации которых система за N шагов переходит из начального состояния $\bar{x}_0$ в конечное $\bar{x}_N$ и при этом общий выигрыш S принимает наибольшее значение.

Условие отсутствия последействия позволяет сформулировать принцип оптимальности Белмана.

Принцип оптимальности. Каково бы ни было допустимое состояние системы $\bar{x}_{i-1} \in X_{i-1}$ перед очередным *i*-м шагом, надо выбрать допустимое УВ $\bar{u}_i \in U_i$ на этом шаге так, чтобы выигрыш W_i на *i*-м шаге плюс оптимальный выигрыш на всех последующих шагах был максимальным.

В качестве примера постановки задачи оптимального управления продолжим рассмотрение задачи управления финансированием группы предприятий. Пусть в начале i-го года группе предприятий $П_1, П_2, ..., П_m$ выделяются соответственно средства: $u_{1i}, u_{2i}, ..., u_{mi}$. совокупность этих значений можно считать управлением на i-м шаге, то есть $\bar{u}_i = (u_{1i}, u_{2i}, ..., u_{mi})$. Управление $\bar{u}$ процессом в целом представляет собой совокупность всех шаговых управлений, то есть $\bar{u} = (\bar{u}_1, \bar{u}_2, ..., \bar{u}_N)$.

Управление может быть хорошим или плохим, эффективным или неэффективным. Эффективность управления $\bar{u}$ оценивается показателем *S*. Возникает вопрос: как выбрать шаговые управления $\bar{u}_1, \bar{u}_2, ..., \bar{u}_N$, чтобы величина *S* обратилась в максимум ?

Поставленная задача является задачей оптимального управления, а управление, при котором показатель *S* достигает максимума, называется оптимальным. Оптимальное управление $\bar{u}^*$ многошаговым процессом состоит из совокупности оптимальных шаговых управлений:

$$\bar{u}^* = (\bar{u}_1^*, \bar{u}_2^*, ..., \bar{u}_N^*)$$

Таким образом, перед нами стоит задача: определить оптимальное управление на каждом шаге $\bar{u}_i^*$ (i=1,2,...N) и, значит, оптимальное управление всем процессом $\bar{u}^*$.

1.3 Примеры задач динамического программирования

Задача планирования рабочей силы:

При выполнении некоторых проектов число рабочих, необходимых для выполнения какого-либо проекта, регулируется путем их найма и увольнения. Поскольку как наем, так и увольнение рабочих связано с дополнительными затратами, необходимо определить, каким образом должна регулироваться численность рабочих в период реализации проекта.

Предположим, что проект будет выполнятся в течение n недель и минимальная потребность в рабочей силе на протяжении i-й недели составит b_i рабочих. При идеальных условиях хотелось бы на протяжении i-й недели иметь в точности b_i рабочих. Однако в зависимости от стоимостных показателей может быть более выгодным отклонение численности рабочей силы как в одну, так и в другую сторону от минимальных потребностей.

Если x_i – количество работающих на протяжении i-й недели, то возможны затраты двух видов: 1) $C_1(x_i - b_i)$-затраты, связанные с необходимостью содержать избыток $x_i - b_i$ рабочей силы и 2) $C_2(x_i - x_{i-1})$-затраты, связанные с необходимостью дополнительного найма $(x_i - x_{i-1})$ рабочих.

Элементы модели динамического программирования определяются следующим образом:

1. Этап i представляется порядковым номером недели i, i=1,2,…n.
2. Вариантами решения на i-ом этапе являются значения x_i – количество работающих на протяжении i-й недели.
3. Состоянием на i-м этапе является x_{i-1} – количество работающих на протяжении (i-1) –й недели (этапа).

Рекуррентное уравнение динамического программирования представляется в виде

$$f_i(x_{i-1}) = \min_{x_i \geq b_i} \{C_1(x_i - b_i) + C_2(x_i - x_{i-1}) + f_{i+1}(x_i)\}, i = 1,2,...,n$$

где $f_{n+1}(x_n) = 0$

Вычисления начинаются с этапа n при $x_n=b_n$ и заканчиваются на этапе 1.

Задача замены оборудования:

Чем дольше механизм эксплуатируется, тем выше затраты на его обслуживание и ниже его производительность. Когда срок эксплуатации механизма достигает определенного уровня, может оказаться более выгодной его замена. Задача замены оборудования, таким образом, сводится к определению оптимального срока эксплуатации механизма.

Предположим, что мы занимаемся заменой механизмов на протяжении n лет. В начале каждого года принимается решение либо об эксплуатации механизма еще один год, либо о замене его новым.

Обозначим через r(t) и c(t) прибыль от эксплуатации t-летнего механизма на протяжении года и затраты на его обслуживание за этот же период. Далее пусть s(t) – стоимость продажи механизма, который эксплуатировался t лет. Стоимость приобретения нового механизма остается неизменной на протяжении всех лет и равна l.

Элементы модели динамического программирования таковы:

1. Этап i представляется порядковым номером года i, i=1,2,...n.
2. Вариантами решения на i-м этапе (т.е. для i-ого года) являются альтернативы: продолжить эксплуатацию или заменить механизм в начале i-ого года.
3. Состоянием на i-м этапе является срок эксплуатации t (возраст) механизма к началу i-ого года.

Пусть $f_i(t)$-максимальная прибыль, получаемая за годы от i до n при условии, что в начале i-ого года имеется механизм t-летнего возраста.

Рекуррентное уравнение имеет следующий вид:

$$f_i(t)=\begin{cases} r(t)-c(t)+f_{i+1}(t+1)(1) \\ r(0)+s(t)-I-c(0)+f_{i+1}(l)(2) \end{cases}$$

(1)-если эксплуатировать механизм,

(2)-если заменить механизм.

Задача инвестирования:

Предположим, что в начале каждого из следующих n лет необходимо сделать инвестиции P_1, P_2,…, P_n соответственно. Вы имеете возможность вложить капитал в два банка: первый банк выплачивает годовой сложный процент r_1, а второй - r_2. Для поощрения депозитов оба банка выплачивают новым инвесторам премии в виде процента от вложенной суммы.

Премиальные меняются от года к году, и для i-ого года равны q_{i1} и q_{i2} в первом и втором банках соответственно. Они выплачиваются к концу года, на протяжении которого сделан вклад, и могут быть инвестированы в один из двух банков на следующий год. Это значит, что лишь указанные проценты и новые деньги могут быть инвестированы в один из двух банков. Размещенный в банке вклад должен находится там до конца рассматриваемого периода. Необходимо разработать стратегию инвестиции на следующие n лет.

Элементы модели динамического программирования следующие:

1. Этап i представляется порядковым номером года i, i=1,2,...n
2. Вариантами решения на i-м этапе (для i-ого года) являются суммы l_i и $\bar{l}_i$ инвестиций в первый и второй банк соответственно.

3. Состоянием x_i на i-м этапе является сумма денег на начало i-ого года, которые могут быть инветсированы.

Заметим, что по определению $\bar{l}_i = x_i - l_i$. Следовательно,

$$x_i = P_i + q_{i-1} l_{i-1} + q_{i-1,2}(x_{i-1} - l_{i-1}) = P_i + (q_{i-1,i} - q_{i-1,2}) l_{i-1} + q_{i-1,2} x_{i-1}$$

где i=2,3,…n, $x_1=P_1$. Сумма денег x_i, которые могут быть инвестированы, включает лишь новые деньги и премиальные проценты за инвестиции, сделанные на протяжении (i-1)-го года.

Пусть $f_i(x_i)$- оптимальная сумма инвестиций для интервала от i-го до n-го года при условии, что в начале i-го года имеется денежная сумма x_i. Далее обозначим через s_i накопленную сумму к концу n-го года при условии, что l_i и (x_i-l_i)-объемы инвестиций на протяжении i-го года в первый и второй банк соответственно. Обозначая $\alpha_i = (1+r_i)$, i=1,2, мы можем сформулировать задачу в следующем виде.

Максимизировать $z=s_1+s_2+\ldots+s_n$, где

$$s_i = l_i\alpha_1^{n+1-i} + (x_i - l_i)\alpha_2^{n+1-i} = (\alpha_1^{n+1-i} - \alpha_2^{n+1-i})l_i + \alpha_2^{n+1-i}x_i, i = 1,2,...,n-1$$
$$s_n = (\alpha_1 + q_{n1} - \alpha_2 - q_{n2})l_n + (\alpha_2 + q_{n2})x$$

Так как премиальные за n-й год являются частью накопленной денежной суммы от инвестиций, в выражения для s_n добавлены q_{n1} и q_{n2}.

Итак, в данном случае рекуррентное уравнение для обратной прогонки в алгоритме динамического программирования имеет вид

$$f_i(x_i) = \max_{0\le l_i\le x_i}\{s_i + f_{i+1}(x_{i+1})\ \}, i = 1,2,...,n-1$$

где x_{i+1} выражается через x_i в соответствии с приведенной выше формулой, а $f_{n+1}(x_{n+1})=0$.

Задача о загрузке

Задача о загрузке – это задача о рациональной загрузке судна (самолета, автомашины и т.п.), которое имеет ограничения по объему или грузоподъемности. Каждый помещенный на судно груз приносит определенную прибыль. Задача состоит в определении загрузки судна такими грузами, которые приносят наибольшую суммарную прибыль.

Рекуррентное уравнение процедуры обратной прогонки выводится для общей задачи загрузки судна грузоподъемностью W предметов (грузов) n наименований. Пусть m_i-количество предметов i-го наименования, подлежащих загрузке, r_i-прибыль, которую приносит один загруженный предмет i-го наименования, w_i-вес одного предмета i-го

наименования. Общая задача имеет вид следующей целочисленной задачи линейного программирования.

Максимизировать $z=r_1m_1+r_2m_2+\ldots+r_nm_n$.

при условии, что

$w_1m_1+w_2m_2+\ldots+w_nm_n \le W$,

$m_1,m_2,\ldots,m_n \ge 0$ и целые.

Три элемента модели динамического программирования определяются следующим образом:

1. Этап i ставится в соответствии предмету i-го наименования, i=1,2,…n.

2. Варианты решения на этапе i описываются количеством m_i предметов i-го наименования, подлежащих загрузке. Соответствующая прибыль равна r_im_i. Значение m_i заключено в пределах от 0 до $[W/w_i]$, где $[W/w_i]$ – целая часть числа W/w_i.

3. Состояние x_i на этапе i выражает суммарный вес предметов, решения о погрузке которых приняты на этапах i,i+1,...n. Это определение отражает тот факт, что ограничения по весу является единственным, которое связывает n этапов вместе.

Пусть $f_i(x_i)$-максимальная суммарная прибыль от этапов i,i+1,...,n при заданном состоянии x_i. Проще всего рекуррентное уравнение определяется с помощью следующей двухшаговой процедуры.

Шаг 1. Выразим $f_i(x_i)$ как функцию $f_{i+1}(x_{i+1})$ в виде

$$f_i(x_i) = \max_{\substack{m_i=0,1,\ldots,[W/w_i] \\ x_i=0,1,\ldots,W}} \{r_i m_i + f_{i+1}(x_{i+1})\}, i = 1,2,\ldots,n$$

где $f_{n+1}(x_{n+1})=0$.

Шаг 2. Выразим x_{i+1} как функцию x_i для гарантии того, что левая часть последнего уравнения является функцией лишь x_i. По определению x_i-x_{i+1} представляет собой вес, загруженный на этапе i, т.е. $x_i-x_{i+1}=w_im_i$ или $x_{i+1}=x_i-w_im_i$. Следовательно, рекуррентное уравнение приобретает следующий вид:

$$f_i(x_i) = \max_{\substack{m_i=0,1,...,[W/w_i] \\ x_i=0,1,...,W}} \{r_i m_i + f_{i+1}(x_i - w_i m_i)\}, i = 1,2,...,n$$

1.4 Рекуррентные соотношения для процедур прямой и обратной прогонки

Фермеру принадлежит стадо овец, насчитывающее *k* голов. Один раз в год фермер принимает решение о том, сколько овец продать и сколько оставить. Прибыль от продажи одной овцы в i-м году составляет p_i. Количество оставленных в i-м году овец удваивается в (1+1)-м году. По истечении *n* лет фермер намеревается продать все стадо.

Этот чрезвычайно простой пример приводится для того, чтобы наглядно продемонстрировать преимущества алгоритма обратной прогонки по сравнению с алгоритмом прямой прогонки. Вычислительные схемы процедур прямой и обратной прогонки обладают различной эффективностью в случаях, когда этапы модели нумеруются в некотором специальном порядке. Такая ситуация имеет место в приводимом примере, где этап j ставится в соответствие году j, т. е. этапы должны рассматриваться в хронологическом порядке.

Сначала построим рекуррентные соотношения для процедур прямой и обратной прогонки, а затем проведем сравнение двух вычислительных схем. Важное различие между двумя формулировками непосредственно следует из определения состояния.

Обозначим количества оставленных и проданных в *j-м* году овец через x_j и y_j, соответственно. Положим $Zj,=x_j+y_j$. Из условий задачи следует, что

$z_1=2x_0=2k,$

$z_j=2x_{j-1}, j=1,2, ...,n.$

Состояние на этапе j можно описать с помощью переменной z_j, которая выражает количество имеющихся к концу этапа j овец для

распределения на этапах j+1, j+2, ..., n, или с помощью переменной x_j, которая выражает количество имеющихся к началу этапа j+1 овец, обусловленное принятыми на этапах 1,2,...,j решениями. Первое определение ориентировано на построение рекуррентного соотношения для процедуры обратной прогонки, тогда как второе определение приводит к использованию алгоритма прямой прогонки.

Алгоритм обратной прогонки

Обозначим через $f_i(z_i)$ максимальную прибыль, получаемую на этапах j,j+1,…,n, при заданном z_j. Рекуррентное соотношение имеет следующий вид:

$$f_n(z_n) = \max_{y_n = z_n \le 2^n k} \{p_n y_n\},$$
$$f_j(z_j) = \max_{y_j \le z_j \le 2^j k} \{p_j y_j + f_{j+1}(2[z_j - y_i])\}, j = 1,2,...,n-1$$

Заметим, что y_j и z_j - неотрицательные целые числа. Кроме того, y_j (количество овец, проданных в конце периода j) должно быть меньше или равно z_j. Верхней границей для значений z_j, является величина $2^j k$ (где k-исходный размер стада), которая соответствует отсутствию продажи.

<u>Алгоритм прямой прогонки</u>

Обозначим через $g_j(x_j)$ максимальную прибыль, получаемую на этапах 1,2,...,j при заданном x_j, (где x_j— размер стада к началу этапа J+1). Рекуррентное соотношение записывается в следующем виде:

$$g_1(x_1) = \max_{y_1 = 2k - x_1} \{p_1 y_1\},$$
$$g_j(x_j) = \max_{y_j \le 2^j k - x_j} \left\{ p_j y_j + g_{j-1}\left(\frac{y_j + x_j}{2}\right) \right\}, j = 2,3,...,n,$$

$\left(\frac{x_j + y_j}{2}\right)$ - целое.

Сравнение двух формулировок показывает, что представление x_{j-1} через x_j создает более существенные препятствия для вычислений, чем представление z_{j+1} через z_j.

В замене $x_{j-1}=(x_j+y_j)/2$ подразумевается целочисленность правой части, тогда как на равенство $z_{j+1}=2(z_j-y_j)$ такое требование не накладывается. Таким образом в случае процедуры прямой прогонки значения y_j и x_j, связанные неравенством

$$Y_j <= 2^j k - X_j,$$

должны дополнительно удовлетворять условию целочисленности их полусуммы, связанному с видом зависимости x_{j-1} от x_j,. Рассмотренный пример иллюстрирует трудности вычислительного характера, которые обычно возникают при использовании алгоритма прямой прогонки.

1.5 Общая структура динамического программирования

Отыскание оптимальной стратегии принятия набора последовательных решений, в большинстве случаях, производится следующим образом: сначала осуществляется выбор последнего во времени решения, затем при движении в направлении, обратном течению времени, выбираются все остальные решения вплоть до исходного.

Для реализации такого метода необходимо выяснить все ситуации, в которых может происходить выбор последнего решения. Обычно условия, в которых принимается решение, называют «состоянием» системы. Состояние системы – это описание системы, позволяющее, учитывая будущие решения, предсказать ее поведение. Нет необходимости выяснять, как возникло то ил иное состояние или каковы были предшествующие решения. Это позволяет последовательно выбирать всего по одному решению в каждый момент времени. Независимо от того, отыскивают оптимальные решения с помощью табличного метода и последующего поиска или аналитическим путем, обычно быстрее и выгоднее производить выбор по одному решению в один момент времени, переходя затем к следующему моменту и т.д. К сожалению, таким методом можно исследовать не все процессы принятия решений. Необходимым

условием применения метода динамического программирования является аддитивность цен всех решений, а также независимость будущих результатов от предыстории того или иного состояния.

Если число решений очень велико, то можно построить относительные оценки состояний так, чтобы оценки, отвечающие каждой паре последовательных решений, отличались друг от друга на постоянную величину, представляющую собой средний «доход» на решение. Также можно выполнять дисконтирование доходов от будущих решений. Необходимость в этом иногда появляется в том случае, когда решение принимаются редко, скажем раз в году. Тогда уже не нужно рассматривать последовательно 1,2,3…решения, чтобы достичь решения с большим номером. Вместо этого можно непосредственно оперировать функциональным уравнением, что, как правило, дает существенную выгоду с точки зрения сокращения объема вычислений.

1.6. Составление математической модели динамического программирования

Дополнительно введем следующие условные обозначения:

s – состояние процесса;

Si – множество возможных состояний процесса перед i-м шагом;

W – выигрыш с i-го шага до конца процесса, $i = \overline{1,m}$.

Можно определить следующие основные этапы составления математической модели задачи динамического программирования.

1. Разбиение задачи на шаги (этапы). Шаг не должен быть слишком мелким, чтобы не проводить лишних расчетов и не должен быть слишком большим, усложняющим процесс шаговой оптимизации.

2. Выбор переменных, характеризующих состояние s моделируемого процесса перед каждым шагом, и выявление налагаемых на них ограничений. В качестве таких переменных следует брать факторы,

представляющие интерес для исследователя, например годовую прибыль при планировании деятельности предприятия.

3. Определение множества шаговых управлений xi, $i = \overline{1,m}$ и налагаемых на них ограничений, т. е. области допустимых управлений X.

4. Определение выигрыша

$$\varphi_i(s, x_i) \qquad (14.3)$$

который принесет на i-м шаге управление xi, если система перед этим находилась в состоянии s'.

5. Определение состояния s', в которое переходит система из состояния s под влиянием управления xi,

$$s' = f_i(s, x_i), \qquad (14.4)$$

где f_i – функция перехода на i-м шаге из состояния s в состояние s' .

6. Составление уравнения, определяющего условный оптимальный выигрыш на последнем шаге, для состояния s моделируемого процесса

$$W_m(S) = \max_{x_m \in X} \{\varphi_m(s, x_m)\} \qquad (14.5)$$

7. Составление основного функционального уравнения динамического программирования, определяющего условный оптимальный выигрыш для данного состояния s с i-го шага и до конца процесса через уже известный условный оптимальный выигрыш с (i+1)-го шага и до конца:

$$W_i(S) = \max_{x_m \in X} \{\varphi_i(s, x_i) + W_{i+1}(f_i(s, x_i))\}. \qquad (14.6)$$

В уравнении (14.6) в уже известную функцию Wi+1(s), характеризующую условный оптимальный выигрыш с (i+1)-го шага до конца процесса, вместо состояния s подставлено новое состояние $s' = f_i(s, x_i)$, в которое система переходит на i-м шаге под влиянием управления xi.

Заметим, что структура модели динамического программирования отличается от статической модели линейного программирования. Действительно, в моделях линейного программирования управляющие переменные – это одновременно и переменные состояния моделируемого процесса, а в динамических моделях отдельно вводятся переменные управления xi,, и переменные, характеризующие изменение состояния s под влиянием управления. Таким образом, структура динамических моделей более сложная, что естественно, так как в этих моделях дополнительно учитывается фактор времени.

1.7. Этапы решения задачи динамического программирования

После того как выполнены пункты 1–7, изложенные в предыдущем параграфе, и математическая модель составлена, приступают к ее расчету. Укажем основные этапы решения задачи динамического программирования.

1. Определение множества возможных состояний Sm для последнего шага.

2. Проведение условной оптимизации для каждого состояния $s \in S_m$ на последнем m-м шаге по формуле (14.5) и определение условного оптимального управления x(s), $s \in S_m$.

3. Определение множества возможных состояний Si для i-го шага, $i = 2,3,...,m-1$.

4. Проведение условной оптимизации i-го шага, $i = 2,3,...,m-1$ для каждого состояния $s \in S_i$ по формуле (14.6) и определение условного оптимального управления xi(s), $s \in S_m$, $i = 2,3,...,m-1$.

5. Определение начального состояния системы s1, оптимального выигрыша W1(S1) и оптимального управления x1(S1) по формуле (14.6) при i=1. Это есть оптимальный выигрыш для всей задачи $W^* = W_1(x_1^*)$.

6. Проведение безусловной оптимизации управления. Для проведения безусловной оптимизации необходимо найденное на первом шаге оптимальное управление $x_1^* = x_1(s_1)$ подставить в формулу (14.4) и определить следующее состояние системы $s_2 = f_2(s_1, x_1^*)$. Для измененного состояния найти оптимальное управление $x_2^* = x_2(s_2)$, подставить в формулу (14.4.) и т. д. Для i-го состояния si найти $s_{i+1} = f_{i+1}(s_i, x_i^*)$ и $x_{i+1}^*(s_{i+1})$ и т. д.