Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Алтайский государственный технический университет
им. И. И. Ползунова»

А. С. Киркинский

ЭЛЕМЕНТЫ КОМБИНАТОРНОГО АНАЛИЗА

Конспект лекций

Изд-во АлтГТУ
Барнаул • 2014

УДК 519.1

Киркинский, А. С. Элементы комбинаторного анализа: Конспект лекций / А. С. Киркинский. – Барнаул : Изд-во АлтГТУ, 2014. – 54 с.

Конспект лекций соответствует программе дисциплины «Элементы комбинаторного анализа», включённой в рабочий учебный план подготовки бакалавров по направлению «Программная инженерия». Содержит краткое повторение начальных понятий комбинаторики, сведения о рекуррентных уравнениях и производящих функциях. Примерно половина курса посвящена теории графов. Рассматриваются начальные понятия, маршруты в графах, деревья, плоские графы, раскраски графов. Значительное место в курсе занимают алгоритмические задачи на графах.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В приложениях разных наук встречаются задачи, связанные с построением различных комбинаций некоторых объектов. Могут интересовать существование комбинаций определённого вида, их количество, алгоритмы нахождения всех таких комбинаций. При этом задачи из самых разных предметных областей обнаруживают сходство – и по своей постановке, и по методам решения. Почти всегда необходимость решать такие **комбинаторные** задачи возникает при построении алгоритмов для применения ЭВМ.

Комбинаторный анализ (или **комбинаторика**) – раздел математики, посвящённый решению задач выбора и расположения элементов некоторого конечного (или счётного) множества, в соответствии с заданными ограничениями. Можно сказать, что комбинаторика изучает некоторые конструкции из элементов множества, так называемые ***комбинаторные конфигурации***. Наиболее известные примеры таких конфигураций – размещения, перестановки, сочетания. Изучение комбинаторных конфигураций включает:

- вопросы существования определённых конфигураций;
- алгоритмы построения конфигураций, их оптимизацию;
- перечислительные задачи, в частности, определение числа конфигураций данного класса.

Уже учёным Древнего Востока была известна формула бинома Ньютона для натуральных показателей. Знали они и формулу для числа сочетаний (через биномиальные коэффициенты). С мистическими целями изучались магические квадраты.

В XVII веке во Франции Б.Паскаль, П. Ферма изучали теорию азартных игр, тогда и возникла комбинаторика, как раздел математики, теория вероятностей, устанавливались их связи. Работы Г.Лейбница, Я.Бернулли привели к выделению комбинаторики в самостоятельную часть математики. Название этому разделу математики дал немецкий философ и математик Готфрид Вильгельм Лейбниц (1646–1716), опубликовавший в 1666 г. свою работу «Об искусстве комбинаторики». Важные работы Л.Эйлера (по разбиениям и композициям натуральных чисел на слагаемые) привели к методу производящих функций.

Бурное развитие комбинаторика получила в 50-х годах XX века, в связи с развитием кибернетики, дискретной математики. Активизировался интерес к классическим комбинаторным задачам. Укажем некоторые из них.

1. Построение ***магических квадратов***: расположение чисел $\mathbf{1}, \mathbf{2}, \ldots, \mathbf{n^2}$ в квадрате $\mathbf{n \times n}$ так, чтобы суммы по строкам, столбцам, диагоналям были одинаковы. Например:

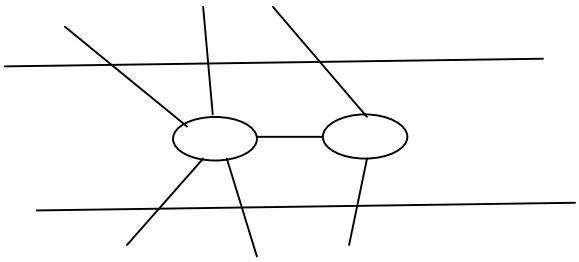
$$\begin{matrix} 4 & 9 & 2 \\ 3 & 5 & 7 \\ 8 & 1 & 6 \end{matrix}$$

Известны некоторые методы построения. Количество магических квадратов при $\mathbf{n>4}$ – нерешённая задача.

2. Построение ***латинских квадратов***. Это квадрат $\mathbf{n \times n}$, содержащий числа $\mathbf{1}, \mathbf{2}, \ldots, \mathbf{n}$, причём числа в каждой строке и каждом столбце различны. Число латинских квадратов порядка $\mathbf{n}$ известно до $\mathbf{n = 9}$ включительно.

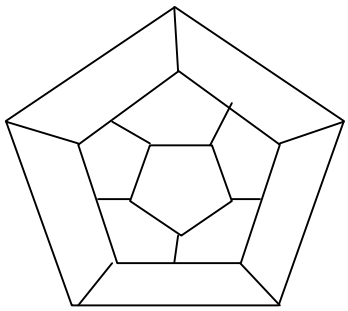
Два латинских квадрата называются ортогональными, если при наложении все $\mathbf{n^2}$ пар элементов различны. В 1959-60 годах доказано: для каждого $\mathbf{n = 4k+2}$, $\mathbf{k = 2, 3}, \ldots$ существует пара ортогональных латинских квадратов, а вот для $\mathbf{n = 6}$ такой пары нет (1900).

3 Задача Л.Эйлера ***о кёнигсбергских мостах***.



Семь мостов соединяют берега реки, протекающей через город, и два острова. Можно ли обойти все мосты, проходя по каждому только один раз, и вернуться в исходную точку? Задача неразрешима, даже если не требовать возвращения в точку выхода. Л.Эйлер обобщил эту задачу, получил критерии существования таких (***эйлеровых***) маршрутов в произвольных графах.

4. ***У. Гамильтон*** в 1859 году придумал игру «Кругосветное путешествие». Требовалось найти в графе



маршрут, проходящий через все вершины по 1 разу и возвращающий в исходную точку. Критерии существования таких (***гамильтоновых***) маршрутов пока неизвестны. Обобщением задачи о гамильтоновых циклах является ***задача коммивояжёра***, имеющая много приложений. Она состоит в следующем. Имеется несколько городов, расстояния между которыми известны. Нужно найти кратчайший путь, проходящий через все города и возвращающийся в исходный.

5. ***Задача о встречах***. Есть 2 одинаковые колоды по **n** карт в каждой. Сколько способов расположить карты во второй колоде так, чтобы при сравнении с первой колодой было ровно **k** совпадений? Эта задача стала исходным пунктом для целого раздела комбинаторики, связанного с исследованием перестановок с различными ограничениями.

6. ***Задача о супружеских парах***. За круглым столом есть **2n** мест. Сколько способов рассадить **n** супружеских пар так, чтобы ни один муж не сидел рядом со своей женой?

7. ***Задача о числе разбиений натурального числа на слагаемые***. Впервые появилась в письме Г.Лейбница к Я.Бернулли в 1669 году. Но разработка общих методов была проведена Л.Эйлером.

8. ***Задача о размещениях элементов по ячейкам***. Классическая задача такого типа – найти число способов разместить **m** различных элементов по **n** различным ячейкам, если **r** ячеек остаются пустыми. Задача эта решена, однако мы будем заниматься более простыми задачами такого типа.

1 Классические задачи комбинаторики

1.1 Важнейшие комбинаторные конфигурации

1.1.1 Правило произведения

Повторим основные правила и формулы элементарной комбинаторики, изученные в курсе дискретной математики.

Наиболее простой комбинаторной операцией является выбор элемента конечного множества. Более сложные операции – упорядочение элементов множества, составление различного рода сочетаний и размещений – основаны, как правило, на простейшей операции выбора. С этим и связана ключевая роль, которую играет в перечислительных задачах так называемое ***правило произведения***:

если комбинаторная операция α может быть выполнена m способами, а другая операция β – n способами (независимо от способа выполнения α) , то сложную операцию «сначала α, потом β » можно выполнить mn способами.

Пример 1. Сколько существует 5-значных чисел, которые делятся на 5?

Решение. Сложную комбинаторную операцию «составление 5-значного числа, кратного 5» представим в виде последовательного выполнения простых операций. Первую слева цифру (старший разряд) можно выбрать 9 способами (любая цифра, кроме 0). Вторая цифра может быть любой – 10 способов для её выбора, причём независимо от того, какая цифра поставлена первой. По правилу произведения, есть 90 вариантов для выбора первых двух цифр. Затем нужно выбрать 3-ю цифру (10 способов), 4-ю цифру (10 способов), 5-ю цифру, младший разряд – для неё всего 2 способа, 0 или 5, так как число должно делиться на 5. Применяем правило произведения:

$$9 \cdot 10 \cdot 10 \cdot 10 \cdot 2 = 18000 .$$

1.1.2 Сочетания, размещения, перестановки

Для выполнения комбинаторных операций, более сложных, чем выбор одного элемента конечного множества, ключевым является понятие выборки. Рассмотрим конечное множество X из n элементов (n-множество). Неупорядоченной выборкой объёма r, или ***r-сочетанием***, называется любое подмножество множества X, содержащее r элементов. Упорядоченной выборкой объёма r, или ***r-размещением***, называется любой упорядоченный набор r элементов множества X. И ***r-сочетания***, и ***r-размещения*** могут включать каждый элемент множества X не более чем по одному разу. Будем использовать стандартные обозначения:

C_n^r – число r-сочетаний элементов n-множества;

A_n^r – число r-размещений элементов n-множества.

Важен частный случай: если в размещении участвуют ***все*** элементы множества X, то такое размещение называется ***перестановкой***. Будем обозначать

P_n – число ***перестановок*** n-множества.

Таким образом, $P_n = A_n^n$. Для числа C_n^r иногда применяется обозначение $\binom{n}{r}$.

Во многих задачах приходится рассматривать выборки, которые могут содержать несколько экземпляров некоторых элементов основного множества. Такие выборки называются выборками ***с повторениями***. Зафиксируем обозначения:

$\overline{C}_n^r$ – число r-сочетаний с повторениями элементов n-множества;

$\overline{A}_n^r$ – число r-размещений с повторениями элементов n-множества.

Пример 2. Пусть $X = \{a, b, c, d, e\}$. Приведём примеры различных выборок.

$\{a,b,e\}$, $\{b,c,e\}$, $\{a,c,e\}$, $\{a,c,d\}$ – примеры 3-сочетаний; фигурные скобки употребляются тогда, когда порядок элементов не играет роли, выборки неупорядоченные;

(a,b,e), (b,e,a), (a,c,e), (d,c,a) – примеры 3-размещений; круглые скобки применяются, когда порядок элементов важен, выборки упорядоченные;

$(a,c,e\,b,d),(c,d,e\,a,b)$ – примеры перестановок множества X;

$\{a,a,e\}$, $\{e,e,e\}$, $\{a,d,e\}$, $\{a,c,d\}$ – примеры 3-сочетаний с повторениями;

(a,b,e), (b,b,a), (b,a,b), (d,c,d) – примеры 3-размещений с повторениями.

С помощью правила произведения нетрудно получить формулы для вычисления C_n^r, A_n^r, P_n, $\overline{C}_n^r$, $\overline{A}_n^r$. Начнём с A_n^r. Чтобы выбрать r-размещение n-множества сначала нужно выбрать элемент, который в размещении будет на первом месте – для этого есть n вариантов. Независимо от того, какой элемент выбран, для выбора второго элемента есть $n-1$ вариант (сейчас мы рассматриваем размещения без повторений). Значит, по правилу произведения, для заполнения первых двух мест в размещении имеется $n(n-1)$ способов. Таким же образом рассуждаем и далее: для выбора 3-го элемента есть $n-2$ варианта; наконец, для выбора последнего элемента (всего в размещении участвуют r элементов) есть $n-r+1$ вариант. Применяем правило произведения:

$$A_n^r = n(n-1)(n-2)\ldots(n-r+1).$$

Применяя операцию «факториал» (напомним: $n! = 1\cdot 2\cdot 3\cdot\ldots\cdot n$), полученную формулу можно записать так:

$$A_n^r = \frac{n!}{(n-r)!}.$$

Так как, по определению, $0! = 1$, то для числа перестановок n-множества получаем:

$$P_n = n!.$$

Эту формулу легко получить и непосредственно, применяя правило произведения.
Для размещений с повторениями те же самые рассуждения приводят к формуле:

$$\overline{A}_n^r = n^r.$$

Чтобы вывести формулу для числа сочетаний, заметим, что из каждого r-сочетания, переставляя его элементы, можно получить $r!$ размещений. Поэтому количество r-размещений в $r!$ раз больше, чем количество r-сочетаний: $A_n^r = r!\cdot C_n^r$. Отсюда получаем очень важную формулу:

$$C_n^r = \frac{n!}{r!(n-r)!}.$$

Немного сложнее выводится формула для числа сочетаний с повторениями. Поэтому оформим этот вывод в виде теоремы.

Теорема 1. Число r-сочетаний с повторениями из элементов n-множества равно

$$\overline{C}_n^r = C_{n+r-1}^r.$$

Доказательство. Рассмотрим некоторое r-сочетание с повторениями из элементов множества $X = \{x_1, x_2, \ldots, x_n\}$. Пусть элемент x_1 в нём участвует r_1 раз, элемент x_2 входит r_2 раз, …, элемент x_n входит r_n раз. Такое сочетание можно записать в виде $\{x_1^{r_1}, x_2^{r_2}, \ldots, x_n^{r_n}\}$, где $r_1 + r_2 + \ldots + r_n = r$. Сопоставим этому сочетанию последовательность из r единиц и $n-1$ нулей, построенную следующим образом:

$$\underbrace{11\ldots1}_{r_1}0\,\underbrace{11\ldots1}_{r_2}0\ldots\ldots0\,\underbrace{11\ldots1}_{r_n}.$$

Нетрудно заметить, что построенное отображения множества сочетаний с повторениями во множество последовательностей указанного вида является взаимно однозначным (биективным). Число таких последовательностей легко подсчитать – оно равно C_{n+r-1}^{n-1} – числу способов выбрать $n-1$ место для нулей в последовательности из $r+n-1$ цифры. Значит, таким же является и число сочетаний с повторениями: $\bar{C}_n^r = C_{n+r-1}^{n-1} = C_{n+r-1}^r$. Теорема доказана.

Числа C_n^r встречаются при решении многих комбинаторных задач. Они называются ***биномиальными коэффициентами*** – в связи с тем, что появляются при возведении в степень бинома $(a+b)$:

$$(a+b)^n = C_n^0 a^n + C_n^1 a^{n-1} b + C_n^2 a^{n-2} b^2 + \ldots + C_n^n b^n = \sum_{r=0}^{n} C_n^r a^{n-r} b^r \ .$$

1.1.3 Упорядоченные разбиения множеств

Рассмотрим n-множество X и упорядоченный набор целых неотрицательных чисел $(r_1, r_2, \ldots, r_k)$, таких, что $r_1 + r_2 + \ldots + r_k = n$. Упорядоченным $(r_1, r_2, \ldots, r_k)$-разбиением множества X называется упорядоченный набор подмножеств $(X_1, X_2, \ldots, X_k)$ такой, что

1) $X_1 \cup X_2 \cup \ldots \cup X_k = X$; 2) $X_i \cap X_j = \varnothing$ при $i \neq j$; 3) $|X_i| = r_i$.

Напомним, что $|X_i|$ – число элементов (мощность) множества X_i. Число упорядоченных $(r_1, r_2, \ldots, r_k)$-разбиений n-множества обозначим $C_n(r_1, r_2, \ldots, r_k)$.

Теорема 2. $C_n(r_1, r_2, \ldots, r_k) = \dfrac{n!}{r_1! \cdot r_2! \cdot \ldots \cdot r_k!}$.

Доказательство. Выберем сначала r_1 элементов для включения в подмножество X_1, для этого имеется $C_n^{r_1}$ способов. Затем из оставшихся выберем r_2 элементов для включения в подмножество X_2, для этого имеется $C_{n-r_1}^{r_2}$ способов. Продолжая, найдём число способов выбора элементов для каждого из множеств X_i. Затем применим правило произведения:

$$C_n(r_1, r_2, \ldots, r_k) = \frac{n!}{r_1!(n-r_1)!} \cdot \frac{(n-r_1)!}{r_2!(n-r_1-r_2)!} \cdot \frac{(n-r_1-r_2)!}{r_3!(n-r_1-r_2-r_3)!} \cdot \ldots$$

$$\ldots \cdot \frac{(n-r_1-r_2-\ldots-r_{k-1})!}{r_k!(n-r_1-r_2-\ldots-r_k)!} = \frac{n!}{r_1! \cdot r_2! \cdot \ldots \cdot r_k!}.$$

В последнем сомножителе учтено, что $(n-r_1-r_2-\ldots-r_k)! = 0! = 1$.

Числа $C_n(r_1, r_2, \ldots, r_k)$ являются коэффициентами в разложении

$$(a_1 + a_2 + \ldots + a_k)^n = \sum_{r_i \geq 0,\, r_1+\ldots+r_k=n} C_n(r_1, r_2, \ldots, r_k) a_1^{r_1} a_2^{r_2} \ldots a_k^{r_k}$$

и поэтому называются ***полиномиальными коэффициентами***.

Числа $C_n(r_1, r_2, \ldots, r_k)$ участвуют в решении ещё одной известной комбинаторной задачи. Пусть имеется n букв, среди которых r_1 букв b_1, r_2 букв b_2, …, r_k букв b_k. Таким образом, $r_1 + r_2 + \ldots + r_k = n$. Сколько существует перестановок этих букв? Или, другими словами, сколько различных слов длины n можно составить из этих букв? Для решения этой задачи, как и в доказательстве теоремы 2, выберем сначала места для

букв b_1 ($C_n^{r_1}$ способов), затем места для букв b_2 ($C_{n-r_1}^{r_2}$ способов) и так далее. Как и в теореме 2, получим, что число различных перестановок равняется $C_n(r_1,r_2,\ldots,r_k)$.

Пример 3. Сколько различных (в том числе и бессмысленных) слов можно составить, переставляя буквы слова «математика»?

Решение. $\dfrac{10!}{3!\cdot 2!\cdot 2!\cdot 1!\cdot 1!\cdot 1!} = \dfrac{2\cdot 3\cdot \cancel{4}\cdot 5\cdot \cancel{6}\cdot 7\cdot 8\cdot 9\cdot 10}{\cancel{6}\cdot \cancel{2}\cdot \cancel{2}} = 30\cdot 56\cdot 90 = 151\,200$.

1.2 Задачи о размещении элементов по ячейкам

Теорема 3. Существует взаимно-однозначное соответствие между множеством всех **r**-размещений с повторениями элементов **n**-множества $X=\{x_1,x_2,\ldots,x_n\}$ и множеством способов разместить **r** различных элементов по **n** различным ячейкам.

Доказательство. Пусть мы раскладываем элементы $a_1,a_2,\ldots,a_r$ по ячейкам $H_1,H_2,\ldots,H_n$. Сопоставим произвольному **r**-размещению с повторениями $(x_{i_1},x_{i_2},\ldots,x_{i_r})$ следующий способ раскладки: элемент a_1 поместим в ячейку H_{i_1}, элемент a_2 – в ячейку H_{i_2} и так далее. Тогда разным **r**-размещениям соответствуют разные способы раскладки (отображение инъективно), и каждому способу раскладки соответствует некоторое **r**-размещение (отображение сюръективно). Следовательно, отображение биективно, что и требовалось доказать.

Пример 4. Пусть $X=\{x,y,z\}$. 5-размещению (z,z,y,z,x) при указанном отображении соответствует следующая раскладка элементов a_1,a_2,a_3,a_4,a_5 по трём ячейкам: элементы a_1,a_2,a_4 попадают в третью ячейку, a_3 во вторую, a_5 в первую.

Теорема 4. Существует взаимно-однозначное соответствие между множеством всех **r**-сочетаний с повторениями элементов **n**-множества $X=\{x_1,x_2,\ldots,x_n\}$ и множеством способов разместить **r** одинаковых элементов по **n** различным ячейкам.

Доказательство. Пусть мы раскладываем элементы $a,a,\ldots,a$ (**r** штук) по ячейкам $H_1,H_2,\ldots,H_n$. Произвольное **r**-сочетание с повторениями обозначим так: $\{x_1^{r_1},x_2^{r_2},\ldots,x_n^{r_n}\}$, где $r_1+r_2+\ldots+r_n=r$. Сопоставим такому **r**-сочетанию следующий способ раскладки: r_1 элементов поместим в ячейку H_1, r_2 элементов – в ячейку H_2 и так далее. Нетрудно видеть, что построенное отображение является биективным.

Пример 5. Пусть $X=\{x,y,z\}$. 5-сочетанию $\{y^4,z\}$ при указанном отображении соответствует следующая раскладка элементов a,a,a,a,a по трём ячейкам: первая ячейка остаётся пустой, во вторую попадают 4 элемента, в третью ячейку – один.

Замечание. Различные ограничения на размещения или сочетания (обычно, ограничения на число вхождений в конфигурацию различных элементов множества X) соответствуют аналогичным ограничениям на вместимость ячеек. Задачи, где рассматриваются такие ограничения, обычно решают с помощью производящих функций.

1.3 Генерирование комбинаторных конфигураций

При решении многих комбинаторных задач применяется перебор всех возможных вариантов. Поэтому необходимо иметь алгоритмы, позволяющие получать списки всех конфигураций определённого типа. Например, список всех перестановок элементов данного множества. При этом порядок конфигураций в списке часто играет существенную роль, т.е. нужны алгоритмы, генерирующие конфигурации в последовательности с определёнными ограничениями.

1.3.1 Генерирование всех подмножеств конечного множества

Простейший алгоритм использует двоичную кодировку подмножеств. Пусть $X=\{x_1,x_2,\ldots,x_n\}$ – конечное множество. Каждому его подмножеству Y сопоставим последовательность $a_1,a_2,\ldots,a_n$ из нулей и единиц по правилу:

$$a_i=\begin{cases}0, & \text{если } x_i\notin Y;\\ 1, & \text{если } x_i\in Y.\end{cases}$$

Легко понять, что построенное соответствие является биективным. Отсюда, в частности, следует, что **n**-элементное множество имеет 2^n подмножеств. Теперь алгоритм, генерирующий все подмножества множества X, строится просто, в виде цикла по j от 1 до 2^n-1. Нужно перевести j в двоичную запись, дополнить впереди нулями (чтобы число цифр равнялось n) и построить подмножество, соответствующее полученной двоичной последовательности. Например, для множества $X=\{1,2,3\}$ получим следующую последовательность подмножеств:

$$\varnothing,\ \{3\},\ \{2\},\ \{2,3\},\ \{1\},\ \{1,3\},\ \{1,2\},\ \{1,2,3\}.$$

В некоторых случаях важно, чтобы каждое следующее подмножество мало отличалось от предыдущего. Рассмотрим способ генерирования подмножеств, при котором каждое следующее подмножество получается из предыдущего добавлением или удалением одного элемента. На языке двоичных кодов это значит, что нужно построить последовательность всех двоичных наборов длины **n** так, чтобы соседние наборы отличались только в одной позиции. Допустим, что мы умеем генерировать ***такую*** последовательность из наборов длины **k**:

$$C_1,\quad C_2,\quad \ldots,\quad C_m,\ \text{где}\quad m=2^k-1.$$

Тогда требуемым свойством будет, очевидно, обладать и следующая последовательность:

$$0C_1,\quad 0C_2,\quad \ldots,\quad 0C_m,\quad 1C_m,\quad 1C_{m-1},\quad \ldots,\quad 1C_1.$$

Заметим, что описанный алгоритм уже позволяет решить поставленную задачу. Однако он является ***рекуррентным*** (возвратным). Чтобы получить все подмножества **n**-элементного множества нужно сначала получить все подмножества **(n-1)**-элементного множества. Алгоритмы такого типа требуют очень больших объёмов памяти, желательно построить ***прямой*** алгоритм. Оказывается, ту же самую последовательность двоичных кодов можно построить и по-другому, не применяя рекурсию.

Пусть Z – стандартная последовательность двоичных кодов (двоичные числа от 0 до 2^n-1 в порядке возрастания). Построим другую (с требуемым свойством) последовательность Ω этих же кодов. Первый её элемент – $0,0,\ldots,0$ (как и в исходной последовательности). Для каждого $i=1,2,\ldots,2^n-1$ найдём в i-ом элементе последовательности Z первый справа нулевой разряд. Перейдём в последовательности Ω от i-го элемента к $(i+1)$-му, изменяя цифру в этом же разряде.

Покажем работу алгоритма на примере. Пусть требуется сгенерировать подмножества 4-х-элементного множества. В левой колонке будем располагать двоичные коды в стандартной последовательности, а в правой – результат.

Z	Ω
0000	0000
0001	0001
0010	0011

0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

1.3.2 Генерирование k-элементных подмножеств в лексикографическом порядке

Пусть элементы **n**-множества X упорядочены. Без потери общности можно считать, что $X=\{1,2,\ldots,n\}$. Тогда множество его **k**-элементных подмножеств (т.е. **k**-сочетаний) можно упорядочить лексикографически (так упорядочиваются слова в словарях). Рассмотрим алгоритм, генерирующий такие подмножества в лексикографическом порядке. Для простоты записи будем записывать подмножества без скобок и запятых, а их элементы располагать в порядке возрастания.

В качестве первого подмножества возьмём первые **k** элементов множества X. Чтобы перейти от уже построенного подмножества к следующему, нужно найти в нём самый правый элемент, который можно увеличить. Этот элемент следует увеличить на **1**, а все последующие (если таковые имеются) получаются последовательным добавлением **1**. Точнее, если подмножество $a_1a_2\ldots a_n$ уже построено, самый правый элемент, допускающий увеличение есть a_i, то следующее подмножество будет таким:

$$a_1,\ a_2,\ldots,a_{i-1},\ a_i+1,\ a_i+2,\ \ldots,\ a_i+n-i+1.$$

Пример 6. Применим указанный алгоритм для получения в лексикографическом порядке 4-х-элементных подмножеств множества $X=\{1,2,3,4,5,6\}$. Подмножества записываем без скобок и запятых:

1234, 1235, 1236, 1245, 1246, 1256, 1345, 1346, 1356, 1456, 2345, 2346, 2356, 2456, 3456.

1.3.3 Генерирование перестановок в лексикографическом порядке

Во многих случаях необходимо генерировать не сочетания (т.е. подмножества), а размещения (упорядоченные подмножества), в частности – перестановки всего множества. Рассмотрим алгоритм, генерирующий перестановки элементов конечного множества $X=\{1,2,\ldots,n\}$ в лексикографическом порядке.

1. Начинаем с перестановки **123…n**. Далее будут описаны действия, необходимые для перехода от уже построенной перестановки к следующей.
2. Найдём упорядоченный по убыванию «хвост» (возможно, длины 1).
3. Элемент, предшествующий «хвосту», заменим на наименьший элемент «хвоста», больший его.
4. Оставшиеся элементы упорядочим по возрастанию.

Проиллюстрируем начало работы этого алгоритма для **n = 6**.

123456, 123465, 123546, 123564, 123645, 123654,
124356, 124365, 124536, 124563, 124635, 124653,
125346, 125364, 125436, 125463, 125634, 125643,

126345, 126354, 126435, 126453, 126534, 126543, 132456, …..

Обратим внимание: алгоритм имеет рекурсивный характер. Первые 120 из этих 720 перестановок на первом месте имеют **1**. Если вычеркнуть у них **1**, то получим перестановки множества **{2,3,4,5,6}**, расположенные в лексикографическом порядке.

Замечание. Программы на псевдокоде, реализующие рассмотренные алгоритмы, можно найти в книгах [5], [7].

2 Возвратные (рекуррентные) задачи

2.1 Примеры возвратных задач

2.1.1 Задача о Ханойской башне

Головоломку придумал французский математики Эдуард Люка в 1883 году. Башня – восемь дисков, нанизанных в порядке уменьшения размеров на один из трёх колышков. Задача: переместить всю башню на один из других колышков, перенося каждый раз только один диск и не помещая больший диск на меньший.

Нетрудно понять, что задача разрешима. Вопрос: какой способ самый оптимальный, самый короткий?

Часто очень полезным оказывается **обобщение**. Пусть дисков **n**. Теперь можно рассмотреть простые случаи: **n=1**, **n=2** . А это легко. Обозначим $\mathbf{T_n}$ – минимальное число перекладываний, необходимое для решения задачи. Ясно, что $\mathbf{T_1=1}$, $\mathbf{T_2=3}$. Можно, конечно, считать, что $\mathbf{T_0=0}$.

Предположим, что мы умеем перемещать на соседний колышек **n-1** диск, т.е. знаем число $\mathbf{T_{n-1}}$. Решаем задачу для **n**. Переместим верхние **n-1** диск на соседний колышек ($\mathbf{T_{n-1}}$ перекладываний), переместим самый большой диск на свободный колышек (1 перекладывание), затем переместим **n-1** диск на колышек, где уже находится большой (ещё $\mathbf{T_{n-1}}$ перекладываний). Итак, задача решена за $\mathbf{2T_{n-1}+1}$ перекладываний. А можно ли быстрее? Нет, так как всё равно, при любом способе придётся перекладывать большой диск, а это разрешается делать только на пустой колышек. Остальные диски в это время должны быть на третьем колышке, а чтобы их туда переместить нужно минимум $\mathbf{T_{n-1}}$ перекладываний. И ещё столько же перекладываний необходимо, чтобы поместить эти диски на большой.

Итак, $\mathbf{T_n= 2T_{n-1}+1}$. Добавим сюда начальное соотношение $\mathbf{T_0=0}$. Теперь мы можем определить $\mathbf{T_n}$ для любого **n**. Заметим, что значения $\mathbf{T_1=1}$, $\mathbf{T_2=3}$ не противоречат полученному соотношению. Соотношения такого типа:

$$\mathbf{T_n = 2T_{n-1}+1}, \quad \mathbf{T_0=0}$$

называются возвратной, или рекуррентной задачей. Употребляются также термины рекуррентное (или рекурсивное) соотношение, рекуррентное уравнение.

Хотя рекуррентное уравнение и позволяет найти $\mathbf{T_n}$ для любого **n**, но сделать это сложно. Действительно, чтобы вычислить, например, $\mathbf{T_{18}}$ нужно сначала найти $\mathbf{T_{17}}$. Нельзя ли получить формулу, позволяющую вычислять $\mathbf{T_n}$ непосредственно по номеру **n** ? Такая формула называется ***решением*** рекуррентного уравнения, найдём её для нашего примера.

Чтобы найти зависимость $\mathbf{T_n}$ от **n**, выпишем начальные значения:

$$T_0 = 0, \quad T_1 = 1, \quad T_2 = 3, \quad T_3 = 7, \quad T_4 = 15, \quad T_5 = 31, \quad T_6 = 63, \ldots$$

Нетрудно заметить закономерность: $T_n = 2^n - 1$. Чтобы убедиться, что эта формула справедлива при любом **n**, докажем её с помощью математической индукции.

База индукции имеется, для малых **n** формула верна. Предположим, что для некоторого **k** справедливо: $T_k = 2^k - 1$. Тогда

$$T_{k+1} = 2 \cdot T_k + 1 = 2\left(2^k - 1\right) + 1 = 2^{k+1} - 1,$$

т.е. получили, что и для $k+1$ формула также верна. Следовательно, для любого **n** верно:

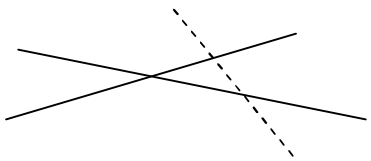
$$T_n = 2^n - 1,$$

мы получили решение рекуррентного уравнения.

2.1.2 Задача о разрезании пиццы

На какое максимальное число частей можно разрезать пиццу, делая n прямолинейных разрезов? Так в книге [3] сформулирована известная задача о максимальном числе областей, на которые делится плоскость n прямыми линиями.

Обозначим искомую величину E_n. Тогда $E_0 = 1$ (если прямых нет, то область только одна), $E_1 = 2$ (один разрез – 2 области), $E_2 = 4$ (конечно, чтобы получить максимальное число областей, прямые не должны быть параллельны). Переход от E_2 к E_3 покажем на рисунке.



Добавленная прямая пересекает 2 прямые и добавляет 3 области, т.е. $E_3 = 7$. Опираясь на это геометрическое представление, постараемся вывести рекуррентное соотношение для E_n. Пусть плоскость разделена на E_{n-1} областей с помощью $n-1$ прямой. Для получения максимального числа областей проведём ещё одну прямую так, чтобы она не была параллельна остальным и не проходила через точки их пересечения. Тогда новая прямая будет иметь ровно $n-1$ точку пересечения, т.е. пройдёт через n областей, разделяя каждую из них на 2 части. Итак, появится n новых областей:

$$E_n = E_{n-1} + n.$$

Рекуррентное соотношение (уравнение) для E_n получено.

Для того, чтобы найти решение уравнения (т.е. выразить E_n непосредственно через n), запишем найденное соотношение для $k = 1, 2, 3, \ldots, n$, предварительно перенося одно из слагаемых в левую часть:

$$E_1 - E_0 = 1,$$
$$E_2 - E_1 = 2,$$
$$E_3 - E_2 = 3,$$
$$\ldots\ldots\ldots\ldots\ldots,$$
$$E_{n-1} - E_{n-2} = n-1,$$
$$E_n - E_{n-1} = n.$$

Складывая все эти равенства, получим: $E_n - E_0 = E_n - 1 = 1 + 2 + 3 + \ldots + n = \dfrac{n(n+1)}{2}$.

Значит, $E_n = \dfrac{n(n+1)}{2} + 1$. Задача о разрезании пиццы решена.

2.1.3 Числа Фибоначчи.

Наиболее известное рекуррентное соотношение – это соотношение

$$u_{n+2} = u_{n+1} + u_n, \quad u_0 = 0, \; u_1 = 1.$$

Оно определяет последовательность чисел Фибоначчи:

$$0,\ 1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89,\ 144,\ 233,\ldots$$

Впервые эти числа были приведены в работе Леонардо Пизанского (Фибоначчи) в 1228 году. Фибоначчи получил первые 14 чисел этой последовательности, решая задачу о размножении кроликов. А впоследствии оказалось, что числа Фибоначчи появляются при решении очень многих задач. Поэтому их свойства хорошо изучены. В частности, была получена формула для n-го члена последовательности Фибоначчи:

$$u_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}.$$

Это так называемая ***формула Бинэ***, получить её не так просто, как в первых двух наших примерах. Мы сможем вывести эту формулу после того, как познакомимся с общей теорией решения линейных рекуррентных уравнений.

2.2 Линейные однородные рекуррентные уравнения

Мы будем рассматривать линейные однородные рекуррентные уравнения (ЛОРУ) с постоянными действительными коэффициентами:

$$a_{n+k} + p_1 a_{n+k-1} + p_2 a_{n+k-2} + \ldots + p_k a_n = 0. \qquad (*)$$

Здесь $p_1, p_2, \ldots, p_k$ – действительные числа, a_i – i-й член некоторой последовательности $\{a_i \mid i = 0,1,2,3,\ldots\}$. Число k называется порядком уравнения.

Последовательность $\{a_i\}$ называется ***решением*** уравнения (*), если для любого $n = 0,1,2,3,\ldots$ её члены $a_{n+k}, a_{n+k-1}, a_{n+k-2}, \ldots, a_n$ удовлетворяют уравнению.

Пример 1. Рассмотрим уравнение $a_{n+1} - 3a_n = 0$ – ЛОРУ 1-го порядка. Его решением является любая числовая последовательность, у которой каждый член, начиная со второго, получается из предыдущего умножением на 3. Например, последовательность $2, 6, 18, 54, 162, \ldots$ является решением, последовательность $\frac{1}{5}, \frac{3}{5}, \frac{9}{5}, \frac{27}{5}, \frac{81}{5}, \ldots$ тоже является, очевидно, решением.

Пример показывает, что ЛОРУ может иметь много решений. Множество всех решений называется ***общим решением*** ЛОРУ. Для того, чтобы из общего решения выбрать одно, конкретное, ***частное*** решение, нужны дополнительные условия. В качестве таких условий обычно задают значения первых k членов искомой последовательности.

Если $a_0, a_1, \ldots, a_{k-1}$ заданы, то из соотношения (*) можно найти сначала a_k, затем a_{k+1} и так далее, можно определить любой член искомой последовательности. Итак, уравнение (*) и значения $a_0, a_1, \ldots, a_{k-1}$ полностью определяют последовательность. Однако задана она ***рекуррентно***. Под термином «решить» уравнение (*) понимается получение ***явного*** задания последовательности, получение формулы, выражающей величину a_n через номер n.

Если начальные значения $a_0, a_1, \ldots, a_{k-1}$ являются действительными числами, то и вся последовательность, заданная уравнением (*), состоит из действительных чисел (так как $p_1, p_2, \ldots, p_k$ – действительные числа). Однако можно рассматривать и комплексные решения, это не вызывает никаких дополнительных трудностей. Поэтому говоря о числах, мы будем допускать, что числа могут быть и комплексными.

Числовые последовательности, как известно, можно складывать, можно умножать на числа, причём эти операции выполняются «поэлементно».

Лемма 1. Если последовательности $\{a_n\}$, $\{b_n\}$ являются решениями уравнения (*), α – число, то последовательности $\{a_n + b_n\}$, $\{\alpha a_n\}$ тоже являются решениями.

Доказательство. Подставим каждую последовательность в уравнение:

$$(a_{n+k}+b_{n+k})+p_1(a_{n+k-1}+b_{n+k-1})+\ldots+p_k(a_n+b_n)=$$
$$=(a_{n+k}+p_1a_{n+k-1}+\ldots+p_ka_n)+(b_{n+k}+p_1b_{n+k-1}+\ldots+p_kb_n)=0+0=0,$$
$$(\alpha a_{n+k})+p_1(\alpha a_{n+k-1})+\ldots+p_k(\alpha a_n)=\alpha(a_{n+k}+p_1a_{n+k-1}+\ldots+p_ka_n)=\alpha\cdot 0=0.$$

Лемма доказана. Так как последовательность $0,0,0,0,\ldots$, очевидно, удовлетворяет уравнению (*), то в лемме фактически говорится, что множество решений уравнения (*) образует линейное пространство над полем комплексных чисел (подпространство линейного пространства всех числовых последовательностей).

Важным инструментом для решения ЛОРУ является понятие «характеристическое уравнение». ***Характеристическим уравнением*** для ЛОРУ (*) называется уравнение

$$x^k+p_1x^{k-1}+p_2x^{k-2}+\ldots+p_{k-1}x+p_k=0.$$

Лемма 2. Если число r является корнем характеристического уравнения ***(характеристическим корнем)***, то последовательность $\{r^n\}$ является решением рекуррентного уравнения (*).

Доказательство. Подставим последовательность $\{r^n\}$ в уравнение (*):

$$r^{n+k}+p_1r^{n+k-1}+p_2r^{n+k-2}+\ldots+p_{k-1}r^{n+1}+p_kr^n=0,$$
$$r^n\left(r^k+p_1r^{k-1}+p_2r^{k-2}+\ldots+p_{k-1}r+p_k\right)=0,$$
$$r^k+p_1r^{k-1}+p_2r^{k-2}+\ldots+p_{k-1}r+p_k=0.$$

Последнее равенство является тождеством, так как r – характеристический корень. В случае $r=0$ утверждение очевидно, нулевая последовательность является решением. Лемма доказана.

Теорема 1. Если все характеристические корни $r_1,r_2,\ldots,r_k$ различны, то общее решение уравнения (*) имеет вид:

$$a_n=C_1r_1^n+C_2r_2^n+\ldots+C_kr_k^n,$$

где C_i могут быть любыми числами.

Доказательство. Леммы 1 и 2 показывают, что любая последовательность такого вида является решением. Мы должны убедиться, что это ***общее*** решение, т.е. других решений нет. Пусть $\{b_n\}$ – какое либо решение. Требуется доказать, что существуют числа $\alpha_1,\alpha_2,\ldots,\alpha_k$ такие, что $b_n=\alpha_1r_1^n+\alpha_2r_2^n+\ldots+\alpha_kr_k^n$. Рассмотрим систему уравнений относительно неизвестных $C_1,C_2,\ldots,C_k$:

$$\begin{cases} C_1+C_2+\ldots+C_k=b_0, \\ C_1r_1+C_2r_2+\ldots+C_kr_k=b_1, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ C_1r_1^{k-1}+C_2r_2^{k-1}+\ldots+C_kr_k^{k-1}=b_{k-1}. \end{cases}$$

Её определитель V называется определителем Вандермонда. По индукции можно доказать, что

$$V=\begin{vmatrix} 1 & 1 & \ldots & 1 \\ r_1 & r_2 & \ldots & r_k \\ \ldots & \ldots & \ldots & \ldots \\ r_1^{k-1} & r_2^{k-1} & \ldots & r_k^{k-1} \end{vmatrix}=\prod_{1\le j<i\le k}(r_i-r_j).$$

В правой части равенства – произведение всех скобок вида (r_i-r_j). Значит, если числа $r_1,r_2,\ldots,r_k$ различны, то $V\neq 0$. Следовательно, система уравнений имеет единственное решение. Обозначим это решение $\alpha_1,\alpha_2,\ldots,\alpha_k$ и убедимся, что соотношение $b_n=\alpha_1r_1^n+\alpha_2r_2^n+\ldots+\alpha_kr_k^n$ выполнено. Действительно, при $n=0,1,2,\ldots,k-1$ это следует

непосредственно из системы уравнений. Так как первые k членов полностью определяют последовательность, удовлетворяющую уравнению (*), то это выполнено и при любом n. Теорема доказана.

Пример 2. Вывести формулу Бинэ, т.е. решить рекуррентное уравнение Фибоначчи: $u_{n+2} = u_{n+1} + u_n,\ \ u_0 = 0,\ u_1 = 1$.

Решение. Запишем данное линейное однородное рекуррентное уравнение 2-го порядка в виде (*): $u_{n+2} - u_{n+1} - u_n = 0$. Характеристическое уравнение для него имеет вид: $x^2 - x - 1 = 0$. Найдём корни: $x_{1,2} = \frac{1 \pm \sqrt{5}}{2}$. По теореме 1, общее решение рекуррентного уравнения имеет вид:

$$u_n = C_1\left(\frac{1+\sqrt{5}}{2}\right)^n + C_2\left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Найдём C_1, C_2. По условию

$$u_0 = C_1 + C_2 = 0, \qquad u_1 = C_1\frac{1+\sqrt{5}}{2} + C_2\frac{1-\sqrt{5}}{2} = 1.$$

Подставляя $C_2 = -C_1$ во второе соотношение, получим: $C_1\left(\frac{1+\sqrt{5}}{2} - \frac{1-\sqrt{5}}{2}\right) = C_1\sqrt{5} = 1$.

Отсюда $C_1 = \frac{1}{\sqrt{5}},\ \ C_2 = -\frac{1}{\sqrt{5}}$. Формула Бинэ доказана: $u_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$.

Из доказательства теоремы 1 следует, что если характеристические корни $r_1, r_2, \ldots, r_k$ различны, то решения $\{r_1\}, \{r_2\}, \ldots, \{r_k\}$ линейно независимы и образуют базис в линейном пространстве всех решений.

Мы рассматриваем ЛОРУ с действительными коэффициентами, однако характеристические корни могут быть комплексными. Это не требует никаких изменений в процессе решения. Рассмотрим соответствующий пример.

Пример 3. Решить реккурентное уравнение:

$$a_{n+2} - 2a_{n+1} + 2a_n = 0,\ \ a_0 = 0,\ a_1 = 1.$$

Решение. Составим характеристическое уравнение: $x^2 - 2x + 2 = 0$. Его корни: $x_{1,2} = \frac{2 \pm \sqrt{4-8}}{2} = 1 \pm i$. По теореме 1, общее решение имеет вид: $a_n = C_1(1+i)^n + C_2(1-i)^n$. Используем начальные данные, чтобы найти C_1, C_2:

$$\begin{cases} C_1 + C_2 = 0, \\ C_1(1+i) + C_2(1-i) = 1. \end{cases}$$

Подставляя $C_2 = -C_1$ во второе уравнение, получим: $C_1 = \frac{1}{2i} = -\frac{i}{2}$. Значит, $C_2 = \frac{i}{2}$, и решение найдено: $a_n = -\frac{i}{2}(1+i)^n + \frac{i}{2}(1-i)^n$.

Заметим, что так как коэффициенты уравнения и начальные данные являются действительными числами, то и все a_n – действительные числа. Можно ещё раз в этом убедиться, вычислив сопряжённое к a_n:

$$\overline{a_n} = \overline{-\frac{i}{2}(1+i)^n + \frac{i}{2}(1-i)^n} = -\frac{\bar{i}}{2}\left(\overline{1+i}\right)^n + \frac{\bar{i}}{2}\left(\overline{1-i}\right)^n = \frac{i}{2}(1-i)^n - \frac{i}{2}(1+i)^n = a_n.$$

Замечание. Можно доказать, то если $r, \bar{r}$ – сопряжённые комплексные ($r \neq \bar{r}$) характеристические корни ЛОРУ с действительными коэффициентами, то действитель-

ные последовательности $\left\{Re\left(r^n\right)\right\}, \left\{Im\left(r^n\right)\right\}$ также являются решениями. Их применение вместо комплексных решений $\left\{r^n\right\}, \left\{\bar{r}^n\right\}$ не нарушает линейной независимости системы решений и позволяет записать решение без комплексных чисел.

В нашем примере 3, с помощью формулы Муавра, получим:

$$(1+i)^n = \left(\sqrt{2}\left(\cos\frac{\pi}{4} + i\sin\frac{\pi}{4}\right)\right)^n = 2^{\frac{n}{2}}\left(\cos\frac{n\pi}{4} + i\sin\frac{n\pi}{4}\right).$$

Линейно независимыми решениями являются $\left\{2^{\frac{n}{2}}\cos\frac{n\pi}{4}\right\}, \left\{2^{\frac{n}{2}}\sin\frac{n\pi}{4}\right\}$. Общее решение:

$$a_n = D_1 2^{\frac{n}{2}}\cos\frac{n\pi}{4} + D_2 2^{\frac{n}{2}}\sin\frac{n\pi}{4},$$

где D_1, D_2 произвольные числа. Используем начальные данные:

$$\begin{cases} a_0 = D_1 = 0, \\ a_1 = D_1\sqrt{2}\cos\frac{\pi}{4} + D_2\sqrt{2}\sin\frac{\pi}{4} = 1, \quad D_2 = 1. \end{cases}$$

Получили решение, записанное без комплексных чисел:

$$a_n = 2^{\frac{n}{2}}\sin\frac{n\pi}{4}.$$

Можно выписать несколько первых членов найденной последовательности:

0, 1, 2, 2, 0, -4, -8, -8, 0, 16, 32, 32, 0, …

Осталось рассмотреть случай, когда характеристическое уравнение имеет кратные корни. Здесь результаты приводятся без доказательств.

Теорема 2. Если r – характеристический корень кратности m, то ему соответствуют m линейно независимых решений: $\left\{r^n\right\}, \left\{nr^n\right\}, \left\{n^2r^n\right\}, \ldots, \left\{n^{m-1}r^n\right\}$. Общее решение уравнения (*) является линейной комбинацией всех решений, соответствующих характеристическим корням.

Пример 4. Найти общее решение рекуррентного уравнения:

$$a_{n+3} + 3a_{n+2} + 3a_{n+1} + a_n = 0.$$

Решение. Составим характеристическое уравнение: $x^3 + 3x^2 + 3x + 1 = 0$, или $(x+1)^3 = 0$. Его корни $r_1 = r_2 = r_3 = -1$. По теореме 2, общее решение имеет вид

$$a_n = C_1(-1)^n + C_2 n(-1)^n + C_3 n^2(-1)^n = (-1)^n\left(C_1 + C_2 n + C_3 n^2\right).$$

Пример 5. Решить рекуррентное уравнение:

$$a_{n+2} - 8a_{n+1} + 16a_n = 0, \quad a_0 = 1, \; a_1 = 3.$$

Решение. Составим характеристическое уравнение: $x^2 - 8x + 16 = 0$. Его корни $r_1 = r_2 = 4$. Значит, общее решение имеет вид: $a_n = C_1 \cdot 4^n + C_2 \cdot n \cdot 4^n$. Используя начальные данные, найдём константы C_1, C_2:

$$\begin{cases} C_1 = 1 \\ 4C_1 + 4C_2 = 3 \end{cases} \Rightarrow \begin{cases} C_1 = 1 \\ C_2 = -\frac{1}{4} \end{cases} \Rightarrow a_n = 4^n - n\cdot 4^{n-1} = 4^{n-1}(4-n).$$

2.3 Линейные неоднородные рекуррентные уравнения

Теперь научимся решать (в простейших случаях) линейные неоднородные рекуррентные уравнения – ЛНРУ:

$$a_{n+k} + p_1 a_{n+k-1} + p_2 a_{n+k-2} + \ldots + p_k a_n = f(n). \qquad (**)$$

Теорема 3 (о структуре общего решения ЛНРУ). Общее решение уравнения (**) можно найти в виде:

$$\{a_n\} = \{\overline{a_n}\} + \{t_n\},$$

где $\{\overline{a_n}\}$ – общее решение соответствующего ЛОРУ (правая часть $f(n)$ заменена нулём), $\{t_n\}$ – какое-либо частное решение уравнения (**).

Доказательство приведём только для уравнения 2-го порядка, причём ограничимся случаем разных характеристических корней.

Итак, рассмотрим уравнение $a_{n+2} + p_1 a_{n+1} + p_2 a_n = f(n)$. Пусть корни характеристического уравнения $x^2 + p_1 x + p_2 = 0$ различны: $r_1 \neq r_2$. По теореме 1 общее решение ЛОРУ имеет вид $C_1 r_1^n + C_2 r_2^n$. Пусть t_n – какое либо частное решение данного ЛНРУ. Требуется доказать, что $a_n = C_1 r_1^n + C_2 r_2^n + t_n$ – общее решение. Подставляя это выражение в уравнение, убедимся, что оно является решением при любых константах C_1, C_2:

$$C_1 r_1^{n+2} + C_2 r_2^{n+2} + t_{n+2} + p_1\left(C_1 r_1^{n+1} + C_2 r_2^{n+1} + t_{n+1}\right) + p_2\left(C_1 r_1^n + C_2 r_2^n + t_n\right) = f(n),$$

$$C_1\left(r_1^{n+2} + p_1 r_1^{n+1} + p_2 r_1^n\right) + C_2\left(r_2^{n+2} + p_1 r_2^{n+1} + p_2 r_2^n\right) + \left(t_{n+2} + p_1 t_{n+1} + p_2 t_n\right) = f(n),$$

$$0 + 0 + f(n) = f(n).$$

Пусть $\{b_n\}$ – какое-либо решение уравнения $a_{n+2} + p_1 a_{n+1} + p_2 a_n = f(n)$. Нам нужно убедиться, что оно совпадает с $a_n = C_1 r_1^n + C_2 r_2^n + t_n$ при некоторых значениях C_1, C_2. Возьмём $n = 0, n = 1$ и составим систему уравнений:

$$\begin{cases} C_1 + C_2 + t_0 = b_0, \\ C_1 r_1 + C_2 r_2 + t_1 = b_1. \end{cases}$$

Она имеет единственное решение, так как её определитель $\begin{vmatrix} 1 & 1 \\ r_1 & r_2 \end{vmatrix} = r_2 - r_1 \neq 0$. Так как решения полностью определяются своими первыми двумя членами, то отсюда следует, что $C_1 r_1^n + C_2 r_2^n + t_n = b_n$ при любом n. Значит, $a_n = C_1 r_1^n + C_2 r_2^n + t_n$ – общее решение. Теорема в рассматриваемом случае доказана.

Мы научились находить общее решение ЛОРУ (в случае, если удаётся решить характеристическое уравнение). Но для нахождения частного решения ЛНРУ общих правил нет. Мы рассмотрим 2 случая, соответствующие различным функциям $f(n)$ в правой части уравнения.

Случай 1. Правая часть является многочленом: $f(n) = P(n)$. Тогда существует частное решение ЛНРУ в виде $t_n = n^m Q(n)$, где m – кратность корня $x = 1$ в характеристическом уравнении, $Q(n)$ – некоторый многочлен той же степени, что и $P(n)$. Если число 1 не является характеристическим корнем, то $m = 0$ и частное решение следует искать в виде $t_n = Q(n)$. Многочлен $Q(n)$ сначала записывается с неопределёнными коэффициентами. Затем коэффициенты нужно найти, подставляя $Q(n)$ в уравнение.

Пример 6. Найти общее решение рекуррентного уравнения:

$$a_{n+2} - 9a_{n+1} + 14a_n = 12n + 4.$$

Решение. Составим характеристическое уравнение: $x^2 - 9x + 14 = 0$, его корни $r_1 = 7, r_2 = 2$. Общее решение соответствующего однородного уравнения имеет вид $C_1 \cdot 7^n + C_2 \cdot 2^n$. Так правая часть есть многочлен 1-й степени, а число 1 не является характеристическим корнем, то будем искать частное решение исходного уравнения в виде $Q(n) = An + B$. Чтобы найти коэффициенты, подставим $Q(n)$ в уравнение:

$$A(n+2)+B-9\big(A(n+1)+B\big)+14\big(An+B\big)=12n+4\,.$$

Это равенство должно быть тождеством, т.е. должно выполняться при любых значениях n. Можно найти A, B сравнивая коэффициенты при одинаковых степенях n в левой и правой частях равенства:

$$\begin{cases} A-9A+14A=12, \\ 2A+B-9A-9B+14B=4, \end{cases} \Rightarrow \begin{cases} A=2, \\ -7A+6B=4, \end{cases} \Rightarrow \begin{cases} A=2, \\ B=3. \end{cases}$$

Итак, общее решение исходного ЛНРУ найдено:

$$a_n=C_1\cdot 7^n+C_2\cdot 2^n+2n+3\,.$$

Пример 7. Решить рекуррентное уравнение:

$$a_{n+2}-6a_{n+1}+5a_n=32,\quad a_0=1,\ a_1=-15\,.$$

Решение. Составим характеристическое уравнение: $x^2-6x+5=0$. Его корни $r_1=1, r_2=5$. Значит, общее решение однородного уравнения $a_{n+2}-6a_{n+1}+5a_n=0$ имеет вид: $C_1+C_2\cdot 5^n$. В правой части исходного уравнения – константа, т.е. многочлен нулевой степени. Число 1 является характеристическим корнем кратности 1. Поэтому будем искать частное решение в виде $t_n=n^m Q(n)=An$. Чтобы найти константу A, подставим An в исходное уравнение:

$$A(n+2)-6A(n+1)+5An=32 \quad\Rightarrow\quad 2A-6A=32 \quad\Rightarrow\quad A=-8\,.$$

Общее решение исходного уравнения найдено:

$$a_n=C_1+C_2\cdot 5^n-8n\,.$$

Пользуясь начальными данными, найдём значения коэффициентов C_1, C_2:

$$\begin{cases} C_1+C_2=1, \\ C_1+5C_2-8=-15 \end{cases} \Rightarrow \begin{cases} C_1=3, \\ C_2=-2. \end{cases}$$

Итак, искомое решение получено: $a_n=3-2\cdot 5^n-8n$.

Случай 2. Правая часть ЛНРУ имеет вид: $f(n)=c\cdot s^n$, где c, s – числа. Тогда существует частное решение ЛНРУ в виде $t_n=A\cdot n^m\cdot s^n$, где m – кратность корня $x=s$ в характеристическом уравнении, A – некоторое число, его нужно найти, подставляя выражение t_n в исходное уравнение.

Пример 8. Найти общее решение рекуррентного уравнения:

$$2a_{n+2}+7a_{n+1}+3a_n=(-3)^n\,.$$

Решение. Составим характеристическое уравнение: $2x^2+7x+3=0$, его корни $r_1=-3, r_2=-\frac{1}{2}$. Значит, общее решение однородного уравнения $C_1(-3)^n+C_2\left(-\frac{1}{2}\right)^n$.

Так как -3 является характеристическим корнем (кратности 1), то будем искать частное решение исходного ЛНРУ в виде $An(-3)^n$. Подставим это выражение в уравнение, чтобы найти коэффициент A:

$$2A(n+2)(-3)^{n+2}+7A(n+1)(-3)^{n+1}+3A(-3)^n=(-3)^n\,.$$

Сокращая на $(-3)^n$ и приравнивая коэффициенты в левой и правой частях, получим:

$$18A(n+2)-21A(n+1)+3An=1 \quad\Rightarrow\quad 36A-21A=1 \quad\Rightarrow\quad A=\frac{1}{15}\,.$$

Общее решение исходного уравнения найдено:

$$a_n=C_1(-3)^n+C_2\left(-\frac{1}{2}\right)^n+\frac{n}{15}(-3)^n\,.$$

3 Производящие функции

3.1 Задача о размене

Рассмотрим один из вариантов известной комбинаторной задачи: сколько существует способов заплатить 50 копеек монетами достоинством 1 копейка, 5 копеек и 10 копеек? Метод, который мы будем применять, позволяет найти подходы к ответу на более общий вопрос: сколько существует способов размена этими монетами не только 50 копеек, но и любой суммы?

Составим «сумму», в которой «слагаемыми» будут все возможные способы размена n копеек. Наиболее просто она выглядит, если разменивать только монетами по 1 копейке:

$$A = \varnothing + \odot + \odot\odot + \odot\odot\odot + \odot\odot\odot\odot + \ldots = \varnothing + \odot + \odot^2 + \odot^3 + \odot^4 + \ldots$$

Здесь значок $\odot$ обозначает монетку 1 копейка, а символ $\varnothing$ – единственный способ заплатить $n = 0$ копеек, т.е. не платить ничего. Пусть теперь $\oplus$ – пятикопеечная монета, а $\otimes$ – гривенник (10 копеек). Тогда «сумма»

$$B = A + \oplus A + \oplus\oplus A + \oplus\oplus\oplus A + \ldots = \left(\varnothing + \oplus + \oplus^2 + \oplus^3 + \oplus^4 + \ldots\right) A$$

содержит все способы размена копейками и пятаками, а «сумма»

$$C = B + \otimes B + \otimes\otimes B + \otimes\otimes\otimes B + \ldots = \left(\varnothing + \otimes + \otimes^2 + \otimes^3 + \otimes^4 + \ldots\right) B$$

включает все способы размена (произвольной денежной суммы) копейками, пятаками и гривенниками. Если раскрыть все скобки в выражении для C:

$$C = \left(\varnothing + \otimes + \otimes^2 + \otimes^3 + \otimes^4 + \ldots\right)\left(\varnothing + \oplus + \oplus^2 + \oplus^3 + \oplus^4 + \ldots\right)\left(\varnothing + \odot + \odot^2 + \odot^3 + \odot^4 + \ldots\right),$$

то получится «сумма», в которой каждое «слагаемое» имеет, например, такой вид:

$$\otimes^5 \oplus^8 \odot^2 .$$

Ясно, что это «слагаемое» символизирует один из способов разменять (заплатить) 92 копейки.

Перейдём теперь от суммирования символов к функциям, а точнее – к формальным степенным рядам. Заменим каждый символ $\odot$ символом переменной t, символ $\oplus$ заменим на t^5, символ $\otimes$ – на t^{10}. Тогда каждое слагаемое в сумме C будет иметь вид t^n. Например, $\otimes^5 \oplus^8 \odot^2 \Rightarrow t^{92}$. А если сложить все слагаемые, соответствующие размену 92 копеек, т.е. «привести подобные», то получим $P_{92}t^{92}$, где коэффициент P_{92} – количество слагаемых t^{92}, т.е. количество способов заплатить 92 копейки. Символ $\varnothing$ нужно заменять на $t^0 = 1$, он используется для обозначения одного – «пустого» – способа оплаты. Итак, мы получили функцию

$$C(t) = \left(1 + t + t^2 + t^3 + \ldots\right)\left(1 + t^5 + t^{10} + t^{15} + \ldots\right)\left(1 + t^{10} + t^{20} + t^{30} + \ldots\right).$$

Это и есть производящая функция для последовательности $\{P_n\}$. Она содержит информацию о ***всех*** членах последовательности $\{P_n\}$, так как если раскрыть скобки и записать $C(t)$ в виде степенного ряда, то

$$C(t) = P_0 + P_1 t + P_2 t^2 + P_3 t^3 + \ldots .$$

Конечно, ещё остаются значительные трудности при определении конкретных коэффициентов. Напомним, в исходной задаче требуется найти P_{50}. Простое раскрытие скобок и приведение подобных оказывается слишком трудоёмким. Можно заметить, что бесконечные ряды можно заменить конечными суммами, так как слагаемые $t^{51}, t^{52}, \ldots$ не могут участвовать в получении коэффициента P_{50}. Однако и конечные суммы оказываются слишком длинными.

Другой подход состоит в использовании формулы

$$1+t+t^2+t^3+\ldots=\frac{1}{1-t}.$$

В школьном курсе математики эта формула используется для суммирования убывающей геометрической прогрессии. Бесконечная сумма оказывается равной конечному числу (сходится), если $|t|<1$. Для нас вопросы сходимости не играют существенной роли, мы работаем с формальными степенными рядами. Но по-прежнему: если $S=1+t+t^2+t^3+\ldots$, то $St=t+t^2+t^3+t^4+\ldots$ и, значит, $S-St=1$, т.е. $S=\frac{1}{1-t}$.

Полученная формула позволяет записать производящую функцию для последовательности $\{P_n\}$ в «конечном» виде: $C(t)=\frac{1}{(1-t)(1-t^5)(1-t^{10})}$, однако это ещё слишком сложная формула для представления $C(t)$ в виде степенного ряда.

Привлечём дополнительные соображения, связанные с нашей конкретной задачей поиска коэффициента P_{50}. Можно упростить множитель $(1+t+t^2+t^3+\ldots)$ в записи функции $C(t)$, так как для получения t^{50} из этой скобки могут участвовать только степени, делящиеся на 5. Оставляя только такие слагаемые, получим новую функцию

$$\tilde{C}(t)=\frac{1}{(1-t^5)(1-t^5)(1-t^{10})}=\frac{1}{(1-t^5)^2(1-t^{10})}$$

с тем же самым коэффициентом при t^{50} в её разложении в степенной ряд. Кроме того, обозначая $t^5=z$, можно перейти к функции $\tilde{\tilde{C}}(z)=\frac{1}{(1-z)^2(1-z^2)}$, в разложении которой нас интересует коэффициент при z^{10}.

Правило умножения степенных рядов обсуждается в следующем пункте, однако нетрудно заметить (раскрывая скобки и приводя подобные), что

$$\frac{1}{(1-z)^2}=(1+z+z^2+\ldots)(1+z+z^2+\ldots)=1+2z+3z^2+\ldots=\sum_{n\ge 0}(n+1)z^n.$$

Теперь уже несложно найти коэффициент при z^{10} в разложении функции $\tilde{\tilde{C}}(z)$:

$$\tilde{\tilde{C}}(z)=\frac{1}{(1-z)^2(1-z^2)}=\left(\sum_{n\ge 0}(n+1)z^n\right)\left(\sum_{n\ge 0}z^{2n}\right),$$

необходимые нам подобные члены легко выписать:

$$11z^{10}z^0+9z^8z^2+7z^6z^4+5z^4z^6+3z^2z^8+z^0z^{10}=36z^{10}.$$

Задача решена, существует 36 способов заплатить 50 копеек монетами достоинством 1 копейка, 5 копеек и 10 копеек.

Метод производящих функций оказывается очень полезным потому, что одна функция содержит информацию о всей числовой последовательности. «Производящая функция является устройством, отчасти напоминающим мешок. Вместо того, чтобы нести отдельно много предметов, что могло бы оказаться затруднительным, мы собираем их вместе, и тогда нам нужно нести лишь один предмет – мешок» (Д.Пойа, Математика и правдоподобные рассуждения, М.: Наука, 1975).

3.2 Начальные сведения о производящих функциях

Пусть $a_0, a_1, \ldots, a_n, \ldots$ – некоторая числовая последовательность. Производящей функцией для неё называется степенной ряд

$$F(z)=\sum_{n=0}^{\infty}a_n z^n .$$

Если последовательность конечна, то она дополняется нулями: $a_0, a_1, \ldots, a_n, 0, 0, \ldots$ и производящая функция для неё – многочлен $a_0 + a_1 z + a_2 z^2 + \ldots + a_n z^n$.

Пусть $\{a_n\}$, $\{b_n\}$ – некоторые последовательности, $F(z)$, $G(z)$ их производящие функции. Тогда сумма производящих функций

$$F(z)+G(z)=\sum_{n=0}^{\infty}a_n z^n+\sum_{n=0}^{\infty}b_n z^n=\sum_{n=0}^{\infty}(a_n+b_n)z^n$$

является производящей функцией последовательности $\{a_n + b_n\}$. Немного сложнее выполняется умножение производящих функций:

$$F(z)\cdot G(z)=\left(\sum a_n z^n\right)\left(\sum b_n z^n\right)=\sum c_n z^n ,$$

где $c_0 = a_0 b_0$, $c_1 = a_0 b_1 + a_1 b_0$, $c_2 = a_0 b_2 + a_1 b_1 + a_2 b_0$, …,

$$c_n = a_0 b_n + a_1 b_{n-1} + a_2 b_{n-2} + \ldots + a_n b_0 \ , \ldots .$$

Последовательность $\{c_n\}$ называется свёрткой последовательностей $\{a_n\}$, $\{b_n\}$.

Определённые операции сложения и умножения коммутативны, ассоциативны, умножение дистрибутивно относительно сложения. Имеются нейтральные элементы: $F(z) \equiv 0$ – производящая функция нулевой последовательности, $F(z) \equiv 1$ – производящая функция последовательности $1, 0, 0, \ldots, 0, \ldots$. Очевидно, $-F(z)$ – противоположный элемент: $F(z) + (-F(z)) \equiv 0$. Таким образом, множество производящих функций вместе с определёнными операциями сложения и умножения образует коммутативное кольцо с единицей.

Если $a_0 \neq 0$, то для производящей функции $F(z)=\sum_{n=0}^{\infty}a_n z^n$ существует обратная производящая функция $F^{-1}(z)$: $F(z)\cdot F^{-1}(z) \equiv 1$. Действительно, если $F^{-1}(z)=\sum_{n=0}^{\infty}d_n z^n$, то по определению произведения

$$a_0 d_0 = 1 , \ a_0 d_1 + a_1 d_0 = 0 , \ a_0 d_2 + a_1 d_1 + a_2 d_0 = 0 , \ldots .$$

Если $a_0 \neq 0$, то из первого равенства можно найти d_0, из второго d_1, и так далее. Ясно, что если $a_0 = 0$, то производящая функция не имеет обратной.

Можно умножать производящие функции на числа: если $F(z)=\sum_{n=0}^{\infty}a_n z^n$, то

$$cF(z)=\sum_{n=0}^{\infty}(ca_n)z^n .$$

Сделаем важное замечание о работе с производящими функциями. Можно было бы ограничиться записью формальной бесконечной суммы: $\sum_{n=0}^{\infty}a_n z^n$, так как эта запись содержит всю информацию о последовательности $\{a_n\}$. Однако удобно пользоваться тем, что обычно в некоторой окрестности нуля (при малых $|z|$, а иногда и при любых z) ряд сходится, т.е. представляет собой некоторую функцию, определённую на области сходимости. Мы обозначили её $F(z)$. В таком случае, как известно, этот ряд является рядом Маклорена функции $F(z)$, т.е. $a_n = \dfrac{F^{(n)}(0)}{n!}$. Другими словами, бесконечно диффе-

ренцируемая в некоторой окрестности нуля функция $F(z)$, при достаточно слабых дополнительных условиях (грубо говоря, почти всегда) является производящей функцией для последовательности $a_n = \frac{F^{(n)}(0)}{n!}$:

$$F(z) = \sum_{n\geq 0} \frac{F^{(n)}(0)}{n!} z^n .$$

Мы уже пользовались тем, что $1 + t + t^2 + t^3 + \ldots = \frac{1}{1-t}$, т.е. $\frac{1}{1-z}$ является производящей функцией для последовательности $1, 1, 1, \ldots, 1, \ldots$. Было получено также

$$\frac{1}{(1-z)^2} = (1 + z + z^2 + \ldots)(1 + z + z^2 + \ldots) = 1 + 2z + 3z^2 + \ldots = \sum_{k\geq 0} (k+1) z^k .$$

Справедлива более общая формула:

$$\frac{1}{(1-z)^n} = \sum_{k\geq 0} C_{n+k-1}^k z^k . \qquad (*)$$

Действительно, k-я производная функции $\frac{1}{(1-z)^n}$ при $z = 0$ равна $n(n+1)(n+2)\ldots(n+k-1)$, а значит коэффициент при z^k в разложении этой функции в ряд Маклорена равен

$$\frac{n(n+1)(n+2)\ldots(n+k-1)}{k!} = \frac{(n+k-1)!}{k!(n-1)!} = C_{n+k-1}^k .$$

3.3 Производящие функции для основных типов сочетаний с повторениями

Мы уже получили производящую функцию для последовательности

$$\bar{C}_n^0, \bar{C}_n^1, \bar{C}_n^2, \bar{C}_n^3, \ldots .$$

Действительно, так как $\bar{C}_n^k = C_{n+k-1}^k$, то $\frac{1}{(1-z)^n} = \sum_{k\geq 0} C_{n+k-1}^k z^k = \sum_{k\geq 0} \bar{C}_n^k z^k$.

Многие комбинаторные задачи требуют рассмотрения сочетаний с ограничениями на число повторений. Обозначим: $\bar{C}_n^k(j)$ – число k-сочетаний элементов n-множества, в которых каждый элемент повторяется не более j раз. Построим производящую функцию для последовательности $\bar{C}_n^0(j), \bar{C}_n^1(j), \bar{C}_n^2(j), \bar{C}_n^3(j), \ldots$. Поступим так же, как при решении задачи о размене. Пусть выборка производится из множества $X = \{x_1, x_2, \ldots, x_n\}$. Сначала учитываем сочетания, содержащие только x_1:

$$A_1 = 1 + x_1 + x_1^2 + \ldots + x_1^j .$$

Это формальная сумма. Каждое слагаемое – сочетание, содержащее только элементы x_1. Слагаемое $1 = x^0$ – пустое сочетание, не содержащее никаких элементов. Теперь запишем такую же сумму для сочетаний, в которых могут участвовать x_1 и x_2:

$$A_2 = A_1 + A_1 x_2 + A_1 x_2^2 + \ldots + A_1 x_2^j = A_1 (1 + x_2 + x_2^2 + \ldots + x_2^j) =$$
$$= (1 + x_1 + x_1^2 + \ldots + x_1^j)(1 + x_2 + x_2^2 + \ldots + x_2^j).$$

Продолжая эти рассуждения, заключаем, что все рассматриваемые сочетания представимы слагаемыми в сумме, полученной при раскрытии скобок в выражении:

$$(1 + x_1 + x_1^2 + \ldots + x_1^j)(1 + x_2 + x_2^2 + \ldots + x_2^j)\ldots(1 + x_n + x_n^2 + \ldots + x_n^j).$$

Действительно, каждое слагаемое имеет вид $x_1^{k_1}x_2^{k_2}\ldots x_n^{k_n}$ и представляет сочетание, в которое x_1 входит k_1 раз, …, x_n входит k_n раз. При этом $k_1 \le j,\ \ldots,\ k_n \le j$ и $k_1 + k_2 + \ldots + k_n = k$ – объём сочетания.

Так как нас интересует лишь ***число*** сочетаний данного объёма, то заменим все x_i на переменную z. После раскрытия скобок в выражении

$$\left(1 + z + z^2 + \ldots + z^j\right)\ldots\left(1 + z + z^2 + \ldots + z^j\right) = \left(1 + z + z^2 + \ldots + z^j\right)^n$$

слагаемые, соответствующие сочетаниям объёма k, будут подобными. Приводя подобные, получим искомые коэффициенты $\bar{C}_n^k(j)$. Таким образом, производящая функция для последовательности $\bar{C}_n^0(j), \bar{C}_n^1(j), \bar{C}_n^2(j), \bar{C}_n^3(j), \ldots$ построена:

$$F(z) = \left(1 + z + z^2 + \ldots + z^j\right)^n.$$

Действуя по аналогии, можно построить производящие функции для сочетаний с другими ограничениями на число повторений элементов. Например, для последовательности сочетаний элементов n-множества, в которых x_1 может повторяться не более j_1 раз, x_2 не более j_2 раз, …, x_n не более j_n раз производящая функция имеет вид: $F(z) = \left(1 + z + z^2 + \ldots + z^{j_1}\right)\left(1 + z + z^2 + \ldots + z^{j_2}\right)\ldots\left(1 + z + z^2 + \ldots + z^{j_n}\right)$.

Далее, в примерах, встречаются и другие виды ограничений.

Пример 1. Найти число 10-сочетаний элементов множества $M = \{a, b, c\}$, если элемент a нельзя повторять более 5 раз, b повторяется не более 3 раз, а c повторяется в каждом сочетании или 2, или 4 раза.

Решение. Обозначим C_k – число k-сочетаний указанного вида. Производящая функция для последовательности $C_0, C_1, C_2, C_3, \ldots$ имеет вид:

$$F(z) = \left(1 + z + z^2 + z^3 + z^4 + z^5\right)\left(1 + z + z^2 + z^3\right)\left(z^2 + z^4\right).$$

В задаче требуется найти коэффициент при z^{10}. В случае, когда раскрытие всех скобок затруднительно, можно привлекать другие приёмы. В нашем примере ясно, что слагаемые $1, z, z^2$ в первой скобке, а также слагаемое 1 во второй скобке не могут внести вклад в искомый коэффициент. Поэтому у функции

$$F_1(z) = \left(z^3 + z^4 + z^5\right)\left(z + z^2 + z^3\right)\left(z^2 + z^4\right)$$

коэффициент при z^{10} такой же, как и у функции $F(z)$. Далее,

$$F_1(z) = z^6\left(1 + z + z^2\right)\left(1 + z + z^2\right)\left(1 + z^2\right) = z^6 F_2(z).$$

Коэффициент при z^{10} у функции $F_1(z)$ совпадает, очевидно, с коэффициентом при z^4 у функции $F_2(z)$. Теперь найти этот коэффициент уже нетрудно:

$$F_2(z) = \left(1 + z + z^2\right)^2(1 + z) = \left(1 + z^2 + z^4 + 2z + 2z^2 + 2z^3\right)\left(1 + z^2\right) =$$

$$= \ldots + 3z^2 \cdot z^2 + z^4 \cdot 1 + \ldots = \ldots + 4z^4 + \ldots.$$

Замечание. В разделе 1 (теорема 4) было установлено взаимно однозначное соответствие между k-сочетаниями с повторениями элементов n-множества и способами размещения k одинаковых элементов в n различных ячейках. Ограничениям на число повторений элементов соответствуют аналогичные ограничения на вместимость ячеек.

Поэтому многочисленные задачи о размещении элементов по ячейкам успешно решаются с помощью производящих функций. В следующем примере рассмотрена известная задача.

Пример 2. Сколько существует способов разместить r одинаковых элементов в n различных ячейках так, чтобы не было пустых ячеек?

Решение. Пусть D_n^r – число указанных размещений. Тогда производящая функция для последовательности $D_n^0, D_n^1, D_n^2, \ldots$ имеет вид:

$$F(z) = \left(z + z^2 + z^3 + \ldots\right)^n = z^n \left(1 + z + z^2 + \ldots\right)^n = \frac{z^n}{(1-z)^n}.$$

Пользуясь формулой (*), получаем:

$$F(z) = \frac{z^n}{(1-z)^n} = z^n \sum_{k \geq 0} C_{n+k-1}^k z^k = \sum_{k \geq 0} C_{n+k-1}^k z^{k+n}.$$

Обозначим: $r = n + k$. Тогда $F(z) = \sum_{r \geq n} C_{r-1}^{r-n} z^r$. Следовательно, $D_n^r = C_{r-1}^{r-n}$, если $r \geq n$, и, конечно, $D_n^r = 0$, если $r < n$.

3.4 Решение рекуррентных уравнений с помощью производящих функций

Одно из самых важных применений производящих функций – решение рекуррентных уравнений. Мы уже познакомились с методами решения некоторых уравнений. Однако рекуррентности встречаются так часто, что требуются и более общие подходы.

Для решения рекуррентных уравнений можно пользоваться следующим алгоритмом.

1. Записать рекуррентное уравнение так, чтобы выразить n-й член последовательности a_n через предыдущие.
2. Умножить обе части уравнения на z^n. Просуммировать по всем n, для которых справедливо уравнение. В левой части получим $F(z) = \sum a_n z^n$ – производящую функцию для нашей последовательности (без нескольких первых слагаемых). Правую часть нужно преобразовать так, чтобы тоже получилось выражение, включающее $F(z)$.
3. Выразить $F(z)$ из полученного равенства.
4. Разложить $F(z)$ в степенной ряд. Коэффициент при z^n – искомое выражение для a_n.

Пример 3. Решить рекуррентное уравнение

$$a_n = a_{n-1} + 2a_{n-2} + (-1)^n \text{ при } n \geq 2, \quad a_0 = a_1 = 1.$$

Решение. Применим указанный алгоритм.

$$\sum_{n \geq 2} a_n z^n = \sum_{n \geq 2} a_{n-1} z^n + 2 \sum_{n \geq 2} a_{n-2} z^n + \sum_{n \geq 2} (-1)^n z^n,$$

$$F(z) - 1 - z = z\left(F(z) - 1\right) + 2z^2 F(z) + \sum_{n \geq 2} (-1)^n z^n.$$

Здесь $F(z) = \sum_{n \geq 0} a_n z^n$ – производящая функция последовательности $\{a_n\}$. Мы воспользовались тем, что $a_0 = a_1 = 1$. Далее, так как

$$\sum_{n \geq 2} (-1)^n z^n = z^2 - z^3 + z^4 - \ldots = z^2 \left(1 + (-z) + (-z)^2 + (-z)^3 + \ldots\right) = \frac{z^2}{1+z},$$

то получаем:

$$F(z)-1=zF(z)+2z^2F(z)+\frac{z^2}{1+z},$$

$$F(z)\left(1-z-2z^2\right)=\frac{z^2}{1+z}+1=\frac{z^2+z+1}{1+z},$$

$$F(z)=\frac{z^2+z+1}{(1+z)\left(1-z-2z^2\right)}=\frac{z^2+z+1}{(1+z)^2(1-2z)}.$$

Производящая функция получена. Для разложения в ряд представим её в виде суммы простейших дробей:

$$\frac{z^2+z+1}{(1+z)^2(1-2z)}=\frac{A}{1+z}+\frac{B}{(1+z)^2}+\frac{C}{1-2z},$$

$$z^2+z+1=A(1+z)(1-2z)+B(1-2z)+C(1+z)^2.$$

Полученное равенство является тождеством. Можно найти A, B, C подставляя различные значения z, или сравнивая коэффициенты при одинаковых степенях z.

При $z=-1$: $1=3B$, $B=\frac{1}{3}$,

при $z=\frac{1}{2}$: $\frac{7}{4}=\frac{9}{4}C$, $C=\frac{7}{9}$,

коэффициент при z^2: $1=-2A+C$, $A=\frac{C-1}{2}=-\frac{1}{9}$.

Итак, $F(z)=-\frac{1}{9}\frac{1}{1+z}+\frac{1}{3}\frac{1}{(1+z)^2}+\frac{7}{9}\frac{1}{1-2z}=$

$$=-\frac{1}{9}\sum_{n\ge0}(-1)^n z^n+\frac{1}{3}\sum_{n\ge1}(-1)^{n+1}nz^{n-1}+\frac{7}{9}\sum_{n\ge0}2^n z^n=$$

$$=\sum_{n\ge0}\left(\frac{(-1)^{n+1}}{9}+\frac{(-1)^n}{3}(n+1)+\frac{7}{9}\cdot2^n\right)z^n.$$

Следовательно, $a_n=\frac{(-1)^{n+1}+3(-1)^n(n+1)+7\cdot2^n}{9}$.

Этот пример, возможно, проще было бы решить методом, рассмотренным в разделе 2, но нам важен новый, более универсальный подход. Применим его для решения ещё одной задачи.

Пример 4. Сколько имеется способов расставить скобки в произведении

$$x_0\cdot x_1\cdot x_2\cdot\ldots\cdot x_n$$

так, чтобы порядок умножений был полностью определён?

Решение. Обозначим искомую величину V_n. Ясно, что в «произведении» x_0 скобки не нужны; произведение $x_0\cdot x_1$ тоже можно вычислить единственным способом. Поэтому $V_0=1$, $V_1=1$. Для $n=2$ есть уже 2 способа: $(x_0\cdot x_1)\cdot x_2$, $x_0\cdot(x_1\cdot x_2)$, т.е. $V_2=2$. Постараемся получить рекуррентное соотношение для последовательности V_n.

Какое-то из умножений выполняется последним. Допустим, это умножение, расположенное между x_k и x_{k+1}. Перемножаются произведения $x_0\cdot x_1\cdot\ldots\cdot x_k$ (а для него имеется V_k способов вычисления) и $x_{k+1}\cdot x_{k+2}\cdot\ldots\cdot x_n$ (для него имеется V_{n-k-1} спосо-

бов вычисления). Ясно, что всего способов вычисления $V_k \cdot V_{n-k-1}$ (если последнее умножение расположено на указанном месте). Меняя k, получим общее число способов:

$$V_n = V_0 \cdot V_{n-1} + V_1 \cdot V_{n-2} + \ldots + V_{n-1} \cdot V_0 .$$

Получено рекуррентное уравнение, однако V_n выражается через все предыдущие члены последовательности. Нам поможет то, что в правой части соотношения – $(n-1)$-й член последовательности, которая является ***свёрткой*** последовательности $V_0, V_1, V_2, \ldots$ с ней же самой. Как было показано в пункте 3.2, производящая функция свёртки совпадает с произведением производящих функций. Поэтому, умножая найденное соотношение на z^n и суммируя по $n = 1, 2, 3, \ldots$, получим:

$$F(z) - V_0 = zF^2(z).$$

Так как $V_0 = 1$, то можно записать $zF^2(z) - F(z) + 1 = 0$. Решая квадратное уравнение, находим: $F(z) = \dfrac{1 \pm \sqrt{1-4z}}{2z}$. Здесь нужно взять знак «–», так как должно быть $F(0) = V_0 = 1$, но $\lim\limits_{z\to 0} \dfrac{1+\sqrt{1-4z}}{2z} = \infty$, $\lim\limits_{z\to 0} \dfrac{1-\sqrt{1-4z}}{2z} = \lim\limits_{z\to 0} \dfrac{-\frac{-4}{2\sqrt{1-4z}}}{2} = \lim\limits_{z\to 0} \dfrac{1}{\sqrt{1-4z}} = 1$.

Итак, производящая функция найдена: $F(z) = \dfrac{1-\sqrt{1-4z}}{2z}$.

Для разложения $F(z)$ в ряд, воспользуемся биномиальным разложением:

$$(1+x)^{\alpha} = 1 + \alpha x + \frac{\alpha(\alpha-1)}{2!}x^2 + \frac{\alpha(\alpha-1)(\alpha-2)}{3!}x^3 + \ldots .$$

Подставим $x = -4z, \quad \alpha = \dfrac{1}{2}$:

$$(1-x)^{\frac{1}{2}} = 1 + \frac{1}{2}(-4z) + \frac{\frac{1}{2}\left(\frac{1}{2}-1\right)}{2!}(-4z)^2 + \frac{\frac{1}{2}\left(\frac{1}{2}-1\right)\left(\frac{1}{2}-2\right)}{3!}(-4z)^3 + \ldots .$$

Заметим: все знаки, кроме знака первого слагаемого, отрицательны. Поэтому

$$1 - \sqrt{1-4z} = \frac{1}{2}(4z) + \frac{1}{2!\,2^2}(4z)^2 + \frac{1\cdot 3}{3!\,2^3}(4z)^3 + \frac{1\cdot 3\cdot 5}{4!\,2^4}(4z)^4 + \ldots ,$$

$$\frac{1-\sqrt{1-4z}}{2z} = 1 + \frac{1}{2!}2z + \frac{1\cdot 3}{3!}2^2 z^2 + \frac{1\cdot 3\cdot 5}{4!}2^3 z^3 + \ldots .$$

Итак, задача решена, для $n \geq 1$

$$V_n = \frac{1\cdot 3\cdot 5\cdot \ldots \cdot (2n-1)}{(n+1)!}2^n .$$

Если умножить числитель и знаменатель на $2\cdot 4\cdot 6\cdot \ldots \cdot 2n$, то можно получить более компактную запись:

$$V_n = \frac{1\cdot 3\cdot 5\cdot \ldots \cdot (2n-1)\cdot 2\cdot 4\cdot \ldots \cdot 2n}{(n+1)!\cdot 2\cdot 4\cdot \ldots \cdot 2n}2^n = \frac{(2n)!}{(n+1)!\,n!\,2^n}2^n = \frac{(2n)!}{(n+1)(n!)^2} = \frac{1}{n+1}C_{2n}^n .$$

Замечание. Числа $\dfrac{1}{n+1}C_{2n}^n$ называются ***числами Каталана***, они встречаются во многих комбинаторных задачах.

3.5 Экспоненциальные производящие функции

Аппарат производящих функций включает разнообразные инструменты. Для любой линейно независимой системы функций

$$\varphi_0(z), \varphi_1(z), \varphi_2(z), \varphi_3(z), \ldots$$

можно ввести понятие производящей функции для последовательности $a_0, a_1, \ldots, a_n, \ldots$ по этой системе:

$$A(z) = \sum_{n=0}^{\infty} a_n \varphi_n(z).$$

Мы рассмотрели случай $\varphi_n(z) = z^n$ (обычные производящие функции). Часто применяются также экспоненциальные производящие функции – для $\varphi_n(z) = \frac{z^n}{n!}$. Итак, экспоненциальной производящей функцией (эпф) для последовательности $a_0, a_1, \ldots, a_n, \ldots$ называется ряд

$$E(z) = \sum_{n=0}^{\infty} a_n \frac{z^n}{n!}.$$

Как и обычные производящие функции, эпф можно складывать, умножать на числа: $E_1(z) + E_2(z) = \sum_{n=0}^{\infty} a_n \frac{z^n}{n!} + \sum_{n=0}^{\infty} b_n \frac{z^n}{n!} = \sum_{n=0}^{\infty} (a_n + b_n) \frac{z^n}{n!}$ – эпф для последовательности $a_0 + b_0, a_1 + b_1, \ldots, a_n + b_n, \ldots$;

$\alpha E(z) = \sum_{n=0}^{\infty} (\alpha a_n) \frac{z^n}{n!}$ – эпф для последовательности $\alpha a_0, \alpha a_1, \ldots, \alpha a_n, \ldots$.

Сложнее определяется произведение эпф:

$$E_1(z) \cdot E_2(z) = \left(a_0 + a_1 \frac{z}{1!} + a_2 \frac{z^2}{2!} + a_3 \frac{z^3}{3!} + \ldots \right) \left(b_0 + b_1 \frac{z}{1!} + b_2 \frac{z^2}{2!} + b_3 \frac{z^3}{3!} + \ldots \right) =$$

$$= a_0 b_0 + (a_0 b_1 + a_1 b_0) \frac{z}{1!} + (a_0 b_2 + 2a_1 b_1 + a_2 b_0) \frac{z^2}{2!} + \ldots = \sum_{n \geq 0} c_n \frac{z^n}{n!},$$

где $c_n = C_n^0 a_0 b_n + C_n^1 a_1 b_{n-1} + C_n^2 a_2 b_{n-2} + \ldots + C_n^n a_n b_0 = \sum_{k=0}^{n} C_n^k a_k b_{n-k}$. Такая последовательность c_n называется биномиальной свёрткой последовательностей a_n и b_n. Таким образом, эпф для биномиальной свёртки совпадает с произведением эпф для свёртываемых последовательностей.

Можно проверить, что операции сложения и умножения эпф ассоциативны и коммутативны, умножение дистрибутивно относительно сложения. Если $a_0 \neq 0$, то для $E(z) = \sum_{n=0}^{\infty} a_n \frac{z^n}{n!}$ существует обратная эпф: $E(z) \cdot E^{-1}(z) = \left(\sum_{n=0}^{\infty} a_n \frac{z^n}{n!} \right) \left(\sum_{n=0}^{\infty} d_n \frac{z^n}{n!} \right) = 1$. Здесь 1 – эпф для последовательности $1, 0, 0, \ldots, 0, \ldots$.

Можно сказать, что эпф для последовательности $a_0, a_1, \ldots, a_n, \ldots$ – это обычная производящая функция для последовательности $a_0, \frac{a_1}{1!}, \frac{a_2}{2!}, \frac{a_3}{3!}, \ldots, \frac{a_n}{n!}, \ldots$, а последовательности такого типа часто встречаются при решении комбинаторных задач.

3.6 Экспоненциальные производящие функции для основных типов размещений с повторениями

Известная формула бинома Ньютона

$$(1+z)^n = C_n^0 + C_n^1 z + C_n^2 z^2 + \ldots + C_n^n z^n = A_n^0 + A_n^1 \frac{z}{1!} + A_n^2 \frac{z^2}{2!} + \ldots + A_n^n \frac{z^n}{n!}$$

показывает, что многочлен $(1+z)^n$ является производящей функцией для последовательности $C_n^0, C_n^1, C_n^2, \ldots, C_n^n$ сочетаний без повторений, а также является экспоненциальной производящей функцией для последовательности $A_n^0, A_n^1, A_n^2, \ldots, A_n^n$ размещений без повторений (напомним: $C_n^k = \frac{n!}{k!(n-k)!} = \frac{1}{k!} A_n^k$).

Для последовательности $\bar{A}_n^0, \bar{A}_n^1, \bar{A}_n^2, \bar{A}_n^3, \ldots$ размещений с повторениями, зная формулу $\bar{A}_n^k = n^k$, легко записать и обычную производящую функцию:

$$F(z) = \sum_{k \geq 0} n^k z^k = \sum_{k \geq 0} (nz)^k = \frac{1}{1-nz},$$

и экспоненциальную производящую функцию:

$$E(z) = \sum_{k \geq 0} n^k \frac{z^k}{k!} = \sum_{k \geq 0} \frac{(nz)^k}{k!} = e^{nz} .$$

Однако, как и в случае сочетаний, многие задачи требуют рассмотрения размещений с ограничениями на число повторений. Обозначим: $\bar{A}_n^k(j)$ – число k-размещений элементов n-множества, в которых каждый элемент повторяется не более j раз. Построим экспоненциальную производящую функцию для последовательности $\bar{A}_n^0(j), \bar{A}_n^1(j), \bar{A}_n^2(j), \bar{A}_n^3(j), \ldots$.

Пусть выборка производится из множества $X = \{x_1, x_2, \ldots, x_n\}$. Напомним: при составлении производящей функции для последовательности сочетаний с аналогичными ограничениями мы использовали символическое выражение

$$\left(1 + x_1 + x_1^2 + \ldots + x_1^j\right)\left(1 + x_2 + x_2^2 + \ldots + x_2^j\right)\ldots\left(1 + x_n + x_n^2 + \ldots + x_n^j\right).$$

При раскрытии скобок каждое произведение $x_1^{k_1} x_2^{k_2} \ldots x_n^{k_n}$ представляет сочетание, в которое x_1 входит k_1 раз, …, x_n входит k_n раз, причём $k_1 + k_2 + \ldots + k_n = k$ – объём сочетания. Как известно, переставляя элементы, из этого k-сочетания можно получить $\frac{k!}{k_1! k_2! \ldots k_n!}$ k-размещений. Мы не можем сразу заменить все x_i на z (как это было сделано для сочетаний), потому что разные k-сочетания будут давать разное количество k-размещений. Поэтому рассмотрим выражение:

$$E = \left(1 + \frac{x_1}{1!} + \frac{x_1^2}{2!} + \ldots + \frac{x_1^j}{j!}\right)\left(1 + \frac{x_2}{1!} + \frac{x_2^2}{2!} + \ldots + \frac{x_2^j}{j!}\right)\ldots\left(1 + \frac{x_n}{1!} + \frac{x_n^2}{2!} + \ldots + \frac{x_n^j}{j!}\right).$$

Теперь, после раскрытия скобок, каждое произведение $\frac{x_1^{k_1}}{k_1!} \cdot \frac{x_2^{k_2}}{k_2!} \cdot \ldots \cdot \frac{x_n^{k_n}}{k_n!}$ символизирует k-сочетание $x_1^{k_1} x_2^{k_2} \ldots x_n^{k_n}$ с коэффициентом $\frac{1}{k_1! k_2! \ldots k_n!}$. После замены всех x_i на z и приведения подобных мы получим сумму, в которой коэффициент при z^k в $k!$ раз меньше числа k-размещений:

$$E(z) = \left(1 + \frac{z}{1!} + \frac{z^2}{2!} + \ldots + \frac{z^j}{j!}\right)^n = \sum_{k \geq 0} \frac{\bar{A}_n^k(j)}{k!} z^k = \sum_{k \geq 0} \bar{A}_n^k(j) \frac{z^k}{k!}.$$

Экспоненциальная производящая функция для последовательности $\bar{A}_n^k(j)$, $k = 0, 1, 2, \ldots$ построена. Аналогичным образом строятся экспоненциальные производящие функции для размещений с другими ограничениями на число повторений. Например, если для каждого элемента x_i имеется своё ограничение, число повторений ограничено сверху числом j_i, то экспоненциальная производящая функция для последовательности размещений имеет вид:

$$E(z) = \left(1 + \frac{z}{1!} + \frac{z^2}{2!} + \ldots + \frac{z^{j_1}}{j_1!}\right) \cdot \left(1 + \frac{z}{1!} + \frac{z^2}{2!} + \ldots + \frac{z^{j_2}}{j_2!}\right) \cdot \ldots \cdot \left(1 + \frac{z}{1!} + \frac{z^2}{2!} + \ldots + \frac{z^{j_n}}{j_n!}\right).$$

Часто встречается требование, чтобы каждый элемент хотя бы один раз участвовал в размещении. Аналогичные рассуждения приводят в этом случае к функции

$$E(z) = \left(\frac{z}{1!} + \frac{z^2}{2!} + \frac{z^3}{3!} + \ldots\right)^n = \left(e^z - 1\right)^n.$$

Замечание. В разделе 1 (теорема 3) было установлено взаимно однозначное соответствие между ***k***-размещениями с повторениями элементов ***n***-множества и способами размещения ***k*** различных элементов в ***n*** различных ячейках. Как и в случае сочетаний, ограничениям на число повторений элементов соответствуют аналогичные ограничения на вместимость ячеек. Поэтому задачи о размещении одинаковых элементов по ячейкам решаются с помощью обыкновенных производящих функций, а задачи о размещении различных элементов – с помощью экспоненциальных производящих функций.

Пример 5. Сколько различных 6-буквенных слов можно составить из букв ***A***, ***B***, ***C***, если буквы ***A*** и ***B*** можно повторять не более 2-х раз?

Решение. Пусть W_k – число ***k***-буквенных слов указанного типа. Экспоненциальная производящая функция для последовательности $W_0, W_1, W_2, \ldots$ имеет вид:

$$E(z) = \left(1 + \frac{z}{1!} + \frac{z^2}{2!}\right) \cdot \left(1 + \frac{z}{1!} + \frac{z^2}{2!}\right) \cdot \left(1 + \frac{z}{1!} + \frac{z^2}{2!} + \frac{z^3}{3!} + \ldots\right) = \sum_{k \geq 0} W_k \frac{z^k}{k!}.$$

Чтобы найти W_6 мы должны раскрыть скобки, привести подобные, найти коэффициент при z^6 и умножить его на $6!$.

Проведём упрощения. Первые 2 слагаемые в последней скобке не могут участвовать в произведении, содержащем z^6. Не будут участвовать и слагаемые, содержащие z в 7-й и более высоких степенях. Поэтому коэффициент при z^6 у функции $E(z)$ такой же, как и у функции

$$E_1(z) = \left(1 + \frac{z}{1!} + \frac{z^2}{2!}\right) \cdot \left(1 + \frac{z}{1!} + \frac{z^2}{2!}\right) \cdot \left(\frac{z^2}{2!} + \frac{z^3}{3!} + \frac{z^4}{4!} + \frac{z^5}{5!} + \frac{z^6}{6!}\right).$$

Вынесем z^2 из последней скобки: $E_1(z) = z^2 E_2(z)$, где

$$E_2(z) = \left(1 + \frac{z}{1!} + \frac{z^2}{2!}\right) \cdot \left(1 + \frac{z}{1!} + \frac{z^2}{2!}\right) \cdot \left(\frac{1}{2!} + \frac{z}{3!} + \frac{z^2}{4!} + \frac{z^3}{5!} + \frac{z^4}{6!}\right).$$

Нам нужно найти коэффициент при z^4. Выпишем все подобные слагаемые, содержащие z^4:

$$1 \cdot 1 \cdot \frac{z^4}{6!} + 1 \cdot z \cdot \frac{z^3}{5!} + 1 \cdot \frac{z^2}{2!} \cdot \frac{z^2}{4!} + z \cdot 1 \cdot \frac{z^3}{5!} + z \cdot z \cdot \frac{z^2}{4!} + z \cdot \frac{z^2}{2!} \cdot \frac{z}{3!} + \frac{z^2}{2!} \cdot 1 \cdot \frac{z^2}{4!} +$$

$$+\frac{z^2}{2!}\cdot 1\cdot\frac{z^2}{4!}+\frac{z^2}{2!}\cdot z\cdot\frac{z}{3!}+\frac{z^2}{2!}\cdot\frac{z^2}{2!}\cdot\frac{1}{2}=$$

$$=z^4\left(\frac{1}{6!}+\frac{1}{5!}+\frac{1}{2\cdot 4!}+\frac{1}{5!}+\frac{1}{4!}+\frac{1}{2\cdot 3!}+\frac{1}{2\cdot 4!}+\frac{1}{2\cdot 3!}+\frac{1}{8}\right).$$

Умножим коэффициент на $6!=720$ и получим требуемый результат:

$$W_6=1+6+15+6+30+60+15+60+90=283\ .$$

4 Элементы теории графов

4.1 Бинарные отношения

Пусть A, B – непустые множества, $A\times B$ – их декартово произведение. Любое подмножество $\alpha\subseteq A\times B$ называется ***бинарным отношением*** на множествах A, B. Если $A=B$, то бинарное отношение называется ***однородным***.

Укажем основные способы задания бинарных отношений.

1) Бинарное отношение может быть задано перечислением всех своих элементов.

Пример 1. Пусть $A=\{a,b,c,d\}$. Например,

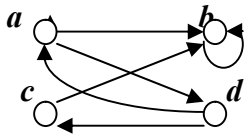
$$\alpha=\{(a,b),(a,d),(b,b),(b,d),(c,b),(d,a),(d,c)\}-$$

однородное бинарное отношение на множестве A.

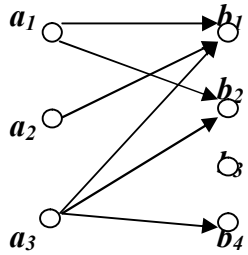
2) Бинарное отношение можно задать матрицей из нулей и единиц. Строки матрицы обычно соответствуют элементам множества A, столбцы – элементам множества B. Для однородного отношения из примера 1 получается матрица

$$\begin{array}{c} \\ a\\ b\\ c\\ d\end{array}\begin{array}{c}\begin{array}{cccc}a & b & c & d\end{array}\\ \begin{pmatrix}0&1&0&1\\0&1&0&1\\0&1&0&0\\1&0&1&0\end{pmatrix}\end{array}.$$

3) Бинарное отношение может быть задано графически, т.е. графом. Элементам множества A соответствуют точки – вершины графа. Если пара (a,b) входит в данное отношение, то из точки a в точку b проводится стрелка (дуга). Для рассмотренного примера получим граф:



Неоднородное бинарное отношение (на множествах A, B) тоже можно задавать графически. Получается так называемый ***двудольный*** граф:



Рассмотрим наиболее важные свойства однородных бинарных отношений.

1. Рефлексивность. Отношение $\alpha\subseteq A\times A$ называется ***рефлексивным***, если

$$(a,a)\in\alpha\quad\forall a\in A\ .$$

2. Симметричность. Отношение $\alpha \subseteq A \times A$ называется ***симметричным***, если

$$(a,b) \in \alpha \quad \Leftrightarrow \quad (b,a) \in \alpha .$$

3. Транзитивность. Отношение $\alpha \subseteq A \times A$ называется ***транзитивным***, если

$$(a,b) \in \alpha, \quad (b,c) \in \alpha \quad \Rightarrow \quad (a,c) \in \alpha .$$

Если бинарное отношение является одновременно рефлексивным, симметричным и транзитивным, то оно называется ***отношением эквивалентности***. Отношение эквивалентности на множестве A задаёт разбиение A на непересекающиеся классы эквивалентных элементов: $A = A_1 \cup A_2 \cup \ldots \cup A_k$, $A_i \cap A_j = \varnothing$ при $i \neq j$. Внутри каждого класса A_i все элементы попарно эквивалентны, а элементы из разных классов не эквивалентны.

Важную роль в математике играют также отношения порядка. Однородное бинарное отношение $\alpha \subseteq A \times A$ называется ***отношением частичного порядка*** на множестве A (или просто частичным порядком на A), если оно рефлексивно, транзитивно и ***антисимметрично***, т.е.

$$(a,b) \in \alpha, \quad (b,a) \in \alpha \quad \Rightarrow \quad a = b .$$

Если, кроме того, $\forall a,b \in A \quad (a,b) \in \alpha$ или $(b,a) \in \alpha$, то α называется ***полным порядком***. Множество, на котором определено отношение полного порядка называется ***вполне упорядоченным***. Примером может служить множество действительных чисел R: отношение $\geq$ («больше») является полным порядком на R.

Важным классом бинарных отношений (не обязательно однородных) являются отображения. Бинарное отношение $f \subseteq A \times B$ называется ***отображением*** множества A во множество B, если

$$1) \ \forall a \in A \quad \exists b \in B : \quad (a,b) \in f ,$$

$$2) \ (a,b_1) \in f, \quad (a,b_2) \in f \quad \Rightarrow \quad b_1 = b_2 .$$

Для отображений вместо $(a,b) \in f$ обычно пишут $f(a) = b$. Среди отображений выделяют инъективные, сюръективные, биективные. Эти понятия подробно обсуждаются в основном вузовском курсе математики, мы будем считать их известными.

4.2 Начальные понятия теории графов

Ориентированным графом (***орграфом***) называется пара множеств

$$G = \langle A, \bar{V} \rangle ,$$

где A – произвольное непустое множество (его элементы называются ***вершинами*** графа), $\bar{V} \subseteq A \times A$ – множество ***рёбер***. В орграфе рёбра называются также ***дугами***. Фактически $\bar{V}$ – некоторое бинарное отношение на множестве вершин, оно называется ***отношением смежности***.

Неориентированным графом (***неорграфом***) называется пара множеств

$$G = \langle A, V \rangle ,$$

где $A \neq \varnothing$ (множество вершин), $V = \left\{ \{a_i, a_j\} \,\middle|\, a_i, a_j \in A \right\}$ – некоторое множество неупорядоченных пар вершин (множество рёбер). Как и в случае ориентированного графа, если в графе имеется ребро $\{a_i, a_j\}$, то вершины a_i, a_j называются ***смежными***. Для неорграфа таким образом определённое бинарное ***отношение смежности*** является, очевидно, симметричным.

Ясно, что отношение смежности (заданное, например, своей матрицей – ***матрицей смежности***) полностью определяет граф.

Ребро $\{a_i, a_j\}$ называется ***инцидентным*** вершинам a_i, a_j, которые это ребро соединяет. Неорграф можно задать ***матрицей инцидентности*** – это матрица из 0 и 1, строки которой соответствуют вершинам графа, а столбцы – рёбрам. Элемент матрицы равен 1, если соответствующие ребро и вершина инцидентны, и 0 в противном случае.

В орграфе дуга (a_i, a_j) также называется инцидентной своему началу a_i и концу a_j. Матрица инцидентности для орграфа в столбце, соответствующем дуге (a_i, a_j), содержит 1 – в строке, соответствующей a_i, –1 – в строке, соответствующей a_j, 0 на остальных местах. Для петли условимся применять значение 2. (Заметим, что могут применяться и другие правила построения матрицы инцидентности для орграфа).

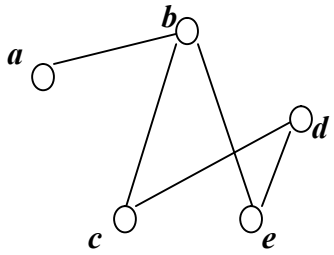
Две вершины в графе могут быть соединены несколькими рёбрами. Однако мы будем рассматривать только графы ***без кратных рёбер***. Ребро $\{a, a\}$ (или дуга (a, a)) называется ***петлёй***. Граф, в котором нет кратных рёбер и петель, называется ***простым***. В дальнейшем почти всегда мы будем рассматривать простые графы.

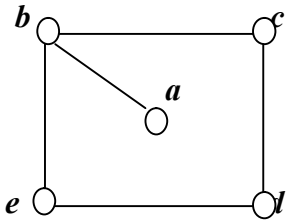
Большое количество приложений использует взвешенные графы. Граф называется ***взвешенным*** (или ***нагруженным***), если каждому ребру сопоставлено число (вес). Взвешенными могут быть как ориентированные, так и неориентированные графы.

Число рёбер, инцидентных данной вершине графа, называется ***степенью*** этой вершины. При этом каждая петля учитывается дважды. Вершины, степень которых равна 0, называются ***изолированными***, а вершины степени 1 называются ***концевыми*** (или ***висячими***). Для вершины орграфа определяется ***полустепень исхода*** – число дуг, выходящих из этой вершины, а также ***полустепень захода*** – число дуг, оканчивающихся в этой вершине.

Граф, в котором нет ни одного ребра, называется ***нуль-графом***. ***Полным*** графом называется неориентированный граф без петель, в котором каждые 2 вершины соединены ребром.

Один и тот же граф может быть изображён по-разному. Можно по-другому расположить вершины. Рёбра не обязательно должны быть прямолинейными.





Это разные графы или один и тот же? Можно заметить, что оба графа содержат одну и ту же информацию о вершинах, их связях с помощью рёбер. Такие графы называются изоморфными. Дадим точное определение.

Неориентированные графы $G_1 = \langle A_1, V_1 \rangle$, $G_2 = \langle A_2, V_2 \rangle$ называются ***изоморфными***, если существует биективное отображение

$$\varphi: \quad A_1 \to A_2,$$

сохраняющее отношение смежности:

$$\forall a, b \in A_1 \quad \{a, b\} \in V_1 \Leftrightarrow \{\varphi(a), \varphi(b)\} \in V_2.$$

Аналогично определяется изоморфизм орграфов: $\langle A_1, \overline{V_1} \rangle$, $\langle A_2, \overline{V_2} \rangle$ изоморфны, если существует биективное отображение $\varphi: \quad A_1 \to A_2$, такое, что

$$\forall a, b \in A_1 \quad (a, b) \in \overline{V_1} \Leftrightarrow (\varphi(a), \varphi(b)) \in \overline{V_2}.$$

В большинстве задач изоморфные графы не различаются.

4.3 Маршруты в графах

Сначала рассмотрим неориентированные графы. ***Маршрутом*** в графе $G = \langle A, V \rangle$ называется последовательность рёбер вида $\{a_0, a_1\}, \{a_1, a_2\}, \{a_2, a_3\}, \ldots, \{a_{k-1}, a_k\}$. Вершина a_0 – начало маршрута, a_k – конец маршрута. ***Длиной*** маршрута называется число рёбер в нём (если граф не является взвешенным). Маршрут называется ***цепью***, если все рёбра в нём различны. Если в цепи все вершины различны, то это ***простая*** цепь. Цепь, у которой начало и конец совпадают, называется ***циклом***. Если все остальные вершины различны, то цикл называется ***простым***.

Если цикл содержит все рёбра графа, то он называется ***эйлеровым*** циклом. Граф, в котором есть такой цикл, также называется ***эйлеровым***. Простой цикл, содержащий все вершины графа, называется ***гамильтоновым***. Так же называется и граф, в котором есть такой цикл.

На множестве вершин графа можно определить бинарное ***отношение достижимости*** – это множество $\{(a_i, a_j)\}$ таких пар, что существует маршрут с началом a_i и концом a_j. Это отношение является отношением эквивалентности. Действительно, симметричность и транзитивность очевидны. Рефлексивностью отношение достижимости также обладает, так как из a в a всегда есть маршрут – пустой, рёбер в нём нет, длина равна нулю.

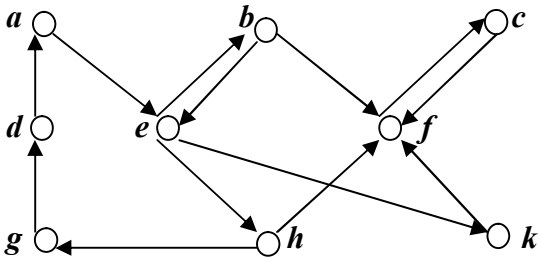
Отношение достижимости разбивает множество вершин графа на классы эквивалентных вершин. Каждый такой класс называется ***компонентой связности*** графа. Граф называется ***связным***, если у него всего одна компонента связности. Другими словами, граф является связным, если любые 2 его вершины соединены маршрутом.

Для ***ориентированных*** графов применяется несколько иная терминология. Вместо термина «цепь» употребляется термин «***путь***» (это маршрут, в котором все дуги различны). Иногда вместо термина «цикл» применяется термин «***контур***». Простой путь, простой контур (или простой цикл) определяются так же, как и для неориентированных графов.

Отношение достижимости уже не является эквивалентностью для ориентированного графа, так как оно не симметрично. Вместо него рассматривается отношение ***взаимной достижимости***: вершины a_i и a_j взаимно достижимы, если существуют маршруты из a_i в a_j и из a_j в a_i. Это отношение является эквивалентностью, классы эквивалентных вершин называются ***сильными компонентами связности***. Если граф состоит всего из одной такой компоненты, то он называется ***сильно связным***.

Иногда и для ориентированных графов используется понятие «связность» (или слабая связность), в этом случае ориентация рёбер игнорируется.

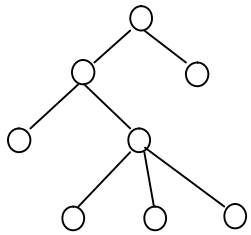
Пример 2.



Это связный граф, но сильно связным он не является. Имеются 3 компоненты сильной связности: $\{a, b, d, e, g, h\}$, $\{c, f\}$, $\{k\}$.

4.4 Деревья и каркасы

Связный неориентированный граф без циклов называется ***деревом***. Деревья – наиболее простой вид графов, в то же время они служат хорошими моделями для различных приложений. Удобно изображать дерево следующим образом:



Рассмотрим простейшие свойства деревьев.

Теорема 1. Если дерево имеет n вершин, то у него $n-1$ ребро.

Доказательство. Применим индукцию по числу вершин n. Для $n=1$ утверждение очевидно. При $n>1$ в дереве есть концевая вершина. Действительно, рассмотрим произвольный маршрут: $\{a_0,a_1\},\{a_1,a_2\},\ldots,\{a_{k-1},a_k\}$. Так как циклов нет, то все вершины в нём разные. Если вершина a_k не является концевой, то маршрут можно продолжить. Продолжая маршрут, мы обязательно найдём концевую вершину.

Теперь удалим концевую вершину вместе с инцидентным ей ребром. Полученный граф – дерево с $n-1$ вершиной. По предположению индукции оно имеет $n-2$ ребра. Значит, исходное дерево имеет $n-1$ ребро. Теорема доказана.

Справедливо и утверждение, в некотором смысле обратное.

Теорема 2. Если связный граф имеет n вершин и $n-1$ ребро, то это дерево.

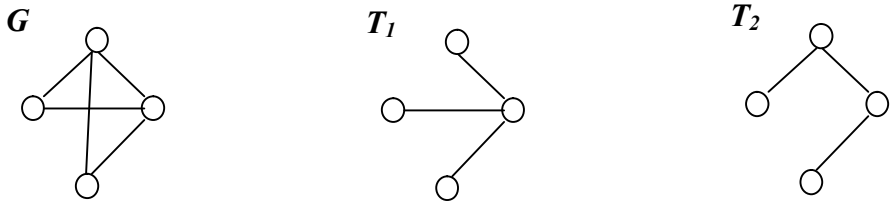
Доказательство. Допустим, в таком графе есть цикл. Удалим любое его ребро, при этом связность графа, очевидно, не нарушится. Будем удалять рёбра, пока в графе есть циклы. В результате получим дерево, в котором n вершин и меньше, чем $n-1$ ребро. По теореме 1, это невозможно.

Теорема 3. При добавлении к дереву одного ребра в графе появляется один цикл. При удалении из дерева одного ребра получается граф с двумя компонентами связности.

Доказательство очень просто.

Для каждого связного графа можно определить ***каркас*** (***стягивающее***, или ***остовное дерево***) – дерево, полученное из графа удалением рёбер.

Пример 3. На рисунке изображён граф G и два его каркаса T_1 и T_2.



Большое значение имеет построение ***минимального каркаса*** для взвешенных связных графов. Минимальным называется каркас с наименьшей суммой весов рёбер. Опишем ***алгоритм Краскала***, позволяющий строить минимальный каркас.

Пусть дан связный взвешенный граф с n вершинами.

Шаг 1: выберем ребро с наименьшим весом.

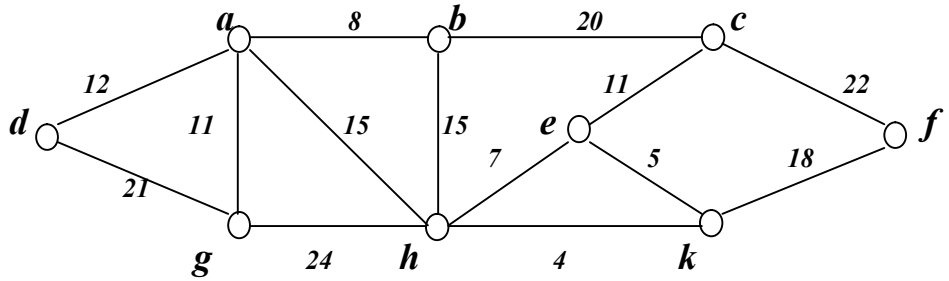
Шаг 2: среди оставшихся рёбер выберем ребро с наименьшим весом.

Шаг 3: среди оставшихся рёбер выберем ребро с наименьшим весом, не образующее цикла с рёбрами, выбранными ранее. Шаг 3 повторяется до тех пор, пока не будет выбрано $n-1$ ребро. Выбранные $n-1$ ребро образуют минимальный каркас.

Обоснование алгоритма. Необходимо убедиться, что шаг 3 возможен, пока не выбрано $n-1$ ребро. Рассмотрим произвольный момент в работе алгоритма. Допустим, уже выбранные рёбра вместе с инцидентными им вершинами образуют несвязный граф. Тогда добавление ребра, соединяющего компоненты связности (такое существует, ведь исходный граф связен), не приведёт к образованию цикла. Если уже выбранные рёбра образуют связный граф, то обязательно есть свободная вершина – так как если заняты все вершины, то, по теореме 1, выбрано уже $n-1$ ребро. Добавление ребра, инцидентного свободной вершине, конечно, не приведёт к образованию цикла. Итак, шаг 3 всегда возможен.

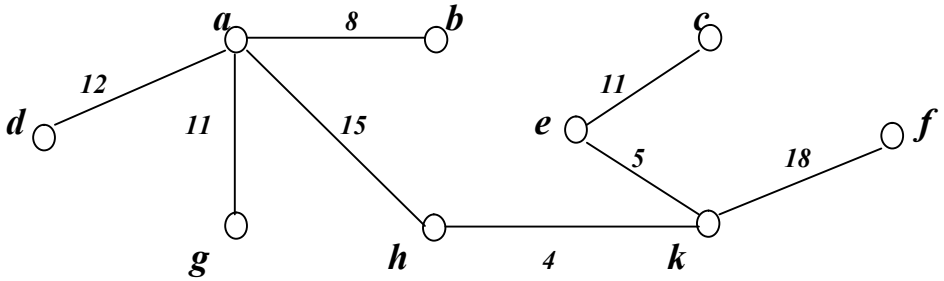
Если выбрано $n-1$ ребро, то они обязательно образуют связный граф – иначе, добавляя рёбра, мы получим дерево с n вершинами и большим, чем $n-1$ числом рёбер.

Пример 4. Необходимо соединить 9 компьютеров в единую сеть. Стоимость прокладки линий связи задана с помощью взвешенного графа. Отсутствие ребра означает невозможность (или слишком высокую стоимость) прокладки соответствующей линии. Требуется найти наименее дорогую схему соединения, определить её стоимость.



Решение. На языке теории графов: требуется найти минимальный каркас в графе. Будем строить минимальный каркас с помощью алгоритма Краскала. Реализацию алгоритма оформим в виде таблицы. Кроме того, будем изображать графически выбранные рёбра – для того, чтобы избежать циклов. Итоговое дерево расположим после таблицы.

Ребро	Вес	Отсутствие циклов	Счётчик рёбер
(h,k)	***4***	+	***1***
(e,k)	***5***	+	***2***
(h,e)	***7***	-	-
(a,b)	***8***	+	***3***
(c,e)	***11***	+	***4***
(a,g)	***11***	+	***5***
(a,d)	***12***	+	***6***
(a,h)	***15***	+	***7***
(b,h)	***15***	-	-
(k,f)	***18***	+	***8 → конец***



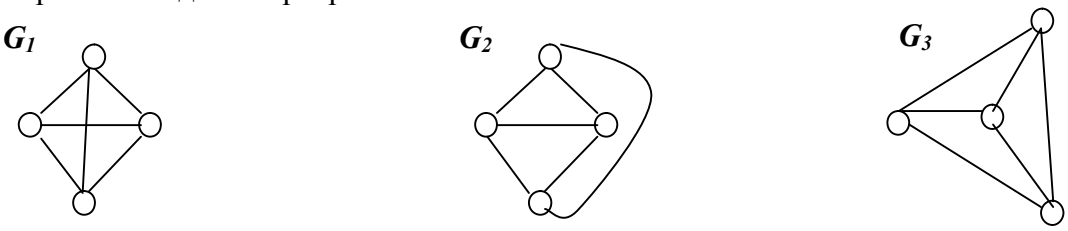
Стоимость найденного оптимального плана: ***4+5+8+11+11+12+15+18 = 84***.

Замечание. При машинной реализации этого алгоритма представляет трудность определение наличия или отсутствия цикла. Поэтому применяются несколько иные варианты алгоритма Краскала.

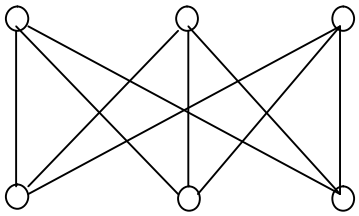
4.5 Плоские графы

Граф называется ***плоским***, или ***планарным***, если его можно изобразить на плоскости так, чтобы рёбра пересекались только в вершинах.

Пример 5. Граф G_1 является плоским. Действительно, графы G_2 и G_3 есть лишь другие изображения, другие реализации графа G_1 на плоскости. Говоря более строго, эти графы изоморфны: каждый из них – полный граф с четырьмя вершинами, так как каждая вершина соединена рёбрами со всеми остальными.



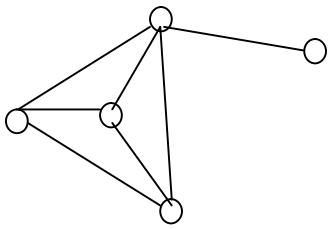
Известна старинная задача о трёх домах и о трёх колодцах. Три соседа пользуются водой, маслом и повидлом из трёх колодцев. Соседи поссорились и решили провести тропинки от каждого дома к каждому колодцу так, чтобы они не пересекались. Можно ли это сделать?



На языке графов задача формулируется так: является ли плоским двудольный граф $K_{3,3}$?

Эта задача оказалась не только любопытной и занимательной, но и очень важной для теории графов. Поэтому мы подробно рассмотрим её решение. Но сначала докажем замечательную теорему Эйлера, которая и поможет нам разобраться с колодцами.

Пусть имеется плоский связный граф. Рассмотрим его некоторую реализацию на плоскости – такую, что рёбра пересекаются только в вершинах. На множестве точек плоскости можно задать отношение эквивалентности: две точки плоскости (не лежащие на рёбрах и не совпадающие с вершинами) назовём эквивалентными, если существует непрерывная кривая, соединяющая эти точки и не пересекающая вершины и рёбра графа. Классы эквивалентных между собой точек будем называть ***гранями*** графа.



У графа на рисунке 4 грани, одна из них неограниченна (бесконечна).

Теорема 4 (теорема Эйлера). Пусть B, P, $Г$ – число вершин, рёбер и граней соответственно. Тогда для любого плоского связного графа

$$B - P + Г = 2.$$

Доказательство. Проведём индукцию по числу рёбер P. Если $P = 1$, то либо это петля, и тогда $B = 1$, $Г = 2$, $B - P + Г = 2$; либо это ребро, соединяющее 2 разные вершины, и тогда $B = 2$, $Г = 1$, $B - P + Г = 2$. База индукции имеется.

Предполагаем, что для графов с числом рёбер $P - 1$ теорема верна. Добавим к такому графу одно ребро так, чтобы граф остался связным. Это можно сделать тремя способами.

1) Новое ребро – петля у одной из вершин. Добавляется 1 ребро и 1 грань, величина $В - Р + Г$ не изменяется.
2) Новое ребро соединяет 2 разные вершины графа. Но граф плоский, и это ребро можно провести так, что оно не пересекает других рёбер, лежит в одной грани и делит её на две. Снова добавляется 1 ребро и 1 грань, величина $В - Р + Г$ не изменяется.
3) Новое ребро инцидентно только одной из вершин графа. Тогда приходится добавлять новую вершину. Если она не лежит на ребре графа, то добавляется 1 ребро и 1 вершина, число граней не изменяется, величина $В - Р + Г$ не изменяется. Если же новая вершина лежит на ребре, то это ребро делится на 2, грань делится на 2 грани. В этом случае добавляется 2 ребра, 1 грань и 1 вершина, величина $В - Р + Г$ опять не изменяется.

Следствие 1. Задача о трёх домах и о трёх колодцах неразрешима.

Доказательство. Допустим, граф $К_{3,3}$ является плоским. Так как $В = 6$, $Р = 9$, то

$$Г = 2 - В + Р = 2 - 6 + 9 = 5.$$

Обозначим g_i – число граней, ограниченных i рёбрами. В нашем графе $g_1 = 0$, так как петель нет, $g_2 = 0$ (кратных рёбер нет), $g_3 = 0$ (между домами и между колодцами дорожек нет). Поэтому

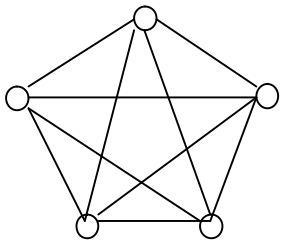
$$Г = g_4 + g_5 + g_6 + \ldots . \qquad (*)$$

Кроме того, если подсчитать все рёбра, ограничивающие эти грани, то

$$2Р = 4g_4 + 5g_5 + 6g_6 + \ldots , \qquad (**)$$

так как каждое ребро считается дважды. Полученные соотношения приводят к противоречию. Из (*): $Г = 5 = g_4 + g_5 + g_6 + \ldots$, значит $20 = 4g_4 + 4g_5 + 4g_6 + \ldots$. Но из (**): $2Р = 18 = 4g_4 + 5g_5 + 6g_6 + \ldots$. Получается, $20 \le 18$, что не так.

Следствие 2. Полный граф с 5 вершинами не является плоским.



Доказательство. Напомним: у полного графа петель нет, а каждые 2 вершины соединены ребром. Для данного графа $В = 5$, $Р = 10$. Если предположить, что он плоский, то, по теореме Эйлера, $Г = 2 - В + Р = 7$.

Запишем соотношения, аналогичные равенствам (*), (**). Так как имеются грани, ограниченные тремя рёбрами, то

$$Г = g_3 + g_4 + g_5 + \ldots , \qquad 2Р = 3g_3 + 4g_4 + 5g_5 + \ldots .$$

Отсюда следует:

$$21 = 3Г = 3g_3 + 3g_4 + 3g_5 + \ldots \le 3g_3 + 4g_4 + 5g_5 + \ldots = 2Р = 20 ,$$

т.е. противоречие.

Графы, рассмотренные в следствиях из теоремы Эйлера, являются очень важными примерами неплоских графов. Они позволяют сформулировать необходимое и достаточное условие «плоскости» (планарности) графа.

Теорема 5 (теорема Понтрягина–Куратовского). Граф является плоским тогда и только тогда, когда в нём нет подграфа, который можно «сжать» до одного из графов, рассмотренных в следствиях 1 и 2.

Сжатие графа – это удаление вершины степени 2 и замена двух инцидентных ей рёбер одним ребром:

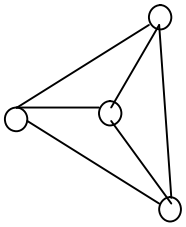
Конечно, это действие может быть повторено несколько раз. Напомним также, что ***подграфом*** называется граф, полученный удалением некоторых вершин и (или) рёбер.

4.6 Раскраска графов

Исторически задачи о раскраске графов связаны с раскраской географических карт. Каждой стране на карте можно сопоставить вершину графа. Если страны имеют общую границу (являются смежными), то и вершины графа будут смежными, между ними следует провести ребро. Нетрудно понять, что таким образом построенный граф является плоским и не имеет петель. Раскрашивая карту, обычно следят за тем, чтобы соседние страны получали разные цвета. Такую раскраску называют правильной. Сколько необходимо красок для правильной раскраски любой карты? Дадим теперь более строгие определения на языке графов.

Пусть $G = \langle A, V \rangle$ – неориентированный граф без петель, $C = \{c_1, c_2, \ldots, c_k\}$ – некоторое конечное множество – множество «цветов». ***Правильной раскраской*** графа называется такое отображение $A \to C$, что смежные вершины получают разные цвета. Наименьшее число k, при котором существует правильная раскраска, называется ***хроматическим числом*** графа: $\chi(G) = k$.

Ясно, что для полного графа K_n с n вершинами $\chi(K_n) = n$. Возникает вопрос: сколько необходимо красок, чтобы раскрасить любой плоский граф? Трёх красок, конечно, мало – легко привести пример плоского графа, требующего 4-х красок:



Мы докажем, что 6 красок достаточно. Но сначала нужно установить ещё одно свойство плоских графов.

Лемма. В плоском графе без петель всегда есть вершина, степень которой не больше 5.

Доказательство. Пусть, как и в предыдущем пункте, g_i – число граней, ограниченных i рёбрами. Запишем соотношения, которыми мы уже пользовались в доказательстве следствия 2:

$$Г = g_3 + g_4 + g_5 + \ldots, \qquad 2P = 3g_3 + 4g_4 + 5g_5 + \ldots.$$

Из этих равенств, очевидно, следует: $2P \geq 3Г$.

Теперь допустим, что существует плоский граф, у которого степени всех вершин больше 6. Так как сумма степеней всех вершин равна $2P$, то для этого графа $2P \geq 6B$. Применим теорему Эйлера: $2P \geq 6B = 6(2 + P - Г) \Rightarrow 2P \leq 3Г - 6$. Неравенства $2P \leq 3Г - 6$ и $2P \geq 3Г$ противоречат друг другу. Лемма доказана.

Теорема 6. Для любого плоского графа без петель $\chi(G) \leq 6$, т.е. для правильной раскраски достаточно 6 красок.

Доказательство. Проведём индукцию по числу вершин B. Если $B = 1$ или $B = 2$, то ясно, что много красок не нужно. Допустим, что для графов с числом вершин меньше B теорема доказана. Рассмотрим плоский граф без петель с B вершинами. По лемме, в нём есть вершина степени не больше 5. Удалим её вместе с инцидентными ей рёбрами. По предположению индукции, полученный граф можно раскрасить 6 красками. Для удалённой вершины выберем краску, не совпадающую с окраской смежных вершин. Это возможно, так как смежных вершин не более 5, а красок 6. Теорема доказана.

Заметим, что хотя мы доказали теорему для графов без петель, но отсутствие петель для возможности правильной раскраски графа является требованием формальным, к содержанию задачи о раскраске не относящимся.

Справедлива и более сильная теорема, которая здесь приводится без доказательства.

Теорема 7. Для любого плоского графа $\chi(G) \le 5$, т.е. для правильной раскраски достаточно 5 красок.

Остаётся вопрос: достаточно ли 4-х красок для правильной раскраски любого плоского графа? Это так называемая «проблема 4-х красок». Первая научная статья, связанная с ней, появилась в 1879 году. Было много попыток решения проблемы, но она оказалась трудной. Наконец, в 1976 году появилось сообщение, что проблема решена и 4-х красок достаточно. Однако доказательство содержит сотни страниц текста и очень сложную программу для ЭВМ. С тех пор его несколько раз упрощали другие авторы, но все доказательства остаются сложными и выполняются с применением ЭВМ.

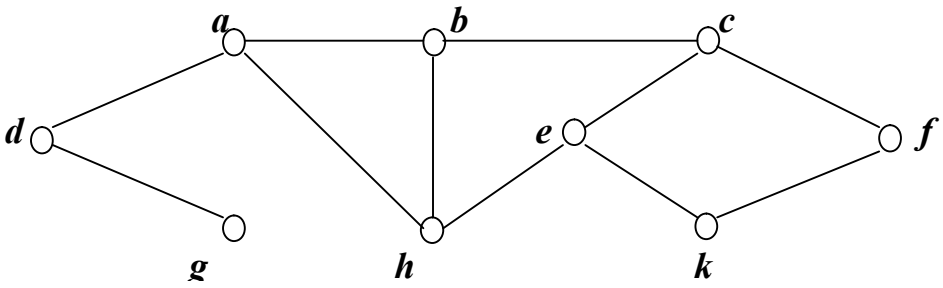
5 Алгоритмы на графах

Многие алгоритмические задачи, которые ставятся и решаются на графах, требуют систематического перебора всех его вершин – так, чтобы каждая вершина проверялась один раз. Рассмотрим два общих подхода к организации такого перебора, известных под названиями «поиск в глубину» и «поиск в ширину». На основе этих подходов часто строятся алгоритмы решения конкретных задач на графах.

5.1 Поиск «в глубину»

Начинаем обход графа с произвольной вершины. На каждом шаге алгоритма осуществляется переход от текущей вершины к смежной с ней. Если все смежные уже просмотрены, то происходит возврат к предыдущей вершине и поиск смежных с ней. Удобно организовать это с помощью стека, помещая в него «проверенную» вершину и удаляя «использованную» – вершину, для которой все смежные уже проверены. Покажем работу алгоритма на примере.

Пример 1. Провести, начиная с вершины ***a***, «проверку» методом поиска «в глубину» всех вершин графа



Решение. Оформим протокол работы алгоритма в виде таблицы

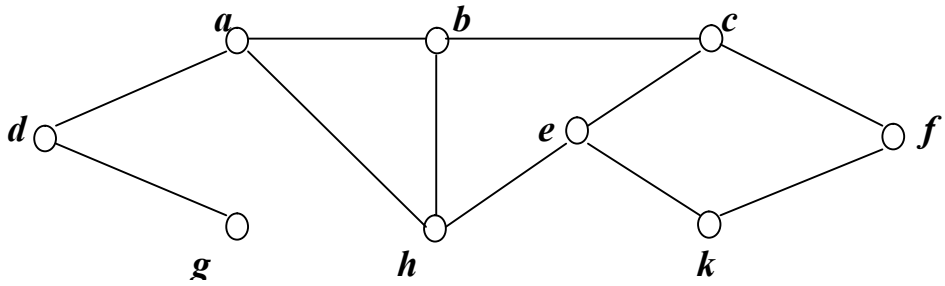
Шаг	Непроверенные вершины	Стек	Комментарии
1	***b,c,d,e,f,g,h,k***	***a***	Проверяется ***a***
2	***c,d,e,f,g,h,k***	***ab***	Проверяется ***b***, смежная с ***a***
3	***d,e,f,g,h,k***	***abc***	Проверяется ***c***, смежная с ***b***
4	***d,f,g,h,k***	***abce***	Проверяется ***e***, смежная с ***c***
5	***d,f,g,k***	***abceh***	Проверяется ***h***, смежная с ***e***
6	***d,f,g,k***	***abce***	Нет непроверенных, смежных с ***h***; ***h*** удаляется из стека
7	***d,f,g***	***abcek***	Проверяется ***k***, смежная с ***e***
8	***d,g***	***abcekf***	Проверяется ***f***, смежная с ***k***
9	***d,g***	***abcek***	Нет непроверенных, смежных с ***f***; ***f*** удаляется из стека
10	***d,g***	***abce***	Нет непроверенных, смежных с ***k***; ***k*** удаляется из стека
11	***d,g***	***abc***	Нет непроверенных, смежных с ***e***; ***e*** удаляется из стека
12	***d,g***	***ab***	Нет непроверенных, смежных с ***c***; ***c*** удаляется из стека
13	***d,g***	***a***	Нет непроверенных, смежных с ***b***; ***b*** удаляется из стека
14	***g***	***ad***	Проверяется ***d***, смежная с ***a***
15		***adg***	Проверяется ***g***, смежная с ***d***. Все вершины проверены, конец работы алгоритма.

Может оказаться, что стек пуст, но ещё не все вершины проверены. Значит, проверенные вершины образуют компоненты связности. Алгоритм продолжает работу, начиная с любой непроверенной вершины. Таким образом, этот алгоритм позволяет находить компоненты связности. Заметим также, что алгоритм находит маршруты от начальной вершины до каждой вершины текущей компоненты связности. Однако это не обязательно самые короткие маршруты.

5.2 Поиск «в ширину»

Основная идея поиска «в ширину»: для данной вершины рассматриваются ***все*** смежные с ней, а затем происходит переход к другой вершине. Чтобы организовать такой процесс, нужен не стек, а ***очередь***. Сначала заносится в очередь («проверяется») начальная вершина. Затем проверяются, заносятся в очередь смежные с ней. Когда все смежные проверены, вершина удаляется из очереди. Начинается проверка вершин, смежных с первой в очереди вершиной. Алгоритм заканчивает работу, когда все вершины проверены. Можно использовать алгоритм поиска «в ширину» для построения кратчайших маршрутов от начальной вершины до всех остальных. Такой маршрут можно зафиксировать сразу, как только вершина попадает в очередь, так как известна «предыдущая» (она первая в очереди), и кратчайший маршрут до неё уже построен. Проведём поиск «в ширину» на том же самом графе.

Пример 2. Провести, начиная с вершины ***a***, «проверку» методом поиска «в ширину» всех вершин графа



Решение. Оформим протокол работы алгоритма в виде таблицы.

Шаг	Непроверенные вершины	Очередь	Комментарии	Кратчайшие маршруты от ***a***
1	***b,c,d,e,f,g,h,k***	***a***	Проверяется ***a***	
2	***c,d,e,f,g,h,k***	***ab***	Проверяется ***b***, смежная с ***a***	***ab***
3	***c,e,f,g,h,k***	***abd***	Проверяется ***d***, смежная с ***a***	***ad***
4	***c,e,f,g,k***	***abdh***	Проверяется ***h*** смежная с ***a***	***ah***
5	***c,e,f,g,k***	***bdh***	Нет непроверенных, смежных с ***a***; ***a*** удаляется из очереди	
6	***e,f,g,k***	***bdhc***	Проверяется ***c***, смежная с ***b***	***abc***
7	***e,f,g,k***	***dhc***	Нет непроверенных, смежных с ***b***; ***b*** удаляется из очереди	
8	***e,f,k***	***dhcg***	Проверяется ***g***, смежная с ***d***	***adg***
9	***e,f,k***	***hcg***	Нет непроверенных, смежных с ***d, d*** удаляется из очереди	
10	***f,k***	***hcge***	Проверяется ***e***, смежная с ***h***	***ahe***
11	***f,k***	***cge***	Нет непроверенных, смежных с ***h***; ***h*** удаляется из очереди	
12	***k***	***cgef***	Проверяется ***f***, смежная с ***c***	***abcf***
13	***k***	***gef***	Нет непроверенных, смежных	

			с ***c***; ***c*** удаляется из очереди	
14	***k***	***ef***	Нет непроверенных, смежных с ***g***; ***g*** удаляется из очереди	
15		***efk***	Проверяется ***k***, смежная с ***e***. Все вершины проверены, конец работы алгоритма.	***ahek***

Так же, как и при поиске «в глубину», очередь может оказаться пустой в тот момент, когда ещё не все вершины проверены. Это говорит о том, что граф несвязен, обход следующей компоненты связности можно начинать с любой непроверенной вершины.

Алгоритмы поиска «в глубину» и «в ширину» можно применять и для ориентированных графов. В этом случае за один цикл работы алгоритма проверяется не связная компонента, а все вершины графа, достижимые из начальной.

5.3 Алгоритм Дейкстры

Длиной маршрута во взвешенном графе называется сумма весов всех рёбер этого маршрута. Ставится задача: найти кратчайшие маршруты от одной из вершин (она называется ***источником***) до всех остальных вершин графа. Для решения этой задачи мы рассмотрим ***алгоритм Дейкстры***, который можно применять, если нет рёбер с отрицательными весами. При этом граф может быть как ориентированным, так и неориентированным.

Алгоритм Дейкстры требует присваивания каждой вершине графа (кроме источника) двойной метки ***s-x***, которая изменяется по определённому правилу на каждом шаге алгоритма. Первая часть метки ***s*** – длина маршрута от источника до данной вершины, найденного на предыдущем шаге алгоритма. Вторая часть метки ***x*** – обозначение вершины, предшествующей данной на найденном маршруте.

Шаг алгоритма состоит в том, что метка с минимальным значением ***s*** становится окончательной, «замораживается». А метки тех вершин, в которые идут рёбра от вершины с «замороженной» меткой, пересчитываются по правилу:

$$s(B) = min\{s(B), s(A) + p(A,B)\}.$$

Здесь A – вершина, метка которой только что была «заморожена», B – вершина, для которой метка пересчитывается, $p(A,B)$ – вес ребра с началом A и концом B. Таким образом, значения s временных меток могут лишь уменьшаться. При уменьшении s меняется, конечно, и вторая часть метки.

Алгоритм Дейкстры заканчивает свою работу, когда метки всех вершин становятся окончательными. Окончательное значение параметра s для каждой вершины и есть длина кратчайшего маршрута от источника до этой вершины. Благодаря вторым частям окончательных меток можно восстановить и сами маршруты.

Поясним работу алгоритма Дейкстры на примере.

Пример 3. Найти кратчайшие маршруты от вершины ***e*** до всех остальных вершин взвешенного ориентированного графа с вершинами a,b,c,d,e,f,g, заданного своей матрицей весов:

	a	***b***	***c***	***d***	***e***	***f***	***g***
a	∞	∞	***5***	***4***	***2***	***2***	∞
b	∞	∞	∞	∞	***1***	***3***	***4***
c	***2***	***2***	∞	***1***	***1***	***1***	***1***
d	∞	∞	∞	∞	***3***	***3***	***2***
e	***4***	∞	∞	***1***	∞	***4***	***1***
f	***3***	∞	∞	***3***	∞	∞	***1***
g	***5***	∞	***2***	***1***	∞	***2***	∞

В каждой строке указаны веса дуг, выходящих из соответствующей вершины. Знак ∞ поставлен, если дуга отсутствует.

Решение. Метки вершин на каждом шаге алгоритма будем заносить в таблицу. На первом шаге метки получают вершины, в которые идут дуги от источника ***e***. Источнику метки не присваиваются.

Шаг	***a***	***b***	***c***	***d***	***e***	***f***	***g***
1	***4-e***	∞	∞	***1-e***		***4-e***	***1-e***

Шаг 2. Выбираем вершину с наименьшим значением ***s***. Так как $s(d)=s(g)$, то можно взять любую из них. Пусть метка вершины ***d*** «замораживается». Из ***d*** идут дуги в ***f*** и ***g***.

$$s(f)=4,\quad \min\{s(f),s(d)+p(d,f)\}=\min\{4,1+3\}=4;$$

$$s(g)=1,\quad \min\{s(g),s(d)+p(d,g)\}=\min\{1,1+2\}=1.$$

Метки не изменились.

Шаг	***a***	***b***	***c***	***d***	***e***	***f***	***g***
1	***4-e***	∞	∞	***1-e***		***4-e***	***1-e***
2	***4-e***	∞	∞			***4-e***	***1-e***

Шаг 3. Метка вершины ***g*** «замораживается». Из ***g*** идут дуги в «незамороженные» ***a***, ***c***, ***f***.

$$s(a)=4,\quad \min\{s(a),s(g)+p(g,a)\}=\min\{4,1+5\}=4;$$

$$s(c)=\infty,\quad \min\{s(c),s(g)+p(g,c)\}=\min\{\infty,1+2\}=3;$$

$$s(f)=4,\quad \min\{s(f),s(g)+p(g,f)\}=\min\{4,1+2\}=3.$$

Шаг	***a***	***b***	***c***	***d***	***e***	***f***	***g***
1	***4-e***	∞	∞	***1-e***		***4-e***	***1-e***
2	***4-e***	∞	∞			***4-e***	***1-e***
3	***4-e***	∞	***3-g***			***3-g***	

Шаг 4. Метка вершины ***c*** (например) «замораживается». Из ***c*** идут дуги в «незамороженные» ***a***, ***b***, ***f***.

$$s(a)=4,\quad \min\{s(a),s(c)+p(c,a)\}=\min\{4,3+2\}=4;$$

$$s(b)=\infty,\quad \min\{s(b),s(c)+p(c,b)\}=\min\{\infty,3+2\}=5;$$

$$s(f)=3,\quad \min\{s(f),s(c)+p(c,f)\}=\min\{3,3+1\}=3.$$

Шаг	***a***	***b***	***c***	***d***	***e***	***f***	***g***
1	***4-e***	∞	∞	***1-e***		***4-e***	***1-e***
2	***4-e***	∞	∞			***4-e***	***1-e***
3	***4-e***	∞	***3-g***			***3-g***	
4	***4-e***	***5-c***				***3-g***	

Шаг 5. Метка вершины ***f*** «замораживается». Из ***f*** идёт дуга в «незамороженную» ***a***.

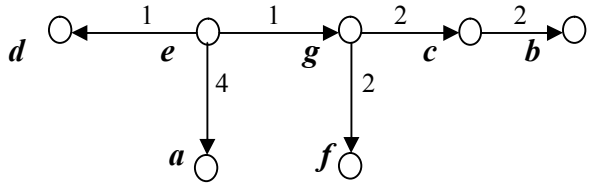
$$s(a)=4,\quad \min\{s(a),s(f)+p(f,a)\}=\min\{4,3+3\}=4.$$

Шаг	***a***	***b***	***c***	***d***	***e***	***f***	***g***
1	***4-e***	∞	∞	***1-e***		***4-e***	***1-e***
2	***4-e***	∞	∞			***4-e***	***1-e***
3	***4-e***	∞	***3-g***			***3-g***	
4	***4-e***	***5-c***				***3-g***	
5	***4-e***	***5-c***					

Шаг 6 Метка вершины ***a*** «замораживается». Из ***a*** нет дуги в «незамороженную» ***b***, метка не меняется, алгоритм заканчивает работу.

Шаг	*a*	*b*	*c*	*d*	*e*	*f*	*g*
1	***4-e***	∞	∞	***1-e***		***4-e***	***1-e***
2	***4-e***	∞	∞			***4-e***	***1-e***
3	***4-e***	∞	***3-g***			***3-g***	
4	***4-e***	***5-c***				***3-g***	
5	***4-e***	***5-c***					
6		***5-c***					

Итоговая таблица позволяет легко восстановить кратчайший маршрут от источника ***e*** до любой вершины. Можно построить дерево кратчайших маршрутов:

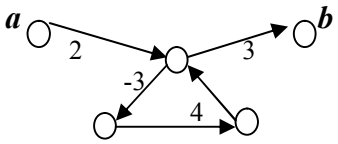


Замечание. На каждом шаге алгоритма рассматриваются (и могут изменить свои метки) только вершины, смежные с той, метка которой «заморожена». Однако при машинной реализации алгоритма на каждом шаге может работать цикл по всем вершинам графа, кроме источника. Так как $p(A,B)=\infty$, если ребро между A и B отсутствует, то остальные вершины, не смежные с «замороженной», не могут изменить свои метки.

5.4 Алгоритм Форда-Беллмана

Если рассматривать графы, у которых, возможно, есть рёбра с отрицательными весами, то для решения той же задачи приходится применять другие алгоритмы. Мы познакомимся с алгоритмом Форда-Беллмана, позволяющим находить кратчайшие пути от одной из вершин (источника) до всех остальных вершин в ориентированном взвешенном графе без циклов отрицательной длины.

Требование отсутствия циклов отрицательной длины легко объяснимо. Например, в графе



-2

нет кратчайшего пути от вершины ***a*** до вершины ***b***. Действительно, цикл длины (-1) позволяет находить пути от a до b любой отрицательной длины. Аналогичные трудности возникают и при работе с неориентированными графами, имеющими рёбра с отрицательными весами.

Алгоритм Форда-Беллмана использует те же самые двойные метки для вершин графа, что и алгоритм Дейкстры, применяется то же самое правило для изменения меток:

$$s(B)=min\{s(B),s(A)+p(A,B)\}.$$

Но на каждом шаге алгоритма Дейкстры вершина A была зафиксирована, это правило применялось к вершинам B, смежным с A. Теперь же нам придётся для каждой вершины B рассматривать все вершины A, в которых начинаются дуги, оканчивающиеся в B. В случае машинной реализации алгоритма Форда-Беллмана на каждом шаге вычисляются метки всех вершин (цикл по B), причём для каждой из них используются все остальные вершины (цикл по A).

Алгоритм заканчивает свою работу, если на некотором шаге ни одна из меток не изменилась.

Рассмотрим пример работы алгоритма Форда-Беллмана.

Пример 4. Найти кратчайшие маршруты от вершины ***a*** до всех остальных вершин взвешенного ориентированного графа с вершинами a,b,c,d,e,f,g,h, заданного своей матрицей весов:

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
a	∞	***2***	***3***	***6***	∞	∞	∞	∞
b	∞	∞	***-2***	∞	***7***	∞	∞	∞
c	∞	∞	∞	***4***	***3***	***8***	∞	∞
d	∞	∞	***-3***	∞	∞	∞	***3***	∞
e	∞	∞	∞	∞	∞	***4***	∞	***3***
f	∞	∞	∞	∞	***5***	∞	∞	***2***
g	∞	∞	∞	∞	∞	***2***	∞	***-2***
h	∞	∞	∞	∞	∞	***1***	∞	∞

Решение.

Шаг 1: метки получают вершины, смежные с источником. Метка источника $s(a)=0$.

Шаг	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*
1	***0***	***2-a***	***3-a***	***6-a***	∞	∞	∞	∞

Шаг 2: пересчитываем метки всех вершин, кроме источника. При этом используется ***текущее*** значение параметра ***s*** каждой вершины.

$$s(b)=min\{s(b),s(a)+p(a,b),s(b)+p(b,b),\ldots,s(h)+p(h,b)\}=$$

$$=min\{2,0+2,2+\infty,3+\infty,6+\infty,\infty+\infty,\ldots,\infty+\infty\}=2.$$

Однако так поступает машина. Будем применять более короткую запись. При пересчёте метки для некоторой вершины не будем пытаться использовать источник, саму вершину, а также вершины, из которых не выходят дуги, идущие в данную.

$$s(c)=min\{s(c),s(b)+p(b,c),s(d)+p(d,c)\}=min\{2,2+(-2),6+(-3)\}=0.$$

Заносим найденное значение в таблицу и продолжаем вычисления.

$$s(d)=min\{s(d),s(c)+p(c,d)\}=min\{6,0+4\}=4,$$

$$s(e)=min\{s(e),s(b)+p(b,e),s(c)+p(c,e),s(f)+p(f,e)\}=$$

$$=min\{\infty,2+7,0+3,\infty+5\}=3,$$

$$s(f)=min\{s(f),s(c)+p(c,f),s(e)+p(e,f),s(g)+p(g,f),s(h)+p(h,f)\}=$$

$$=min\{\infty,0+8,3+4,\infty+2,\infty+1\}=7,$$

$$s(g)=min\{s(g),s(d)+p(d,g)\}=min\{\infty,4+3\}=7,$$

$$s(h)=min\{s(h),s(e)+p(e,h),s(f)+p(f,h),s(g)+p(g,h)\}=$$

$$min\{\infty,3+3,7+2,7+(-2)\}=5.$$

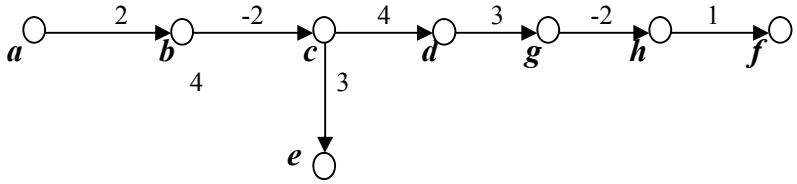
Шаг	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*
1	***0***	***2-a***	***3-a***	***6-a***	∞	∞	∞	∞
2	***0***	***2-a***	***0-b***	***4-c***	***3-c***	***7-e***	***7-d***	***5-g***

Аналогично выполняется шаг 3, на этом шаге изменяется только метка вершины ***f***. На шаге 4 метки не изменяются. Итоговая таблица имеет вид:

Шаг	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*
1	***0***	***2-a***	***3-a***	***6-a***	∞	∞	∞	∞

2	***0***	***2-a***	***0-b***	***4-c***	***3-c***	***7-e***	***7-d***	***5-g***
3	***0***	***2-a***	***0-b***	***4-c***	***3-c***	***6-h***	***7-d***	***5-g***
4	***0***	***2-a***	***0-b***	***4-c***	***3-c***	***6-h***	***7-d***	***5-g***

Построим дерево кратчайших маршрутов:



5.5 Алгоритм Флойда

Рассмотрим теперь задачу определения кратчайших маршрутов для всех пар вершин графа. Задача решается для ориентированных взвешенных графов без циклов отрицательной длины.

Обозначим вершины графа: $a_1, a_2, a_3, \ldots, a_n$. Пусть $p(i,j)$ – вес дуги (a_i, a_j). Если такая дуга отсутствует, то, как и раньше, считаем $p(i,j) = \infty$. Кроме того, для всех i полагаем $p(i,i) = 0$, так как петель в графе нет, и длина кратчайшего маршрута от вершины до самой себя равна нулю.

Введём обозначение: $s_m(i,j)$ – длина маршрута между a_i и a_i, кратчайшего среди маршрутов с промежуточными вершинами, принадлежащими множеству $\{a_1, a_2, \ldots, a_m\}$. Другими словами, $s_m(i,j)$ – длина наиболее короткого маршрута от a_i до a_i, не проходящего через вершины $a_{m+1}, a_{m+2}, \ldots, a_n$. Ясно, что $s_0(i,j) = p(i,j)$, а цель поставленной задачи – вычисление величин $s_n(i,j)$.

Докажем важное равенство:

$$s_{m+1}(i,j) = min\{s_m(i,j), s_m(i,m+1) + s_m(m+1,j)\}. \qquad (*)$$

Действительно, если самый короткий маршрут из не проходящих через $a_{m+2}, \ldots, a_n$ не проходит и через a_{m+1}, то $s_{m+1}(i,j) = s_m(i,j)$. А если этот маршрут проходит через a_{m+1}, то его части до a_{m+1} и после неё являются кратчайшими маршрутами между соответствующими парами вершин. Часть кратчайшего маршрута всегда является кратчайшим маршрутом. Значит, в этом случае $s_{m+1}(i,j) = s_m(i,m+1) + s_m(m+1,j)$.

Доказанное соотношение позволяет последовательно, начиная с матрицы весов данного графа $S_0 = \{s_0(i,j)\}$, построить матрицы $S_1, S_2, \ldots, S_n$. Матрица $S_n = \{s_n(i,j)\}$ будет содержать искомые длины кратчайших маршрутов, для которых разрешены любые промежуточные вершины.

Рассмотренный алгоритм позволяет находить длины кратчайших маршрутов, но не сами маршруты. Для определения маршрутов существуют различные подходы. Можно, например, для каждого элемента матрицы S_m сохранять предпоследнюю вершину соответствующего маршрута (аналогично второй части метки в алгоритмах Дейкстры и Форда-Беллмана). Зная предпоследние вершины всех маршрутов в итоговой матрице S_n, легко восстановить любой маршрут.

Пример 5. Найти кратчайшие маршруты между всеми парами вершин взвешенного ориентированного графа с вершинами $a_1, a_2, a_3, a_4, a_5, a_6$, заданного матрицей весов:

$$S=\begin{pmatrix} 0 & \infty & 4 & 2 & 6 & \infty \\ -2 & 0 & 1 & 3 & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & -1 \\ \infty & \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & \infty & 1 & 0 & 3 \\ \infty & \infty & 2 & \infty & -2 & 0 \end{pmatrix}.$$

Решение. Матрица $S_0 = S$ дана в условии. При машинной реализации алгоритма достаточно применить равенство (*) в цикле по переменным i, j, m. При работе «вручную» нужно помнить, что переход от S_0 к S_1 состоит в построении более коротких путей через промежуточную вершину a_1. В a_1 входит только 1 дуга – из a_2(это видно по первому столбцу матрицы), выходят из a_1 дуги в вершины a_3, a_4, a_5 (это видно по первой строке). Значит, измениться могут только элементы $s_0(2,3), s_0(2,4), s_0(2,5)$.

$s_1(2,3) = min\{s_0(2,3), s_0(2,1) + s_0(1,3)\} = min\{1, -2+4\} = 1$;

$s_1(2,4) = min\{s_0(2,4), s_0(2,1) + s_0(1,4)\} = min\{3, -2+2\} = 0$;

$s_1(2,5) = min\{s_0(2,5), s_0(2,1) + s_0(1,5)\} = min\{\infty, -2+6\} = 4$.

Получаем матрицу S_1: $$S_1=\begin{pmatrix} 0 & \infty & 4 & 2 & 6 & \infty \\ -2 & 0 & 1 & 0_1 & 4_1 & \infty \\ \infty & \infty & 0 & \infty & \infty & -1 \\ \infty & \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & \infty & 1 & 0 & 3 \\ \infty & \infty & 2 & \infty & -2 & 0 \end{pmatrix}.$$

Шаг 2: использование промежуточной вершины a_2. Но входящих в a_2 дуг нет, поэтому $S_2 = S_1$.

Шаг 3: использование промежуточной вершины a_3. «Входящие» вершины: a_1, a_2, a_6. «Выходящие» – только a_6. Могут измениться $s_2(1,6), s_2(2,6)$.

$s_3(1,6) = min\{s_2(1,6), s_2(1,3) + s_2(3,6)\} = min\{\infty, 4+(-1)\} = 3$;

$s_3(2,6) = min\{s_2(2,6), s_2(2,3) + s_2(3,6)\} = min\{\infty, 1+(-1)\} = 0$.

Получаем матрицу S_3: $$S_3=\begin{pmatrix} 0 & \infty & 4 & 2 & 6 & 3_3 \\ -2 & 0 & 1 & 0_1 & 4_1 & 0_3 \\ \infty & \infty & 0 & \infty & \infty & -1 \\ \infty & \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & \infty & 1 & 0 & 3 \\ \infty & \infty & 2 & \infty & -2 & 0 \end{pmatrix}.$$

Шаг 4: использование промежуточной вершины a_4. «Входящие» вершины: a_1, a_2, a_5. «Выходящие» – только a_5. Могут измениться $s_3(1,5), s_3(2,5)$.

$s_4(1,5) = min\{s_3(1,5), s_3(1,4) + s_3(4,5)\} = min\{6, 2+4\} = 6$;

$s_4(2,5) = min\{s_3(2,5), s_3(2,4) + s_3(4,5)\} = min\{4, 0+4\} = 4$.

Метки не изменились, поэтому $S_4 = S_3$.

Шаг 5: использование промежуточной вершины a_5. «Входящие» вершины: a_1, a_2, a_4, a_6. «Выходящие» – a_4, a_6. Могут измениться $s_4(1,4), s_4(1,6)$, $s_4(2,4), s_4(2,6)$, $s_4(4,6), s_4(6,4)$.

$$s_5(1,4)=\min\{s_4(1,4),s_4(1,5)+s_4(5,4)\}=\min\{2,6+1\}=2;$$
$$s_5(1,6)=\min\{s_4(1,6),s_4(1,5)+s_4(5,6)\}=\min\{3,6+3\}=3;$$
$$s_5(2,4)=\min\{s_4(2,4),s_4(2,5)+s_4(5,4)\}=\min\{0,4+1\}=0;$$
$$s_5(2,6)=\min\{s_4(2,6),s_4(2,5)+s_4(5,6)\}=\min\{0,4+3\}=0;$$
$$s_5(4,6)=\min\{s_4(4,6),s_4(4,5)+s_4(5,6)\}=\min\{\infty,4+3\}=7;$$
$$s_5(6,4)=\min\{s_4(6,4),s_4(6,5)+s_4(5,4)\}=\min\{\infty,-2+1\}=-1.$$

У изменённых элементов матрицы ставим метку – «предпоследняя вершина»:

$$S_5=\begin{pmatrix} 0 & \infty & 4 & 2 & 6 & 3_3 \\ -2 & 0 & 1 & 0_1 & 4_1 & 0_3 \\ \infty & \infty & 0 & \infty & \infty & -1 \\ \infty & \infty & \infty & 0 & 4 & 7_5 \\ \infty & \infty & \infty & 1 & 0 & 3 \\ \infty & \infty & 2 & -1_5 & -2 & 0 \end{pmatrix}.$$

Шаг 6: использование промежуточной вершины a_6. «Входящие» вершины: a_1, a_2, a_3, a_4, a_5. «Выходящие» – a_3, a_4, a_5. Могут измениться $s_5(1,3), s_5(1,4)$, $s_5(1,5), s_5(2,3), s_5(2,4), s_5(2,5)$, $s_5(3,4), s_5(3,5)$, $s_5(4,3), s_5(4,5)$, $s_5(5,3), s_5(5,4)$.

$$s_6(1,3)=\min\{s_5(1,3),s_5(1,6)+s_5(6,3)\}=\min\{4,3+2\}=4;$$
$$s_6(1,4)=\min\{s_5(1,4),s_5(1,6)+s_5(6,4)\}=\min\{2,3+(-1)\}=2;$$
$$s_6(1,5)=\min\{s_5(1,5),s_5(1,6)+s_5(6,5)\}=\min\{6,3+(-2)\}=1;$$
$$s_6(2,3)=\min\{s_5(2,3),s_5(2,6)+s_5(6,3)\}=\min\{1,0+2\}=1;$$
$$s_6(2,4)=\min\{s_5(2,4),s_5(2,6)+s_5(6,4)\}=\min\{0,0+(-1)\}=-1;$$
$$s_6(2,5)=\min\{s_5(2,5),s_5(2,6)+s_5(6,5)\}=\min\{4,0+(-2)\}=-2;$$
$$s_6(3,4)=\min\{s_5(3,4),s_5(3,6)+s_5(6,4)\}=\min\{\infty,-1+(-1)\}=-2;$$
$$s_6(3,5)=\min\{s_5(3,5),s_5(3,6)+s_5(6,5)\}=\min\{\infty,-1+(-2)\}=-3;$$
$$s_6(4,3)=\min\{s_5(4,3),s_5(4,6)+s_5(6,3)\}=\min\{\infty,7+2\}=9;$$
$$s_6(4,5)=\min\{s_5(4,5),s_5(4,6)+s_5(6,5)\}=\min\{4,7+(-2)\}=4;$$
$$s_6(5,3)=\min\{s_5(5,3),s_5(5,6)+s_5(6,3)\}=\min\{\infty,3+2\}=5;$$
$$s_6(5,4)=\min\{s_5(5,4),s_5(5,6)+s_5(6,4)\}=\min\{1,3+(-1)\}=1.$$

Получаем итоговую матрицу $S_6=\begin{pmatrix} 0 & \infty & 4 & 2 & 1_6 & 3_3 \\ -2 & 0 & 1 & -1_6 & -2_6 & 0_3 \\ \infty & \infty & 0 & -2_6 & -3_6 & -1 \\ \infty & \infty & 9_6 & 0 & 4 & 7_5 \\ \infty & \infty & 5_6 & 1 & 0 & 3 \\ \infty & \infty & 2 & -1_5 & -2 & 0 \end{pmatrix}$, содержащую длины кратчайших маршрутов между любыми двумя вершинами. Знак ∞ означает, что такого маршрута нет. Метки у элементов матрицы позволяют восстановить и сами маршруты. Например, найдём кратчайший маршрут от вершины a_1 до вершины a_5. Предпослед-

няя вершина a_6. У маршрута $a_1 \to a_6$ предпоследняя вершина a_3, у маршрута $a_1 \to a_3$ пометки нет, это дуга. Значит, маршрут найден: $a_1 \to a_3 \to a_6 \to a_5$.

5.6 Потоки в транспортных сетях

Сетью будем называть ориентированный взвешенный граф с положительными весами рёбер, у которого имеется ***источник*** – вершина, из которой дуги только выходят, и ***сток*** – вершина, в которую дуги только входят. Источник и сток будем всегда обозначать буквами s и t соответственно. Вес дуги (u,v) будем обозначать $c(u,v)$ и называть ***пропускной способностью*** дуги.

Потоком в сети называется функция f, сопоставляющая каждой дуге число, причём

1) $0 \le f(u,v) \le c(u,v)$ для любой дуги;
2) для любой вершины, кроме s и t, сумма потоков по входящим дугам равна сумме потоков по выходящим из неё дугам.

Разность между выходящим и входящим потоками называется ***дивергенцией*** вершины:

$$\sum_{u:v\to u} f(u,v) - \sum_{u:u\to v} f(u,v) = div(f,v).$$

Таким образом, по определению потока, дивергенция любой вершины, кроме s и t, равна нулю. Дивергенция источника положительна, а стока отрицательна, в сумме они тоже дают 0. ***Величиной*** потока называется число $W(f) = div(f,s)$. Одна из основных задач: для данной сети найти поток максимальной величины.

Потоки в сетях являются хорошей моделью для различных ситуаций. Это может быть поток газа или жидкости в трубопроводе, поток автомобилей, поток грузов по железной дороги, поток информации в сети и т.д.

Для решения задач о потоках в сети используется понятие разреза. Пусть V – множество всех вершин сети, $A \subseteq V$ – некоторое множество вершин, причём $s \in A,\ t \notin A$. ***Разрезом***, соответствующим A, называется множество дуг $P(A)$, связывающих вершины из A с вершинами из $V \setminus A$. Подмножество дуг, начинающихся в A и оканчивающихся в $V \setminus A$, обозначим $P^+(A)$, подмножество дуг, идущих в обратном направлении, обозначим $P^-(A)$. Таким образом, $P(A) = P^+(A) \cup P^-(A)$.

Пропускной способностью разреза $P(A)$ называется величина

$$C(P) = \sum_{(u,v)\in P^+} c(u,v).$$

Потоком через разрез называется величина

$$f(P) = \sum_{(u,v)\in P^+} f(u,v) - \sum_{(u,v)\in P^-} f(u,v).$$

Теорема 1. Поток через разрез не превышает пропускной способности этого разреза.

Доказательство.

$$f(P) = \sum_{(u,v)\in P^+} f(u,v) - \sum_{(u,v)\in P^-} f(u,v) \le \sum_{(u,v)\in P^+} f(u,v) \le \sum_{(u,v)\in P^+} c(u,v) = C(P).$$

Теорема 2. Потоки через все разрезы одинаковы и совпадают с величиной потока в сети.

Доказательство. Рассмотрим произвольный разрез $P(A)$. Вычислим сумму

$$\sum_{v\in A} div(f,v) = \sum_{v\in A}\left(\sum_{u:v\to u} f(u,v) - \sum_{u:u\to v} f(u,v) \right).$$

Эта сумма равна $div(f,s)=W(f)$, так как дивергенции остальных вершин равны нулю. С другой стороны, слагаемые $f(u,v)$, для которых вершины $u,v \in A$, входят парами, причём с разными знаками, т.е. сокращаются. Поэтому

$$W(f)=\sum_{v\in A}\left(\sum_{u\notin A,\, v\to u} f(u,v)-\sum_{u\notin A,\, u\to v} f(u,v)\right)=f(P)=\sum_{(u,v)\in P^{+}} f(u,v)-\sum_{(u,v)\in P^{-}} f(u,v)=f(P)$$

.

Теорема доказана.

Так как ясно, что поток через разрез не может быть больше пропускной способности разреза, то и величина потока в сети не может быть больше пропускной способности любого разреза. Оказывается, других ограничений на величину потока нет, т.е. справедлива

Теорема 3 (теорема Форда-Фалкерсона). Максимальная величина потока в сети равна минимальной пропускной способности разреза этой сети.

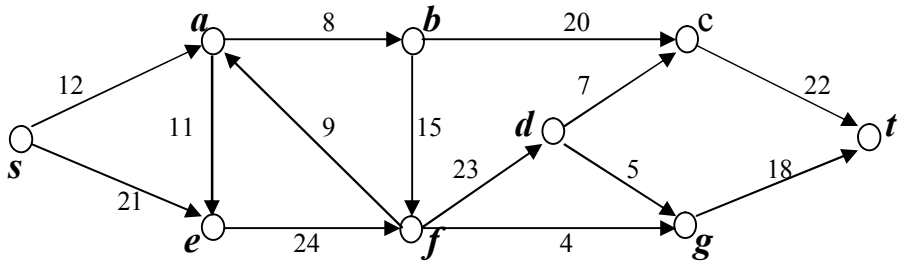
Доказательство теоремы здесь не приводится. Существует алгоритм Форда-Фалкерсона, позволяющий находить максимальный поток для сети с помощью построения так называемой ***увеличивающей цепи***.

Мы рассмотрим наиболее простой алгоритм поиска максимального потока для случая ***плоской*** сети с ***целочисленными*** пропускными способностями дуг.

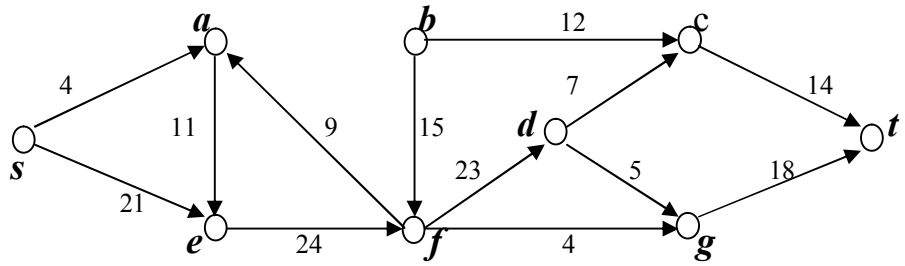
Пусть плоская сеть удовлетворяет дополнительному требованию: если добавить ребро, соединяющее источник ***s*** и сток ***t***, то сеть останется плоской. В таком случае граф можно изобразить на плоскости ***выше*** добавленного ребра (s,t). Точнее, если изобразить добавленное ребро прямолинейным отрезком, то весь граф можно расположить с одной стороны от прямой, являющейся продолжением этого отрезка.

Для такого изображения сети определим ***верхний маршрут*** от источника ***s*** до стока ***t***. Это маршрут, на котором в каждой вершине выбирается ***самая левая*** дуга из ведущих к стоку. По верхнему маршруту пропускается максимально возможный поток – он равен минимальной пропускной способности дуг этого маршрута. Величину потока запоминаем. Пропускные способности всех дуг маршрута уменьшаем на эту величину. Дуги, у которых пропускная способность стала нулевой, убираем. Затем снова выбираем верхний маршрут, пропускаем по нему максимально возможный поток, изменяем сеть. Повторяем такие шаги, пока существует маршрут от источника до стока. Сумма потоков по рассмотренным маршрутам и является максимально возможным потоком в сети.

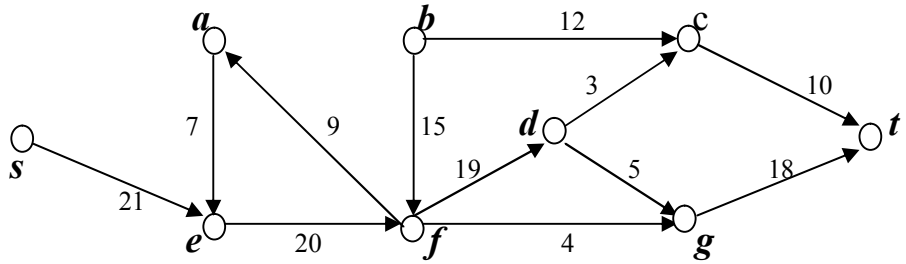
Пример 6. Найти максимальный поток в плоской сети:



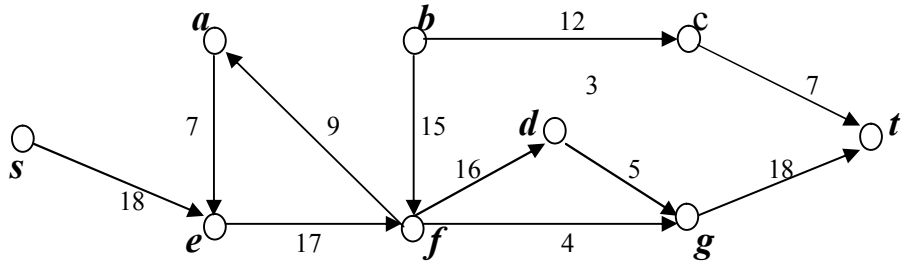
Решение. Выбираем верхний маршрут: ***s, a ,b, c, t***. Минимальная пропускная способность дуги равна 8. Пропускаем поток по этому маршруту: ***W = 8***. Веса дуг этого маршрута уменьшаем на 8 единиц, дугу ***(a, b)*** удаляем.



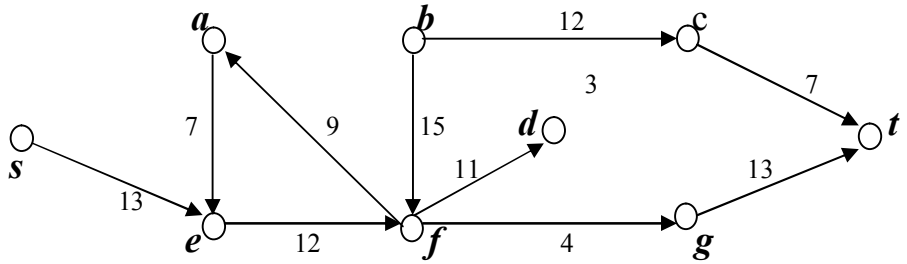
Шаг 2: верхний маршрут: ***s, a, e, f, d, c, t***. Минимальную пропускную способность, равную 4, имеет дуга ***(s, a)***. Удаляем эту дугу, веса остальных дуг маршрута уменьшаем на 4. Достигнута величина потока ***W = 8+4***.



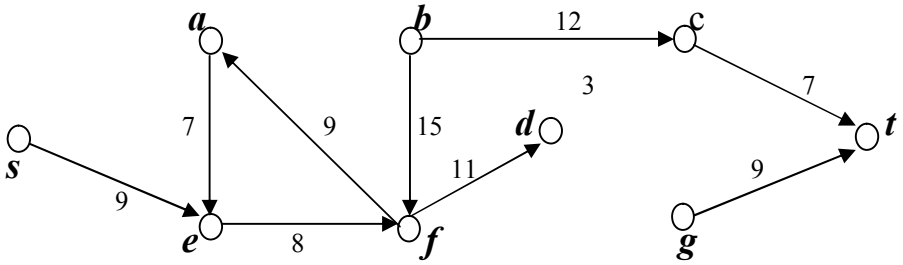
Шаг 3: верхний маршрут: ***s, e, f, d, c, t***. Минимальную пропускную способность, равную 3, имеет дуга ***(d, c)***. Удаляем эту дугу, веса остальных дуг маршрута уменьшаем на 3. Достигнута величина потока ***W = 8+4+3***.



Шаг 4: верхний маршрут: ***s, e, f, d, g, t***. Минимальную пропускную способность, равную 5, имеет дуга ***(d, g)***. Удаляем эту дугу, веса остальных дуг маршрута уменьшаем на 5. Достигнута величина потока ***W = 8+4+3+5***.



Шаг 5: верхний маршрут: ***s, e, f, g, t***. Минимальную пропускную способность, равную 4, имеет дуга ***(f, g)***. Удаляем эту дугу, веса остальных дуг маршрута уменьшаем на 4. Достигнута величина потока ***W = 8+4+3+5+4***.



Маршрута от источника до стока теперь нет. Максимальный поток найден: ***W = 8+4+3+5+4=24***. Используя протокол работы алгоритма, можно определить загрузку каждой дуги, сделать необходимые выводы.

Литература

1 Асанов, М.О. Дискретная математика: графы, матроиды, алгоритмы. Учебное пособие.– 2-е изд., испр. и доп. / М.О.Асанов, В.А.Баранский, В.В. Расин.– СПб.: Издательство «Лань», 2010.– 368 с. (ЭБС «Лань»).

2 Гаврилов, Г.П. Сборник задач по дискретной математике / Г.П. Гаврилов, А.А. Сапоженко. – М.: Наука, 1977.– 368 с.

3 Грэхем, Р. Конкретная математика. Основание информатики / Р. Грэхем, Д.Кнут, О.Паташник. – М.: Мир, 1998.– 703 с.

4 Киркинский, А.С. Элементы дискретной математики: Методические рекомендации и варианты заданий контрольных работ для студентов-заочников.–2-е изд. / А.С.Киркинский.– Алт. гос. техн. ун-т им. И.И.Ползунова.– Барнаул: АлтГТУ, 2014.– 48 с. (электронная библиотека АлтГТУ).

5 Липский, В. Комбинаторика для программистов / Витольд Липский.– М.: Мир, 1988. – 213 с.

6 Павловский, Е.В. Теория графов: конспект лекций / Е.В.Павловский. – Алтайский политехнический институт.– Барнаул, 1975.– 150 с.

7 Поздняков, С.Н. Дискретная математика: учебник для студ. вузов / С.Н.Поздняков, С.В.Рыбин. – М.: Издательский центр «Академия», 2008.– 448 с.

8 Шевелев, Ю.П. Дискретная математика. Учебное пособие /Ю.П.Шевелев.– СПб.: Издательство «Лань», 2008.– 592 с. (ЭБС «Лань»).