НАО «Восточно-Казахстанский технический университет имени Д. Серикбаева»

Дисциплина «Теоретико-числовые методы в криптографии»

ПРАКТИЧЕСКАЯ РАБОТА 7

ПОДСТАНОВКИ, ПОДГРУППЫ И СИММЕТРИИ ТРЕУГОЛЬНИКА

Методические указания и задание к выполнению

Аудиторная нагрузка: 5 академических часов. Сначала изучите методические сведения и разобранные примеры, затем выполните задания своего варианта. Поскольку порядок практических тем в силлабусе отличается от порядка лекций, необходимые понятия теории групп изложены здесь самостоятельно.

1 Цель работы

Освоить вычисления в группе подстановок: композицию, обратные элементы, циклы, чётность и порядок. Научиться строить циклические подгруппы, применять теорему Лагранжа и описывать симметрии правильного треугольника как подстановки его вершин.

2 Планируемые результаты и организация занятия

После выполнения работы студент различает степень подстановки и её порядок, согласует ручное решение с выбранным порядком композиции, проверяет критерий подгруппы, вычисляет смежные классы и объясняет результаты программной проверки.

Рекомендуемое распределение пяти академических часов: первый — композиция и обратные подстановки; второй — циклы и чётность; третий — порядки и циклические подгруппы; четвёртый — теорема Лагранжа и симметрии треугольника; пятый — программная проверка, оформление и защита. При академическом часе 45 минут общий объём составляет 225 минут.

3 Программное обеспечение

Достаточно Python 3 и любой среды, в которой можно запустить файл `.py`. Дополнительные пакеты не нужны. Все блоки Python из раздела 5 помещаются в **один файл в порядке следования**; блок «Ожидаемый вывод» не является кодом. Сначала выполните ключевые действия вручную, затем сверяйте их с программой.

4 Методические сведения и разобранные примеры

4.1 Подстановка и способы её записи

Подстановка степени n — биекция множества $\{1, \dots, n\}$ на себя. Множество всех таких биекций обозначают S_n; оно содержит $n!$ элементов. Для первого образа есть n вариантов, для второго — $n-1$ и так далее, что объясняет формулу.

В работе используется однострочная запись $[a_1, a_2, \dots, a_n]$: элемент i переходит в a_i. Квадратные скобки здесь обозначают список образов, **не класс вычетов**. Например,

$$\sigma = [2,3,1,5,4,6] = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 1 & 5 & 4 & 6 \end{pmatrix}.$$

Числа в нижней строке должны быть различными и содержать все значения от 1 до n. Запись $[2,2,1]$ не задаёт подстановку: один образ повторён, а число 3 не является образом ни одного элемента.

Цикл $(i_1\ i_2\ \dots\ i_\ell)$ означает переходы $i_1 \mapsto i_2$, затем $i_2 \mapsto i_3$, и в конце $i_\ell \mapsto i_1$. Элементы вне цикла остаются неподвижными. Поэтому [2,3,1] и (1 2 3) обозначают одну подстановку, но читаются по разным правилам.

4.2 Композиция: сначала правая подстановка

Во всём комплекте принято соглашение

$$(\sigma\tau)(i) = \sigma(\tau(i)).$$

Сначала действует τ, затем σ. Если результат записывается слева направо как $\sigma\tau$, это не означает, что первой выполняется σ. В таблице умножения строка задаёт левый множитель, столбец — правый.

В книге Л. М. Мартынова, в примере о группе подстановок в разделе 2.3, композиция записана в другом порядке: сначала левое отображение. При сопоставлении его примеров с данной методичкой порядок множителей нужно переводить в наше соглашение.

Пример 1. Пусть $\sigma = [2,3,1,5,4,6]$ и $\tau = [4,6,3,1,5,2]$. Построим таблицу образов.

i	**1**	**2**	**3**	**4**	**5**	**6**
$\tau(i)$	4	6	3	1	5	2
$\sigma(\tau(i))$	5	6	1	2	4	3
$\sigma(i)$	2	3	1	5	4	6
$\tau(\sigma(i))$	6	3	4	5	1	2

Получаем $\sigma\tau = [5,6,1,2,4,3]$, тогда как $\tau\sigma = [6,3,4,5,1,2]$. Достаточно сравнить образ 1: первое произведение переводит его в 5, второе — в 6. Следовательно, множители не перестановочны.

Композиция ассоциативна: обе подстановки $(\alpha\beta)\gamma$ и $\alpha(\beta\gamma)$ переводят i в $\alpha(\beta(\gamma(i)))$. Единица $e = [1,2,\dots,n]$ ничего не меняет. Каждая биекция имеет обратную биекцию. Вместе с замкнутостью композиции эти свойства показывают, что S_n — группа. При $n \geq 3$ она некоммутативна: достаточно подстановок (1 2) и (2 3), фиксирующих остальные элементы.

4.3 Обратная подстановка и уравнение в группе

Если $\sigma(i) = j$, то $\sigma^{-1}(j) = i$. Чтобы найти обратную подстановку в двухстрочной записи, меняют строки местами, затем сортируют столбцы по верхней строке. В примере

$$\sigma^{-1} = [3,1,2,5,4,6], \qquad \tau^{-1} = \tau.$$

Проверка выполняется в обе стороны: $\sigma\sigma^{-1} = \sigma^{-1}\sigma = e$. У произведения обращается также порядок множителей: $(\sigma\tau)^{-1} = \tau^{-1}\sigma^{-1}$, поскольку при таком порядке соседние взаимно обратные множители сокращаются.

Пример 2. Решим уравнение $\sigma x = \tau$. Умножаем **слева** на σ^{-1}, получаем $x = \sigma^{-1}\tau$. Для данных примера 1 это $x = [5,6,2,3,4,1]$. Подстановка в исходное уравнение возвращает τ. Умножение справа дало бы решение другого уравнения.

4.4 Независимые циклы и степени

Для разложения на циклы выберите ещё не использованный элемент и следуйте его образам до возвращения в начало. Повторяйте, пока не будут учтены все элементы. Орбиты не пересекаются: у каждого элемента ровно один образ и один прообраз. Поэтому получаются независимые циклы.

Для σ из примера 1 цепочки имеют вид $1 \mapsto 2 \mapsto 3 \mapsto 1$, $4 \mapsto 5 \mapsto 4$ и $6 \mapsto 6$:

$$\sigma = (1\ 2\ 3)(4\ 5)(6).$$

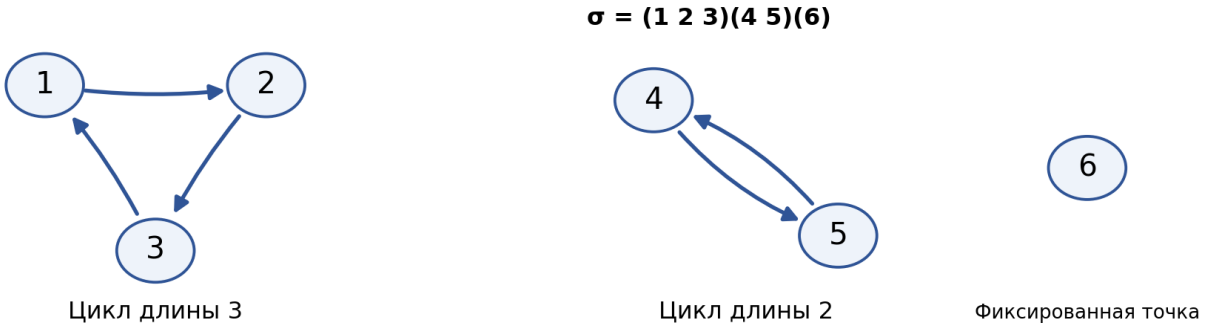


Рисунок 1. Независимые циклы подстановки σ.

Одноэлементные циклы обычно не пишут, но учитывают при подсчёте **общего числа циклов**. Независимые циклы коммутируют. Внутри одного цикла начальный элемент можно менять циклически: (1 2 3) = (2 3 1). Обращать направление без изменения подстановки нельзя: (1 3 2) = $(1\ 2\ 3)^{-1}$.

Степень k сдвигает позицию внутри цикла длины ℓ на $k \bmod \ell$. Например, $\sigma^2 = (1\ 3\ 2)$, $\sigma^3 = (4\ 5)$, а $\sigma^{19} = \sigma$, поскольку $19 \equiv 1 \pmod 3$ и $19 \equiv 1 \pmod 2$. Нулевая степень равна e, отрицательная вычисляется через обратную подстановку.

4.5 Инверсии и чётность

Инверсией однострочной записи называется пара позиций $i < j$, для которых $\sigma(i) > \sigma(j)$. Если число инверсий $I(\sigma)$ чётно, подстановка чётная; если нечётно — нечётная. Знак подстановки равен

$$\operatorname{sgn}(\sigma) = (-1)^{I(\sigma)}.$$

В записи [2,3,1,5,4,6] инверсии образуют пары **позиций** (1,3), (2,3), (4,5). Итого $I(\sigma) = 3$ и знак равен -1. Для $\tau = [4,6,3,1,5,2]$ количество меньших элементов справа последовательно равно 3,4,2,0,1,0; сумма равна 10, знак равен $+1$.

Цикл длины ℓ раскладывается в $\ell - 1$ транспозиций:

$$(a_1\ a_2\ \dots\ a_\ell) = (a_1\ a_\ell)(a_1\ a_{\ell-1}) \cdots (a_1\ a_2).$$

Транспозиция меняет знак. Это видно для обмена соседних позиций: чётность числа инверсий меняется на единицу. Обмен позиций $i < j$ можно выполнить за $2(j - i) - 1$ соседних обменов; это нечётное число. Следовательно, в любом разложении на транспозиции чётность их количества одна и та же, а знак произведения удовлетворяет правилу

$$\operatorname{sgn}(\sigma\tau) = \operatorname{sgn}(\sigma)\operatorname{sgn}(\tau).$$

Если разложение на независимые циклы, включая неподвижные точки, содержит c циклов, то сумма чисел $\ell - 1$ равна $n - c$. Получаем удобную вторую формулу:

$$\operatorname{sgn}(\sigma) = (-1)^{n-c}.$$

В примере $n = 6, c = 3$, знак снова равен -1. Произведение $\sigma\tau$ нечётно. Множество всех чётных подстановок образует подгруппу A_n: единица чётная, знак произведения двух чётных подстановок положителен, обратная к чётной также чётная. Для $n \geq 2$ чётных и нечётных подстановок поровну: умножение на фиксированную транспозицию даёт биекцию между ними. Поэтому $|A_n| = n!/2$.

4.6 Порядок элемента и циклическая подгруппа

Порядок элемента σ — наименьшее положительное t, такое что $\sigma^t = e$. Если длины независимых циклов равны $\ell_1, \dots, \ell_c$, то

$$\operatorname{ord}(\sigma) = \operatorname{lcm}(\ell_1, \dots, \ell_c).$$

Действительно, каждый цикл возвращается в исходное положение ровно на показателях, кратных его длине. Для одновременного возвращения всех циклов нужен наименьший общий кратный. Единичная подстановка имеет порядок 1.

В примере порядок σ равен $\operatorname{lcm}(3,2,1) = 6$. Группа, порождённая элементом, называется циклической и обозначается $\langle\sigma\rangle$. Её элементы — все целые степени σ; при порядке t достаточно степеней от 0 до $t-1$.

j	**Подстановка σ^j**	**Пояснение**
0	e	Единица
1	(1 2 3)(4 5)	Исходная подстановка
2	(1 3 2)	Транспозиция исчезла
3	(4 5)	Трёхцикл исчез
4	(1 2 3)	Остатки показателя 1 и 0
5	(1 3 2)(4 5)	Обратная к σ

Далее $\sigma^6 = e$, и список повторяется. Эти шесть степеней различны: равенство $\sigma^i = \sigma^j$ при $0 \le i < j < t$ означало бы $\sigma^{j-i} = e$ раньше минимального положительного показателя. Значит, $|\langle\sigma\rangle| = \operatorname{ord}(\sigma)$.

Элемент σ^j порождает всю эту группу тогда и только тогда, когда $\gcd(j,t) = 1$. В самом деле, его порядок равен $t/\gcd(j,t)$, поскольку это наименьшее положительное u, при котором $t \mid ju$. При $t = 6$ генераторы — σ и σ^5; их количество $\varphi(6) = 2$. Здесь используется функция Эйлера из предыдущей практической работы.

4.7 Подгруппа и теорема Лагранжа

Непустое подмножество H группы G называется подгруппой, если оно само является группой с той же операцией. Можно проверить наличие единицы, замкнутость произведения и замкнутость обращения. Ассоциативность наследуется от G.

Критерий подгруппы. Непустое $H \subseteq G$ является подгруппой тогда и только тогда, когда $ab^{-1} \in H$ для всех $a, b \in H$. Для достаточности сначала берём $a = b$ и получаем $e \in H$. Затем $eb^{-1} = b^{-1} \in H$. Наконец, применив условие к a и b^{-1}, получаем $ab \in H$. Необходимость следует из замкнутости операций в подгруппе.

Пример 3. Множество $H = \langle\sigma\rangle$ замкнуто: $\sigma^i\sigma^j = \sigma^{i+j}$, показатель приводится по модулю 6; обращение даёт σ^{-i}. Но множество $\{e, \sigma\}$ для данного примера не является подгруппой: σ^2 в него не входит.

Левый смежный класс имеет вид $gH = \{gh : h \in H\}$, правый — $Hg = \{hg : h \in H\}$. Отображение $h \mapsto gh$ биективно, поэтому каждый левый класс имеет столько же элементов, сколько H. Два левых класса либо не пересекаются, либо совпадают. Если $ah_1 = bh_2$, то $a = bh_2h_1^{-1}$, откуда каждый элемент aH принадлежит bH, и наоборот. Все элементы группы входят в такие классы, поскольку $g = ge \in gH$.

Теорема Лагранжа. Для конечной группы G и подгруппы H справедливо

$$|G| = |H|\,[G:H].$$

Здесь $[G:H]$ — количество левых смежных классов, называемое индексом. Равенство следует из разбиения G на непересекающиеся классы одинакового размера $|H|$. В частности, порядок подгруппы делит порядок группы; порядок каждого элемента также делит $|G|$, поскольку равен порядку порождённой им подгруппы.

Для нашего примера $|S_6| = 6! = 720$, $|H| = 6$, поэтому $[S_6 : H] = 120$. Не нужно перечислять все 720 подстановок. При этом число 6 в обозначении S_6 — **степень** подстановок, 720 — порядок группы, а 6 в $\mathrm{ord}(\sigma)$ — порядок конкретного элемента. Числа могут совпадать, но понятия различны.

Теорема даёт необходимое условие для возможного порядка подгруппы. Она сама по себе не утверждает, что для каждого делителя порядка произвольной конечной группы обязательно существует подгруппа такого порядка.

4.8 Группа симметрий правильного треугольника

Пронумеруем вершины: 1 — верхняя, 2 — нижняя левая, 3 — нижняя правая. Положим $r = (1\ 2\ 3)$ — поворот на 120° против часовой стрелки, $s = (2\ 3)$ — отражение относительно оси через вершину 1 и середину противоположной стороны.

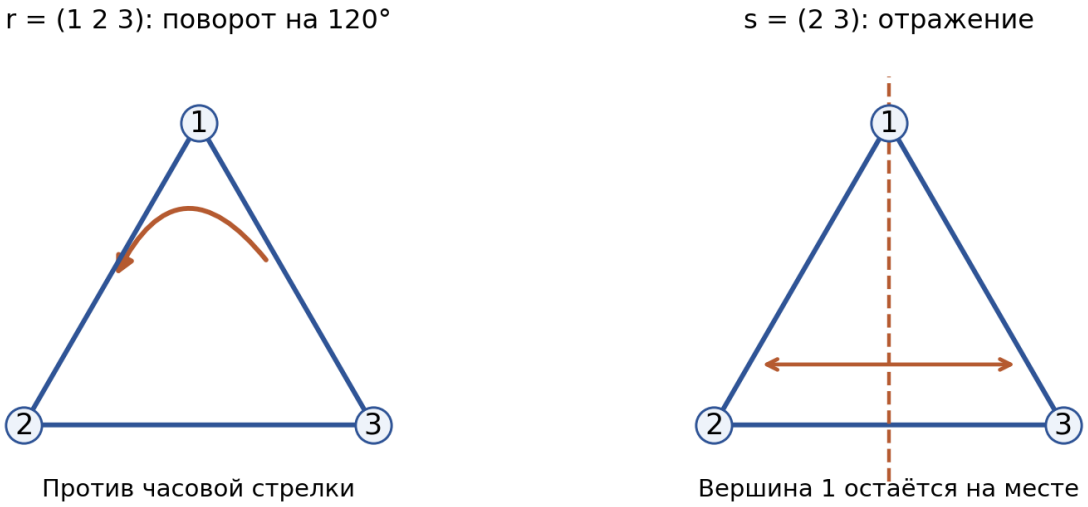


Рисунок 2. Две симметрии, порождающие группу правильного треугольника.

Все симметрии образуют группу D_3 порядка 6. Встречается и обозначение D_6 по числу элементов; здесь индекс 3 означает число вершин. Список элементов:

$$D_3 = \{e, r, r^2, s, rs, r^2 s\}, \qquad r^3 = s^2 = e, \quad sr = r^2 s.$$

Подстановка rs сначала отражает, затем поворачивает: $rs = (1\ 2)$, тогда как $sr = (1\ 3) = r^2 s$. Поэтому группа некоммутативна. Перестановки вершин точно определяют симметрии, и все шесть элементов S_3 реализуются; следовательно, $D_3 \cong S_3$.

Таблица умножения D_3**.** В клетке стоит произведение «элемент строки × элемент столбца»; первым действует элемент столбца.

×	e	r	r^2	s	rs	$r^2 s$
e	e	r	r^2	s	rs	$r^2 s$
r	r	r^2	e	rs	$r^2 s$	s
r^2	r^2	e	r	$r^2 s$	s	rs
s	s	$r^2 s$	rs	e	r^2	r
rs	rs	s	$r^2 s$	r	e	r^2
$r^2 s$	$r^2 s$	rs	s	r^2	r	e

Например, клетка строки rs и столбца r равна s, поскольку $(rs)r = r(sr) = r(r^2s) = r^3s = s$. У единицы порядок 1, у r, r^2 — 3, у трёх отражений — 2. Элемента порядка 6 нет, поэтому группа D_3 не является циклической.

Все её подгруппы: $\{e\}, \{e, r, r^2\}, \{e, s\}, \{e, rs\}, \{e, r^2s\}$ и D_3. Полнота следует из Лагранжа: возможны порядки 1, 2, 3, 6. Подгруппа порядка 2 определяется единственным неединичным элементом порядка 2; их три. Подгруппа порядка 3 циклическая и содержит оба нетождественных поворота; она одна.

Для $K = \langle s \rangle = \{e, s\}$ левые классы равны $K, rK = \{r, rs\}$ и $r^2K = \{r^2, r^2s\}$. Правые классы равны K, $Kr = \{r, r^2s\}$ и $Kr^2 = \{r^2, rs\}$. В обоих случаях индекс равен 3, но сами разбиения различны. Смежный класс, отличный от K, не является подгруппой: в нём нет единицы.

5 Программные примеры

5.1 Проверка входа, композиция и обращение

В коде подстановка хранится кортежем образов: математической записи [2,3,1] соответствует Python-кортеж `(2, 3, 1)`. Это **не запись цикла**. Нумерация элементов начинается с 1, индексы кортежа — с 0, поэтому при обращении к образу используется `- 1`. Функция `compose(p, q)` вычисляет pq: сначала `q`, затем `p`.

```
from math import gcd

def validate(p):
    p = tuple(p)
    if (not p or any(type(x) is not int for x in p)
            or sorted(p) != list(range(1, len(p) + 1))):
        raise ValueError("Нужны все числа от 1 до n без повторов")
    return p

def identity(n):
    return tuple(range(1, n + 1))

def compose(p, q):
    p, q = validate(p), validate(q)
    if len(p) != len(q):
        raise ValueError("Подстановки должны иметь одну степень")
    return tuple(p[q[i] - 1] for i in range(len(p)))

def inverse(p):
    p = validate(p)
    result = [0] * len(p)
    for i, value in enumerate(p, 1):
        result[value - 1] = i
    return tuple(result)
```

5.2 Циклы, инверсии, знак и порядок

Функция `cycles` сохраняет и одноэлементные циклы. Поэтому выражение `len(p) - len(cycles(p))` корректно для формулы знака. НОК вычисляется через НОД целочисленно.

```
def cycles(p):
    p = validate(p)
    seen, result = set(), []
    for start in range(1, len(p) + 1):
        if start in seen:
            continue
        cycle, x = [], start
        while x not in seen:
            seen.add(x)
            cycle.append(x)
            x = p[x - 1]
        result.append(tuple(cycle))
    return tuple(result)
```

Число инверсий определяется сравнением каждой пары позиций. Функция `order` последовательно накапливает НОК длин найденных циклов.

```
def inversion_count(p):
    p = validate(p)
    return sum(p[i] > p[j] for i in range(len(p))
               for j in range(i + 1, len(p)))

def sign(p):
    return (-1) ** inversion_count(p)

def order(p):
    result = 1
    for cycle in cycles(p):
        length = len(cycle)
        result = result * length // gcd(result, length)
    return result
```

5.3 Степени и циклическая подгруппа

Быстрый алгоритм использует ассоциативность композиции. Для отрицательного показателя сначала обращается подстановка. Циклическая подгруппа перечисляется отдельно последовательным умножением, начиная с единицы и до её первого повторения; повторная единица в список не добавляется.

```
def power(p, k):
    p = validate(p)
    if type(k) is not int:
        raise ValueError("Показатель должен быть целым")
    if k < 0:
        p, k = inverse(p), -k
    result = identity(len(p))
    while k:
        if k % 2:
            result = compose(result, p)
        p = compose(p, p)
        k //= 2
    return result

def cyclic_subgroup(p):
    p = validate(p)
    e, result = identity(len(p)), []
    current = e
    while True:
        result.append(current)
        current = compose(current, p)
        if current == e:
            return result
```

5.4 Проверка подгруппы и смежные классы

Для небольших конечных множеств проверяем все произведения и все обратные элементы. Это учебный алгоритм полного перебора: он удобен для наших подгрупп и D_3. Функция `cosets` возвращает множество неизменяемых множеств; равные смежные классы автоматически объединяются.

```
def is_subgroup(elements):
    elements = [validate(p) for p in elements]
    if not elements:
        return False
    n = len(elements[0])
    if any(len(p) != n for p in elements):
        raise ValueError("Элементы имеют разные степени")
    H = set(elements)
    if identity(n) not in H:
        return False
    return (all(inverse(p) in H for p in H)
            and all(compose(p, q) in H for p in H for q in H))

def cosets(G, H, left=True):
    G, H = set(G), set(H)
    if not is_subgroup(G) or not is_subgroup(H) or not H <= G:
        raise ValueError("Требуются конечная группа G и H <= G")
    result = set()
    for g in G:
        result.add(frozenset(compose(g, h) if left
                             else compose(h, g) for h in H))
    return result
```

5.5 Симметрии треугольника в Python

Строки `r2` и `r2s` в выводе программы обозначают r^2 и r^2s. Порядок имён фиксирован, чтобы таблица и классы выводились одинаково при повторном запуске. Функция `cayley_rows()` формирует строки таблицы умножения в порядке `names`.

```
e3 = identity(3)
r = (2, 3, 1)
s = (1, 3, 2)
names = ("e", "r", "r2", "s", "rs", "r2s")
D = dict(zip(names, (e3, r, power(r, 2), s,
                     compose(r, s), compose(power(r, 2), s))))
label = {p: name for name, p in D.items()}
G = set(D.values())
K = {e3, s}

def cayley_rows():
    return [[label[compose(D[a], D[b])] for b in names]
            for a in names]

def named_cosets(left=True):
    rank = {name: i for i, name in enumerate(names)}
    rows = [sorted((label[p] for p in C), key=rank.get)
            for C in cosets(G, K, left)]
    return sorted(rows, key=lambda row: rank[row[0]])
```

5.6 Запуск разобранного примера

Следующий блок использует данные примеров 1–3. После успешного запуска замените исходные подстановки и показатель данными своего варианта; для задачи о треугольнике используйте имена из столбцов g, h.

```
sigma = (2, 3, 1, 5, 4, 6)
tau = (4, 6, 3, 1, 5, 2)
k = 19
H = cyclic_subgroup(sigma)
print("sigma*tau =", compose(sigma, tau))
print("tau*sigma =", compose(tau, sigma))
print("sigma^" + str(k) + " =", power(sigma, k))
print("Циклы sigma:", cycles(sigma))
print("Инверсии и знак:", inversion_count(sigma), sign(sigma))
print("Порядок sigma:", order(sigma), "; размер H:", len(H))
print("r*s =", label[compose(r, s)])
print("s*r =", label[compose(s, r)])
print("Левые классы:", named_cosets(True))
print("Правые классы:", named_cosets(False))
```

Ожидаемый вывод

```
sigma*tau = (5, 6, 1, 2, 4, 3)
tau*sigma = (6, 3, 4, 5, 1, 2)
sigma^19 = (2, 3, 1, 5, 4, 6)
Циклы sigma: ((1, 2, 3), (4, 5), (6,))
Инверсии и знак: 3 -1
Порядок sigma: 6 ; размер H: 6
r*s = rs
s*r = r2s
Левые классы: [['e', 's'], ['r', 'rs'], ['r2', 'r2s']]
Правые классы: [['e', 's'], ['r', 'r2s'], ['r2', 'rs']]
```

5.7 Проверки результата и ошибочного ввода

Равенство знака проверяется двумя способами: через инверсии и через количество циклов. Проверки в группе D_3 включают все $6^3 = 216$ тройки элементов. Перебор подтверждает конкретную таблицу; общее доказательство ассоциативности композиции приведено в разделе 4.2.

```
assert compose(sigma, inverse(sigma)) == identity(6)
assert compose(inverse(sigma), sigma) == identity(6)
assert power(sigma, 0) == identity(6)
assert power(sigma, -1) == inverse(sigma)
assert power(sigma, order(sigma)) == identity(6)
assert sign(compose(sigma, tau)) == sign(sigma) * sign(tau)
assert sign(sigma) == (-1) ** (len(sigma) - len(cycles(sigma)))
assert len(H) == order(sigma) and is_subgroup(H)
assert is_subgroup([identity(6), sigma]) == (order(sigma) <= 2)
assert len(G) == 6 and is_subgroup(G)
assert power(r, 3) == e3 and power(s, 2) == e3
assert compose(s, r) == compose(power(r, 2), s)
assert len(cosets(G, K)) * len(K) == len(G)
assert cosets(G, K, True) != cosets(G, K, False)
for a in G:
    for b in G:
        for c in G:
            assert compose(compose(a, b), c) == compose(a, compose(b, c))
```

Некорректная подстановка и попытка умножить подстановки разных степеней должны вызывать `ValueError`. Пример также проверяет пустой ввод. Исключения здесь перехватываются намеренно: это проверка предусмотренного поведения.

```
for bad in [(1, 1, 3), (0, 1, 2), ()]:
    try:
        validate(bad)
    except ValueError:
        pass
    else:
        raise AssertionError("Ожидался ValueError")
try:
    compose((2, 1), (2, 3, 1))
except ValueError:
    pass
else:
    raise AssertionError("Ожидалась ошибка разных степеней")
print("Все проверки пройдены")
```

После всех проверок появляется строка `Все проверки пройдены`. При подготовке решения запустите файл обычной командой `python имя_файла.py`, без параметра `-O`, отключающего операторы `assert`.

6 Порядок выполнения работы

1. Разберите соглашение о композиции и примеры раздела 4. Вручную проверьте хотя бы по одному образу каждого произведения.
2. Выпишите данные своего варианта из раздела 8. Первые две подстановки принадлежат S_6, а g, h — группе D_3.
3. Выполните задания 1–5, сохраняя промежуточные таблицы и обоснования. Все шесть заданий обязательны; задания 5–6 включают общую для вариантов часть.
4. Запустите программные примеры, затем выполните задание 6 для собственных данных. При расхождении сначала проверьте порядок композиции и индексацию.
5. Оформите отчёт и подготовьтесь пояснить выбранные вычисления без обращения к коду.

7 Индивидуальные задания

Задание 1. Записи и обратные элементы. Для σ, τ составьте двухстрочную запись и разложение на независимые циклы. Найдите σ^{-1}, τ^{-1} в однострочной и циклической записях. Проверьте по одному произведению с каждой обратной подстановкой. Неподвижные точки можно опускать в записи циклов, но отдельно укажите их количество.

Задание 2. Композиция и уравнение. Найдите $\sigma\tau$ и $\tau\sigma$ по таблице образов. Установите, перестановочны ли данные подстановки. Вычислите σ^k по длинам циклов. Решите уравнение $\sigma x = \tau$ и проверьте подстановкой. Итоговые подстановки запишите в однострочной форме.

Задание 3. Чётность и порядок. Подсчитайте числа инверсий $I(\sigma), I(\tau)$; найдите их знаки двумя способами. Определите знак $\sigma\tau$ по правилу произведения и проверьте через его циклы. Найдите порядки $\sigma, \tau, \sigma\tau$; для каждого укажите длины независимых циклов, по которым вычислен НОК.

Задание 4. Циклическая подгруппа. Выпишите все элементы $H = \langle\sigma\rangle$ в порядке степеней от e до σ^{t-1}, где $t = \mathrm{ord}(\sigma)$. Обоснуйте замкнутость и наличие обратных элементов. Найдите все генераторы H в виде σ^j и как конкретные подстановки. Вычислите $[S_6 : H]$, объяснив применение теоремы Лагранжа.

Задание 5. Симметрии треугольника. Для данных g, h найдите gh, hg, g^{-1} и порядок g. Для трёх клеток таблицы D_3 — sr, $(rs)r$ и $(r^2s)(rs)$ — приведите вычисления по образам вершин или по определяющим соотношениям. Перечислите все подгруппы D_3 с их порядками и индексами. Для $K = \{e, s\}$ выпишите все различные левые и правые смежные классы и сравните два разбиения.

Задание 6. Программная проверка. Подставьте свои данные в функции раздела 5 и сопоставьте результаты заданий 1–5 с ручным решением. В отчёт включите исходный код и результаты проверок обратных элементов, нулевой и отрицательной степени, формулы знака, порядка и размера H. Проверьте замкнутость именно своего H; пример множества $\{e, \sigma\}$ из кода является подгруппой тогда и только тогда, когда порядок вашей σ не превышает 2; соответствующая проверка уже учитывает этот случай. Отдельно запустите общие проверки D_3, ошибочных входов и разных степеней подстановок.

8 Варианты индивидуальных заданий

Все строки в столбцах σ, τ — списки образов 1,2,3,4,5,6. Показатель k относится только к σ^k. Обозначения g, h читаются по разделу 4.8; в их произведении также сначала применяется правый множитель.

№	σ	τ	k	g	h
1	[2,3,1,5,4,6]	[4,6,3,1,5,2]	19	r	s
2	[1,5,2,4,6,3]	[3,4,5,6,1,2]	26	rs	r
3	[5,1,3,6,4,2]	[4,2,3,1,5,6]	33	r^2s	rs
4	[4,3,2,1,6,5]	[3,6,5,4,1,2]	40	e	r
5	[2,4,3,1,5,6]	[4,1,3,2,5,6]	47	r	rs
6	[3,5,4,1,6,2]	[6,2,3,1,5,4]	54	r^2	r^2
7	[3,1,6,5,4,2]	[5,6,2,3,1,4]	61	s	r^2s
8	[1,2,3,6,5,4]	[2,1,3,5,4,6]	68	rs	r^2
9	[3,2,1,5,4,6]	[4,5,3,2,1,6]	75	r^2s	r^2s
10	[1,4,3,5,6,2]	[1,6,3,2,4,5]	82	e	r^2
11	[1,3,4,2,6,5]	[4,1,6,2,3,5]	89	r	r^2s
12	[1,3,5,6,4,2]	[1,4,3,2,5,6]	96	r^2	s
13	[6,4,5,2,3,1]	[2,5,3,6,1,4]	103	s	e
14	[1,5,3,4,6,2]	[4,1,2,6,5,3]	110	rs	s
15	[2,5,6,4,3,1]	[6,1,5,4,2,3]	117	r^2s	e
16	[5,4,1,6,3,2]	[6,3,2,1,4,5]	124	e	s
17	[3,4,2,1,6,5]	[4,2,6,1,5,3]	131	r	e
18	[1,4,3,2,5,6]	[2,6,3,4,1,5]	138	r^2	rs
19	[2,1,6,4,5,3]	[3,4,1,2,6,5]	145	s	r
20	[2,3,1,5,4,6]	[3,1,2,5,4,6]	152	rs	rs

9 Требования к отчёту

Отчёт включает номер работы, цель, номер варианта и исходные данные; таблицы образов; разложения на циклы и ручные вычисления; полный список элементов и генераторов H; решения о треугольнике и смежных классах; код и результаты проверок; краткий вывод. В выводе объясните различие между степенью подстановки, порядком элемента и порядком группы.

Текст оформляется по ширине. Формулы должны оставаться редактируемыми. Для таблиц указывайте, какое соглашение о композиции используется. Скриншот выполнения программы дополняет решение, но не заменяет вычислений и математического обоснования.

10 Критерии оценивания

Составляющая	Баллы
Задание 1: записи, циклы и обратные элементы	15
Задание 2: два произведения, степень, уравнение и проверки	20
Задание 3: инверсии, знаки и порядки с обоснованием	15
Задание 4: все элементы и генераторы H, индекс	20
Задание 5: симметрии, подгруппы и смежные классы	20
Задание 6: работающий код и проверки своих данных	10
Итого	100

Арифметическая ошибка в одном образе и систематическое нарушение порядка композиции оцениваются раздельно. За подтверждённый правильный метод сохраняются соответствующие баллы. Для защиты студент должен пояснить хотя бы одно произведение, один расчёт порядка и одно применение теоремы Лагранжа.

11 Контрольные вопросы

1. Чем различаются однострочная запись и запись цикла?
2. Какая подстановка действует первой в произведении $\sigma\tau$?
3. Почему композиция ассоциативна, хотя обычно некоммутативна?
4. Как получить обратную подстановку и решить $\sigma x = \tau$?
5. Почему неподвижные точки нужно учитывать в формуле $(-1)^{n-c}$?
6. Как связаны инверсии, транспозиции и знак подстановки?
7. Почему порядок подстановки равен НОК длин её циклов?
8. Почему размер $\langle\sigma\rangle$ совпадает с порядком σ?
9. Какие степени σ порождают всю циклическую группу?
10. Что утверждает теорема Лагранжа и чего она не утверждает?
11. Почему D_3 некоммутативна и не является циклической?
12. Чем различаются левые и правые смежные классы по $K = \{e, s\}$?

Литература

1. Мартынов Л. М. **Алгебра и теория чисел для криптографии.** 2020. §1.7 — чётность подстановок; раздел 2.3 — группы и примеры, с учётом оговорённого отличия в порядке композиции; п. 2.7.1–2.7.2 — подгруппы и циклические группы; §12.1 — смежные классы и теорема Лагранжа.
2. Trappe W., Washington L. C. **Introduction to Cryptography with Coding Theory.** 2nd ed. 2006. §3.6 — функция Эйлера и степенные соотношения; используется для связи с количеством генераторов циклической группы и предыдущей темой курса.

Примеры, индивидуальные варианты, иллюстрации и программные реализации подготовлены для этой работы. Ручные вычисления и код используют одно и то же соглашение: сначала правая подстановка.