МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Московский институт электронной техники»

Кафедра «Высшей математики №1»

Вологжанин Никита Андреевич

Бакалаврская работа

по направлению 01.03.04 «Прикладная математика »

Представление конечных решеток конгруэнциями унарных алгебр

Студент _____ **/Н.А. Вологжанин/**
группы ПМ-41

Руководитель _____ **/И.Б. Кожухов/**
профессор кафедры ВМ-1.

Москва 2024 г.

Определения :

Отношения эквивалентности и классы эквивалентности :

Пусть K – отношение эквивалентности на заданном множестве R , a n – элемент из R . Будем рассматривать множество всех элементов из R , находящихся в отношении K к элементу n .

Классом эквивалентности R_n будет называться множество всех элементов R , находящихся в отношении K к элементу n , то есть множество

$$R_n = \{x \in R : x\ R\ n\}$$

Свойства классов эквивалентности

Пусть K – отношение эквивалентности на множестве R и $R_a, R_b, \ldots \ldots, R_z, \ldots -$ все классы эквивалентности для K . Тогда эти классы имею следующие свойства .

Свойство 1 .

∀ элемента $n \in R$ выполняется условие

$$n \in R_n$$

Действительно, по определению, класс $R_n = \{x \in R : x\ R\ n\}$. Тогда для элемента n должно выполняться условие $n \in R \leftrightarrow$x R n , которое выполняется в связи с тем, что отношение K рефлексивно по определению отношения эквивалентности. Следовательно, $n \in R$.

Как следствие этого свойства можно сказать, что всякий класс R_n является непустым множеством.

Свойство 2

Пусть R_a и R_b классы эквивалентности для отношения K . Классы R_a и R_b равны тогда и только тогда , когда элемент n находится в отношении K к элементу b

$$R_a = R_b \leftrightarrow\ n\ R\ b$$

Свойство 3

Пусть R_a и R_b классы эквивалентности для отношения K и a $\notin R_b$. Тогда классы R_a и R_b не имеют общих элементов .

$$R_a \neq R_b \rightarrow R_a \cap R_b = \varnothing$$

Свойство 4

Объединение всех классов эквивалентности множества R равно множеству R

$$\cup_{n \in R} \ R_a = R$$

Унарная операция

Под унарной операцией будем понимать операцию с одним операндом. Мы будем рассматривать функцию ϕ A → A , где A –двенадцатиэлементное множество { 0,1,2,3,4,5,6,7,8,9,10,11} , а ϕ унарная операция над A .

Конгруэнция

Будем понимать конгруэнцию , как отношение эквивалентности на носителе алгебраической структуры , которое сохраняет операции этой структуры . Формально, для алгебраической структуры (A,*) , конгруэнция является бинарным отношением ~ на множестве A , которое обладает следующими свойствами :

1. Рефлексивность : ∀ a ∈ A , a~a.
2. Симметричность : Если a~b , то и b~a.
3. Транзитивность : a~b , b~c , то и a~c.
4. Совместимость с операциями : ∀ a,b,c,d ∈ A , если a~b и c~d , то a*c~b*d

Решетка

Частично упорядоченное множество < L; ≤ > - решетка , если sup(H) и inf(H) существует для любого непустого конечного подмножества H множества L .

Конечные решетки

Суть конечных решеток в том , что они состоят из конечного множества элементов. Данные решетки , как и решетки вообще, обязательно обладают sup и inf , причем под sup мы будем понимать 1 , а под inf 0 , для них так же выполняются следующие аксиомы : рефлексивность , симметричность , транзитивность .

Алгеброй с операторами называется универсальная алгебра с дополнительной системой операторов – унарных операций, действующих как эндоморфизмы относительно основных операций.

Отношение $P(x1,\ x2,...,\ xn)$ на множестве A называется стабильным относительно n-арной операции F, определенной на этом множестве, если для любых элементов $a_{i1}, a_{i2}, \ldots, a_{in}$ (i= 1, 2,..., m) множества A из истинности отношений $P(a_{i1}, a_{i2}, \ldots, a_{in})(i = 1, 2,..., m)$ вытекает истинность отношения $P(F(a_{11}, \ldots, a_{m1}),..., F(a_{1n}, \ldots, a_{mn}))$. Класс конгруэнции θ, содержащий элемент a, будем обозначать через $[a]\theta$

Через Con$A$ обозначается решетка конгруэнций алгебры A, через Sub$A$ — решетка ее подалгебр, через End$A$ и Aut$A$ — соответственно, полугруппа ее эндоморфизмов и группа автоморфизмов.

Решетка $\langle L,\ \vee,\ \wedge\rangle$ называется дистрибутивной, если в ней выполняется тождество $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$.

Решетка $\langle L,\ \vee,\ \wedge\rangle$ называется модулярной, если в ней выполняется квазитождество $x \leqslant z \Rightarrow x \vee (y \wedge z) = (x \vee y) \wedge z$.

Если каждый элемент решетки обладает в точности одним дополнением, то она называется решеткой с единственными дополнениями.

Решетка L называется геометрической,если L — полумодулярная алгебраическая решетка, в которой компактными элементами являются конечные объединения атомов и только они.

Гомоморфизм

Отображение между двумя алгебраическими структурами (например, группами, кольцами, полями и т.д.), которое сохраняет операции этих структур. Другими словами, если у нас есть две алгебраические структуры A и B с операциями

(например, умножение, сложение и т.д.), то гомоморфизм от A к B отображает элементы A на элементы B таким образом, что операции сохраняются. Например, если f : A→B является гомоморфизмом групп, то для всех x,y ∈A выполняется f(x*y)=f(x)*f(y), где * - операция группы.

Изоморфизм

Специальный вид гомоморфизма между двумя алгебраическими структурами, который является взаимно однозначным и сохраняет все свойства структур. Другими словами, если у нас есть две алгебраические структуры A и B с операциями, то изоморфизм между A и B обеспечивает взаимно однозначное соответствие между элементами A и B, сохраняя при этом все алгебраические свойства структур. Например, если f : A→B является изоморфизмом групп, то для всех x,y ∈ A выполняется f(x*y)=f(x)*f(y) где * - операция группы, и f является биекцией.

Автоморфизм

Изоморфизм группы на себя , то есть биективное отображение ϕ : G -> G , сохранящее групповую операцию .

Примитивное действие группы G на множестве Ω означает, что G действует на Ω так, что это действие не имеет нетривиальной блоковой структуры. Это можно определить следующим образом:

1. Множество Ω называется множество действия группы G.
2. Блоки в Ω— это подмножества Ω, которые группа G переводит в себя. То есть, если B — блок, то для любого элемента g∈G, g(B)=B.
3. Действие группы G на Ω называется транзитивным, если для любых двух элементов α,β∈Ω существует элемент g∈G, такой что g(α)=β.
4. Действие группы G на Ω называется примитивным, если оно транзитивно и не существует нетривиального разбиения Ω на блоки, где каждый блок содержит больше одного элемента и меньше, чем Ω целиком. То есть единственными блоками в Ω являются само Ω и одноэлементные множества.

Простое транзитивное действие

Например , пусть G – симметричная группа S_n , действующая на множестве {0,1,2,…,n } . Это действие транзитивно и примитивно , так как S_n перевести любой элемент в любой другой.

В теории групп “core” подгруппы H в группе G обозначает пересечение всех сопряженных подгрупп H в G :

$$core_G(H) = \bigcap_{g \in G} gHg^{-1}$$

Данный объект часто называю нормальное ядро подгруппы H в G .

Символы :

$\forall$ - для любого

$\exists$ - существует , найдется

Z – множество целых чисел

A – заданная алгебра

Con(A) – решетка конгруэнции над A

$\Leftrightarrow$ - тогда и только тогда , когда

$\Rightarrow$ - следует

$\sim$ - эквивалентно

$\cap$ - пересечение

$\cup$ - объединение

$\vee$ - или

$\wedge$ - и

$\equiv$ - сравнимость

$\boldsymbol{L_7}$ – семиэлементная решетка

N – множество натуральных чисел

Inn(T) -внутренний автоморфизм группы T

$core_G(H)$ – нормальное ядро подгруппы H в G

Аннотация

Тема : «Представление конечных решеток конгруэнциями унарных алгебр »

Объем дипломной работы 40 страниц , на которых размещены 5 рисунков . При написании дипломной работы использовалось 7 источников .

В дипломную работу входит : определения , список символов , введение , две главы, итоговое заключение и приложение А.

В главе определения , дается определения всем терминам , которые будут использоваться в дипломной работе

В введении раскрывается актуальность исследования по выбранной теме , ставится проблема , цель и задачи исследования , определяется объект и предмет научных поисков .

В первой главе рассматривается подход к решению поставленной задачи доктором физико-математических наук Уильямом Дж.ДеМео , рассматриваются методы решения для схожих задач .

Во второй главе происходит поэтапное решение задачи , в конце главы описывается краткий вывод .

Заключение посвящено полученным в ходе работы результатам .

В приложении А находится весь код , который был написан для решения поставленной задачи

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность работы

В мире математики одними из фундаментальных объектов изучения являются алгебры. Алгебра, обозначаемая как A = <A, F>, состоит из непустого множества A и операций F. Среди наиболее важных примеров алгебр можно выделить решетки, группы, кольца и модули. Для более глубокого понимания конкретной алгебры A часто изучаются её представления, которые представляют собой гомоморфизмы из A в другую алгебру B. Одной из ключевых характеристик такого гомоморфизма является его ядро, определяемое как множество {(x, y) ∈ A^2 | ϕ(x) = ϕ(y)}, которое представляет собой конгруэнтное отношение алгебры A. Это отношение указывает на то, как изменяется A при представлении её образа в B через ϕ. Каждый гомоморфизм порождает конгруэнтное отношение, и множество всех таких отношений, Con A, образует решетку. Например, в случае, если A является группой, Con A изоморфно решетке нормальных подгрупп A.

Каждому конгруэнтному отношению θ ∈ Con A соответствует естественный гомоморфизм A на факторалгебру A/θ, который имеет θ в качестве ядра. Следовательно, существует взаимно однозначное соответствие между множеством Con A и естественными гомоморфизмами. Форма решетки Con A предоставляет полезную информацию о самой алгебре и её представлениях, например, о возможности разложения или вложения A в произведение более простых алгебр.

В контексте произвольной алгебры необходимо рассмотреть, существуют ли какие-либо априорные ограничения на форму её решетки конгруэнций. Однако известный результат Грэтцера и Шмидта указывает на то, что таких ограничений практически нет.Солгасно теореме , Гретцера и Шмидта : Каждая конечная распределительная решетка изоморфна решетке соответствия некоторой конечной решетки. Фактически, каждая алгебраическая решетка является решеткой конгруэнций некоторой алгебры. Это приводит к важному вопросу о представимости конечных решеток. Если для произвольной конечной решетки L всегда можно найти конечную алгебру A, у которой L является решеткой

конгруэнций, то говорят, что решетка L представима. Вопрос о том, является ли каждая конечная решетка представимой, известен как проблема конечного представления решетки (FLRP). Это важный вопрос современной алгебры, и его нерешенность представляется удивительной.

Объектом исследования являются унарные алгебры

Предметом исследований является решетка $\boldsymbol{L_7}$.

Для достижения поставленной цели было необходимо решить следующие задачи :

1. Описать данную решетку.
2. Ввести минимальную алгебру на решетке $\boldsymbol{L_7}$.
3. Найти все классы эквивалентности для заданной алгебры на заданной решетке.
4. Найти отображения сохраняющие классы.
5. Найти все разбиения множества A на классы , чтобы все ϕ_i сохраняли эти классы.
6. Построить решетку конгруэнций.

Научная новизна : в ходе исследования были получены алгоритмы, облегчающие работу с заданным типом решеток .

Теоретическая и практическая значимость . Данная работа носит сугубо теоретический характер , все методы и результаты работы могут быть использованы для исследований связанных с изучением конечных решеток.

Методы исследования . В ходе работы были использованы методы теории решеток , универсальной алгебры и программирования.

1.Постановка задачи .

1.1 Общие сведения

В своей докторской диссертации Уильям Дж.ДеМео занимается рассмотрением конечных решеток и методов их получения . Очень важная и давняя проблема универсальной алгебры заключается в том - каждая ли конечная решетка изоморфна решетке конгруэнций конечной алгебры . Работы в данной области ведутся уже давно , но однозначного ответа пока не удалось дать . Суть в том , что пока ответ на этот вопрос не будет дан , наше понимае конечных алгебр так и останется неполным , мы не можем сказать , существует ли какие-нибудь ограничения на форму решетки конгруэнций . Если будет найдена конечная решетка , которая не встречается в решетке конгруэнций конечной алгебры , то можно будет заявлять о существовании ограничений .

В данной работе будет рассмотрена семиэлементная решетка и методы нахождения конечной алгебры . Также будут рассмотрены методы нахождения решения с помощью компьютера , а именно языка программирования Python .

Стоит упомянуть , что все решетки , которые содержат не более шести элементов , являются представимыми .

Вообще , если L_7 представима в виде решетки конгруэнции конечной алгебры , то она долна появиться , как интервал в решетке подгрупп конечной группы .

Всего существует 53 решетки , которые содержат не более семи элементов , представления для большинста из них найти достаточно просто , взять, например, дистрибутивные решетки , котороые являются представимыми по следующей теореме .

Теорема 1.1 (Berman , Quackenbush и Walk) Любая конечная дистрибутивная решетка является представимой .

Теореема 1.2

Если $L \leq Eq(x)$, то $L = Con(A)$ для некоторой алгебры $A = \langle X,F \rangle$ тогда и только тогда , когда L закрытая

Пользуясь теоремой 1.2 , докажем теорему 1.1

В контексте Теоремы 1.2 , решетка L называется закрытой , если для любого отношения эквивалентности $\theta \in Eq(X)$, если все отношения в L ниже θ пересекаются (то есть их пересечение есть в L) , то и само θ находится в L.

Доказательство .

Примим , что L≤ Eq(X) . Без ограничения общности , считаем , что L≤ Eq(X) , то есть L состоит из отношений эквивалентности на множестве X.

Выберем θ ∉ L . Зафиксируем $\theta \in Eq(X) \setminus L$.

Определим α и β : $\alpha = \bigwedge \{\rho \in L \mid \rho \geq \theta\}$, $\beta = \bigvee \{\rho \in L \mid \rho \leq \theta\}$

Здесь α – наибольший элемент в L , который меньше или равен θ , а β - наименьший элемент в L , который больше или равен θ.

Неприводимый γ . Пусть γ неприводима в L и лежит между α и β , то есть $\alpha < \gamma < \beta$, заметим , что $\theta \not\leq \gamma$.

Определим δ : $\delta = \bigvee \{\rho \in L \mid \rho \leq \gamma\}$

Если $\delta > \theta$, то δ будет больше γ , что противоречит предположению о простоте γ . Следовательно , $\delta \leq \theta$.

Выберем пары (u,v) и (x,y) , для которых (u,v) ∈ $\delta \setminus \theta$, заметим , что u ≠v . Также выберим (x,y) ∈ $\theta \setminus \gamma$ и заметим , что x≠ y .

Построим множество B . Пусть B состоит из блоков по y и определяется как h ∈ B. Тогда ясно , что нарушает θ . h включает в себя все элементы множеств :

$$B_1 = \{\rho \in L | \rho \leq \gamma\}, \ B_2 = \{\rho \in L | \rho \leq \beta\}$$

и L = $B_1 \cup B_2$.

С учетом того , что $\theta \in Eq(X) \setminus L$, мы можем построить такое h для каждого $\theta \in Eq(X) \setminus L$.

$$H = \{ \theta \setminus L \}$$

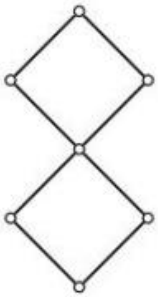
и пусть A будет алгеброй < X,H> . Тогда L=Con(A) .

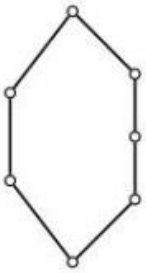
Теорема доказана ■

Таким образом, доказательство теоремы 1.1 использует теорему 1.2 для демонстрации, что любая конечная дистрибутивная решетка может быть представлена как решетка конгруэнций некоторой алгебры A.

Некоторые другие решетки могут быть представлены путем поиска (с помощью компьютера) замкнутых представлений L ≤ Eq(X) над некоторым

небольшим множеством , скажем , |X| < 8 . Третья находятся путем проверки того , что они получены в результате применения операции , при которых L_3 замкнуты . Например , решетка слева , это порядковая сумма двух копий дистрибутивной решетки 2*2 , справа параллельная сумма дистрибутивных решеток 2 и 3 .

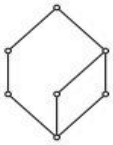




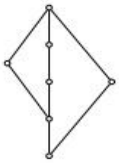
Используя данные методы не трудно доказать существование решетки конгруэнций .

1.2.1Подходы к решению похожих задач

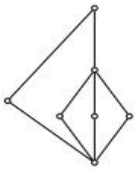
На рисунке ниже будут показаны семиэлементные решетки без очевидного представления решетки конгруэнции .



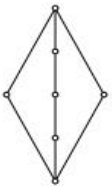
L_{19}



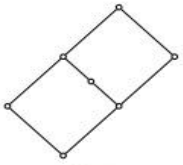
L_{20}



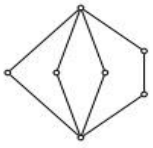
L_{17}



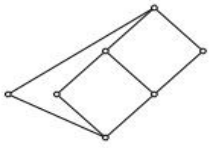
L_{13}



L_{11}



L_9



L_7

Хотя на рисунке показано 7 решеток , 4 из этих семи решеток являются самодвойственными , поэтому всего существут 10 решеток , для которых найти представление не так просто .

Теперь рассмотрим методы , которыми были решены данные решетки , все , кроме $\boldsymbol{L_7}$ **.** Решетки L_{19} и L_{20} были найдены с помощью Sage[1] путем замыкания . Решетка L_{13} является интервалом в решетке подгрупп , с помощью GAP[2] ,было получено , что G = $(C_2 * C_2 * C_2 * C_2) * A_5$ имеет подгруппу H ~ A_4, такую что [H,G] = L_{13} .В итоге получается , что действие G на смежных классах G/H является алгеброй состоящей из 80 элементов . L_{17} – была решена алгебраическим путем , в силу того , что она является подрешеткой Sub(A_4) и подгруппы A_4 (группа всех четных перестановок четырехэлементного множества) . Решение для L_{11}и L_9 , так же были найдены алгебраическим путем .

1.2.2Результаты полученные ДеМео

Итак, из всех выше представленных решеток , осталась только решетка $\boldsymbol{L_7}$ **.** Это наименьшая решетка **,** для которой неизвестно существует ли представление .

На самом деле , данныя решетка является в какой-то мере исключительной решеткой . ДеМео в своей работе не удалось найти конечную алгебру , имеющую решетку конгруэнций, изоморфную решетке $\boldsymbol{L_7}$ **.** О наших результатах будет изложено в конце второй главы .

Далее рассмотрим результат , который получил ДеМео в свой работе и приступим к собственному решению . Предположим , что A – конечная алгебра в которой ConA ≅ $\boldsymbol{L_7}$. Тогда можно утверждать , что сама алгебра A должна быть изоморфна транзитивному G-множеству . Следовательно , если $\boldsymbol{L_7}$ представима , то мы можем предположить существование кончной группы G с подгруппой H < G без ядра , такой , что $\boldsymbol{L_7}$ изоморфна интервальной подрешетке [H,G]Sub(G) .

Теорема 1.3 (О`Нен-Скотт) HS (голоморф простой группы)

Пусть *T* — конечная неабелева простая группа. Тогда M = T*T действует на

Ω = T помощью $t^{t_1,t_2} = t_1^{-1}tt_2$. Теперь M имеет две минимальные нормальные подгруппы N_1, N_2 , каждая из которых изоморфна T и каждая действует регулярно на Ω , одна с помощью правого умножения, а другая с помощью левого умножения. Действие группы M является примитивным и если мы возьмём $\alpha = 1_T$, мы получим $M_\alpha = \{(t,t)|t \in T\}$, которая включает Inn($T$) из Ω . Фактически любой автоморфизм будет группы T будет действовать на Ω . Примитивная группа типа HS является тогда любой группой G, такой, что M ≅ T.Inn(T)≤G≤T.Aut(T) . Все такие группы имеют N_1 и N_2 в качестве минимальных нормальных подгрупп.

Теперь поговорим об ограничениях , которые дествуют на решетку $\boldsymbol{L_7}$. Важным ограничением будет то , что G должна действовать примитивно на смежных классах одной из максимальных подгрупп . Так же должна существовать возможность описать описания решетки $\boldsymbol{L_7}$ в терминах теоремы O`Нена-Скотта , характерезующая примитивные группы перестановок .

Напишим важную теорему , полученную ДеМео :

Теорема 1.4

Предположим , что H < G – конечные группы с $core_G(H) = 1$, и пусть $\boldsymbol{L_7}$ ≅[H,G] . Тогда справедливы следующие утверждения :

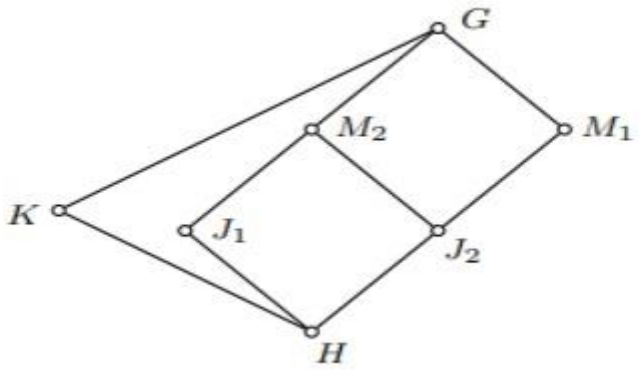
1. G – примитивная группа перестановок
2. Если N $\triangleleft$ G , то $C_G(N) = 1$
3. G содержит нетревиальные абелевы подгруппы
4. G неразложима
5. G попарно неразложима
6. За возможным одним исключением не более , чем одной максимальной подгруппы , все собственные подгруппы в интервале [H,G] свободны от ядра .

Далее в своей работе ДеМео приводит доказательство теоремы 1.3 .

Подведем итог полученный ДеМео. G действует примитивно на смежных классах K , а так же примитивно действует на смежных классах хотя бы одного из M_1 или M_2 . Предположим , что M_1 не имеет ядер , так что G – примитивная перестановка группы на смежные классы группы M_1 и пусть N – минимальная

нормальная подгруппа группы G . Как было установлено N имеет тривиальный стабилизатор , поэтому она не является абелевой и является единственной минимальной подгруппой группы G . В том случае $M_2 \leq$ NH, тогда получим H < J_2 <NH влечет за собой N$\bigcap M_1 \neq 1$. Совершенно такой же результат мы получили бы , если сказали , что M_2 не имеет ядра ,тогда $M_1 \leq$ NH и H < J_2 <NH влечет N$\bigcap M_2 \neq 1$.

Чтобы лучше понять о чем идет речь , в абзаце выше , приложим следующий рисунок



Полученный результат говорит о взаимодействии классов между собой .

2.Решение задачи .

Решение поставленной задачи .

Решение задачи будет разделено на 3 этапа :

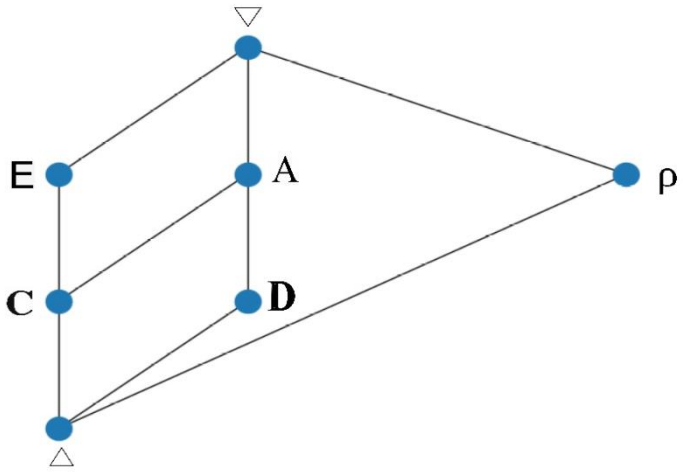
1 . Нахождение всех элементов решетки .

2 . Нахождение отображений , сохраняющих классы и проверка , что найдены все отображения

3 . Нахождение разбиений , которые будут сохранять классы .

Этап 2.1 .

Нам дана семиэлементная решетка



Для простоты дальнейшей работы будем использовать буквы , под которыми подразумеваются классы эквивалентности . Мы вводим алгебру A = {0,1,2,3,4,5,6,7,8,9,10,11} , для заданной решетки . Выпишем все классы эквивалентности для данной решетки :

A = {0,2,4,6,8,10},{1,3,5,7,9,11}

D = {0,4,8},{1,5,9},{2,6,10},{3,7,11}

E = {0,3,6,9},{1,4,7,10},{2,5,8,11}

C = {0,6},{1,7},{2,8},{3,9},{4,10},{5,11}

Δ = {0},{1},{2},{3},{4},{5},{6},{7},{8},{9},{10},{11}

∇ = {0,1,2,3,4,5,6,7,8,9,10,11}

Теперь поговорим об элементе ρ . Чтобы найти ρ ,должны быть выполнены 2 условия : 1 . При объединении ρ с A,D,E,C должно получаться ∇ . 2 . При пересечении ρ с A,D,E,C должно получаться Δ .

Это можно переписать так :

$P \cup A = \nabla$ $P \cap A = \Delta$

$P \cup C = \nabla$ $P \cap C = \Delta$

$P \cup D = \nabla$ $P \cap D = \Delta$

$P \cup E = \nabla$ $P \cap E = \Delta$

Рассмотрим подробнее варианты , которые нас будут удовлетворять . Для этого обратим наше внимание уже на имеющиеся у нас классы эквивалентности. Понятно , что ρ должно быть разбито на 6 классов , так как только в этом случае мы можем получить пересечение , которое даст нам Δ , это будет следовать из пересечения с A . Так же очевидно , что в классах ρ должны лежать четное и нечетное число , в противном случае пересечение с A не даст Δ . Так же при дальнейшем рассмотрении мы получим еще одно важное требование : Не может быть двух элементов с одинаковым остатком по mod .

Приведем пример разбиения , которое не будет удовлетворять условиям : {0,1},{2,3},{4,5},{6,7},{8,9},{10,11} , в данном случае $P \cup D \neq \nabla$.

Теперь рассмотрим ρ, который будет удовлетворять всем требованиям : {0,1},{2,3},{4,11},{8,7},{6,5},{10,9}

Стоит отметить , что приведенное выше ρ не является единственным , в ходе решения задачи было найдено 62 таких ρ . Ниже они будут все перечислены .

{0,7},{2,1},{4,3},{6,5},{8,9},{10,11} {0,7},{2,9},{4,5},{6,11},{8,1},{10,3}

{0,7},{2,1},{4,9},{6,5},{8,3},{10,11} {0,7},{2,9},{4,11},{6,5},{8,1},{10,3}

{0,7},{2,1},{4,5},{6,11},{8,3},{10,9} {0,11},{2,1},{4,5},{6,7},{8,3},{10,9}

{0,7},{2,1},{4,3},{6,11},{8,9},{10,5} {0,11},{2,1},{4,9},{6,7},{8,3},{10,5}

{0,7},{2,1},{4,9},{6,11},{8,3},{10,5} {0,11},{2,1},{4,3},{6,7},{8,9},{10,5}

{0,7},{2,1},{4,5},{6,11},{8,9},{10,3} {0,11},{2,1},{4,5},{6,7},{8,9},{10,3}

{0,7},{2,3},{4,9},{6,5},{8,1},{10,11} {0,11},{2,3},{4,5},{6,7},{8,1},{10,9}

{0,7},{2,3},{4,5},{6,11},{8,1},{10,9} {0,11},{2,3},{4,5},{6,1},{8,7},{10,9}

{0,7},{2,3},{4,11},{6,5},{8,1},{10,9} {0,11},{2,3},{4,9},{6,7},{8,1},{10,5}

{0,7},{2,3},{4,9},{6,11},{8,1},{10,5}
{0,7},{2,9},{4,3},{6,5},{8,1},{10,11}
{0,7},{2,9},{4,3},{6,11},{8,1},{10,5}

{0,11},{2,3},{4,9},{6,1},{8,7},{10,5}
{0,11},{2,7},{4,5},{6,1},{8,3},{10,9}
{0,11},{2,7},{4,9},{6,1},{8,3},{10,5}

{0,11},{2,7},{4,3},{6,1},{8,9},{10,5}
{0,11},{2,7},{4,5},{6,1},{8,9},{10,3}
{0,11},{2,9},{4,3},{6,7},{8,1},{10,5}
{0,11},{2,9},{4,5},{6,1},{8,7},{10,3}
{0,11},{2,9},{4,5},{6,7},{8,1},{10,3}
{0,1},{2,3},{4,11},{6,5},{8,7},{10,9}
{0,1},{2,3},{4,9},{6,5},{8,7},{10,11}
{0,1},{2,3},{4,9},{6,11},{8,7},{10,5}
{0,1},{2,3},{4,5},{6,11},{8,7},{10,9}
{0,1},{2,7},{4,11},{6,5},{8,3},{10,9}
{0,1},{2,7},{4,9},{6,5},{8,3},{10,11}
{0,1},{2,7},{4,9},{6,11},{8,3},{10,5}

{0,1},{2,7},{4,5},{6,11},{8,3},{10,9}
{0,1},{2,7},{4,3},{6,5},{8,9},{10,11}
{0,1},{2,7},{4,3},{6,11},{8,9},{10,5}
{0,1},{2,7},{4,11},{6,5},{8,9},{10,3}
{0,1},{2,9},{4,3},{6,5},{8,7},{10,11}
{0,1},{2,9},{4,3},{6,11},{8,7},{10,5}
{0,1},{2,9},{4,5},{6,11},{8,7},{10,3}
{0,1},{2,7},{4,5},{6,11},{8,3},{10,9}
{0,1},{2,3},{4,11},{6,5},{8,7},{10,9}
{0,5},{2,9},{4,11},{6,7},{8,1},{10,3}
{0,1},{2,3},{4,9},{6,5},{8,7},{10,11}
{0,1},{2,3},{4,9},{6,11},{8,7},{10,5}

{0,1},{2,3},{4,9},{6,11},{8,7},{10,5}
{0,1},{2,3},{4,5},{6,11},{8,7},{10,9}
{0,1},{2,7},{4,11},{6,5},{8,3},{10,9}
{0,1},{2,7},{4,9},{6,5},{8,3},{10,11}
{0,1},{2,7},{4,9},{6,11},{8,3},{10,5}
{0,1},{2,7},{4,5},{6,11},{8,3},{10,9}
{0,1},{2,7},{4,3},{6,5},{8,9},{10,11}
{0,1},{2,7},{4,3},{6,11},{8,9},{10,5}
{0,1},{2,7},{4,11},{6,5},{8,9},{10,3}
{0,1},{2,9},{4,3},{6,5},{8,7},{10,11}
{0,1},{2,9},{4,3},{6,11},{8,7},{10,5}
{0,1},{2,9},{4,11},{6,5},{8,7},{10,3}

{0,5},{2,9},{4,11},{6,7},{8,1},{10,3}
{0,1},{2,3},{4,9},{6,5},{8,7},{10,11}
{0,1},{2,3},{4,9},{6,11},{8,7},{10,5}

{0,1},{2,9},{4,5},{6,11},{8,7},{10,3}
{0,5},{2,1},{4,3},{6,7},{8,9},{10,11}
{0,5},{2,1},{4,9},{6,7},{8,3},{10,11}
{0,5},{2,1},{4,11},{6,7},{8,3},{10,9}
{0,5},{2,1},{4,11},{6,7},{8,9},{10,3}
{0,5},{2,3},{4,9},{6,1},{8,7},{10,11}
{0,5},{2,3},{4,11},{6,1},{8,7},{10,9}
{0,5},{2,3},{4,11},{6,7},{8,1},{10,9}
{0,5},{2,7},{4,3},{6,1},{8,9},{10,11}
{0,5},{2,7},{4,9},{6,1},{8,3},{10,11}
{0,5},{2,7},{4,11},{6,1},{8,3},{10,9}
{0,5},{2,7},{4,11},{6,1},{8,9},{10,3}

Перед тем , как мы перейдем ко второму этапу решения задачи , выберем одно из представленных ρ . Пусть это будет : {0,1},{2,3},{4,11},{8,7},{6,5},{10,9} . Нам будет достаточно продемонстрировать решение задачи только для одного ρ , так как решение не будет зависеть от конкретного ρ . Для нас важно только то , что для него выполнено это условие :

$P \cup A = \nabla$ $P \cap A = \Delta$

$P \cup C = \nabla$ $P \cap C = \Delta$

$P \cup D = \nabla$ $P \cap D = \Delta$

$P \cup E = \nabla$ $P \cap E = \Delta$

Этап 2 .

После того , как мы нашли все классы нашей решетки , необходимо найти все отображения , которые будут сохранять эти классы . Вообще мы ищем все отображения из Z12 -> Z12 , которые будут сохранять абсолютно все классы . Поясним , как будут строиться отображения , которые не будут нарушать целостность классов .

Для понятности будем использовать следующую конструкцию :

0 -> n

1 ->n

2 ->n

3 ->n

4 ->n

5 ->n

6 ->n

7 ->n

8 ->n

9 ->n

10 ->n

11 ->n

Числа от 0 до 11 это числа из исходной Алгебры A , n тоже число из исходной A , для которого классы сохраняются . Соответственно , n – это не одно и

тоже число . Самыми очевидными отображениями , которые сохраняют классы будут :

[0,0,0,0,0,0,0,0,0,0,0,0]

[0,1,2,3,4,5,6,7,8,9,10,11]

[1,1,1,1,1,1,1,1,1,1,1,1]

[2,2,2,2,2,2,2,2,2,2,2,2]

[3,3,3,3,3,3,3,3,3,3,3,3]

[4,4,4,4,4,4,4,4,4,4,4,4]

[5,5,5,5,5,5,5,5,5,5,5,5]

[6,6,6,6,6,6,6,6,6,6,6,6]

[7,7,7,7,7,7,7,7,7,7,7,7]

[8,8,8,8,8,8,8,8,8,8,8,8]

[9,9,9,9,9,9,9,9,9,9,9,9]

[10,10,10,10,10,10,10,10,10,10,10,10]

[11,11,11,11,11,11,11,11,11,11,11,11]

Выше перечислено 13 тривиальных отображений , которые сохраняют классы A,p,C,D,E, Δ,∇.

Подходя к решению этой задачи , я хотел перебрать всевозможные комбинации и проверить выполнение сохранения каждого класса на компьютере , но , к сожалению , это невозможно . Всего существует 12^{12} различных отображений , что равно 8916100448256 . Проверка столь большого числа отображений слишком затратна по времени , поэтому было получено аналитическое решение данной задачи .

Аналитическое решение задачи об отображениях :

Первым делом рассмотрим все классы , чтобы понимать , с чем имеем дело .

1 . Дан класс : C = [0,6] , [1,7] ,[2,8] , [3,9], [4,10] , [5,11]

Для C у нас есть 6 классов => 6^6 соответствий . В каждом классе по 2 элемента , соответственно получаем : $6^6 * 2^{12}$ = 191102976 образов . Теперь построим матрицу отображений для [0,6] , т.е такие наборы в которые [0,6] может перейти :

0,0 1,1 2,2 3,3 4,4 5,5

0,6 1,7 2,8 3,9 4,10 5,11

6,0 7,1 8,2 9,3 10,4 11,5

6,6 7,7 8,8 9,9 10,10 11,11

Для [1,7] ,[2,8] , [3,9], [4,10] , [5,11] – будут выполняться те же самые варианты .

2 . Дан класс A =[0,2,4,6,8,10],[1,3,5,7,9,11]

Для A у нас есть 2 класса => 2^2 = 4соответствия . В каждом классе по 6 элементов , соответственно получаем : $2^2 * 6^{12}$ = 8707129344 образов . Теперь построим матрицу отображений для [0,2,4,6,8,10] , т.е такие наборы в которые [0,2,4,6,8,10] может перейти :

0,0,0,0,0,0 1,1,1,1,1,1

0,0,0,0,0,2 1,1,1,1,1,3

…… …….

10,10,10,10,10,10 11,11,11,11,11,11

Для [1,3,5,7,9,11]– будут выполняться те же самые варианты .

3 . Дан класс E = [0,3,6,9],[1,4,7,10],[2,5,8,11]

Для E у нас есть 3 класса => 3^3 = 27соответствия . В каждом классе по 4 элементов , соответственно получаем : $3^3 * 4^{12}$ = 452984832 образов . Теперь построим матрицу отображений для [0,3,6,9] , т.е такие наборы в которые [0,3,6,9] может перейти :

0,0,0,0	1,1,1,1	2,2,2,2
0,0,0,3	1,1,1,4	2,2,2,5
……	……	……
9,9,9,9	10,10,10,10	11,11,11,11

Для [1,4,7,10],[2,5,8,11] - будут выполняться те же самые варианты .

4 . Дан класс p = [0,1],[2,3],[4,11],[8,7],[6,5],[10,9]

Для p у нас есть 6 классов => 6^6 соответствий . В каждом классе по 2 элемента , соответственно получаем : $6^6 * 2^{12}$ = 191102976 образов . Теперь построим матрицу отображений для [0,1] , т.е такие наборы в которые [0,1] может перейти :

0,0 2,2 4,4 7,7 5,5 9,9

0,1 2,3 4,11 7,8 5,6 9,10

1,0 3,2 11,4 8,7 6,5 10,9

1,1 3,3 11,11 7,7 6,6 10,10

Для],[2,3],[4,11],[8,7],[6,5],[10,9]- будут выполняться те же самые варианты .

5 . Дан класс D = [0,4,8],[1,5,9],[2,6,10],[3,7,11]

Для D у нас есть 4 класса => $4^4 = 256$ соответствий . В каждом классе по 3 элемента , соответственно получаем : $4^4 * 3^{12} = 136048896$ образов . Теперь построим матрицу отображений для [0,4,8], т.е такие наборы в которые [0,4,8]может перейти :

0,0,0 1,1,1 2,2,2 3,3,3

0,0,4 1,1,5 2,2,6 3,3,7

…… …… …… …….

8,8,8 9,9,9 10,10,10 11,11,11

Для [1,5,9],[2,6,10],[3,7,11]- будут выполняться те же самые варианты .

Теперь определим следующее , какие из отображений являются гомоморфизмами :

0 должен превратиться в 0

0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11

0 , a , 2a , 3a , 4a , 5a , 6a , 7a , 8a , 9a , 10a , 11a

Пункт 1.

a=0 подходит

a=1 подходит

a=2 подходит

a=4 подходит

Возможные варианты :

0 1 2 3 4 5 6 7 8 9 10 11

0 0 0 0 0 0 0 0 0 0 0 0

0 1 2 3 4 5 6 7 8 9 10 11

0 2 4 6 8 10 0 2 4 6 8 10

0 3 6 9 0 3 6 9 0 3 6 9

В результате получаем [x , x +6] -> [ax , ax + 6a] = [ax , ax] или же [ax , ax + 6]

Пункт 2 .

a=0 подходит

a=1 подходит

a=2 подходит

…..

Получаем : Четное число при умножении на любое число будет четным , поэтому всегда подойдет .

Пункт 3

Аналогичен пункту 2

Пункт 5

Аналогичен пункту 2

Пункт 4 .

a=0 подходит

a=1 подходит

a=2 не подходит

a=3 не подходит

остальные тоже не подходят

Сделаем замечание : Из классов 3 и 5 будут следовать классы 1 и 2 .

Теперь рассмотрим классы 3 и 4 , они однозначно определяют число , рассматривать будем в виде таблицы :

	[0,3,6,9]	[1,4,7,10]	[2,5,8,11]
[0,1]	0	1	-
[2,3]	3	-	2
[4,11]	-	4	11
[5,6]	6	-	5
[7,8]	-	7	8
[9,10]	9	10	-

В первой строке может быть 24 варианта . Далее смотрим по столбцам . Рассмотрим следующие варианты :

0 0 0 0 0 0

0 Так можно 0 0 0 и так далее

0 0 0 0

0 3 3 3

Для первой строки (0,0) получаем пересечение класса 3 и класса 4 , которое равно 4^3 =64 варианта .

Мы можем выбрать любую из 24 двоек из пункта 4 и четверку из пункта 3 , начинающуюся с того же элемента . Получаем : 24*2*6^11 отображений .

Теперь рассмотрим Пункт 4 и Пункт 5

	[0,1]	[2,3]	[4,11]	[5,6]	[7,8]	[9,10]
[0,4,8]	0	-	4	-	8	-
[1,5,9]	1	-	-	5	-	9
[2,6,10]	-	2	-	6	-	10
[3,7,11]	-	3	11	-	7	-

Теперь рассмотрим , что получится в результате пересечения Пунктов 3,4 и 5:

0 0 Вариант тривиальный

0 0

0 0

0 0

0 0

Рассмотрим пару случаев и выявим закономерность :

00 0 0 -> 0 0 -> 0 0 - > 0 0 -> 0 0 ->

00 0/3/6/9 0/6 0/6 0/6 0/6 0/6 0 0/6

00

00 0/3/6/9 0/6 0/6 0/6 0 0 0 0

00

00 0/3/6/9 0/6 0/6 0/6 0 0 0 0

По E по A по p и A по D по p

- > 0 0 -> 0 0 -> 0 0 -> 0 0

0 0 0 0 0 0 0 0

0/4/8 0/4/8 0/4/8 0 0

0 0 0 0 0 0 0 0

0/4/8 0/4/8 0/4/8 0 0

0 0 0 0 0 0 0 0

По р по D по р и А по Е

1 1 -> 1 1 -> 1 1 -> 1 1 -> 1 1 ->

1/7 1/7 1/7 1/7 1/7 1 1

1/7 1/7 1/7 1 1 1 1

1/7 1/7 1/7 1 1 1 1

По Е и А по р и А по р и D по р и D

1 1 -> 1 1 -> 1 1

1 1 1 1 1 1

1/5/9 1/5/9 1/5/9 1 1

1 1 1 1 1 1

1/5/9 1/5/9 1/5/9 1 1

1 1 1 1 1 1

по D по р и А по Е

2 2 -> 2 2 - > 2 2 -> 2 2 -> 2 2 ->

2/8 2/8 2/8 2/8 2/8 2 2

2/8 2/8 2/8 2 2 2 2

2/8 2/8 2/8 2 2 2 2

По Е и А по р и А по р и D по р и D

2 2 -> 2 2 -> 2 2

2 2 2 2 2 2

2/6/10 2/6/10 2/6/10 2 2

2 2 2 2 2 2

2/6/10 2/6/10 2/6/10 2 2

2 2 2 2 2 2

По D по р и A по E

Аналогичный результат мы получим и для остальных пар 3 3 , 4 4 , 5 5 , 6 6, 7 7 , 8 8 , 9 9 , 10 10 и 11 11. Для всех таких наборов получаем только тривиальные случаи . Продолжим рассматривать дальше :

2 3 -> 2 3 -> 2 3 -> 2 3 -> не подходит поE

5/11 6/4 5/11 6/4 5/11

2/8 2/8 3/7 2/8 3/7

5/11 6/4 5/11 4 11

По E и A по A и p по p и D

3 2 -> 3 2 -> 3 2 -> 3 2 -> 3 2 ->

0/6 1/5 0/6 1/5 0/6 1/5 0/6

3/7/11 2/8/4

3/9 3/9 2/10 9 10 9 10

3/7/11 2/8/4

0/6 1/5 0/6 5 6 5 6

По A и E по p и A по p и D по D,p и A

->3 2 невозможно представить отображение нарушается либо E , либо p

5 6

11

4 11 -> 4 11 -> 4 11 -> 4 11 -> 4 11

1/7 0/8 1/7 0/8 1/7 0/8 1/7

0/4/8 1/11/7

4/10 4/10 11/9 4 11 4 11

0/4/8 1/11/7

1/7 0/8 1/7 8 7 8 7

По A и E по p и A по p и D по D,p и A

-> невозможно представить отображение нарушается либо E , либо p

11 4 -> 11 4 -> 11 4 -> 11 4 -> 11 4

2/8 3/7 2/8 3/7 2/8 3/7 2/8

3/7/11 2/8/4

5/11 5/11 6/4 11 4 11 4

3/7/11 2/8/4

2/8 3/7 2/8 7 8 7 8

По A и E по p и A по p и D по D,p и A

-> невозможно представить отображение нарушается либо E , либо p

6 5 -> 6 5 - > 6 5 -> 6 5 -> 6 5

3/9 2/10 3/9 2/10 3/9 2/10 3/9

2/6/10 3/5/9

0/6 0/6 1/5 6 5 6 5

2/6/10 3/5/9

3/9 2/10 3/9 10 9 10 9

По A и E по p и A по p и D по D,p и A

->не сходится с E

5 6 -> 5 6 -> 5 6 -> 5 6 -> 5 6 ->

2/8 3/7 2/8 3/7 2/8 3/7 2/8

1/5/9 0/6/10

5/11 5/11 6/4 5 6 5 6

1/5/9 0/6/10

2/8 3/7 2/8 3 2 3 2

По A и E по p и A по p и D по D,p и A

-> 5 6

7 8

9 10

5 6

->не сходится с E

0 1 -> 0 1 -> 0 1 -> 0 1 -> 0 1 ->

0/3/6/9 3/9 2/10 3/9 2/10 3/9

0/3/6/9 0/6 0/6 1/5 0/6 1/5

0/3/6/9 3/9 2/10 3/9 2/10 9

По E по A по p и A по D

0 1 -> 0 1 -> 0 1 -> 0 1 -> 0 1 ->

2/10 3/19 2/10 3/9 2/10 3/9 2/10 3/9 2 3/9

0/4/8 0/4/8 1/7/11 4 11

0/6 1/5 6 5 6 5 6 5 6 5

0/4/8 0/4/8 1/7/11 8 1/7

10 9 10 9 10 9 10 9 10 9

По p по p и D по D по p и A по E

-> 0 1

2 3

4 11

6 5

7 8

9 10

1 0 -> 1 0 ->1 0 -> 1 0 -> 1 0 -> 1 0

1/4/7/10 4/10 11/9 4/10 11/9 4/10 11/9 4/10

1/4/7/10 1/7 1/7 0/8 1/7 0/8 7 8

1/4/7/10 4/10 11/9 4/10 11 4 11 4

->1 0 -> 1 0 -> не сходится с E

11 4 11 4

1/5/9 1/5/9 0/6/10

7 8 7 8

1/5/9 1/5/9 0/6/10

11 4 11 4

8 7 –> 8 7 -> 8 7 -> 8 7 -> 8 7 -> 8 7

5/11 6/4 5/11 6/4 5/11 4 11 4 11

0/4/8 1/11/7

2/8 2/8 3/7 2/8 3/7 8 7 8 7

0/4/8 1/11/7

5/11 6/4 5/11 4 11 4 11 4 11

По A и E по p и A по p и D по D,p и A по p

-> 8 7 - невозможно представить отображение нарушается либо E , либо p

4

7 8 -> 7 8 -> 7 8 -> 7 8 -> 7 8 -> 7 8 ->

4/10 11/9 4/10 11/9 4/10 11/9 4/10 11 4

1/7 1/7 0/8 1/7 0/8 7 8 7 8

4/10 11/9 4/10 11 4 11 4 11 4

По A и p по p и A по p и D по p и D по p, D ,A

-> 7 8 -> 7 8

11 4 11 4

3/7/11 2/8/4 11

7 8

3/7/11 2/8/4

11 4

по p, D ,A

невозможно представить отображение нарушается либо E , либо p

10 9 –> 10 9 -> 10 9 -> 10 9

1/7 0/8 1/7 0/8 1/7

4/10 4/10 11/9 10 9

1/7 0/8 1/7 0 1

По A и E по p и A по D и p

-> невозможно представить отображение нарушается D

9 10 - > 9 10 -> 9 10 -> 9 10 -> 9 10 -> 9 10

0/6 1/5 0/6 1/5 0/6 1/5 0/6 1/5 0/6

1/5/9 0/6/10 1

3/9 3/9 2/10 9 10 9 10

1/5/9 0/6/10

0/6 1/5 0/6 5 6 5 6

По A и E по A и p по D и p по D,p,A

невозможно представить отображение нарушается либо E , либо p

В результате проверки , было обнаружено , что только тривиальные отображения будут сохранять наши классы . Всевозможные отображения для всех пар p были написаны выше :

k1 = [0,0,0,0,0,0,0,0,0,0,0,0]

k2 = [0,1,2,3,4,5,6,7,8,9,10,11]

k3 = [1,1,1,1,1,1,1,1,1,1,1,1]

k4 = [2,2,2,2,2,2,2,2,2,2,2,2]

k5 = [3,3,3,3,3,3,3,3,3,3,3,3]

k6 = [4,4,4,4,4,4,4,4,4,4,4,4]

k7 = [5,5,5,5,5,5,5,5,5,5,5,5]

k8 = [6,6,6,6,6,6,6,6,6,6,6,6]

k9 = [7,7,7,7,7,7,7,7,7,7,7,7]

k10 = [8,8,8,8,8,8,8,8,8,8,8,8]

k11 = [9,9,9,9,9,9,9,9,9,9,9,9]

k12 = [10,10,10,10,10,10,10,10,10,10,10,10]

k13 = [11,11,11,11,11,11,11,11,11,11,11,11]

Где kn – индекс отображения .

Проверим , что наши отображения на самом деле сохраняют наши классы , для этого воспользуемся следующим кодом , стоит сказать , что отображения являются тривиальными и не требуют проверки , но данный код является универсальным и может проверить любое отображение для любых классов , именно поэтому я оставил его в своей работе .

После проверки данного кода получаем , что все отображения верны , как следствие мы можем продолжать решать нашу задачу .

В результате мы получаем 13 тривиальных отображений , которые будут сохранять заданные классы .

Для того , чтобы убедиться в том , что мы нашли все отображения , построим таблицу отображений . Соответственно , наша таблица будет размером 13x13 . Напишем , как будем строить таблицу : Сначала берем ki потом kj . Пример :

1 2 3 1 2 3 1 2 3

* =

3 1 1 2 1 2 2 2 2

	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
k1	k1	k1	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
k2	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
k3	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
k4	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
k5	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
k6	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
k7	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
k8	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
k9	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
k10	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
k11	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
k12	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
k13	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13

Для получения данной таблицы был написан следующий код , который упросил работу в разы , код в приложении А.

Наша проверка прошла успешно , мы получили все имеющиеся отображения .

Этап 3.

Теперь решим следующую задачу : У нас есть алгебра A = {0,1,2,3,4,5,6,7,8,9,10,11} , состоящая из 12 элементов с 13 унарными операциями (A , k0 , …., k12) . Нужно построить решетку ConA , найти все такие разбиения двенадцати элементного множества A на классы , что все ki будут сохранять эти классы .

Стоит учитывать , что возможно мы получим класс K , которого нет в исходных классах : A , p , C,D,E . Данный результат будет говорить нам о том , что алгебра для данной решетки найдена неверно и нужно применять более изощренные методы . Если же все классы совпадут и не будет ни одного лишнего , то задача может считаться решенной .

На втором этапе решения задачи мы получили то, что у нас существуют только тривиальные отображения , которые сохраняют наши классы . Перечислим их :

[0,0,0,0,0,0,0,0,0,0,0,0]

[0,1,2,3,4,5,6,7,8,9,10,11]

[1,1,1,1,1,1,1,1,1,1,1,1]

[2,2,2,2,2,2,2,2,2,2,2,2]

[3,3,3,3,3,3,3,3,3,3,3,3]

[4,4,4,4,4,4,4,4,4,4,4,4]

[5,5,5,5,5,5,5,5,5,5,5,5]

[6,6,6,6,6,6,6,6,6,6,6,6]

[7,7,7,7,7,7,7,7,7,7,7,7]

[8,8,8,8,8,8,8,8,8,8,8,8]

[9,9,9,9,9,9,9,9,9,9,9,9]

[10,10,10,10,10,10,10,10,10,10,10,10]

[11,11,11,11,11,11,11,11,11,11,11,11]

Первое , что нам нужно сделать , это разделить наше множество A ={0,1,2,3,4,5,6,7,8,9,10,11} пополам для того , чтобы проверить какое разбиение будет сохранять классы . Это можно сделать 12! способами , что равно 479 001 600. Это число можно уменьшить , если учесть , что мы не будем рассматривать перестановку внутри класса . Это значит , что следующие отображения для нас будут идентичны {[0,1,2,3,4,5], [6,7,8,9,10,11]} ~ {[5,0,1,2,3,4],[10,11,6,7,8,9]} , связано это с тем , что элементы обязаны либо перейти , либо остаться в своем классе , а если этого не произойдет , то для нас не имеет значения порядок в котором написаны элементы . Так же будем считать эквивалентными зеркальные отображения , то есть { [0,1,2,3,4,5],[6,7,8,9,10,11]} ~{[6,7,8,9,10,11], [0,1,2,3,4,5]}, в силу того , что мы в любой момент можем поменять их местами $

Таким образом мы получаем следующие варианты , для разбиения пополам :

[0,1,2,3,4,5],[6,7,8,9,10,11]

[0,2,4,6,8,10],[1,3,5,7,9,11]

[1,0,2,4,6,8],[10,3,5,7,9,11]

[3,0,2,4,6,8],[10,3,5,7,9,11]

….

Пока все четные и нечетные числа не поменяются местами .

Напишем код для проверки данного условия , код находится в приложении А .

В результате получаем 924 набора , которые удовлетворяют нашему условию. Теперь нам необходимо проверить , какие из полученных разбиений будут сохранять сразу все 14 отображений .

Во второй главе мы делили отображения на следующие виды :

1. Тривиальные.

   К ним относятся такие отображения , как [0,0,0,0,0,0,0,0,0,0,0,0] и так далее.

2. Наборы , которые содержат все числа такие , как [0,1,2,3,4,5,6,7,8,9,10,11].

Далее будем строить аналитическое решение . Как было описано выше , для нахождения разбиения пополам нам необходимо рассмотреть только наборы из третьего типа , напишем их еще раз :

k1= [0,1,2,3,4,5,6,7,8,9,10,11]

Теперь скажем пару слов о каждом из этих отображений

k1- исходный порядок элементов

Мы можем использовать следующий подход:

Возьмем половину элементов из каждой перестановки ki в качестве первого класса k1.

Оставшуюся половину элементов из каждой перестановки ki добавим во второй класс k2.

Поскольку каждая перестановка содержит все элементы множества A, и каждый элемент встречается в каждой перестановке, этот подход гарантирует, что ни один элемент не будет пропущен.

Давайте применим этот подход к каждой из перестановок ki от k1 до k8.

k1=[0,1,2,3,4,5,6,7,8,9,10,11]:

k1=[0,1,2,3,4,5]

k2=[6,7,8,9,10,11]

Таким образом, универсальное разбиение множества A на два класса будет:

k1=[0,1,2,3,4,5],k2=[6,7,8,9,10,11]

Это разбиение удовлетворяет всем перестановкам ki и обеспечивает, что в каждом классе содержится по 6 элементов.

Данное утверждение можно проверить следующим образом :

k1 и k2 это исходное множество A , в случае k2 оно является перевернутым , все же остальные так же являются множеством A , но только со сдвигом , как следствие данное разбиение будет удовлетворять нашему условию .

Первый полученный результат , а именно разбиение k1=[0,1,2,3,4,5], k2=[6,7,8,9,10,11] сильно отличается от класса эквивалентности A=[0,2,4,6,8,10],[1,3,5,7,9,11] . Что говорит нам о том , что применяя методы мы можем установить алгебру для заданной решетки .

В результате выполнения кода получаем , что данный набор сохраняет отображение . Теперь нам необходимо рассмотреть еще 3 разбиения , а именно разбиение A на три. Соответственно , для 3х у нас будет 34650 разбиений . Теперь найдем такие разбиения , которые будут удовлетворять разбиению A на 3 , 4 и 6 частей .

Приведенный выше метод , является слишком затратным по времени , ведь количество вариантов с каждым разом растет и найти класс для C будет достаточно проблематично , поэтому нужно пересмотреть подход к решению и проверке на правильность задачи .

Для решения этой задачи был предложен метод обратного поиска , на втором этапе решения задачи был написан код , который проверяет отображения , сохраняющие классы , если его модифицировать , то можно решить поставленную задачу , код представлен в приложении А.

Стоит заметить , что все наши отображения являются тривиальными , как следствие они будут сохранять любое разбиение . Чтобы найти количество всех разбиений двенадцатиэлементного множества A , воспользуемся числом Белла . Рекуррентная формула для числа Белла имеет следующий вид :

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} \mathrm{B_k}$$

Для нахождения числа разбиений для множества A , было написано программное решение с использованием формулы Белла . В результате мы получаем 4213597 .

Все эти разбиения нам подойдут , соответственно наша решетка конгруэнций будет содержать все 4213597 разбиения . Построить данную решетку не представляется возможным в ввиду её объема .

2.4 Выводы по полученным результатам :

В данной главе мы рассмотрели методы ,которые позволяют найти все разбиения двенадцатиэлементного множества A на классы , при том , что все ki сохраняют эти классы . Было показано , что все разбиения множества A на классы , будут сохранять эти классы . Так же мы получили решетку , которая отличается от решетки L_7 , что говорит нам о невозможности нахождения этой решетки данным методом .

Заключение

В ходе работы получены следующие результаты :

Используя данный метод невозможно найти представление для решетки L_7 .

Получен алгоритм позволяющий проверить отображения , которые будут сохранять классы .

Написано программное решение для разбиения заданного множества A с отображениями (A,k0,…,k12) и проверка на сохранение классов эквивалентности .

Задел на будущее :

Применяя подобные методы , можно попытаться найти представления и для других решеток , которые имеют более семи элементов .

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Винберг Э.Б. Курс алгебры 2021 – 591 с.
2. Д.Хобби , Р.Маккензи Строение конечных алгебр – 286 с.
3. William J. DeMeo CONGRUENCE LATTICES OF FINITE ALGEBRAS –124 с.
4. Г.Гретцер Общая теория решеток - 454 с.
5. Лата Александр Николаевич – Производные структуры унарных алгебр - 78 с.
6. А.Клиффорд , Г.Престон Алгебраическая теория полугрупп Том 1 – 283с.
7. Berman, J. On the congruence lattices of unary algebras / J. Berman // Proc. Amer. Math. Soc. — 1972 — Vol. 36, No. 1 — P. 34—38.

ПРИЛОЖЕНИЕ А

[1] – Sage .Система для экспериментов с алгеброй и геометрией"[3]) представляет собой систему компьютерной алгебры (CAS) с функциями, охватывающими многие аспекты математики, включая алгебру, комбинаторику, теорию графов, теорию групп, дифференцируемые многообразия, численный анализ, теорию чисел, математический анализ и статистику.

[2] - GAP (Группы, алгоритмы и программирование) - это система компьютерной алгебры с открытым исходным кодом для вычислительной дискретной алгебры с особым упором на теорию вычислительных групп.

С помощью этого кода можно проверить будут ли отображения сохранять классы :

```
A = [[0, 2, 4, 6, 8, 10], [1, 3, 5, 7, 9, 11]]
B = [[0, 1], [2,3] ,[4,11],[5,6],[7,8],[9,10]]
C = [[0, 6], [1, 7], [2, 8], [3, 9], [4, 10], [5, 11]]
D = [[0, 4, 8], [1, 5, 9], [2, 6, 10], [3, 7, 11]]
E = [[0, 3, 6, 9], [1, 4, 7, 10], [2, 5, 8, 11]]

def check_mapping_for_E(E1, E2, E3, mapping):
    mapping_set = set(mapping)
    E1_set = set(E1)
    E2_set = set(E2)
    E3_set = set(E3)

    if len(mapping_set) == 1:
        return True

    if all(x in E2_set or x in E3_set or x not in mapping_set for x in E1):
        return True

    if all(x in E1_set or x in E3_set or x not in mapping_set for x in E2):
```

```
            return True

        if all(x in E1_set or x in E2_set or x not in mapping_set for x in E3):
            return True

        if all((x == y) or (x in E1_set and y in E1_set) or (x in E2_set and y in E2_set)
               or (x in E3_set and y in E3_set)
               for x, y in enumerate(mapping)):
            return True

        return False


    def check_mapping_for_D(D1, D2, D3, D4, mapping):
        mapping_set = set(mapping)
        D1_set = set(D1)
        D2_set = set(D2)
        D3_set = set(D3)
        D4_set = set(D4)

        if len(mapping_set) == 1:
            return True

        if all(x in D2_set or x in D3_set or x in D4_set or x not in mapping_set for x in
D1):
            return True

        if all(x in D1_set or x in D3_set or x in D4_set or x not in mapping_set for x in
D2):
            return True
```

```
        if all(x in D1_set or x in D2_set or x in D4_set or x not in mapping_set for x in
D3):
            return True

        if all(x in D1_set or x in D2_set or x in D3_set or x not in mapping_set for x in
D4):
            return True

        if all((x == y) or (x in D1_set and y in D1_set) or (x in D2_set and y in D2_set)
               or (x in D3_set and y in D3_set) or (x in D4_set and y in D4_set)
               for x, y in enumerate(mapping)):
            return True

        return False

    def check_mapping_for_C(C1, C2, C3, C4, C5, C6, mapping):
        mapping_set = set(mapping)
        C1_set = set(C1)
        C2_set = set(C2)
        C3_set = set(C3)
        C4_set = set(C4)
        C5_set = set(C5)
        C6_set = set(C6)

        if len(mapping_set) == 1:
            return True

        if all(x in C2_set or x in C3_set or x in C4_set or x in C5_set or x in C6_set or x
not in mapping_set for x in C1):
            return True
```

```
            if all(x in C1_set or x in C3_set or x in C4_set or x in C5_set or x in C6_set or x
not in mapping_set for x in C2):
                return True

            if all(x in C2_set or x in C1_set or x in C4_set or x in C5_set or x in C6_set or x
not in mapping_set for x in C3):
                return True

            if all(x in C2_set or x in C3_set or x in C1_set or x in C5_set or x in C6_set or x
not in mapping_set for x in C4):
                return True

            if all(x in C2_set or x in C3_set or x in C4_set or x in C1_set or x in C6_set or x
not in mapping_set for x in C5):
                return True

            if all(x in C2_set or x in C3_set or x in C4_set or x in C5_set or x in C1_set or x
not in mapping_set for x in C6):
                return True

            if all((x == y) or (x in C1_set and y in C1_set) or (x in C2_set and y in C2_set)
                   or (x in C3_set and y in C3_set) or (x in C4_set and y in C4_set)
                   or (x in C5_set and y in C5_set) or (x in C6_set and y in C6_set)
                   for x, y in enumerate(mapping)):
                return True


        def check_mapping_for_A(A1, A2, mapping):
            mapping_set = set(mapping)
            A1_set = set(A1)
            A2_set = set(A2)
```

```
        if len(mapping_set) == 1:
            return True

        if all(x in A2_set or x not in mapping_set for x in A1):
            return True

        if all(x in A1_set or x not in mapping_set for x in A2):
            return True

        if all((x == y) or (x in A1_set and y in A1_set) or (x in A2_set and y in A2_set)
               for x, y in enumerate(mapping)):
            return True

        return False


    def check_mapping_for_B(B1, B2, B3, B4, B5, B6, mapping):
        mapping_set = set(mapping)
        B1_set = set(B1)
        B2_set = set(B2)
        B3_set = set(B3)
        B4_set = set(B4)
        B5_set = set(B5)
        B6_set = set(B6)

        if len(mapping_set) == 1:
            return True

        if all(x in B2_set or x in B3_set or x in B4_set or x in B5_set or x in B6_set or x
not in mapping_set for x in B1):
            return True
```

```
        if all(x in B1_set or x in B3_set or x in B4_set or x in B5_set or x in B6_set or x not in mapping_set for x in B2):
            return True

        if all(x in B1_set or x in B2_set or x in B4_set or x in B5_set or x in B6_set or x not in mapping_set for x in B3):
            return True

        if all(x in B1_set or x in B2_set or x in B3_set or x in B5_set or x in B6_set or x not in mapping_set for x in B4):
            return True

        if all(x in B2_set or x in B3_set or x in B4_set or x in B1_set or x in B6_set or x not in mapping_set for x in B5):
            return True

        if all(x in B2_set or x in B3_set or x in B4_set or x in B5_set or x in B1_set or x not in mapping_set for x in B6):
            return True

        if all((x == y) or (x in B1_set and y in B1_set) or (x in B2_set and y in B2_set)
               or (x in B3_set and y in B3_set) or (x in B4_set and y in B4_set)
               or (x in B5_set and y in B5_set) or (x in B6_set and y in B6_set)
               for x, y in enumerate(mapping)):
            return True

        return check(B1, mapping) and check(B2, mapping) and check(B3, mapping) and check(B4, mapping) and check(B5, mapping) and check(B6, mapping)
        return False

    mappings = [
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
```

```
        [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],

        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2],
        [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3],
        [4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4],
        [5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5],
        [6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6],
        [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],
        [7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7],
        [9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9],
        [10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10],
        [11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11],

            ]

    def check_all_mappings():
        mappings_result = {}

        for mapping in mappings:
            mapping_result = {}

            mapping_result['A'] = all(check_mapping_for_A(A[0], A[1], mapping) for A
in zip(A, B))
            mapping_result['B'] = check_mapping_for_B(*B, mapping)
            mapping_result['C'] = check_mapping_for_C(*C, mapping)
            mapping_result['D'] = check_mapping_for_D(*D, mapping)
            mapping_result['E'] = check_mapping_for_E(*E, mapping)

            mappings_result[str(mapping)] = mapping_result

        return mappings_result
```

```
        results = check_all_mappings()

        for mapping, mapping_result in results.items():
            print("Mapping:", mapping)
            for matrix, is_correct in mapping_result.items():
                print(f"Matrix {matrix}: {'Correct' if is_correct else 'Incorrect'}")
```

Дописав код выше , сможем найти разбиения , которые будут сохранять классы :

```
def check_all_mappings():
    mappings_result = {}

    for mapping in mappings:
        mapping_result = {}

        mapping_result['A'] = all(check_mapping_for_A(A[0], A[1], mapping) for A in
zip(A, B))
        mapping_result['B'] = check_mapping_for_B(*B, mapping)
        mapping_result['C'] = check_mapping_for_C(*C, mapping)
        mapping_result['D'] = check_mapping_for_D(*D, mapping)
        mapping_result['E'] = check_mapping_for_E(*E, mapping)

        mappings_result[str(mapping)] = mapping_result

    return mappings_result


results = check_all_mappings()
print(results)
```

```
def find_valid_mappings():
    valid_mappings = []
    for mapping in mappings:
        if (check_mapping_for_A(A[0], A[1], mapping) and
            check_mapping_for_B(B[0], B[1], B[2], B[3], B[4] , B[5] , mapping) and
            check_mapping_for_C(C[0], C[1], C[2], C[3], C[4], C[5], mapping) and
            check_mapping_for_D(D[0], D[1], D[2], D[3], mapping) and
            check_mapping_for_E(E[0], E[1], E[2], mapping)):
            valid_mappings.append(mapping)
    return valid_mappings

def get_ABCDE_from_mapping(mapping):
    A_indices = [i for i, x in enumerate(mapping) if x == 0]
    B_indices = [i for i, x in enumerate(mapping) if x == 1]
    C_indices = [i for i, x in enumerate(mapping) if x == 2]
    D_indices = [i for i, x in enumerate(mapping) if x == 3]
    E_indices = [i for i, x in enumerate(mapping) if x == 4]

    A_values = [A[0][i] if i < len(A[0]) else A[1][i - len(A[0])] for i in A_indices]
    B_values = [B[0][i] if i < len(B[0]) else B[1][i - len(B[0])] if i < len(B[0]) + len(B[1])
                else B[2][i - len(B[0]) - len(B[1])] if i < len(B[0]) + len(B[1]) + len(B[2])
                else B[3][i - len(B[0]) - len(B[1]) - len(B[2])] if i < len(B[0]) + len(B[1]) +
len(B[2]) + len(B[3])
                else B[4][i - len(B[0]) - len(B[1]) - len(B[2]) - len(B[3])] if i < len(B[0]) +
len(B[1]) + len(B[2]) + len(B[3]) + len(B[4])
                else B[5][i - len(B[0]) - len(B[1]) - len(B[2]) - len(B[3]) - len(B[4])] for i in
B_indices]
    C_values = [C[0][i] if i < len(C[0]) else C[1][i - len(C[0])] if i < len(C[0]) + len(C[1])
                else C[2][i - len(C[0]) - len(C[1])] if i < len(C[0]) + len(C[1]) + len(C[2])
                else C[3][i - len(C[0]) - len(C[1]) - len(C[2])] if i < len(C[0]) + len(C[1]) +
len(C[2]) + len(C[3])
```

```
            else C[4][i - len(C[0]) - len(C[1]) - len(C[2]) - len(C[3])] if i < len(C[0]) +
len(C[1]) + len(C[2]) + len(C[3]) + len(C[4])
            else C[5][i - len(C[0]) - len(C[1]) - len(C[2]) - len(C[3]) - len(C[4])] for i in
C_indices]
    D_values = [D[0][i] if i < len(D[0]) else D[1][i - len(D[0])] if i < len(D[0]) + len(D[1])
            else D[2][i - len(D[0]) - len(D[1])] if i < len(D[0]) + len(D[1]) + len(D[2])
            else D[3][i - len(D[0]) - len(D[1]) - len(D[2])] for i in D_indices]
    E_values = [E[0][i] if i < len(E[0]) else E[1][i - len(E[0])] if i < len(E[0]) + len(E[1])
            else E[2][i - len(E[0]) - len(E[1])] for i in E_indices]

    return A_values, B_values, C_values, D_values, E_values

valid_mappings = find_valid_mappings()
ABCDE_combinations = [get_ABCDE_from_mapping(mapping) for mapping in
valid_mappings]



print(valid_mappings)
print(ABCDE_combinations)



with open("output11.txt", "w") as file:
    for mapping in valid_mappings:
        A_values, B_values, C_values, D_values, E_values =
get_ABCDE_from_mapping(mapping)
        if len(A_values) > 1:
            file.write(f"A = {[A_values[:6], A_values[6:]]}\n")
        if len(B_values) > 1:
            file.write(f"B = {[B_values[i:i+2] for i in range(0, len(B_values), 2)]}\n")
        if len(C_values) > 1:
            file.write(f"C = {[C_values[i:i+2] for i in range(0, len(C_values), 2)]}\n")
        if len(D_values) > 1:
```

```
            file.write(f"D = {[D_values[i:i+3] for i in range(0, len(D_values), 3)]}\n")
        if len(E_values) > 1:
            file.write(f"E = {[E_values[i:i+4] for i in range(0, len(E_values), 4)]}\n")

output_strings = []

with open("output11.txt", "r") as file:
    for line in file:
        output_strings.append(line.strip())

sorted_output_strings = sorted(output_strings, key=len, reverse=True)
for i in range(5):
    print(sorted_output_strings[i])
```

Данный код находит количество вариантов , для каждого класса , в задаче с разбиением множества A :

```
def divide_array(array):
    n = len(array)
    half = n // 2
    combinations_set = set()

    for indices in combinations(range(n), half):
        first_half = [array[i] for i in indices]
        second_half = [elem for j, elem in enumerate(array) if j not in indices]
        first_half.sort()
        second_half.sort()
        combination = (tuple(first_half), tuple(second_half))
        combinations_set.add(combination)

    return combinations_set

A = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

```
combinations = divide_array(A)

print("Количество комбинаций:", len(combinations))
print("Список комбинаций:")
for combination in combinations:
   print(combination)
```

Код , с помощью которого мы строим таблицу отображений :

```
def multiply_permutations(perm1,perm2):
        if len(perm1) != len(perm2):
            raise ValueError("Длины перестановок должны быть одинаковыми")

            perm1_dict = dict(enumerate(perm1))
            perm2_dict = dict(enumerate(perm2))

            result_perm_dict = {}
            for i in range(len(perm1)):
               result_perm_dict[i] = perm2_dict[perm1[i]]

            result_perm = [result_perm_dict[i] for i in range(len(perm1))]

            return result_perm

k1 = [0,0,0,0,0,0,0,0,0,0,0,0]
k2 = [0,1,2,3,4,5,6,7,8,9,10,11]
k3 = [1,1,1,1,1,1,1,1,1,1,1,1]
k4 = [2,2,2,2,2,2,2,2,2,2,2,2]
k5 = [3,3,3,3,3,3,3,3,3,3,3,3]
k6 = [4,4,4,4,4,4,4,4,4,4,4,4]
k7 = [5,5,5,5,5,5,5,5,5,5,5,5]
k8 = [6,6,6,6,6,6,6,6,6,6,6,6]
```

```
        k9 = [7,7,7,7,7,7,7,7,7,7,7,7]
        k10 = [8,8,8,8,8,8,8,8,8,8,8,8]
        k11 = [9,9,9,9,9,9,9,9,9,9,9,9]
        k12 = [10,10,10,10,10,10,10,10,10,10,10,10]
        k13 = [11,11,11,11,11,11,11,11,11,11,11,11]

        permutations = [k0 ,k1, k2, k3, k4, k5, k6, k7, k8, k9, k10,
                k11, k12]

        with open("matrix2.txt", "w") as file:
            for i, perm1 in enumerate(permutations):
                for j, perm2 in enumerate(permutations):
                    result = multiply_permutations(perm1, perm2)
                    file.write(f"k{i+1} * k{j+1} = {result}\n")
```

Код для нахождения числа Белла :

```
bell = [[0 for _ in range(n+1)] for _ in range(n+1)]
    bell[0][0] = 1
    for i in range(1, n+1):
        bell[i][0] = bell[i-1][i-1]
        for j in range(1, i+1):
            bell[i][j] = bell[i-1][j-1] + bell[i][j-1]
    return bell[n][0]
n = 12
print(f"Число Белла для {n} равно {bell_number(n)}")
```