Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"

профиль "Программное обеспечение средств вычислительной техники и автоматизированных систем"

Кафедра прикладной математики и кибернетики

**Курсовая работа по дисциплине  
 Алгоритмы и вычислительные методы оптимизации**

**Симплекс-метод**

Вариант 12

Выполнил:

студент гр.ИП-912 \_\_\_\_\_\_\_\_\_\_\_\_\_/Кадырбаев А.И./

ФИО студента

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Проверил

\_\_\_\_\_\_\_\_\_\_\_\_\_/Галкина М.Ю./

ФИО преподавателя

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Новосибирск 2022 г.

Содержание

Задание на курсовую работу 3

Теоретические материалы 4

Переход к канонической форме 7

Написание программы 8

Работа программы 9

Решение исходной задачи графически 10

Решение двойственной задачи на основании теоремы равновесия 11

Список используемых источников 12

Приложение 13

**Задание на курсовую работу**

1. Перейти к канонической форме записи задачи линейного программирования.

2. Написать программу, решающую задачу линейного программирования в канонической форме (с выводом всех промежуточных таблиц) двойственным симплекс-методом.

Программа правильно работать для различных случаев решений задачи: единственное решение, бесконечно много решений, функция не ограничена.

3. Решить исходную задачу графически и отметить на чертеже точки, соответствующие симплексным таблицам, полученным при выполнении программы из п.2.

4. Составить двойственную задачу к исходной и найти ее решение на основании теоремы равновесия.

Таблица вариантов

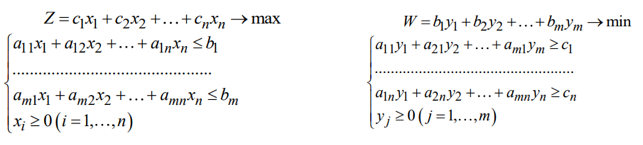
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер Варианта | a | b | c |  |  |  |  |  |  |  |  |
| 12. | 13 | 6 | 9 | 3 | 2 | 4 | 2 | 5 | 1 | 3 | 5 |

**Теоретические материалы**

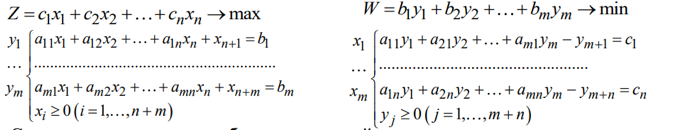
Задача линейного программирования задана в канонической форме, если требуется максимизировать целевую функцию, все ограничения системы – уравнения (m1 =m2 =0), и на все переменные наложено условие неотрицательности (k = n)

Симметричная → каноническая Переход осуществляется путем добавления в левую часть каждого неравенства дополнительной неотрицательной переменной. Если неравенство было «≤», то дополнительная переменная добавляется в левую часть неравенства со знаком «+»

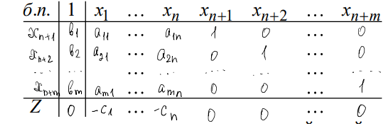
Пусть имеется пара двойственных задач в симметричной форме:



(Рис. 1) Пара двойственных задач в симметричной форме



(Рис. 2) Канонические формы



(Рис. 3) Симплексная таблица для первой задачи

Можно заметить, что построение самой двойственной задачи не обязательно, в столбцах таблицы записана исходная задача, а в строках – двойственная. При этом оценками решения исходной задачи являются коэффициенты Z-строки cij, а оценками решения двойственной задачи – коэффициенты столбца свободных членов bi . Симплексная таблица, в которой все коэффициенты Z-строки неотрицательны, а в столбце свободных членов имеются отрицательные, называется двойственно допустимой, а соответствующее решение – псевдопланом.

Двойственный симплекс-метод позволяет решать задачи линейного программирования, системы ограничений которых при положительном базисе содержат свободные члены любого знака. Этот метод позволяет уменьшить количество преобразований системы ограничений, а также размера симплексной таблицы.

Как и обычный симплексный метод, рассматриваемый метод решения основан на использовании условий допустимости и оптимальности.

**Условие допустимости**. В качестве исключаемой переменной выбирается наибольшая по абсолютной величине отрицательная базисная переменная (при наличии альтернатив выбор делается произвольно). Если все базисные переменные неотрицательные, процесс вычислений заканчивается, так как полученное решение допустимое и оптимальное.

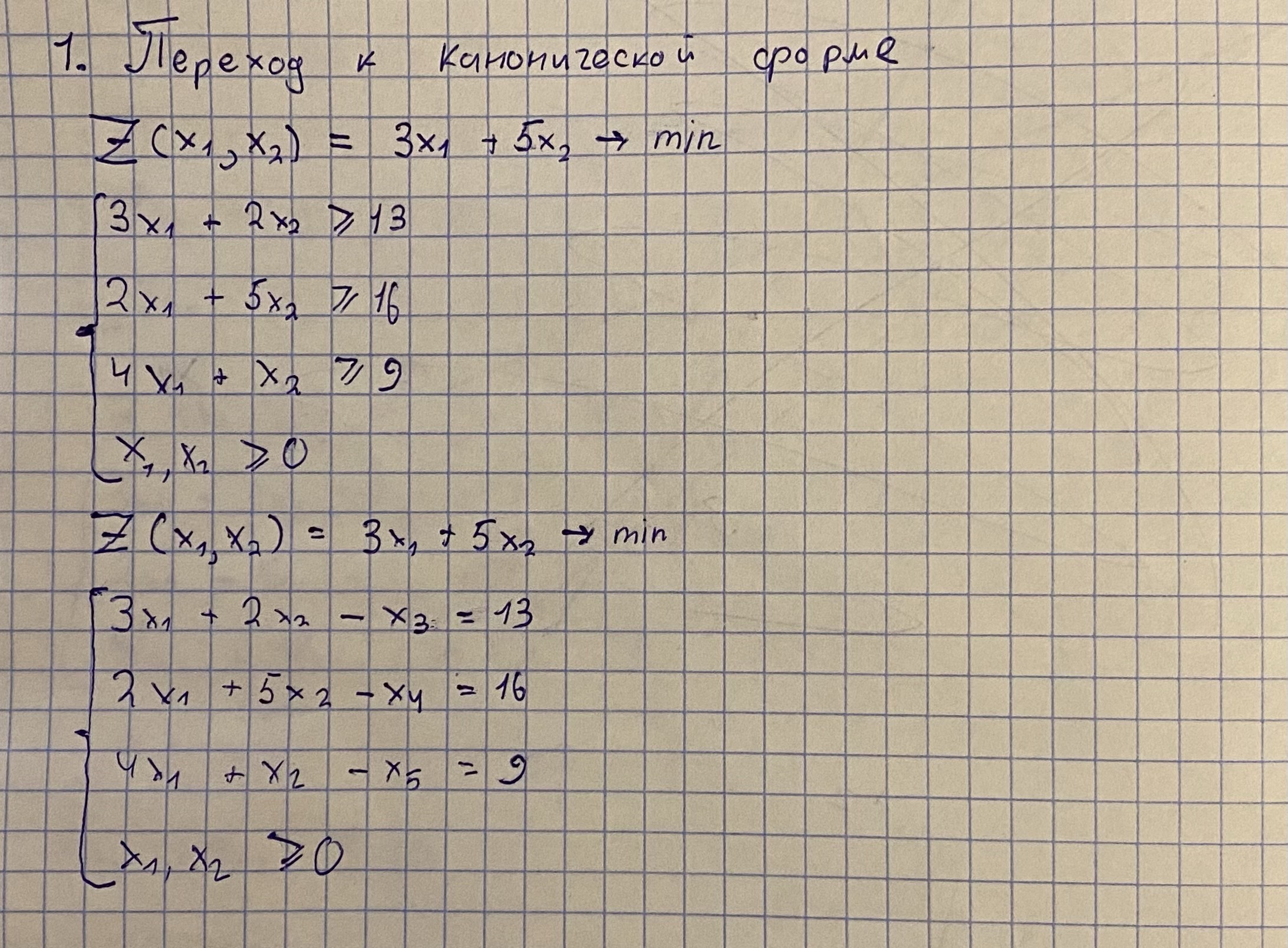
**Условие** **оптимальности**. Включаемая в базис переменная выбирается из числа небазисных переменных следующим образом. Вычисляются отношения коэффициентов левой части Рисунок 1-уравнения к соответствующим коэффициентам уравнения, ассоциированного с исключаемой переменной. Отношения с положительным или нулевым значением знаменателя не учитываются. В задаче минимизации вводимой переменной должно соответствовать наименьшее из указанных отношений, а в задаче максимизации – отношение, наименьшее по абсолютной величине (при наличии альтернатив выбор делается произвольно). Если знаменатели всех отношений равны нулю или положительные, задача не имеет допустимых решений.

Алгоритм двойственного симплекс-метода (применяется в случае двойственно допустимой симплексной таблицы):

1. В столбце свободных членов выбирают среди отрицательных минимальный. Это определяет разрешающую строку.
2. Для отрицательных элементов разрешающей строки находим симплексные отношения: отношения элементов Z-строки к отрицательным элементам разрешающей строки, взятые по модулю.
3. Выбираем минимальное симплексное отношение, соответствующий столбец – разрешающий.
4. Выполняют шаг симплексных преобразований таблицы.
5. Если в столбце свободных членов нет отрицательных, то решение оптимально, иначе переход на п.1.

Обычно с задачей линейного программирования (ЗЛП) связана другая линейная задача, называемая двойственной. Обе эти задачи можно считать двойственными одну по отношению к другой, считать равносильными. Первая задача называется обычно исходной, или прямой, другая - обратной. Переменные, используемые в двойственной задаче называются двойственными или множителями Лагранжа. На них не накладывается ограничений по знаку. Рассматриваются двойственные критерии оптимальности. Специальные случаи называют симметричными двойственными задачами линейного программирования. Связь между оптимальными решениями двойственных задач устанавливается теоремой двойственности.

**Переход к канонической форме**

Переход осуществляется путем добавления в левую часть каждого неравенства дополнительной неотрицательной переменной. Если неравенство было «≤», то дополнительная переменная добавляется в левую часть неравенства со знаком «+». Если неравенство было «», то дополнительная переменная добавляется в левую часть неравенства со знаком «–».

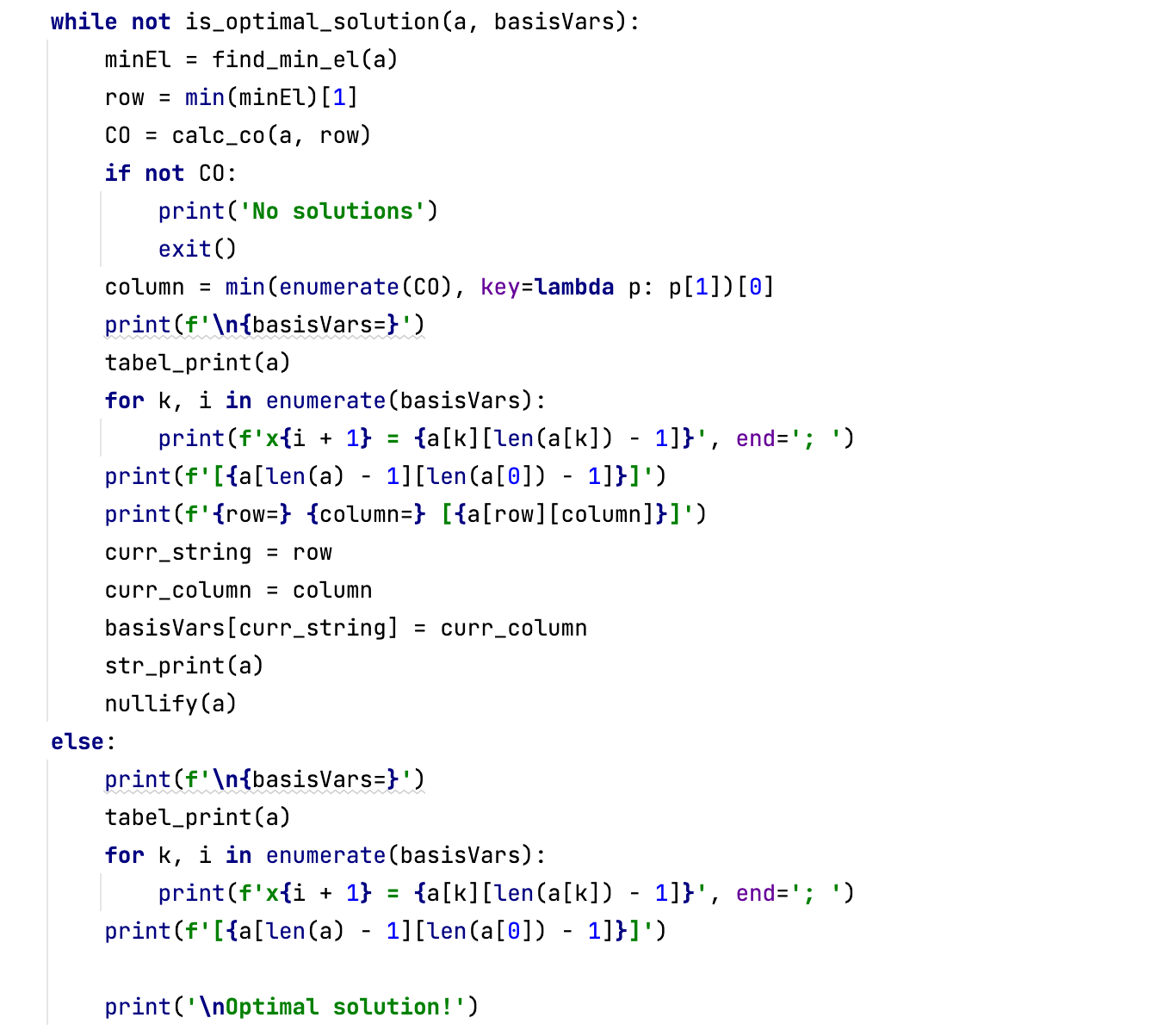
(Рис. 4) Решение исходной задачи графически

**Написание программы**

За основу программы взята лабораторная работа 2, в которой уже реализован класс простых дробей (Fraction), алгоритм нахождения сочетаний, функция нахождения опорных решений.

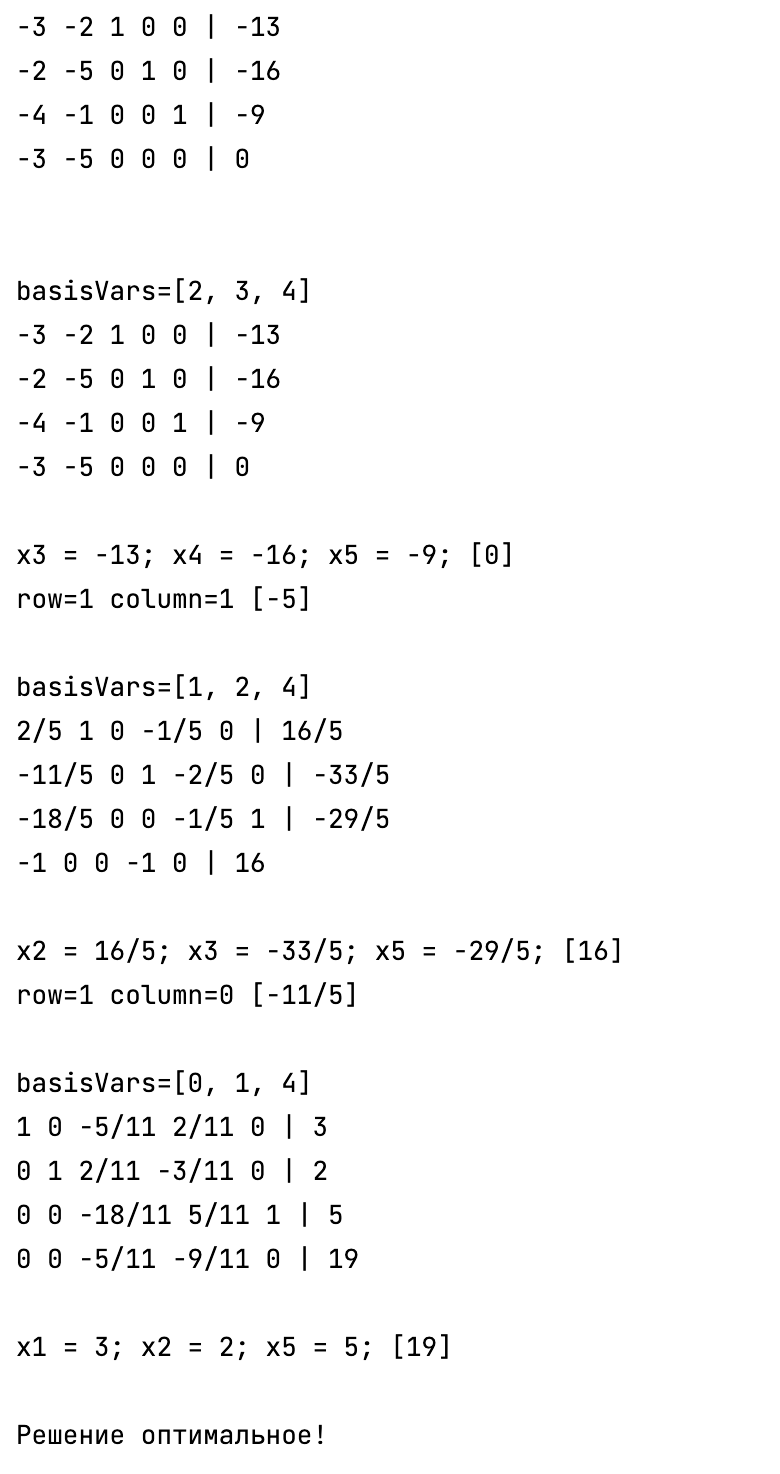
Для работы алгоритма двойственного симплекс-метода написаны следующие ключевые функции:

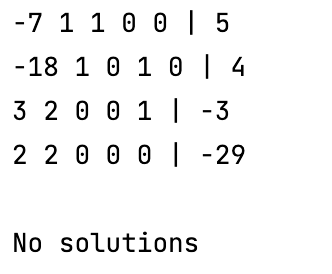
1. calc\_co – функция, проверяющая условие допустимости. Если решение недопустимо, функция возвращает номер строки, для которой требуется найти симплексные отношения. Если решение допустимо, исполнение алгоритма прекращается.
2. is\_optimal\_solution – преобразует матрицу и Z в соответствии с базисом.
3. tabel\_print – выводит на экран матрицу и Z в табличном виде. При необходимости заполняет строку симплексных отношений.

Основной алгоритм программы:

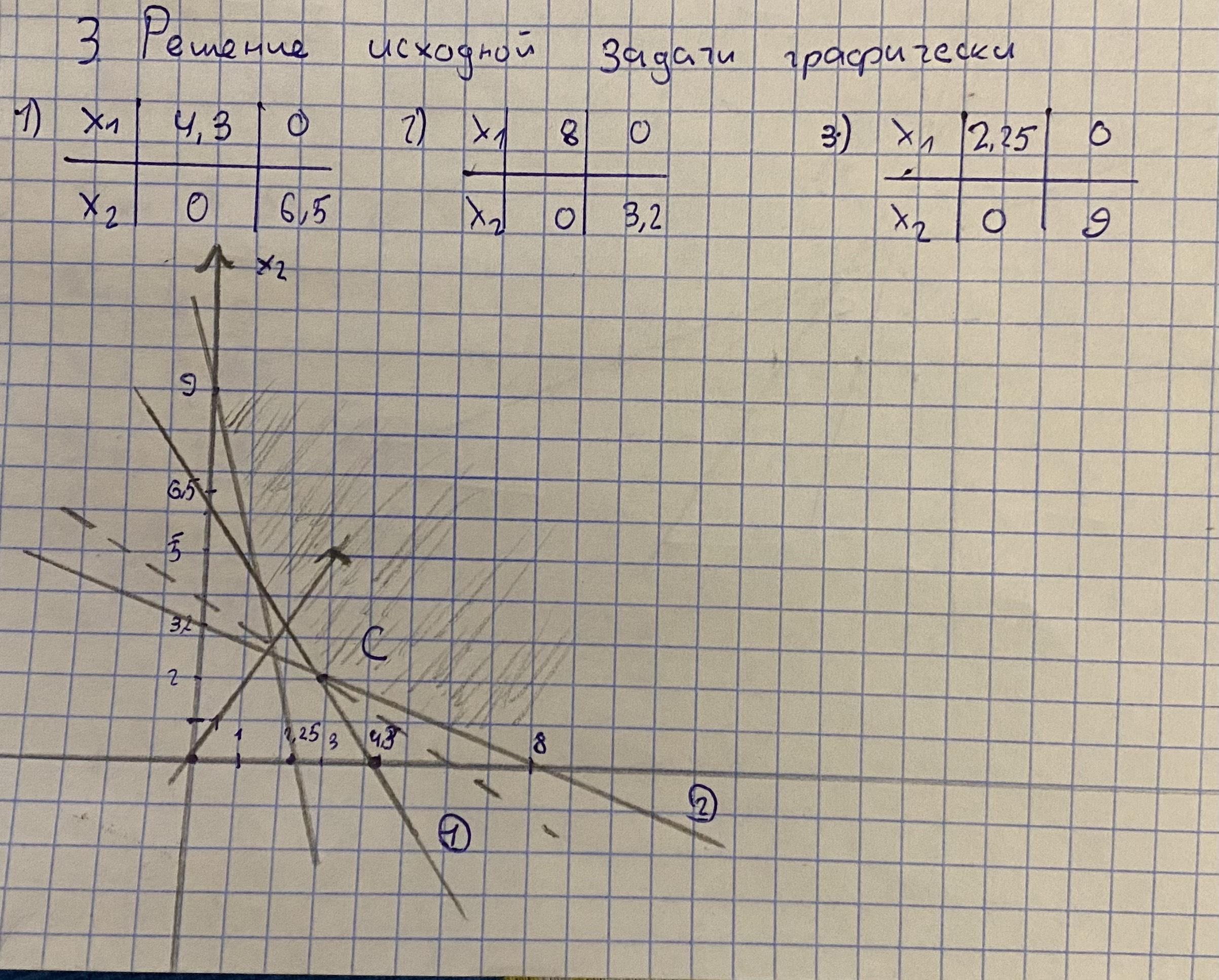
(Рис. 5) Основной алгоритм

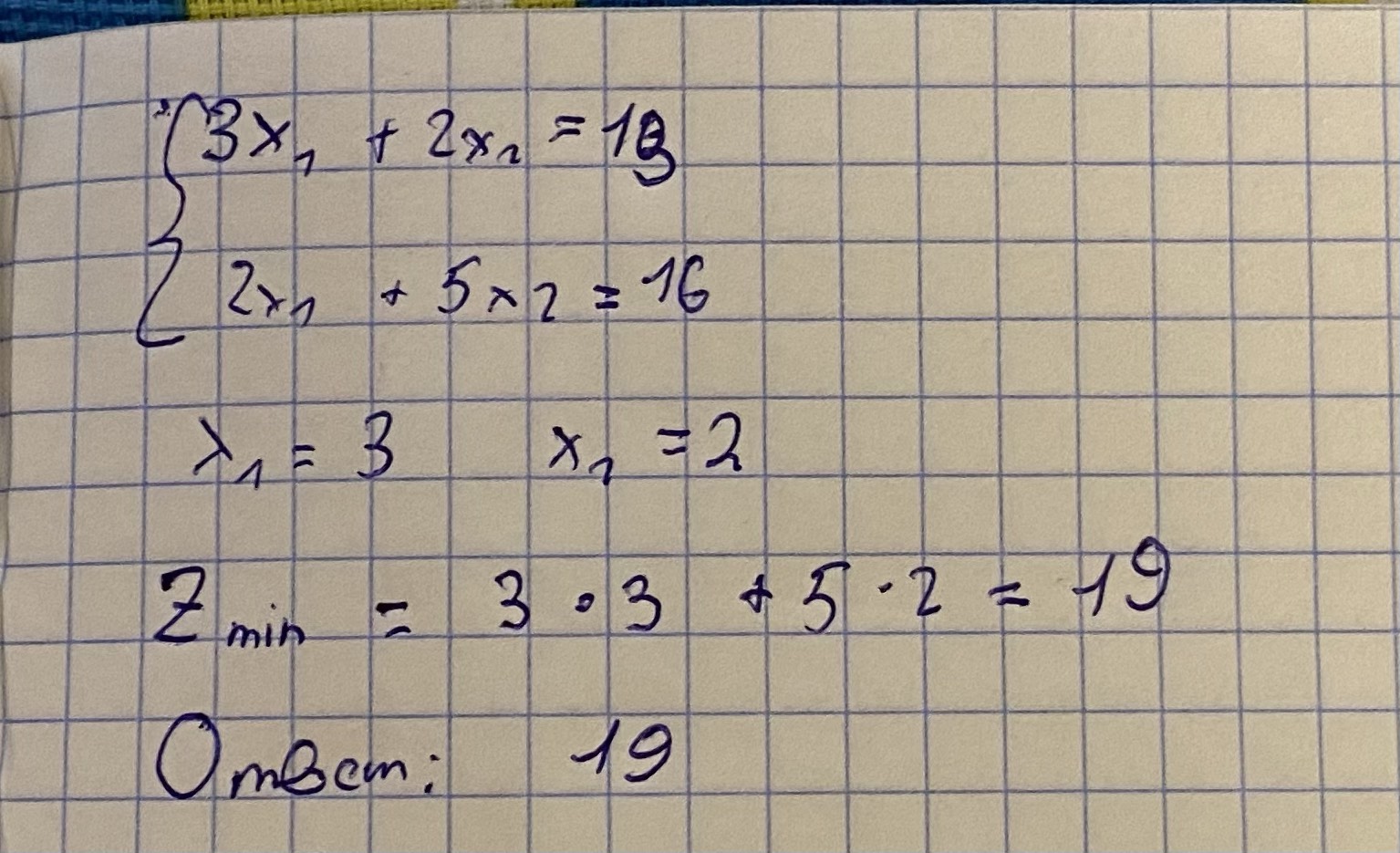
**Работа программы**

Программа выводит каждую итерацию симплекс-метода и вердикт – единственное решение или нет решений.

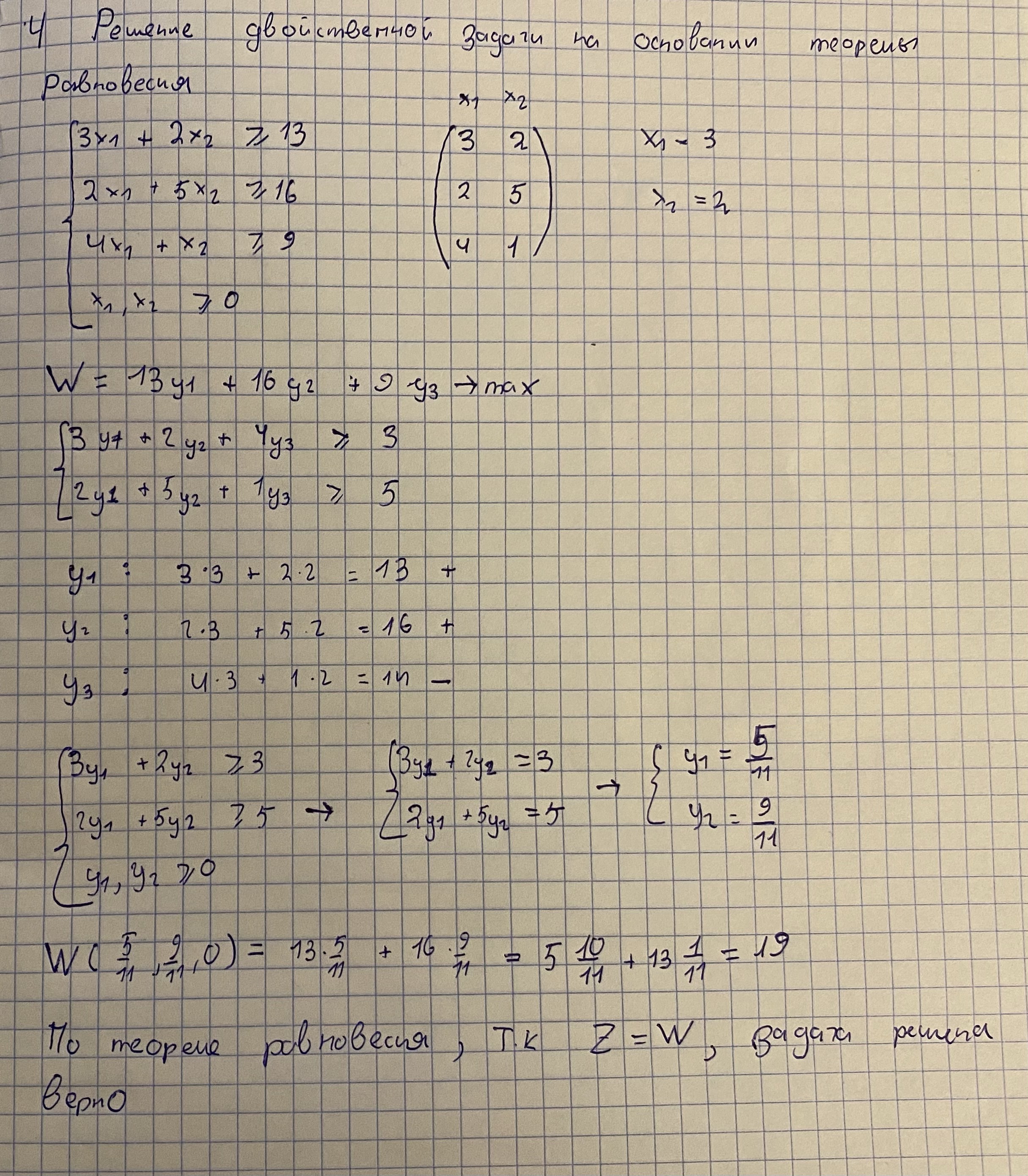
(Рис. 6) Пример работы программы (оптимальное решение)

(Рис. 7) Пример работы программы (нет решений)

**Решение исходной задачи графически**

(Рис. 8) Решение исходной задачи графически

(Рис. 9) Решение исходной задачи графически (ответ)

**Решение двойственной задачи на основании теоремы равновесия**

(Рис. 9) Решение двойственной задачи на основании теоремы равновесия

**Список используемых источников**

1. Галкина М. Ю Алгоритмы и вычислительные методы оптимизации: конспект лекций для студ. дневной формы обучения [Электронный ресурс] Новосибирск: Сиб. гос. ун-т телекоммуникации и информатики, 2022 - Режим доступа: https://eios.sibsutis.ru/mod/folder/view.php?id=118806
2. Электронный учебник "Экономико-математические методы" 11.4.  ДВОЙСТВЕННЫЙ СИМПЛЕКС-МЕТОД [Электронный ресурс] http://www.math.mrsu.ru/text/courses/method/dvoistvennii\_simplex.htm

**Приложение**

Код программы

**import** math

**class** Fraction:

**def** \_\_init\_\_(self, numerator, denominator=1):

self.numerator = numerator

self.denominator = denominator

self.canonize()

@staticmethod

**def** fromstr(string):

**return** Fraction(\*(int(i.strip()) **for** i **in** string.split(**'/'**)))

**def** \_\_str\_\_(self):

**if** self.denominator == 1:

**return f'{**self.numerator**}'**

**elif** self.denominator == -1 **and** self.numerator != 0:

**return f'{**-self.numerator**}'**

**elif** self.numerator != 0:

**if** self.numerator < 0 **or** self.denominator < 0:

**return f'-{**abs(self.numerator)**}/{**abs(self.denominator)**}'**

**else**:

**return f'{**self.numerator**}/{**self.denominator**}'**

**else**:

**return f'{**self.numerator**}'**

**def** canonize(self):

**def** gcd(a, b):

f = 0

**if** a < 0:

a = -a

f += 1

**if** b < 0:

b = -b

f += 1

**while** a != 0 **and** b != 0:

**if** a > b:

a %= b

**else**:

b %= a

**if** f == 2:

**return** (a + b) \* (-1)

**return** a + b

g = gcd(self.numerator, self.denominator)

self.numerator //= g

self.denominator //= g

**def** \_\_float\_\_(self):

**return** self.numerator / self.denominator

**def** \_\_add\_\_(self, value):

**return** Fraction(

self.numerator \* value.denominator + value.numerator \* self.denominator,

self.denominator \* value.denominator

)

**def** \_\_sub\_\_(self, value, /):

**return** Fraction(

self.numerator \* value.denominator - value.numerator \* self.denominator,

self.denominator \* value.denominator

)

**def** \_\_mul\_\_(self, value):

**return** Fraction(

self.numerator \* value.numerator,

self.denominator \* value.denominator

)

**def** \_\_truediv\_\_(self, value):

**return** Fraction(

self.numerator \* value.denominator,

self.denominator \* value.numerator

)

**def** \_\_neg\_\_(self):

**return** Fraction(

-self.numerator,

self.denominator

)

**def** tabel\_print(arr):

**for** i **in** range(len(arr)):

**for** j **in** range(len(arr[i])):

**if** j == len(arr[i]) - 1:

print(**'|'**, end=**' '**)

print(arr[i][j], end=**' '**)

print()

print()

**def** str\_print(a, flag=0):

**global** basisVars

k = 0

basisVars[curr\_string], basisVars[k] = basisVars[k], basisVars[curr\_string]

curr = a[curr\_string][curr\_column]

a[curr\_string], a[k] = a[k], a[curr\_string]

**if** flag != 1:

a[0][:] = [a[0][i].\_\_truediv\_\_(curr) **for** i **in**

range(0, len(a[0]))]

**def** is\_fully\_zero(a):

count = 0

**for** i **in** range(len(a)):

**if** a[i].numerator == 0.0:

count += 1

**return** count == len(a)

**def** nullify(a):

**global** curr\_string

n = len(a)

**for** i **in** range(len(a)):

**if** i != 0:

factor = a[i][curr\_column]

a[i][:] = [a[i][j].\_\_sub\_\_(a[0][j].\_\_mul\_\_(factor)) **for** j **in** range(len(a[i]))]

**def** is\_optimal\_solution(a, basis\_vars):

optimal = **True**

**for** i **in** range(len(a) - 1):

**if** a[i][- 1].\_\_float\_\_() < 0:

optimal = **False**

p = 0

res[0] = [0] \* 5

**for** j **in** basisVars:

res[0][j] = a[p][len(a[0]) - 1].\_\_float\_\_()

p += 1

**for** j **in** range(len(a[0]) - 1):

**if** j **not in** basis\_vars **and** a[len(a) - 1][j].\_\_float\_\_() == 0:

print(**'Many solution'**)

many\_solutions(a, j)

**break**

**return** optimal

**def** find\_min\_el(a):

min\_el = []

**for** i **in** range(len(a) - 1):

**for** j **in** range(len(a[i])):

**if** j == len(a[i]) - 1:

min\_el.append((a[i][j].\_\_float\_\_(), i))

**return** min\_el

**def** calc\_co(a, row):

co = [math.inf] \* (len(a[0]) - 1)

flag = **False**

**for** j **in** range(len(a[row]) - 1):

**if** a[row][j].\_\_float\_\_() < 0:

flag = **True**

**if not** flag:

**return** flag

**for** j **in** range(len(a[row]) - 1):

**if** a[len(a) - 1][j].\_\_float\_\_() != 0 **and** a[row][j].\_\_float\_\_() != 0:

co[j] = abs(a[len(a) - 1][j].\_\_truediv\_\_(a[row][j]).\_\_float\_\_())

**return** co

**def** many\_solutions(a, col):

**global** curr\_string

**global** curr\_column

co = [math.inf] \* (len(a) - 1)

**for** i **in** range(len(a) - 1):

**if** a[i][col].\_\_float\_\_() > 0:

co[i] = a[i][len(a[i]) - 1].\_\_truediv\_\_(a[i][col]).\_\_float\_\_()

minem = min(enumerate(co), key=**lambda** p: p[1])[0]

curr\_string = minem

curr\_column = col

basisVars[curr\_string] = curr\_column

str\_print(a)

nullify(a)

tabel\_print(a)

**for** k, i **in** enumerate(basisVars):

print(**f'x{**i + 1**} = {**a[k][len(a[k]) - 1]**}'**, end=**'; '**)

p = 0

res[1] = [0] \* 5

**for** j **in** basisVars:

res[1][j] = a[p][len(a[0]) - 1]

p += 1

lambda\_return(res)

exit()

**def** lambda\_return(res):

ret = []

**for** j **in** range(len(res[0])):

ret.append(**f'(1-L){**res[0][j]**}+L\*{**res[1][j]**}'**)

**for** i **in** ret:

print(i)

a = [[-3, -2, 1, 0, 0, -13], [-2, -5, 0, 1, 0, -16], [-4, -1, 0, 0, 1, -9], [-3, -5, 0, 0, 0, 0]]

res = [[], []]

**if** \_\_name\_\_ == **'\_\_main\_\_'**:

**for** i **in** range(len(a)):

**for** j **in** range(len(a[i])):

a[i][j] = Fraction(a[i][j])

tabel\_print(a)

curr\_column = 0

curr\_string = 0

n = len(a) - 1

start\_position = (len(a[0]) - len(a))

basisVars = [0] \* (len(a) - 1)

basisVars[:] = [a **for** a **in** range(start\_position, start\_position + (len(a) - 1))]

**try**:

**while not** is\_optimal\_solution(a, basisVars):

minEl = find\_min\_el(a)

row = min(minEl)[1]

CO = calc\_co(a, row)

**if not** CO:

print(**'No solutions'**)

exit()

column = min(enumerate(CO), key=**lambda** p: p[1])[0]

print(**f'\n{**basisVars=**}'**)

tabel\_print(a)

**for** k, i **in** enumerate(basisVars):

print(**f'x{**i + 1**} = {**a[k][len(a[k]) - 1]**}'**, end=**'; '**)

print(**f'[{**a[len(a) - 1][len(a[0]) - 1]**}]'**)

print(**f'{**row=**} {**column=**} [{**a[row][column]**}]'**)

curr\_string = row

curr\_column = column

basisVars[curr\_string] = curr\_column

str\_print(a)

nullify(a)

**else**:

print(**f'\n{**basisVars=**}'**)

tabel\_print(a)

**for** k, i **in** enumerate(basisVars):

print(**f'x{**i + 1**} = {**a[k][len(a[k]) - 1]**}'**, end=**'; '**)

print(**f'[{**a[len(a) - 1][len(a[0]) - 1]**}]'**)

print(**'\nOptimal solution!'**)

**except** Exception **as** ex:

print(ex)

print(**'Exception!'**)