

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Факультет 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
Метод искусственного базиса

Вариант 8

Выполнил:

студент гр.ИП-814

Краснов И.В.

«14» апреля 2021 г.

Подпись студента _____

Проверил:

ассистент кафедры ПМиК

Новожилов Д. И.

«14» апреля 2021 г.

Подпись преподавателя _____

Оценка _____

Новосибирск 2021 г.

Содержание

Задание	3
Переход к канонической форме записи	4
Решение задачи линейного программирования методом искусственного базиса	5
Графическое решение исходной задачи	7
Решение двойственной задачи	8
Код программы	10
Примеры решения задач	23

Задание

1. Перейти к канонической форме записи задачи линейного программирования.

$$\begin{cases} Z(x_1, x_2) = p_1x_1 + p_2x_2 \rightarrow \min \\ a_1x_1 + a_2x_2 \geq a \\ b_1x_1 + b_2x_2 \geq b \\ c_1x_1 + c_2x_2 \geq c \\ x_1; x_2 \geq 0 \end{cases}$$

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

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.

Номер варианта	a	b	c	a_1	b_1	c_1	a_2	b_2	c_2	p_1	p_2	Метод решения задачи
8.	45	8	30	10	1	3	3	1	5	2	10	2

Рис. 1 – Исходные данные для варианта №8.

Переход к канонической форме записи

Для того, чтобы перейти к канонической форме записи исходной ЗЛП (задачи линейного программирования), необходимо добавить балансовые переменные соответствующего неравенству знака, а также сменить условие минимизации целевой функции на условие максимизации. Так как все знаки в неравенствах – это « \geq », то необходимо в каждое неравенство добавить по одной уникальной балансовой переменной со знаком минус, тогда уравнение можно будет записать со знаком « $=$ ».

$$\begin{aligned} 1) \quad Z &= 2x_1 + 10x_2 \rightarrow \min \\ &\begin{cases} 10x_1 + 3x_2 \geq 45 \\ x_1 + x_2 \geq 8 \\ 3x_1 + 5x_2 \geq 30 \\ x_1, x_2 \geq 0 \end{cases} \\ \\ Z' &= -2x_1 - 10x_2 \rightarrow \max \\ &\begin{cases} 10x_1 + 3x_2 - x_3 = 45 \\ x_1 + x_2 - x_4 = 8 \\ 3x_1 + 5x_2 - x_5 = 30 \\ x_i \geq 0, \quad i = 1..5 \end{cases} \end{aligned}$$

Рис. 2 – Каноническая форма записи исходной ЗЛП.

Решение задачи линейного программирования методом искусственного базиса

Далее будет описан алгоритм работы программы. Сначала пользователь вводит входные данные, т.е. саму ЗЛП в канонической форме записи.

```
Количество переменных:
5
Количество ограничений:
3
Заполните матрицу:
1  столбец
10
3
-1
0
0
45
2  столбец
1
1
0
-1
0
8
3  столбец
3
5
0
0
-1
30
Введите Z:
-2
-10
0
0
0
C = 0
```

Рис. 3 – Входные данные.

После того, как данные введены, вызывается функция `artificial_basis()`, в котором выполняется сам метод искусственного базиса. Сначала происходит добавление нового базиса (в каждую строку добавляется новая переменная). После этого добавляются последовательно строки *Z* и *M*. Таким образом сформировалась симплекс-таблица, над которой будут производиться дальнейшие вычисления.

После того, как сформировалась симплекс-таблица, начинается первый цикл, обрабатывающий строку *M*. В этом цикле из таблицы последовательно выводятся базисные переменные. В функции `check_m()` проверяется строка *M*, если в переменных есть отрицательное значение, цикл продолжается, если вся строка обнулилась, цикл завершается, иначе программа завершается со словами «Система не совместна». После завершения цикла *M*-строка удаляется.

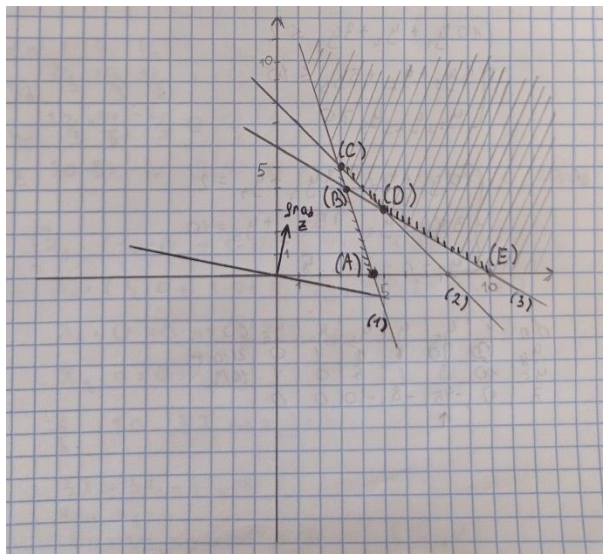
Затем следует цикл симплекс-метода, выход из цикла происходит, когда все отрицательные значения в строке *Z* перестали быть отрицательными (проверка происходит в функции `check_z()`).

Этап 3: Решение методом искусственного базиса									
45	10	3	-1	0	0	1	0	0	
8	1	1	0	-1	0	0	1	0	
30	3	5	0	0	-1	0	0	1	
0	2	10	0	0	0	0	0	0	
-83	-14	-9	1	1	1	0	0	0	
Минимальное CO = (0 , 1) = 45 / 10									
BASIS = [1, 0, 0, 0, 0, 0, -2, 6, 7]									
9/2	1	3/10	-1/10	0	0	0	0	0	
7/2	0	7/10	1/10	-1	0	1	0	0	
33/2	0	41/10	3/10	0	-1	0	1	0	
-9	0	47/5	1/5	0	0	0	0	0	
-20	0	-24/5	-2/5	1	1	0	0	0	
Минимальное CO = (2 , 2) = 33/2 / 41/10									
BASIS = [1, 0, 2, 0, 0, 0, -2, 6, -2]									
135/41	1	0	-5/41	0	3/41	0	0	0	
28/41	0	0	2/41	-1	7/41	1	0	0	
165/41	0	1	3/41	0	-10/41	0	0	0	
-1920/41	0	0	-20/41	0	94/41	0	0	0	
-28/41	0	0	-2/41	1	-7/41	0	0	0	
Минимальное CO = (1 , 5) = 28/41 / 7/41									
BASIS = [1, 5, 2, 0, 0, 0, -2, -2, -3]									
3	1	0	-1/7	3/7	0	0	0	0	
4	0	0	2/7	-41/7	1	0	0	0	
5	0	1	1/7	-10/7	0	0	0	0	
-56	0	0	-8/7	94/7	0	0	0	0	
0	0	0	0	0	0	0	0	0	
Этап 4: Решение симплекс-методом									
3	1	0	-1/7	3/7	0	0	0	0	
4	0	0	2/7	-41/7	1	0	0	0	
5	0	1	1/7	-10/7	0	0	0	0	
-56	0	0	-8/7	94/7	0	0	0	0	
Минимальное CO = (1 , 3) = 4 / 2/7									
BASIS = [1, 3, 2, 0, 0, 0, -2, -2, -3]									
5	1	0	0	-5/2	1/2	0	0	0	
14	0	0	1	-41/2	7/2	0	0	0	
3	0	1	0	3/2	-1/2	0	0	0	
-40	0	0	0	-10	4	0	0	0	
Минимальное CO = (2 , 4) = 3 / 3/2									
BASIS = [1, 3, 4, 0, 0, 0, -2, -2, -3]									
10	1	5/3	0	0	-1/3	0	0	0	
55	0	41/3	1	0	-10/3	0	0	0	
2	0	2/3	0	1	-1/3	0	0	0	
-20	0	20/3	0	0	2/3	0	0	0	
Z -> max = Z(10, 0, 55, 2, 0) = -20									

Рис. 4 – Промежуточные таблицы работы алгоритма

Графическое решение исходной задачи

Для графического решения исходной ЗЛП мы построили линии, соответствующие уравнениям из исходной системы ограничений, определили, где находится область, которую мы ищем, и построили градиент, по которому, с помощью линий уровня, нашли точку минимума нашей функции. Это точка с координатами (8;0).



Этап 3: Решение методом искусственного базиса

45	10	3	-1	0	0	1	0	0
0	1	1	0	-1	0	0	1	0
30	3	5	0	0	-1	0	0	1
0	2	10	0	0	0	0	0	0
-83	-14	-9	1	1	1	0	0	0

Минимальное CO = (0 , 1) = 45 / 10
BASIS = [1, 0, 0, 0, 0, 0, -2, 6, 7]

A

9/2	1	3/10	-1/10	0	0	0	0	0
7/2	0	7/10	1/10	-1	0	1	0	0
33/2	0	41/10	3/10	0	-1	0	1	1
-9	0	47/5	1/5	0	0	0	0	0
-20	0	-24/5	-2/5	1	1	0	0	0

Минимальное CO = (2 , 2) = 33/2 / 41/10
BASIS = [1, 0, 2, 0, 0, 0, -2, 6, -2]

B

135/41	1	0	-5/41	0	3/41	0	0	0
28/41	0	0	2/41	-1	7/41	1	0	0
165/41	0	1	3/41	0	-10/41	0	1	0
-1920/41	0	0	-20/41	0	94/41	0	0	0
-28/41	0	0	-2/41	1	-7/41	0	0	0

Минимальное CO = (1 , 5) = 28/41 / 7/41
BASIS = [1, 5, 2, 0, 0, 0, -2, -2, -3]

C

3	1	0	-1/7	3/7	0	0	0	0
4	0	0	2/7	-41/7	1	0	0	0
5	0	1	1/7	-10/7	0	0	0	0
-56	0	0	-8/7	94/7	0	0	0	0
0	0	0	0	0	0	0	0	0

Этап 4: Решение симплекс-методом

3	1	0	-1/7	3/7	0	0	0	0
4	0	0	2/7	-41/7	1	0	0	0
5	0	1	1/7	-10/7	0	0	0	0
-56	0	0	-8/7	94/7	0	0	0	0

Минимальное CO = (1 , 3) = 4 / 2/7
BASIS = [1, 3, 2, 0, 0, 0, -2, -2, -3]

D

5	1	0	0	-5/2	1/2	0	0	0
14	0	0	0	-41/2	7/2	0	0	0
3	0	1	0	3/2	-1/2	0	0	0
-40	0	0	0	-10	4	0	0	0

Минимальное CO = (2 , 4) = 3 / 3/2
BASIS = [1, 3, 4, 0, 0, 0, -2, -2, -3]

E

10	1	5/3	0	0	-1/3	0	0	0
55	0	41/3	1	0	-10/3	0	0	0
2	0	2/3	0	1	-1/3	0	0	0
-20	0	20/3	0	0	2/3	0	0	0

Z -> max = Z(10, 0, 55, 2, 0) = -20

Рис. 5 – Графическое решение исходной ЗЛП.

Решение двойственной задачи

4) $Z = 2x_1 + 10x_2 \rightarrow \min$

$$y_1 \begin{cases} 10x_1 + 3x_2 \geq 45 \\ x_1 + x_2 \geq 8 \\ 3x_1 + 5x_2 \geq 30 \\ x_1, x_2 \geq 0 \end{cases}$$

$W = 45y_1 + 8y_2 + 30y_3 \rightarrow \max$

$$\begin{cases} 10y_1 + y_2 + 3y_3 \leq 2 \\ 3y_1 + y_2 + 5y_3 \leq 10 \\ y_1, y_2, y_3 \geq 0 \end{cases}$$

④

$$\begin{cases} 10y_1 + y_2 + 3y_3 + y_4 = 2 \\ 3y_1 + y_2 + 5y_3 + y_5 = 10 \\ y_i \geq 0, i = 1..5 \end{cases}$$

Ln	1	y_1	y_2	y_3	y_4	y_5	CO
y_4	10	1	1	1	0	0	2/10 ←
y_5	10	3	1	5	0	1	10/3
Z	0	-45	-8	-30	0	0	

↑

Рис. 6 – Составление двойственной задачи и симплексной таблицы.

d.n	1	y_1	y_2	y_3	y_4	y_5	CO
y_1	$2/10$	1	$1/10$	$3/10$	$1/100$	0	$2/3$ ←
y_5	$9/41$	0	$7/10$	$41/10$	$-3/10$	1	0.4 $41/10$
Z	9	0	$-3.5/10$	$-26.5/10$	$4.5/10$	0	

↑

d.n	y_1	y_2	y_3	y_4	y_5
y_1	1				
y_5	$9/41$	0	$7/41$	1	$-3/41$
Z					

d.n	1	y_1	y_2	y_3	y_4	y_5	CO
y_2	$2/3$	$10/3$	$1/3$	1	$1/3$	0	
y_5	$20/3$	$-41/3$	$-2/3$	0	$-5/3$	1	
Z	20	55	2	0	10	0	

(0; 0; $2/3$; 0; $20/3$)

$$\begin{cases} x_1 \cdot (10 \cdot 0 + 1 \cdot 0 + 3 \cdot 2/3 + 1 \cdot 0 + 2) = 0 \\ x_2 \cdot (3 \cdot 0 + 1 \cdot 0 + 5 \cdot 2/3 + 20/3 + 10) = 0 \end{cases}$$

$$x_1 \cdot 0 = 0 \Rightarrow x_1 \neq 0 \quad \underline{x_1}$$

$$x_2 \cdot 0 = 0 \Rightarrow \underline{x_2}$$

$$\frac{2}{3} \cdot (30 - (3x_1 + 5x_2)) = 0$$

$$30 - 3x_1 - 5x_2 = 0$$

при $x_2 = 0$

$$30 - 3x_1 = 0 \Rightarrow x_1 = 10 \quad Z(10; 0) = Z_{\min} = 20$$

Рис. 7 –Решение двойственной задачи на основании теоремы равновесия.

Код программы

SimpleFrac.py – файл с описанием класса дроби

```
class SimpleFrac:
    a = 0
    b = 1

    def __init__(self):
        self.a = 0
        self.b = 1

    def write(self, x: int, y: int):
        self.a = x
        self.b = y

    def __str__(self):
        if self.b != 1:
            return '{}/{ {}'.format(int(self.a), int(self.b))
        else:
            return '{}'.format(int(self.a))

    def sf_reduce(self):
        temp_a = abs(self.a)
        temp_b = abs(self.b)

        if self.a == 0:
            return self

        while temp_a != temp_b:
            if temp_a > temp_b:
                tmp = temp_a
                temp_a = temp_b
                temp_b = tmp
            temp_b = temp_b - temp_a

        self.a /= temp_a
        self.b /= temp_a
```

```

    return self

def __add__(self, other):
    result = SimpleFrac()
    result.a = self.a * other.b + other.a * self.b
    result.b = self.b * other.b

    if result.b < 0:
        result.a *= -1
        result.b *= -1

    if result.a == 0:
        result.b = 1

    result.sf_reduce()

    return result

def __sub__(self, other):
    result = SimpleFrac()
    result.a = self.a * other.b - other.a * self.b
    result.b = self.b * other.b

    if result.b < 0:
        result.a *= -1
        result.b *= -1

    if result.a == 0:
        result.b = 1

    result.sf_reduce()

    return result

def __mul__(self, other):
    result = SimpleFrac()
    temp1 = SimpleFrac()

```

```

temp2 = SimpleFrac()

temp1.a = self.a
temp1.b = other.b
temp2.a = other.a
temp2.b = self.b
temp1.sf_reduce()
temp2.sf_reduce()
result.a = temp1.a * temp2.a
result.b = temp1.b * temp2.b

if result.b < 0:
    result.a *= -1
    result.b *= -1

if result.a == 0:
    result.b = 1

result.sf_reduce()

return result

def __truediv__(self, other):
    result = SimpleFrac()
    temp1 = SimpleFrac()
    temp2 = SimpleFrac()

    temp1.a = self.a
    temp1.b = other.a
    temp2.a = other.b
    temp2.b = self.b
    temp1.sf_reduce()
    temp2.sf_reduce()
    result.a = temp1.a * temp2.a
    result.b = temp1.b * temp2.b

    if result.b < 0:

```

```

        result.a *= -1
        result.b *= -1

    if result.a == 0:
        result.b = 1

    result.sf_reduce()

    return result

def __gt__(self, other):
    result = self - other
    if result.a > 0:
        return True
    else:
        return False

def __ge__(self, other):
    result = self - other
    if result.a >= 0:
        return True
    else:
        return False

def __lt__(self, other):
    result = self - other
    if result.a < 0:
        return True
    else:
        return False

def __le__(self, other):
    result = self - other
    if result.a <= 0:
        return True
    else:
        return False

```

```

def __abs__(self):
    if self.a < 0:
        self.a *= -1
    return self

```

main.py – главный исполнительный файл

```

from SimpleFrac import SimpleFrac
import math

def print_matrix(matrix, n, m, k):

    print()
    if k == "matrix":
        for i in range(m):
            for j in range(n + 1):
                if j == n:
                    print("|%10s" % matrix[i][j], end='')
                else:
                    print("%10s" % matrix[i][j], end='')
            print()
    elif k == "simplex":
        for i in range(m + 2):
            print("\t", end='')
            for j in range(n + 1):
                if j == 0:
                    print("%10s|" % matrix[i][j], end='')
                else:
                    print("%10s" % matrix[i][j], end='')
            print()
    else:
        print("\nERROR: UNKNOWN SITUATION\n")

def check_m(simplex_list, n):

```

```

k = 0

for i in range(1, n + 1):
    if simplex_list[i].a > 0:
        k = 1

    elif simplex_list[i].a < 0:
        return 0

if k == 0 and simplex_list[0].a == 0:
    return 1
else:
    return 2

def check_z(simplex_list, n):

    k = 0

    for i in range(1, n + 1):
        if simplex_list[i].a < 0:
            return 0

    return 1

def artificial_basis(matrix, n, m, z):

    N = n
    simplex = []
    basis = []

    for i in range(n + m + 1):
        basis.append(i)
    for i in range(n + 1):
        basis[i] = 0

```



```

print("\n", "Этап 1: Расширение матрицы")

for i in range(m):
    k = 0
    for j in range(m):
        matrix[i].insert(N + k, SimpleFrac())
        if j == i:
            matrix[i][N + k].write(1, 1)
        k += 1

N += m
print_matrix(matrix, N, m, "matrix")

print("\n", "Этап 2: Построение Симплекс-таблицы")

for i in range(m):
    simplex.append([])
    for j in range(N + 1):
        if j == 0:
            simplex[i].append(matrix[i][N])
        else:
            simplex[i].append(matrix[i][j - 1])

simplex.append([])
for j in range(n + 1):
    if j == 0:
        simplex[m].append(z[n])
    else:
        temp = z[j - 1]
        temp.a *= -1
        simplex[m].append(temp)

for j in range(n + 1, N + 1):
    simplex[m].append(SimpleFrac())

simplex.append([])
for j in range(n + 1):

```

```

if j == 0:
    temp = SimpleFrac()
    for i in range(m):
        temp = temp + matrix[i][N]
    temp.a *= -1
    simplex[m + 1].append(temp)
else:
    temp = SimpleFrac()
    for i in range(m):
        temp = temp + matrix[i][j - 1]
    temp.a *= -1
    simplex[m + 1].append(temp)

for j in range(n + 1, N + 1):
    simplex[m + 1].append(SimpleFrac())

print_matrix(simplex, N, m, "simplex")

print("\n", "Этап 3: Решение методом искусственного базиса")

print_matrix(simplex, N, m, "simplex")
# print("\n\t", "BASIS = ", basis)
while True:

    stop_kof = check_m(simplex[m + 1], n)                # проверка строки M

    if stop_kof == 0:                # если есть отрицательный элемент в строке M
        maxi = SimpleFrac()
        maxi_i = -1
        for j in range(1, N + 1):                # поиск столбца с
            # максимальным по модулю значением в M
            if int(simplex[m + 1][j].a) < 0 and simplex[m + 1][j] < maxi:
                maxi = simplex[m + 1][j]
                maxi_i = j

        mini = SimpleFrac()
        mini_i = -1

```

```

k = -1

for i in range(m):
    # поиск опорного элемента, чьё
    симплексное отношение будет взято как минимальное
    if simplex[i][maxi_i].a > 0 and simplex[i][0].a > 0:
        mini = simplex[i][0] / simplex[i][maxi_i]
        k = i
        break

if k == -1:
    print("Нет решений")
    return 1

for i in range(k, m):
    # поиск строки с минимальным
    симплексным отношением
    if simplex[i][maxi_i].a > 0 and simplex[i][0].a > 0 and
    simplex[i][0] / simplex[i][maxi_i] <= mini:
        mini = simplex[i][0] / simplex[i][maxi_i]
        mini_i = i

    print("\n\t", "Минимальное CO = (", mini_i, ",", maxi_i, ") = ",
    simplex[mini_i][0], " / ", simplex[mini_i][maxi_i])

    mini = simplex[mini_i][maxi_i]

    for i in range(N + 1):
        # делим строку с минимальным
        симплексным отношением на опорный элемент
        simplex[mini_i][i] = simplex[mini_i][i] / mini

    for i in range(m + 2):
        # шаг Гаусса
        for j in range(N + 1):
            if i != mini_i and j != maxi_i:
                temp_1 = simplex[i][j] * simplex[mini_i][maxi_i]
                temp_2 = simplex[i][maxi_i] * simplex[mini_i][j]
                simplex[i][j] = temp_1 - temp_2

    for i in range(m + 2):
        # обнуляем столбец с опорным
        элементом
        if i != mini_i:
            simplex[i][maxi_i].a = 0
            simplex[i][maxi_i].b = 1

```

```

        if basis[n + 1 + mini_i] >= 0:          # удаление столбца с
искусственной переменной и обновление базиса

            # print("\n\t", "DEL = ", basis[n + 1 + mini_i])
            temp_k = basis[n + 1 + mini_i]
            for i in range(m + 2):
                simplex[i].pop(temp_k)
            N -= 1
            basis[n + 1 + mini_i] = -1
            for i in range(n + 1 + mini_i, n + m + 1):
                basis[i] -= 1
            basis[mini_i] = maxi_i
            print("BASIS = ", basis)

        print_matrix(simplex, N, m, "simplex")

    elif stop_kof == 1:          # если строка M обнулилась
        break

    elif stop_kof == 2:          # если матрица не разрешима
        print("\n", "Система ограничений не совместна")
        return 1

    print("\n", "Этап 4: Решение симплекс-методом")

    simplex.pop(m + 1)
    print_matrix(simplex, N, m - 1, "simplex")
    while True:
        kof = check_z(simplex[m], N)          # проверяем строку z

        if kof == 0:          # если в строке есть отрицательный элемент
            maxi = SimpleFrac()
            maxi_i = -1
            for j in range(1, N + 1): # поиск столбца с максимальным по модулю
значением в Z
                if int(simplex[m][j].a) < 0 and simplex[m][j] < maxi:
                    maxi = simplex[m][j]
                    maxi_i = j

```

```

mini = SimpleFrac()

mini_i = -1

k = -1

for i in range(m): # поиск опорного элемента, чьё симплексное
отношение будет взято как минимальное
    if simplex[i][maxi_i].a > 0 and simplex[i][0].a > 0:
        mini = simplex[i][0] / simplex[i][maxi_i]
        k = i
        break

if k == -1:
    print("Начинайте паниковать")
    print("Но сперва проверьте правильно ли записана строка z")
    return 1

# print("Cur mini = ", mini)

for i in range(k, m): # поиск строки с минимальным симплексным
отношением
    if simplex[i][maxi_i].a > 0 and simplex[i][0].a > 0 and
simplex[i][0] / simplex[i][maxi_i] <= mini:
        mini = simplex[i][0] / simplex[i][maxi_i]
        mini_i = i

    print("\n\t", "Минимальное CO = (", mini_i, ",", maxi_i, ") = ",
simplex[mini_i][0], " / ",
simplex[mini_i][maxi_i])

mini = simplex[mini_i][maxi_i]

for i in range(N + 1): # делим строку с минимальным симплексным
отношением на опорный элемент
    simplex[mini_i][i] = simplex[mini_i][i] / mini

for i in range(m + 1): # шаг Гаусса
    for j in range(N + 1):
        if i != mini_i and j != maxi_i:
            temp_1 = simplex[i][j] * simplex[mini_i][maxi_i]
            temp_2 = simplex[i][maxi_i] * simplex[mini_i][j]
            simplex[i][j] = temp_1 - temp_2

for i in range(m + 1): # обнуляем столбец с опорным элементом

```

```

        if i != mini_i:
            simplex[i][maxi_i].a = 0
            simplex[i][maxi_i].b = 1

        basis[mini_i] = maxi_i          # записываем новый элемент в
базис

    print("\n", "BASIS = ", basis)

    print_matrix(simplex, N, m - 1, "simplex")

    if kof == 1:
        print("Z -> max = Z(", end='')

        for i in range(n):
            k = -1
            for j in range(n):
                if i + 1 == basis[j] and j <= m:
                    print(simplex[j][0], end='')
                    k = 0
                    break;
            if k != 0:
                print(0, end='')
            if i != n - 1:
                print(", ", end='')
        print(") = ", simplex[m][0])
        return 0

    return 1

if __name__ == '__main__':

    matrix = []
    z = []

    print("Количество переменных: ")
    n = int(input())

```

```

print("Количество ограничений: ")
m = int(input())

for i in range(m):
    matrix.append([])
    for j in range(n + 1):
        matrix[i].append(SimpleFrac())

print("Заполните матрицу: ")
for i in range(m):
    print(i + 1, " столбец")
    for j in range(n + 1):
        matrix[i][j].write(int(input()), 1)

print('Введите Z: ')
for i in range(n + 1):
    if i == n:
        print("C = ", end='')
    z.append(SimpleFrac())
    z[i].write(int(input()), 1)

print("\n", "Ваша матрица:")
print_matrix(matrix, n, m, "matrix")

artificial_basis(matrix, n, m, z)

```


Примеры решения задач

```

Windows PowerShell
Количество переменных:
5
Количество ограничений:
3
Заполните матрицу:
1 столбец
4
1
-1
0
0
9
2 столбец
2
0
0
0
0
13
3 столбец
2
0
0
0
0
16
Введите Z:
-1
-5
0
0
0
C = 0

Этап 1: Расширение матрицы
4 1 -1 0 0 0 9
3 2 0 -1 0 0 13
2 5 0 0 -1 0 16

Этап 2: Построение Симплекс-таблицы
9 | 4 1 -1 0 0 0 1 0 0
13 | 3 2 0 -1 0 0 0 1 0
16 | 2 5 0 0 -1 0 0 0 1
0 | 1 5 0 0 0 0 0 0 0
-36 | -9 0 0 1 1 1 0 0 0

Этап 3: Решение методом искусственного базиса
9 | 4 1 -1 0 0 0 1 0 0
13 | 3 2 0 -1 0 0 0 1 0
16 | 2 5 0 0 -1 0 0 0 1
0 | 1 5 0 0 0 0 0 0 0
-36 | -9 0 0 1 1 1 0 0 0

Минимальное CO = ( 0, 1 ) = 9 / 4
BASIS = [ 1, 0, 0, 0, 0, -2, 6, 7 ]

9/4 | 1 1/4 -1/4 0 0 0 0 0
25/4 | 0 5/4 3/4 -1 0 0 1 0
23/2 | 0 9/2 1/2 0 -1 0 0 1
-9/4 | 0 15/4 1/4 0 0 0 0 0
-71/4 | 0 -23/4 -5/4 1 1 0 0 0

Минимальное CO = ( 2, 2 ) = 23/2 / 9/2
BASIS = [ 1, 0, 2, 0, 0, 0, -2, 6, -2 ]

29/18 | 1 0 -5/18 0 1/18 0
55/18 | 0 0 11/18 -1 5/18 1
23/9 | 0 1 1/9 0 -2/9 0
-259/18 | 0 0 -5/18 0 19/18 0
-55/18 | 0 0 -11/18 1 -5/18 0

Минимальное CO = ( 1, 3 ) = 55/18 / 11/18
BASIS = [ 1, 3, 2, 0, 0, 0, -2, -2, -3 ]

3 | 1 0 0 -5/11 2/11
5 | 0 0 1 -18/11 5/11
2 | 0 1 0 -2/11 -3/11
-13 | 0 0 0 -5/11 13/11
0 | 0 0 0 0 0

Этап 4: Решение симплекс-методом
3 | 1 0 0 -5/11 2/11
5 | 0 0 1 -18/11 5/11
2 | 0 1 0 -2/11 -3/11
-13 | 0 0 0 -5/11 13/11
0 | 0 0 0 0 0

Минимальное CO = ( 2, 4 ) = 2 / 2/11
BASIS = [ 1, 3, 4, 0, 0, 0, -2, -2, -3 ]

8 | 1 5/2 0 0 -1/2
23 | 0 9 1 0 -2
11 | 0 11/2 0 1 -3/2
-4 | 0 5/2 0 0 1/2

Z -> max = Z(6, 0, 23, 11, 0) = -8

```

Рис. 8,9 Решение задачи из варианта 2

```

Windows PowerShell
Количество переменных:
5
Количество ограничений:
3
Заполните матрицу:
1 столбец
1
5
3
-1
0
0
30
2 столбец
2
0
0
0
0
26
3 столбец
2
0
0
0
0
54
Введите Z:
-2
-15
0
0
0
C = 0

Ваша матрица:
5 3 -1 0 0 30
2 4 0 -1 0 26
3 11 0 0 -1 54

Этап 2: Построение Симплекс-таблицы
30 | 5 3 -1 0 0 0 1 0 0
26 | 2 4 0 -1 0 0 0 1 0
54 | 3 11 0 0 -1 0 0 0 1
0 | 1 5 0 0 0 0 0 0 0
-110 | -10 -18 1 1 1 0 0 0

Этап 3: Решение методом искусственного базиса
30 | 5 3 -1 0 0 0 1 0 0
26 | 2 4 0 -1 0 0 0 1 0
54 | 3 11 0 0 -1 0 0 0 1
0 | 1 5 0 0 0 0 0 0 0
-110 | -10 -18 1 1 1 0 0 0

Минимальное CO = ( 2, 2 ) = 54 / 11
BASIS = [ 0, 0, 2, 0, 0, 0, 6, 7, -2 ]

168/11 | 46/11 0 -1 0 3/11 1 0
70/11 | 18/11 0 0 -1 4/11 0 1
54/11 | 3/11 1 0 0 -1/11 0 0
-818/11 | -23/11 0 0 0 15/11 0 0
-238/11 | -56/11 0 1 1 -7/11 0 0

Минимальное CO = ( 0, 1 ) = 168/11 / 46/11
BASIS = [ 1, 0, 2, 0, 0, 0, -2, 6, -3 ]

84/23 | 1 0 -11/46 0 3/46 0
70/23 | 0 0 5/23 -1 7/23 1
98/23 | 0 1 3/46 0 -5/46 0
-46 | 0 0 -1/2 0 3/2 0
-70/23 | 0 0 -5/23 1 -7/23 0

Минимальное CO = ( 1, 5 ) = 70/23 / 7/23
BASIS = [ 1, 5, 2, 0, 0, 0, -2, -2, -4 ]

3 | 1 0 -2/7 3/14 0
10 | 0 0 5/7 -23/7 1
5 | 0 1 1/7 -5/14 0
-81 | 0 0 -11/7 69/14 0
0 | 0 0 0 0 0

Этап 4: Решение симплекс-методом
3 | 1 0 -2/7 3/14 0
10 | 0 0 5/7 -23/7 1
5 | 0 1 1/7 -5/14 0
-81 | 0 0 -11/7 69/14 0

Минимальное CO = ( 1, 3 ) = 10 / 5/7
BASIS = [ 1, 3, 2, 0, 0, 0, -2, -2, -4 ]

7 | 1 0 0 -11/10 2/5
14 | 0 0 1 -23/5 7/5
3 | 0 1 0 1/10 -1/5
-59 | 0 0 0 -23/10 11/5

Минимальное CO = ( 2, 4 ) = 3 / 3/10
BASIS = [ 1, 3, 4, 0, 0, 0, -2, -2, -4 ]

18 | 1 11/3 0 0 -1/3
68 | 0 46/3 1 0 -5/3
10 | 0 10/3 0 1 -2/3
-36 | 0 23/3 0 0 2/3

Z -> max = Z(10, 0, 68, 10, 0) = -36

```

Рис. 10,11 Решение задачи из варианта 5

```

Количество переменных:
5
Количество ограничений:
3
Заполните матрицу:
1  столбец
2
5
-1
0
0
20
2  столбец
5
-1
0
12
3  столбец
5
4
-1
33
Введите Z:
-2
-7
C = 0
    
```

```

Этап 1: Расширение матрицы
      2   5   -1   0   0   1   0   0   20
      5   1   0   -1  0   0   1   0   12
      5   4   0   0   -1  0   0   1   33

Этап 2: Построение симплекс-таблицы
      20|   2   5   -1   0   0   1   0   0   0
      12|   5   1   0   -1  0   0   1   0   0
      33|   5   4   0   0   0   -1  0   0   1
      -65|   2   7   0   0   0   0   0   0   0
           -12  -10   1   1   1   0   0   0   0

Этап 3: Решение методом искусственного базиса
      20|   2   5   -1   0   0   1   0   0   0
      12|   5   1   0   -1  0   0   1   0   0
      33|   5   4   0   0   0   -1  0   0   1
      0|   2   7   0   0   0   0   0   0   0
      -65|  -12  -10   1   1   1   0   0   0   0

      Минимальное CO = ( 1 , 1 ) = 12 / 5
      BASIS = [ 0, 1, 0, 0, 0, 0, 0, -2, 7 ]

      76/5|   0   23/5   -1   2/5   0   1   0   0
      12/5|   1   1/5   0   -1/5   0   0   0   0
      21|   0   3   0   1   -1  0   1   1
      -24/5|  0   33/5   0   2/5   0   0   0   0
      -181/5| 0   -38/5   1   -7/5   1   0   0   0

      Минимальное CO = ( 0 , 2 ) = 76/5 / 23/5
      BASIS = [ 2, 1, 0, 0, 0, 0, -2, -3, 6 ]

      76/23|   0   1   -5/23   2/23   0   0   0
      48/23|   1   0   1/23   -5/23   0   0   0
      255/23| 0   0   15/23   17/23   -1   1   1
      -612/23| 0   0   33/23   -4/23   0   0   0
      -255/23| 0   0   -15/23   -17/23   1   0   0

      Минимальное CO = ( 2 , 4 ) = 255/23 / 17/23
      BASIS = [ 2, 1, 4, 0, 0, 0, -2, -3, -2 ]

      2|   0   1   -5/17   0   2/17
      5|   1   0   4/17   0   -5/17
      15|  0   0   15/17   1   -23/17
      -24| 0   0   27/17   0   -4/17
      0|   0   0   0   0   0

Этап 4: Решение симплекс-методом
      2|   0   1   -5/17   0   2/17
      5|   1   0   4/17   0   -5/17
      15|  0   0   15/17   1   -23/17
      -24| 0   0   27/17   0   -4/17
      -20| 0   2   1   0   0

      Минимальное CO = ( 0 , 5 ) = 2 / 2/17
      BASIS = [ 5, 1, 4, 0, 0, 0, -2, -3, -2 ]

      17|   0   17/2   -5/2   0   1
      10|   1   5/2   -1/2   0   0
      38|   0   23/2   -5/2   1   0
      -20| 0   2   1   0   0

      T -> max = Z(10, 0, 0, 38, 17) = -20
    
```

Рис. 12, 13 Решение задачи из варианта 11