МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)

ИИТУС
БГТУ им. В.Г. Шухова

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №2

по дисциплине: Исследования операций
тема: «Симплекс-метод в чистом виде»

Выполнил: ст. группы ПВ-223
Дмитриев Андрей
Проверил:
Вирченко Ю.П.

Белгород 2024 г.

Цель работы: изучить симплекс-метод для решения задачи линейного программирования с использованием симплекс-таблиц. Получить навыки кодирования изученного алгоритма, отладки и тестирования соответствующих программ.

Задания для подготовки к работе

1. Выяснить: какой вид должна иметь задача ЛП, чтобы можно было применять симплекс-метод в чистом виде, а также как составляется первая симплекс-таблица?

2. Изучить алгоритм перехода от одной симплекс-таблицы к другой при решении задачи симплекс-методом.

3. Запрограммировать и отладить изученный алгоритм. В рамках подготовки тестовых данных решить вручную задачу в соответствии с вариантом.

Вариант 2:

$$z = 6x_2 + 8x_4 + 3x_6 \to \max;$$
$$\begin{cases} x_1 - 4x_2 - 5x_4 - 3x_6 = 9, \\ 7x_2 + 5x_4 + x_5 + 4x_6 = 26, \\ 3x_2 + x_3 - 5x_4 - 4x_6 = 10, \end{cases}$$
$$x_i \geq 0 \left(i = \overline{1,6}\right).$$

Задание 1: Выяснить: какой вид должна иметь задача ЛП, чтобы можно было применять симплекс-метод в чистом виде, а также как составляется первая симплекс-таблица?

Для того чтобы можно было применять симплекс-метод в чистом виде, задача линейного программирования (ЛП) должна быть линейной и включать ограничения в виде неравенств и/или равенств. Задача линейного программирования (ЛП) формулируется следующим образом:

1. Целевая Функция – линейная функция, которую необходимо минимизировать или максимизировать.

2. Ограничения – условия в виде линейных уравнений или неравенств.

Задание №2: Изучить алгоритм перехода от одной симплекс-таблицы к другой при решении задачи симплекс-методом.

Решение:

Алгоритм перехода от одной симплекс-таблицы к другой при решении задачи симплекс-методом включает следующие шаги:

1. Выбор ведущего столбца: Ведущий столбец выбирается на основе коэффициентов целевой функции. Выбирается столбец с наибольшим отрицательным коэффициентом. Если все коэффициенты неотрицательны, то текущее решение является оптимальным.
2. Выбор ведущей строки: Ведущая строка выбирается на основе минимального положительного отношения свободного члена к коэффициенту в ведущем столбце. Если все отношения отрицательны или равны нулю, то задача не имеет ограниченного решения.
3. Пересчёт таблицы: Пересчёт таблицы выполняется с использованием операций над строками для приведения ведущего элемента к единице и всех остальных элементов ведущего столбца к нулю.
4. Повторение процесса: Этот процесс повторяется, пока не произойдёт одно из следующих событий:
    a. Все коэффициенты при переменных в строке целевой функции отрицательны или равны нулю.

b. Все элементы в столбце свободных членов неотрицательные.

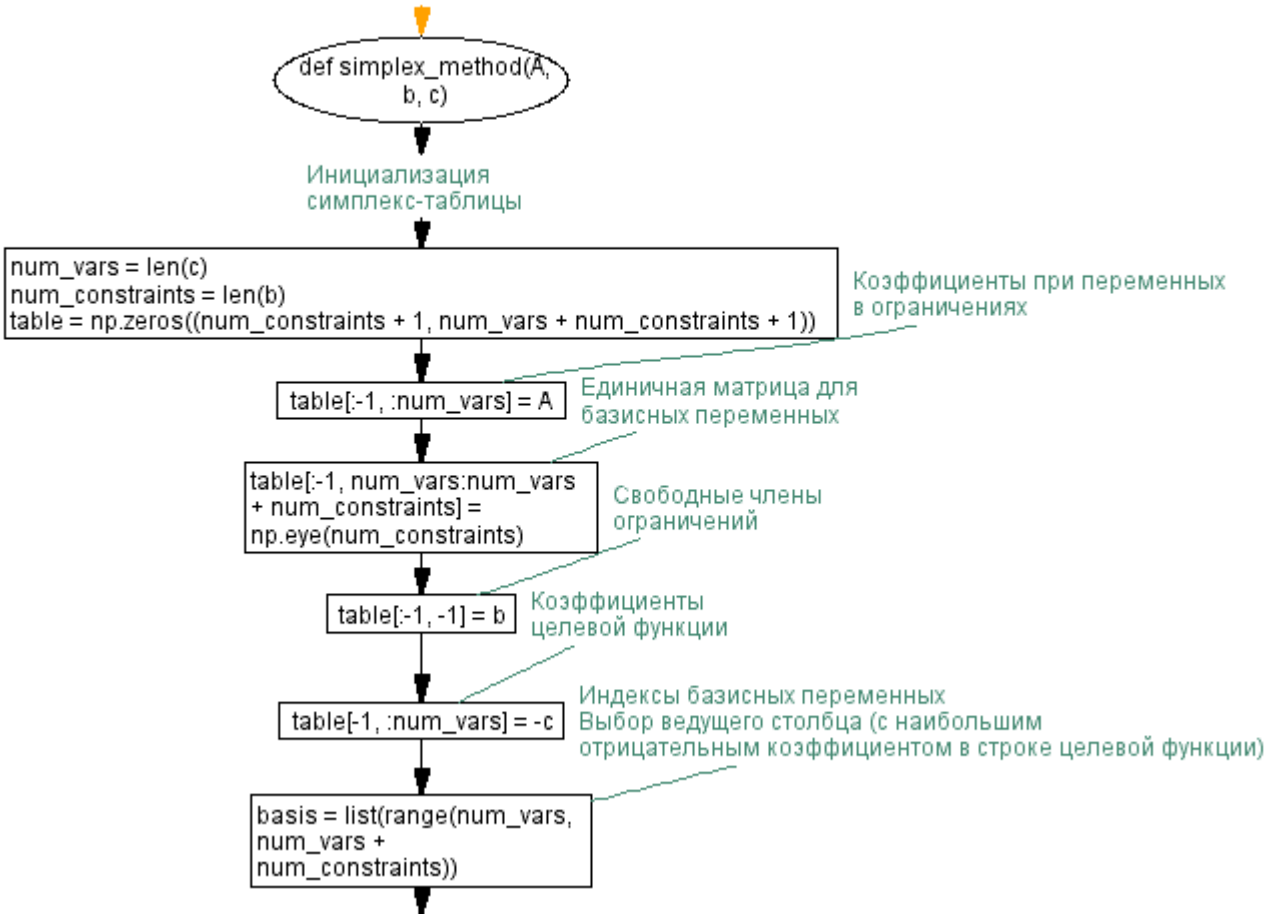
c. В последней строке таблицы целевой функции появляются положительные значения.

d. Нельзя повысить значение целевой функции путем движения к другой базисной точке.

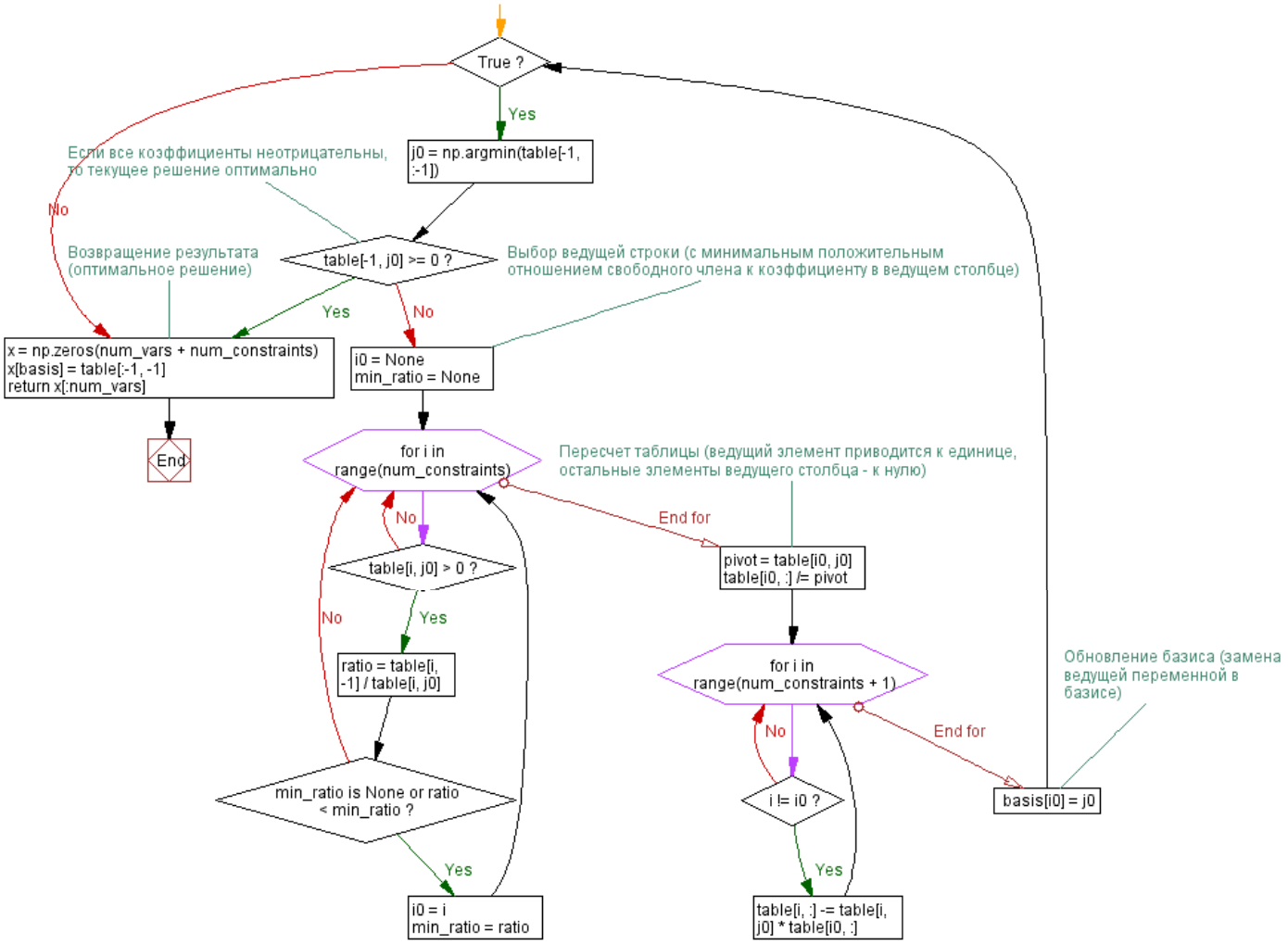
Задание №3: Запрограммировать и отладить изученный алгоритм. В рамках подготовки тестовых данных решить вручную одну из следующих ниже задач.

Реализация

Блок схема (начало, инициализация):



Блок схема (конец, логика алгоритма):



Листинг кода:

```
import numpy as np


def simplex_method(A, b, c):
    # Инициализация симплекс-таблицы
    num_vars = len(c)
    num_constraints = len(b)
    table = np.zeros((num constraints + 1, num vars
                      + num_constraints + 1))
    # Коэффициенты при переменных в ограничениях
    table[:-1, :num_vars] = A
    # Единичная матрица для базисных переменных
    table[:-1, num_vars:num_vars + num_constraints] \
        = np.eye(num_constraints)
    # Свободные члены ограничений
    table[:-1, -1] = b
    # Коэффициенты целевой функции
    table[-1, :num_vars] = -c
    # Индексы базисных переменных
    basis = list(range(num_vars, num_vars + num_constraints))

    while True:
        # Выбор ведущего столбца (с наибольшим
        # отрицательным коэффициентом в строке целевой функции)
        j0 = np.argmin(table[-1, :-1])
        # Если все коэффициенты неотрицательны,
```

```
        # то текущее решение оптимально
        if table[-1, j0] >= 0:
            break
        # Выбор ведущей строки (с минимальным положительным
        # отношением свободного члена к коэффициенту в ведущем столбце)
        i0 = None
        min_ratio = None
        for i in range(num_constraints):
            if table[i, j0] > 0:
                ratio = table[i, -1] / table[i, j0]
                if min_ratio is None or ratio < min_ratio:
                    i0 = i
                    min_ratio = ratio
        # Пересчет таблицы (ведущий элемент приводится к единице,
        # остальные элементы ведущего столбца - к нулю)
        pivot = table[i0, j0]
        table[i0, :] /= pivot
        for i in range(num_constraints + 1):
            if i != i0:
                table[i, :] -= table[i, j0] * table[i0, :]
        # Обновление базиса (замена ведущей переменной в базисе)
        basis[i0] = j0
    # Возвращение результата (оптимальное решение)
    x = np.zeros(num_vars + num_constraints)
    x[basis] = table[:-1, -1]

    return x[:num_vars]
```

Тестовые данные:

```
def main():
    # Пример использования
    A = np.array([[ 1,-4, 0,-5, 0,-3],
                  [ 0, 7, 0, 5, 1, 4],
                  [-0, 3, 1,-5, 0,-4]])
    b = np.array([9, 26, 10])
    c = np.array([0, 6, 0, 8, 0, 3])
    x = simplex_method(A, b, c)
    buf = ""
    for i in range(len(c)):
        if c[i] != 0:
            buf += str(c[i]) + "x_" + str(i + 1) + " + "
    buf += "\b\b-> max"

    print("Целевая функция: ", buf)
    print("Решение: ", x, "; Fmax= ", sum(c * x), sep="")
```

Результат работы программы:

```
C:\Users\dmitr\AppData\Local\Programs\Python\Py
 T:/2kurs2sem/OperRes/lab2/main.py
Целевая функция:  6x_2 + 8x_4 + 3x_6 -> max
Решение: [0.  0.  0.  5.2 0.  0. ]; Fmax= 41.6

Process finished with exit code 0
```

Решение вручную:

Для канонического заданного вида определим таблицу:

Базис	Свободный член	Переменные					
		x_1	x_2	x_3	x_4	x_5	x_6
x_1	9	1	-4	0	-5	0	-3
x_5	26	0	7	0	5	1	4
x_3	10	0	3	1	-5	0	-4
F	0	0	6	0	8	0	3

Инвертируем коэффициенты целевой функции и получим разрешающий элемент, определив новые свободные переменные.

Базис	Свободный член	Переменные					
		x_1	x_2	x_3	x_4	x_5	x_6
x_1	-9/5	1	-4	0	-5	0	-3
x_5	26/5	0	7	0	5	1	4
x_3	-10/5	0	3	1	-5	0	-4
F	0	0	-6	0	-8	0	-3

Проведём замену x_5 на x_4

Базис	Свободный член	Переменные					
		x_1	x_2	x_3	x_4	x_5	x_6
x_1	35	1	3	0	0	1	1
x_4	26	0	7	0	5	1	4
x_3	36	0	10	1	0	1	0
F	0	0	-6	0	-8	0	-3

Базис	Свободный член	Переменные					
		x_1	x_2	x_3	x_4	x_5	x_6
x_1	35	1	3	0	0	1	1
x_4	26/5	0	7/5	0	1	1/5	4/5
x_3	36	0	10	1	0	1	0
F	0-26/5*(-8)	0	-6-7/5*(-8)	0	0	0	1-4/5*(-8)

Получили:

$x_1 = 35$ $x_2 = 0$ $x_3 = 36$ $x_4 = 26/5$ $x_5 = 0$ $x_6 = 0$

$F_{max} = 8*26/5 = 208/5 = 41.6$

Вывод: В ходе лабораторной работы изучили симплекс-метод для решения задачи линейного программирования с использованием симплекс-таблиц. Получили навыки кодирования изученного алгоритма, отладки и тестирования соответствующих программ, как признак – результаты ручного решения и решения программой совпадают.

GitHub: https://github.com/AnDreV133