МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
"МИРЭА - Российский технологический университет"
РТУ МИРЭА

Институт Информационных Технологий
Кафедра Вычислительной Техники

ПРАКТИЧЕСКАЯ РАБОТА №6

по дисциплине
«Теория принятия решений»
Двойственная задача

Студент группы: ИКБО-04-22 __Кликушин В.И.__________
(Ф. И.О. студента)

Преподаватель __Железняк Л.М.__
(Ф.И.О. преподавателя)

Москва 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.. 3
1ДВОЙСТВЕННАЯ ЗАДАЧА 4
1.2 Постановка задачи 4
1.2 Математическая модель исходной задачи... 4
1.3 Соответствующая исходной двойственная задача 5
1.4 Первая теорема двойственности.. 6
1.5 Вторая теорема двойственности... 9
1.6 Третья теорема двойственности 11
ЗАКЛЮЧЕНИЕ 16
СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 17
ПРИЛОЖЕНИЯ.. 18

ВВЕДЕНИЕ

Обычно с задачей линейного программирования (ЗЛП) связана другая линейная задача, называемая двойственной. Тогда первоначальная задача называется исходной или прямой. Математические модели двойственных задач могут быть симметричными или несимметричными. В симметричных задачах система ограничений как исходной, так и двойственной задачи задается неравенствами, причем на двойственные переменные налагается условие не отрицательности. В несимметричных двойственных задачах система ограничений исходной задачи задается в виде равенств, а в двойственной – в виде неравенств, причем в последней переменные могут быть и отрицательными.

1 ДВОЙСТВЕННАЯ ЗАДАЧА

1.1 Постановка задачи

Вариант №13

Задание. Решить прямую ЗЛП с помощью симплексного метода и обратную с помощью теорем двойственности. Определить интервалы устойчивости.

Задача. В кондитерском цехе выпускают печенье двух сортов. В таблице 1 указан расход продуктов для каждого сорта и количество имеющихся продуктов.

Таблица 1.1. Исходные данные задачи.

Сорт	Масло	Яйца	Сахар	Молоко	Цена за 1 кг, ден. ед.
1-й сорт	0,2	0,75	0,15	0,15	1,4
2-й сорт	0,1	0,20	0,20	0,25	0,9
Запасы продуктов	100	150	100	150	

Определить, какое общее количество печенья каждого сорта надо выпекать, чтобы общая стоимость была наибольшей.

1.2 Математическая модель исходной задачи

Пусть x1 – количество печенья первого сорта, x2 – количество печенья второго сорта. Прибыль от продажи печенья составит 1.4x1 + 0.9x2, прибыль требуется максимизировать.

Ограничения задачи:

$$\begin{cases} 0.2x1 + 0.1x2 \leq 100 \\ 0.75x1 + 0.2x2 \leq 150 \\ 0.15x1 + 0.2x2 \leq 100 \\ 0.15x1 + 0.25x2 \leq 150 \end{cases}$$

Таким образом, переходим к задаче линейного программирования:

$$f(x) = 1.4x1 + 0.9x2 \to max$$

$$\begin{cases} 0.2x1 + 0.1x2 \leq 100 \\ 0.75x1 + 0.2x2 \leq 150 \\ 0.15x1 + 0.2x2 \leq 100 \\ 0.15x1 + 0.25x2 \leq 150 \\ x_i \geq 0, i \in [1, 2] \end{cases}$$

1.3 Соответствующая исходной двойственная задача

Найдем соответствующую двойственную задачу. Введем вектор двойственных переменных размерности два $\overline{\text{y}} = (\text{y}_1, \text{y}_2)^{\text{T}}$. Соответствующие векторы и матрица ограничений имеет вид:

$$\overline{\text{c}} = (1.4, 0.9), \overline{b} = (100,150,100,150), A = \begin{pmatrix} 0.2 & 0.1 \\ 0.75 & 0.2 \\ 0.15 & 0.2 \\ 0.15 & 0.25 \end{pmatrix}, A^T = \begin{pmatrix} 0.2 & 0.75 & 0.15 & 0.15 \\ 0.1 & 0.2 & 0.2 & 0.25 \end{pmatrix}.$$

Запишем двойственную задачу. Найти минимум функции.

$$g(\overline{\text{y}}) = (\overline{b}, \overline{\text{y}}) = 100\text{y}_1 + 150\text{y}_2 + 100\text{y}_3 + 150\text{y}_4 \rightarrow min$$

При ограничениях:

$$\begin{pmatrix} 0.2 & 0.75 & 0.15 & 0.15 \\ 0.1 & 0.2 & 0.2 & 0.25 \end{pmatrix} \cdot \begin{pmatrix} \text{y}_1 \\ \text{y}_2 \\ \text{y}_3 \\ \text{y}_4 \end{pmatrix} \geq \begin{pmatrix} 1.4 \\ 0.9 \end{pmatrix}, \text{следовательно}$$

$$\begin{cases} 0.2\text{y}_1 + 0.75\text{y}_2 + 0.15\text{y}_3 + 0.15\text{y}_4 \geq 1.4, \\ 0.1\text{y}_1 + 0.2\text{y}_2 + 0.2\text{y}_3 + 0.25\text{y}_4 \geq 0.9, \\ yi \geq 0, \quad 1 \leq i \leq 4. \end{cases}$$

1.4 Первая теорема двойственности

Если одна из пары двойственных задач имеет оптимальный план, то и другая имеет оптимальный план, причем экстремальные значения целевых функций равны. В ходе решения прямой задачи было определено, что максимальный доход от продажи составляет $f_{max} = 510.417$ тыс. ден. ед., оптимальный план $\overline{\mathrm{x}^*} = (\mathrm{x}1, \mathrm{x}2, \mathrm{x}3, \mathrm{x}4, \mathrm{x}5, \mathrm{x}6) = \left(\frac{250}{3}, 437.5, \frac{475}{12}, 0, 0, \frac{225}{8}\right)$.

Оптимальное решение двойственной задачи может быть получено из оптимального решения прямой задачи. Так как прямая задача имеет решение, то на основании первой теоремы о двойственности задача также разрешима. Ее решение может быть найдено из формулы:

$$\overline{\mathrm{x}^*} = \overline{C_B} \cdot D^{-1},$$

Где D – матрица, составленная из компонентов векторов входящих в последний базис, при котором получен оптимальный план исходной задачи.

В последней симплекс-таблице базисными переменными являются $x3$, $x1$, $x2$, $x6$. Соответствующие этим переменным векторы $\overline{\mathrm{A}_3}, \overline{\mathrm{A}_1}, \overline{\mathrm{A}_2}, \overline{\mathrm{A}_6}$ в разложении используются для формирования столбцов матрицы D.

$$\overline{\mathrm{A}_3} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \overline{\mathrm{A}_1} = \begin{pmatrix} 0.2 \\ 0.75 \\ 0.15 \\ 0.15 \end{pmatrix}, \overline{\mathrm{A}_2} = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.25 \end{pmatrix}, \overline{\mathrm{A}_6} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

Тогда,

$$D = (\overline{\mathrm{A}_3}, \overline{\mathrm{A}_1}, \overline{\mathrm{A}_2}, \overline{\mathrm{A}_6}) = \begin{pmatrix} 1 & 0.2 & 0.1 & 0 \\ 0 & 0.75 & 0.2 & 0 \\ 0 & 0.15 & 0.2 & 0 \\ 0 & 0.15 & 0.25 & 1 \end{pmatrix}$$

Для вычисления обратной матрицы D^{-1} запишем матрицу D дописав к ней справа единичную матрицу.

$$\left(\begin{array}{cccc|cccc} 1 & 0.2 & 0.1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0.75 & 0.2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0.15 & 0.2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0.15 & 0.25 & 1 & 0 & 0 & 0 & 1 \end{array}\right)$$

Для нахождения обратной матрицы D^{-1} используем элементарные преобразования над строками матрицы. Таким образом, преобразуются левая часть полученной матрицы в единичную.

Разделим вторую строку на 0.75;

$$\left(\begin{array}{cccc|cccc} 1 & 0.2 & 0.1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & \frac{4}{15} & 0 & 0 & \frac{4}{3} & 0 & 0 \\ 0 & 0.15 & 0.2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0.15 & 0.25 & 1 & 0 & 0 & 0 & 1 \end{array}\right)$$

от первой строки отнимем вторую строку, умноженную на 0.2; от третьей строки отнимем вторую строку, умноженную на 0.15, от четвертой строки отнимем вторую строку, умноженную на 0.15;

$$\left(\begin{array}{cccc|cccc} 1 & 0 & \frac{7}{150} & 0 & 1 & -\frac{4}{15} & 0 & 0 \\ 0 & 1 & \frac{4}{15} & 0 & 0 & \frac{4}{3} & 0 & 0 \\ 0 & 0 & 0.16 & 0 & 0 & -0.2 & 1 & 0 \\ 0 & 0 & 0.21 & 1 & 0 & -0.2 & 0 & 1 \end{array}\right)$$

разделим третью строку на 0.16;

$$\left(\begin{array}{cccc|cccc} 1 & 0 & \frac{7}{150} & 0 & 1 & -\frac{4}{15} & 0 & 0 \\ 0 & 1 & \frac{4}{15} & 0 & 0 & \frac{4}{3} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1.25 & 6.25 & 0 \\ 0 & 0 & 0.21 & 1 & 0 & -0.2 & 0 & 1 \end{array}\right)$$

от первой строки отнимем третью строку, умноженную на $\frac{7}{150}$; от второй строки отнимем третью, умноженную на $\frac{4}{15}$; от четвертой строки отнимем третью строку, умноженную на 0.21;

$$\left(\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & -\frac{5}{24} & -\frac{7}{24} & 0 \\ 0 & 1 & 0 & 0 & 0 & \frac{5}{3} & -\frac{5}{3} & 0 \\ 0 & 0 & 1 & 0 & 0 & -1.25 & 6.25 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0.0625 & -1.3125 & 1 \end{array}\right)$$

Запишем обратную матрицу.

$$D^{-1} = (y_3^*, y_1^*, y_2^*, y_6^*) = \begin{pmatrix} 1 & -\frac{5}{24} & -\frac{7}{24} & 0 \\ 0 & \frac{5}{3} & -\frac{5}{3} & 0 \\ 0 & -1.25 & 6.25 & 0 \\ 0 & 0.0625 & -1.3125 & 1 \end{pmatrix}$$

Базисными переменными в симплекс-таблице являются $\overline{\mathrm{C_Б}} = (0, 1.4, 0.9, 0)$, тогда

$$\overline{\mathrm{y}^*} = (y_3^*, y_1^*, y_2^*, y_6^*) = \overline{\mathrm{C_Б}} \cdot D^{-1} =$$

$$= \left(0; \left(0 * -\frac{5}{24} + 1.4 * \frac{5}{3} + 0.9 * -1.25 + 0 * 0.0625\right); \left(0 * -\frac{7}{24} + 1.4 * -\frac{5}{3} + 0.9 * 6.25 + 0 * -1.3125\right); 0\right) =$$

$$= (0; \frac{29}{24}; \frac{79}{24}; 0)$$

При этом минимальное значение целевой функции двойственной задачи

$$g_{min} = g\overline{(y^*)} = \left(\bar{b}, \overline{y^*}\right) = 100 * 0 + 150 * \frac{29}{24} + 100 * \frac{79}{24} + 150 * 0 = 510.417 \text{ тыс. ден. ед}$$

совпадает с максимальным значением f_{max} = 510.417 [тыс. ден.ед.] прямой задачи, что является результатом взаимодвойственности. Таким образом,

$$maxf(\bar{x}) = ming(\bar{y}) = 510.417[\text{тыс. ден. ед.}]$$

1.5 Вторая теорема двойственности

Для того, чтобы планы $\overline{x^*} = (x_1^*, x_2^*, \ldots, x_n^*)$ и $\overline{y^*} = (y_1^*, y_2^*, \ldots, y_m^*)$ ЗЛП двойственной пары были оптимальными, необходимо и достаточно, чтобы эти планы удовлетворяли условиям дополняющей нежесткости.

$$\begin{cases} x_j^* \left(\sum_{i=1}^{m} a_{ij} y_i^* - c_j \right) = 0, j = \overline{1, n} \end{cases}$$

$$\begin{cases} y_i^* \left(\sum_{i=1}^{n} a_{ij} x_j^* - b_i \right) = 0, i = \overline{1, m} \end{cases}$$

Итак, имеем оптимальное решение прямой задачи: объем производства печенья первого сорта – $x1 = \frac{250}{3}$; объем производства печенья второго сорта – $x2 = 437.5$; максимальный доход от продажи $f_{max} = 510.417$ [тыс. ден.ед.]. Рассмотрим выполнение неравенств прямой задачи при подстановке $x1$, $x2$ в систему ограничений (Таблица 1.2).

Согласно Таблице 1.2 имеем следующую систему уравнений:

$$\begin{cases} 0.2y1 + 0.75y2 + 0.15y3 + 0.15y4 = 1.4 \\ 0.1y1 + 0.2y2 + 0.2y3 + 0.25y4 = 0.9 \\ y1 = 0 \\ y4 = 0 \end{cases}$$

Решим данную систему уравнений

$$(y1, y2, y3, y4) = (0, \frac{29}{24}, \frac{79}{24}, 0)$$

Решение, найденное из первой теоремы двойственности равнозначно решению из второй теоремы.

$$g(\overline{y^*}) = (\bar{b}, \overline{y^*}) = 100 * 0 + 150 * \frac{29}{24} + 100 * \frac{79}{24} + 150 * 0$$
$$= 510.417 \text{ тыс. ден. ед}$$

$$min\ g(\bar{y}) = 510.417 \text{ [тыс. ден. ед.]}$$

Таким образом, вторая теорема дает нахождение оптимального решения двойственной задачи, пользуясь условием обращения в равенство сопряженных неравенств в системах ограничения.

Таблица 1.2 – Выполнение неравенств прямой задачи

Ограничение	Расчет	Вывод
$0.2x1 + 0.1x2 \leq 100$	$0.2*\frac{250}{3} + 0.1*437.5 < 100$ $60\frac{5}{12} < 100$	Первое ограничение прямой задачи выполняется как строгое неравенство, остается спрос на печенье первого сорта. Значит, этот ресурс не является дефицитным и его оценка в оптимальном плане равна нулю ($y_1 = 0$).
$0.75x1 + 0.2x2 \leq 150$	$0.75*\frac{250}{3} + 0.2*437.5 = 150$ $150=150$	Второе ограничение прямой задачи выполняется как равенство. Это означает, что печенье первого сорта полностью используется в оптимальном плане, является дефицитным и его оценка согласно второй теоремы двойственности отлична от нуля ($y_2 \neq 0$).
$0.15x1 + 0.2x2 \leq 100$	$0.15*\frac{250}{3} + 0.2*437.5 = 100$ $100=100$	Третье ограничение прямой задачи выполняется как равенство. Это означает что печенье второго сорта полностью используется в оптимальном плане, является дефицитным и его оценка согласно второй теоремы двойственности отлична от нуля ($y_3 \neq 0$).
$0.15x1 + 0.25x2 \leq 150$	$0.15*\frac{250}{3} + 0.25*437.5 < 150$ $121\frac{7}{8} < 150$	Четвёртое ограничение прямой задачи выполняется как строгое неравенство, остается спрос на печенье второго сорта. Значит, этот ресурс не является дефицитным и его оценка в оптимальном плане равна нулю ($y4 = 0$).
$x1 \geq 0$	$\frac{250}{3} > 0$	Первое ограничение в двойственной задаче будет равенством $0.2y1 + 0.75y2 + 0.15y3 + 0.15y4 = 1.4$
$x2 \geq 0$	$437.5 > 0$	Второе ограничение в двойственной задаче будет равенством $0.1y1 + 0.2y2 + 0.2y3 + 0.25y4 = 0.9$

1.6 Третья теорема двойственности

Третью теорему двойственности иногда называют теоремой об оценках. Рассматривая ограничения ЗЛП, можно констатировать: изменение правых частей ограничений исходной задачи приводит к изменению максимального значения целевой функции Z_{max}.

Выпишем необходимые элементы из прямой задачи о максимальном доходе. Обратная матрица базиса оптимального плана:

$$D^{-1} = (y_3^*, y_1^*, y_2^*, y_6^*) = \begin{pmatrix} 1 & -\frac{5}{24} & -\frac{7}{24} & 0 \\ 0 & \frac{5}{3} & -\frac{5}{3} & 0 \\ 0 & -1.25 & 6.25 & 0 \\ 0 & 0.0625 & -1.3125 & 1 \end{pmatrix}$$

Индексы базисных переменных оптимального плана:

$$\overline{A_0^*} = (x_3^*, x_1^*, x_2^*, x_6^*) = \begin{pmatrix} \frac{475}{12} \\ \frac{250}{3} \\ 437.5 \\ \frac{225}{8} \end{pmatrix}$$

Свободные члены неравенств (ограничений) прямой задачи:

$$\overline{\mathrm{A}_0} = (b_1, b_2, b_3, b_4) = \begin{pmatrix} 100 \\ 150 \\ 100 \\ 150 \end{pmatrix}$$

Теперь воспользуемся формулами для нахождения нижней и верхней границ интервалов устойчивости оценок по видам ресурсов.

Ресурс 1 (Масло). Найдем нижнюю границу. В четвёртом столбце обратной матрицы один положительный элемент (1), ему соответствует индекс базисной переменной оптимального плана (150).

$$\Delta b_4^H = min\{150/1\} = 150$$

Найдем верхнюю границу. Среди элементов четвёртого столбца отсутствуют отрицательные элементы.

$$\Delta b_4^{\text{В}} = +\infty$$

Таким образом, получаем $\Delta b_1 \in (150; +\infty)$.

Тогда первый ресурс может изменяться в интервале:

$$(b_4 - \Delta b_4^H, b_4 + \Delta b_4^B) = (150 - 150; 150 + \infty) = (0; +\infty)\text{ед.}$$

При таком значении оптимальный план двойственной задачи остается неизменным. Аналогичные рассуждения позволяют найти интервалы устойчивости оценок для остальных ресурсов.

Ресурс 2 (Яйца). Рассматриваем третий столбец обратной матрицы, в котором один положительный элемент (6.25) и три отрицательных ($-\frac{7}{24}$, $-\frac{5}{3}$, -1.3125). Данным элементам соответствуют следующие индексы базисных переменных оптимального плана: положительного элемента – 100; для отрицательных – 100, 150,150.

Тогда находим нижнюю границу.

$$\Delta b_3^H = min\{100/6.25\} = 16$$

Найдем верхнюю границу.

$$\Delta b_3^{\text{В}} = \begin{cases} |max\{100\ x\ (24/7)\}| = \left|\frac{2400}{7}\right| = 342\frac{6}{7} \\ \left|\max\left\{150\ x\ \left(\frac{3}{5}\right)\right\}\right| = |90| = 90 \\ \left|\max\left\{150\ x\ \left(\frac{16}{21}\right)\right\}\right| = \left|\frac{800}{7}\right| = 114\frac{2}{7} \end{cases}$$

Выбираем наибольшее значение, равное $342\frac{6}{7}$.

Получаем $\Delta b_3 \in \left(16; 342\frac{6}{7}\right)$.

Тогда второй ресурс может изменяться в интервале:

$(b_3 - \Delta b_3^H, b_3 + \Delta b_3^B) = \left(100 - 16; 100 + 342\frac{6}{7}\right) == \left(84; 442\frac{6}{7}\right)$ ед.

Ресурс 3 (Сахар). Рассматриваем второй столбец обратной матрицы, в котором два положительных элемента ($\frac{5}{3}$, 0.0625). Данным элементам соответствуют индексы соответствующего базисного переменного оптимального плана – 150, 150.

Находим нижнюю границу.

$$\Delta b_2^H = \begin{cases} |min\{150\ x\ (3/5)\}| = |90| = 90 \\ |\min\{150\ x\ 16\}| = |2400| = 2400 \end{cases}$$

Выбираем наименьшее значение, равное 90.

Найдем верхнюю границу.

$$\Delta b_2^{\text{В}} = \begin{cases} |max\{100\ x\ (24/5)\}| = |480| = 480 \\ |\max\{100\ x\ 0.8\}| = |80| = 80 \end{cases}$$

Выбираем наибольшее значение, равное 480.

Тогда, получаем что $\Delta b_2 \in (90; 480)$.

Получаем интервал устойчивости оценок по отношению к третьему ограничению:

$$(b_2 - \Delta b_2^H, b_2 + \Delta b_2^B) = (150 - 90; 150 + 480) = (60; 630) \text{ ед.}$$

Ресурс 4 (Молоко). Найдем нижнюю границу. В первом столбце обратной матрицы один положительный элемент (1), ему соответствует индекс базисной переменной оптимального плана (100).

$$\Delta b_1^H = min\{100/1\} = 100$$

Найдем верхнюю границу. Среди элементов четвёртого столбца отсутствуют отрицательные элементы.

$$\Delta b_1^{В} = +\infty$$

Таким образом, получаем $\Delta b_1 \in (100; +\infty)$.

Тогда первый ресурс может изменяться в интервале:

$$(b_1 - \Delta b_1^H, b_1 + \Delta b_1^B) = (100 - 100; 100 + \infty) = (0; +\infty)\text{ед.}$$

Далее оценим влияние изменения объема ресурсов на величину максимальной стоимости продукции. Как известно, это дефицитные ресурсы $(y2, y3) = (\frac{29}{24}, \frac{79}{24})$. Введем верхние границы $\Delta b_2^{В}$ и $\Delta b_3^{В}$ в формулу:

$$\Delta G_{max}^i \approx y_i^* \times \Delta b_i$$

$$\Delta G_{max_2} = y_2 \times \Delta b_2^B = \frac{29}{24} \times 342\frac{6}{7} = 414$$

$$\Delta G_{max_3} = y_3 \times \Delta b_3^B = \frac{79}{24} \times 480 = 1580$$

Совместное влияние изменений этих ресурсов приводит к изменению максимальной стоимости продукции G_{max} на величину:

$$\Delta G_{max} = \Delta G_{max_1} + \Delta G_{max_2} = 414 + 1580 = 1994.28$$

Следовательно, оптимальное значение целевой функции при максимальном изменении ресурсов:

$$G_{max} \approx 1994.28 + 510.417 = 2504.70 \text{ [тыс. ден. ед./неделю]}$$

Таким образом, двойственные оценки позволяют судить о чувствительности решения к изменениям.

ЗАКЛЮЧЕНИЕ

Каждая из задач двойственной пары фактически является самостоятельной задачей линейного программирования и может быть решена не зависимо от другой. Связь задач заключается в том, что решение одной из них может быть получено непосредственно из решения другой. Взаимная симметрия прямой и двойственной задач определяет существование определенного соответствия между их оптимальными решениями. Эти соответствия устанавливают теоремы двойственности.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Болотова Л. С. Многокритериальная оптимизация. Болотова Л. С., Сорокин А. Б. [Электронный ресурс] / Метод. указания по вып. курсовой работы — М.: МИРЭА, 2015.

2. Сорокин А. Б. Методы оптимизации: гибридные генетические алгоритмы. Сорокин А. Б. [Электронный ресурс] / Учебно-метод. пособие — М.: МИРЭА, 2016.

3. Сорокин А. Б. Линейное программирование: практикум. Сорокин А. Б., Бражникова Е. В., Платонова О. В. [Электронный ресурс] / Учебно-метод. пособие — М.: МИРЭА, 2017.

ПРИЛОЖЕНИЯ

Приложение А – Код реализации двойственной задачи на языке Python.

Приложение А

Код реализации двойственной задачи на языке Python

Листинг А.1. Реализация двойственной задачи.

```
import re
import math
import sympy
import numpy as np


NUM_CRITERIA = 4  # Количество ограничений в математической моделе
PRECISION = 8  # Количество знаков после запятой при округлении
SEP = 25  # Разделитель для вывода таблицы


def print_table(system, coef_basis, coef_not_basis, basis_values,
not_basis_values):
    print(' '.ljust(SEP), end='')
    print('Cj'.ljust(SEP), end='')
    for i in range(len(coef_not_basis) + 1):
        if i == len(coef_not_basis):
            print(' '.ljust(SEP))
        else:
            print(str(coef_not_basis[i]).ljust(SEP), end='')
    print('Cv'.ljust(SEP), end='')
    for i in range(len(not_basis_values) + 1):
        if i == 0:
            print(''.ljust(SEP), end='')
        else:
            print(str(not_basis_values[i-1]).ljust(SEP), end='')
    print('A0'.ljust(SEP))
    system.insert(0, coef_basis + [' '])
    system.insert(1, basis_values + ['f'])
    for column in range(len(system[0])):
        for row in range(len(system)):
            print(str(system[row][column]).ljust(SEP), end='')
        print()
    del system[0]
    del system[0]


def get_coefficients(data):
    '''Функция для получения списка коэффициентов из системы ограничений'''
    criteria_coefficients, boundaries = list(), list()
    for exp in data:
        if '<=' in exp:
            parse = exp.split('<=')
        elif '>=' in exp:
            parse = exp.split('>=')
        elif '<' in exp:
            parse = exp.split('<')
        elif '>' in exp:
            parse = exp.split('>')
        elif '=' in exp:
            parse = exp.split('=')
        parse = list(map(str.strip, parse))
        boundaries.append(float(parse[1]))
        criteria_coefficients.append(list(map(float, [i.group(1) for i in
re.finditer(
            r'(\d+(\.\d+)?) {0,}[*]? {0,}\w', parse[0])])))
```

Продолжение Листинга А.1.

```
    return criteria_coefficients, boundaries


def count_scalar_product(vec1, vec2):
    '''Функция для рассчёта скалярного произведения двух векторов'''
    res = 0
    for i in range(len(vec1)):
        res += (vec1[i] * vec2[i])
    return res


def create_simplex_table(system, coef_basis, coef_not_basis, basis_values,
not_basis_values):
    F_str = [0] * len(not_basis_values)
    for i in range(len(not_basis_values)):
        F_str[i] = count_scalar_product(
            coef_basis, system[i]) - coef_not_basis[i]
    Q = count_scalar_product(coef_basis, system[-1])
    for i in range(len(F_str)):
        system[i].append(F_str[i])
    system[-1].append(Q)
    return F_str, Q


def simplex_iteration(system, coef_basis, coef_not_basis, basis_values,
not_basis_values, F_str, Q):
    index_column = F_str.index(min(F_str))
    mini = 1e10
    for i in range(len(system[-1]) - 1):
        tmp = system[-1][i] / system[index_column][i]
        if tmp < mini:
            mini = tmp
            index_row = i
    key_element = system[index_column][index_row]
    basis_values.insert(index_row, not_basis_values[index_column])
    not_basis_values.insert(index_column, basis_values.pop(index_row + 1))
    del not_basis_values[index_row]
    coef_basis[index_row], coef_not_basis[index_column] =
coef_not_basis[index_column], coef_basis[index_row]
    new_key_element = round(1 / key_element, PRECISION)
    data = [[0] * (len(coef_basis) + 1)
            for _ in range(len(coef_not_basis) + 1)]
    for i in range(len(system[index_column])):
        data[index_column][i] = - \
            round(system[index_column][i] / key_element, PRECISION)
    for i in range(len(system)):
        data[i][index_row] = round(
            system[i][index_row] / key_element, PRECISION)
    data[index_column][index_row] = new_key_element
    for row in range(len(data[0])):
        for column in range(len(data)):
            if data[column][row] == 0:
                data[column][row] = round(((system[column][row] * key_element) -
(
                    system[index_column][row] * system[column][index_row])) /
key_element, PRECISION)
    F_str = [data[i][-1] for i in range(len(data) - 1)]
    Q = data[-1][-1]
    return data, coef_basis, coef_not_basis, basis_values, not_basis_values,
F_str, Q
```

Продолжение Листинга А.1.

```
def check_inequality(inequality, variables,
индексы_базисных_переменных_оптимального_плана):
    inequality = inequality.replace('*', '')
    for i in range(len(variables)):
        if variables[i] not in inequality:
            continue
        inequality = inequality.replace(
            variables[i], '*' +
str(индексы_базисных_переменных_оптимального_плана[i]))
    # Символьное вычисление неравенства
    inequality += '- 0.1'
    result = str(sympy.sympify(inequality))
    # Возвращение True, если неравенство выполняется, иначе False
    return eval(result)


def dual_task():
    коэффициенты_целевой_функции = np.array(target_coefficients)
    свободные_члены_неравенств = np.array(boundaries)
    матрица_ограничений, _ = get_coefficients(criteria_function)
    транспонированная_матрица_ограничений = np.transpose(матрица_ограничений)
    индексы_базисных_переменных_оптимального_плана = np.array(system[-1][:-1])
    y = np.array([])
    D = list()
    for i in range(len(basis_values)):
        index = int(basis_values[i][1:]) - 1
        if index < len(транспонированная_матрица_ограничений):
            D.append(транспонированная_матрица_ограничений[index])
        else:
            D.append(
                np.array([1 if i == j else 0 for j in range(NUM_CRITERIA)]))
    D_inversed = np.linalg.inv(np.transpose(D))

    def first_duality_theorem():
        y = np.dot(np.array(coef_basis), D_inversed)
        G_min = np.dot(свободные_члены_неравенств, y)
        print(f"Gmin is {G_min} by first_duality_theorem")
        assert abs(G_min - Q) < 0.00001

    def second_duality_theorem():
        nonlocal y
        zeros = list()
        for i in range(NUM_CRITERIA):
            if check_inequality(
                    criteria_function[i], basis_values,
индексы_базисных_переменных_оптимального_плана):
                zeros.append(i)
        система_уравнений = транспонированная_матрица_ограничений.copy()
        for i in range(len(zeros)):
            система_уравнений = np.delete(
                система_уравнений, zeros[i], 1)
            for j in range(i + 1, len(zeros)):
                zeros[j] -= 1
        y = np.linalg.solve(система_уравнений, коэффициенты_целевой_функции)
        for i in range(len(zeros)):
            y = np.insert(y, zeros[i], 0)
            for j in range(i + 1, len(zeros)):
                zeros[j] += 1
        G_min = np.dot(свободные_члены_неравенств, y)
```

Продолжение Листинга А.1.

```
        print(f"Gmin is {G_min} by second_duality_theorem")
        assert abs(G_min - Q) < 0.00001

    def third_duality_theorem():
        нижняя_граница = list()
        верхняя_граница = list()
        b = list()
        for i in range(len(D_inversed) - 1, -1, -1):
            pozitive = list()
            negative = list()
            bH = - math.inf
            bB = math.inf
            for j in range(len(D_inversed)):
                if D_inversed[j][i] > 0:
                    pozitive.append(
                        (свободные_члены_неравенств[j], D_inversed[j][i]))
                elif D_inversed[j][i] < 0:
                    negative.append(
                        (свободные_члены_неравенств[j], D_inversed[j][i]))
            if len(pozitive) > 1:
                elem = min(pozitive, key=lambda x: abs(
                    pozitive[0][0] / pozitive[0][1]))
                нижняя_граница.append(elem[0] / elem[1])
            elif len(pozitive) == 1:
                нижняя_граница.append(pozitive[0][0] / pozitive[0][1])
            else:
                нижняя_граница.append(bH)

            if len(negative) > 1:
                elem = max(negative, key=lambda x: abs(
                    negative[0][0] / negative[0][1]))
                верхняя_граница.append(abs(elem[0] / elem[1]))
            elif len(negative) == 1:
                верхняя_граница.append(negative[0][0] / negative[0][1])
            else:
                верхняя_граница.append(bB)

            b.append(свободные_члены_неравенств[i])
            print(f'Ресурс №{len(D_inversed)-i}')
            print(
                f'b{len(D_inversed)-i} ∈ ({нижняя_граница[-1]};
{верхняя_граница[-1]})')
            print(f'{len(D_inversed)-i}-й ресурс изменяется в интервале: ',
end='')
            if нижняя_граница[-1] == - math.inf:
                print(f'({нижняя_граница[-1]}; ', end='')
            else:
                print(f'({b[-1] - нижняя_граница[-1]}; ', end='')
            if верхняя_граница[-1] == math.inf:
                print(f'{верхняя_граница[-1]})')
            else:
                print(f'{b[-1] + верхняя_граница[-1]})')
        total = 0
        for i in range(len(y)):
            if y[i] != 0:
                total += y[i] * верхняя_граница[i]
                print(f'ΔGmax{i + 1} = y{i+1} * bB{i +
                      1} = {y[i] * верхняя_граница[i]}')
        print(f'Совместное влияние изменений этих ресурсов приводит к изменению
максимальной стоимости продукции Gmax на величину: {total}')
        print(f'Следовательно, оптимальное значение целевой функции при
```

Продолжение Листинга А.1.

```
 максимальном изменении ресурсов: {Q+total}')

    first_duality_theorem()
    second_duality_theorem()
    third_duality_theorem()


with open('TPR_PRACT5.csv', encoding='utf-8') as file:
    target_function = file.readline().rstrip()  # Целевая функция
    target_coefficients = list(map(float, [i.group(1) for i in re.finditer(
        # Список коэффициентов целевой функции
        r'(\d+(\.\d+)?) {0,}[*]? {0,}\w', target_function)]))
    criteria_function = [file.readline().rstrip() for _ in range(NUM_CRITERIA)]
    criteria_coefficients, boundaries = get_coefficients(criteria_function)
    print('Переходим к задаче линейного программирования:',
          target_function, sep='\n')
    for i in criteria_function:
        print("{ " + i)
    system = list(map(list, list(zip(*criteria_coefficients))))
    system.append(boundaries.copy())
    # Вектор коэффициентов целевой функции при базисных переменных
    coef_basis = [0] * NUM_CRITERIA
    # Коэффициенты целевой функции, соответствующие небазисным переменным
    coef_not_basis = target_coefficients.copy()
    not_basis_values = re.findall(r'[A-Za-z]\d{1,}', target_function)
    basis_values = [f'{not_basis_values[-1][0]}{i}' for i in range(
        int(not_basis_values[-1][1]) + 1, NUM_CRITERIA + int(not_basis_values[-
1][1]) + 1)]
    F_str, Q = create_simplex_table(
        system, coef_basis, coef_not_basis, basis_values, not_basis_values)
    num_iteration = 0
    while num_iteration < 50 and min(F_str) < 0:
        print(
            ('\x1b[6;30;42m' + f"Итерация №{num_iteration}" +
'\x1b[0m').center(201))
        system, coef_basis, coef_not_basis, basis_values, not_basis_values,
F_str, Q = simplex_iteration(
            system, coef_basis, coef_not_basis, basis_values, not_basis_values,
F_str, Q)
        print_table(system, coef_basis, coef_not_basis,
                    basis_values, not_basis_values)
        num_iteration += 1
    if num_iteration != 50:
        print(f'Решение найдено! Общая прибыль составила {
              round(Q, 3)} денежных единиц')
    else:
        print('Поставленная задача решения не имеет')
        exit(0)
    dual_task()
```