Министерство Образования Республики Беларусь

Учреждение образования

«Гомельский государственный технический университет

имени П. О. Сухого»

Кафедра «Информатика»

**Лабораторная работа № 1**

по дисциплине: **«Системный анализ и исследование операций»**

Выполнил студент

группы ИП-42

*Суховенко Э. С.*

Проверил преподаватель

*Бородин Н.Н.*

**ЛАБОРАТОРНАЯ РАБОТА №1**

**ПРОГРАММНАЯ РЕАЛИЗАЦИЯ РЕШЕНИЯ ЗАДАЧ ЦЕЛОЧИСЛЕННОГО ПРОГРАММИРОВАНИЯ**

**Цель работы:** получить практические навыки решения задач целочисленного программирования.

**Задание**

1. Решить задачи целочисленного программирования (варианты см. в таблице 1.1); проверить правильность полученных решений средствами пакетов *Mathcad* или *MS Excel*.

2) Решить (см. общую постановку ЗЦП и таблицу вариантов ниже):

а) частично целочисленную задачу ( – любое, – целочисленное);

б) полностью целочисленную задачу ( и  – целочисленные)

средствами пакетов Mathcad (графическим методом и с помощью блока решения) и MS Excel (с помощью надстройки «Поиск решения»). Сравнить полученные результаты.









, 

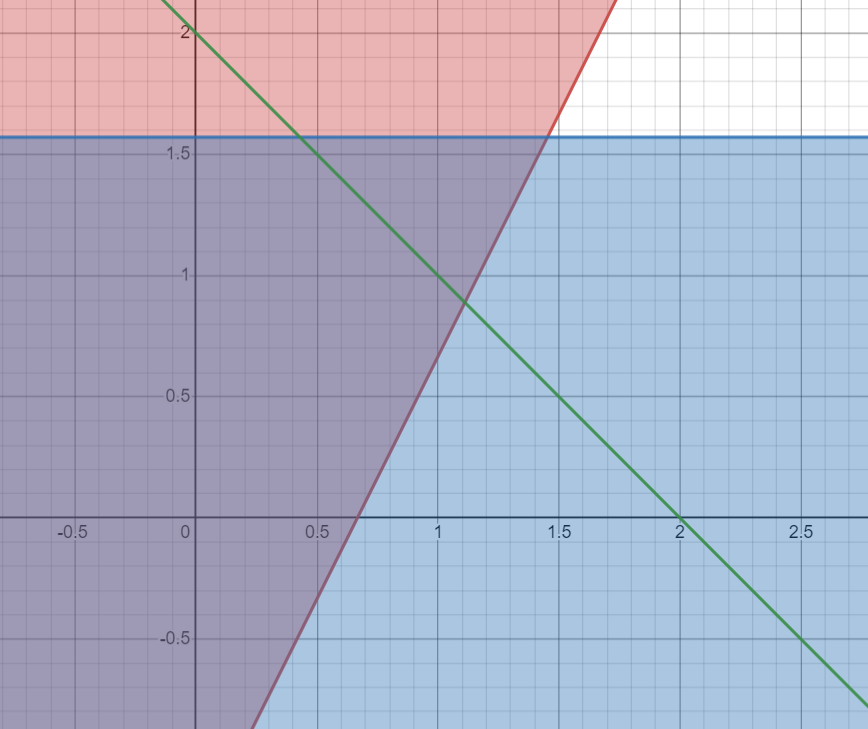
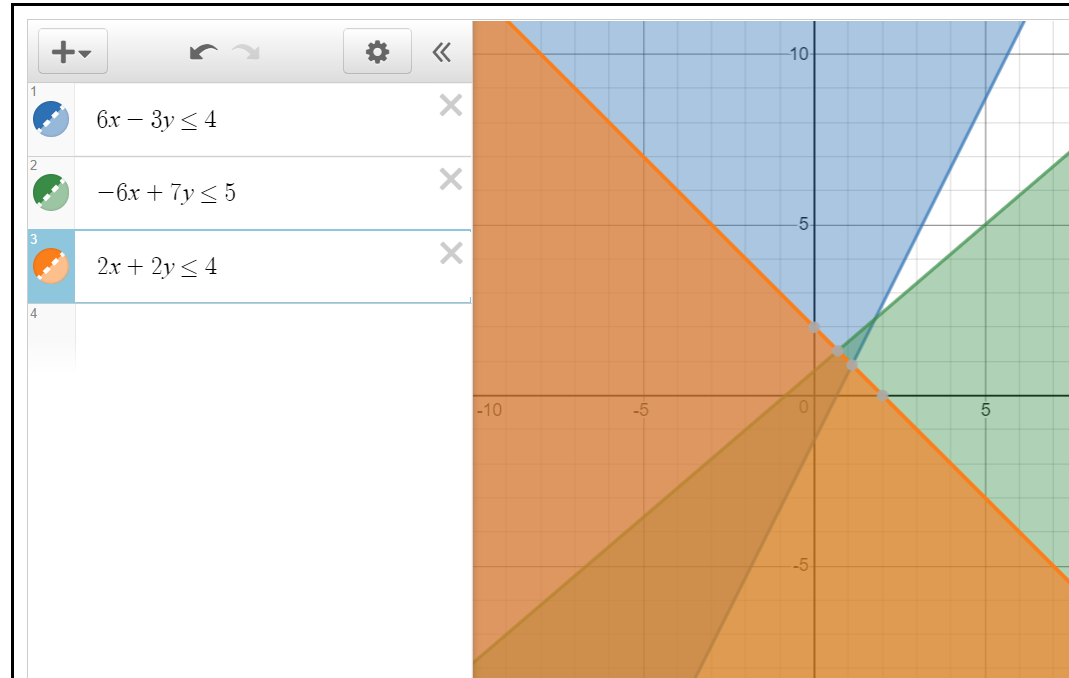
**Вариант задания**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Вариант** | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **p** | **r** | **q** |
| **1** | 6 | 5 | 6 | -3 | 4 | -6 | 7 | 5 | 2 | 2 | 4 |

Система ограничений:

Целевая функция:

График:



Из графика видно, что x1, x2 будут равны 1, 1. Значит F = 6 + 5 = 11.

**Программная реализация:**

import numpy as np

import matplotlib

matplotlib.use('TkAgg')

import matplotlib.pyplot as plt

from pulp import LpMaximize, LpProblem, LpStatus, lpSum, LpVariable

def show\_point(model) -> None:

x = [data\_point.value() for data\_point in model.variables()]

print(f'x1 = {x[0]}\n' + f'x2 = {x[1]}')

plt.scatter(x[0], x[1], color="red")

def show\_info(model) -> None:

print(f"status: {model.status}, {LpStatus[model.status]}")

print(f"objective: {model.objective.value()}")

for name, constraint in model.constraints.items():

print(f"{name}: {constraint.value()}")

def create\_model(\*args: tuple):

a, b, c, d, e, f, g, h, p, r, q = args

model = LpProblem(name="small-problem", sense=LpMaximize)

x\_1 = LpVariable(name="x1", lowBound=0)

x\_2 = LpVariable(name="x2", lowBound=0, cat="Integer")

model += (c \* x\_1 + d \* x\_2 <= e, "first\_constraint")

model += (f \* x\_1 + g \* x\_2 <= h, "second\_constraint")

model += (p \* x\_1 + r \* x\_2 <= q, "third\_constraint")

model += lpSum([a \* x\_1, b \* x\_2])

status = model.solve()

return model

def create\_landscape(\*args):

a, b, c, d, e, f, g, h, p, r, q, x = args

y2 = (e - c \* x) / d

y3 = (h - f \* x) / g

y4 = (q - p \* x) / r

y = [y2, y3, y4]

for y\_variable in y:

plt.plot(x, y\_variable)

plt.plot(x, y\_variable)

plt.plot(x, y\_variable)

return y

def work(args: tuple):

a, b, c, d, e, f, g, h, p, r, q = args

x = np.linspace(0, a, 2000)

y = create\_landscape(\*args, x)

plt.xlim((0, a))

plt.xlim((0, a))

y5 = np.minimum(y[0], y[2])

y6 = y[1]

plt.fill\_between(x, y5, y6, where=y5 > y6, color='grey', alpha=0.5)

model = create\_model(\*args)

show\_point(model)

show\_info(model)

plt.grid()

plt.show()

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

coeffs = (6, 5, 6, -3, 4, -6, 7, 5, 2, 2, 4)

work(coeffs)



Рисунок 2 — график системы в python

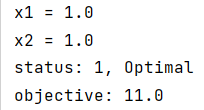


Рисунок 3 — максимум функции

**Вывод:** были получены практические навыки решения задач целочисленного программирования.