

**Федеральное государственное образовательное
бюджетное учреждение
высшего образования**

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ
ФЕДЕРАЦИИ»**

(Финансовый университет)

**Факультет
информационных технологий и анализа больших данных
Кафедра «Бизнес-информатика»**

Домашнее задание № 1

«Задачи линейного программирования: транспортная задача.»

Студенты группы БИ20-8:

Луканина Полина

Аверкин Никита

Филимонова Арина

Совин Владимир

Горшков Георгий

Киселева Евгения

Руководитель:

Аксенов Дмитрий

Андреевич

Москва 2022

Оглавление

Оглавление	2
1. Постановка задачи (физическая модель)	5
2. Математическая модель.....	6
3. Алгоритмы.....	7
3.1. Алгоритм 1	7
3.1.1. Описание входных данных	7
3.1.2. Описание алгоритма решения	7
3.1.3. Описание выходных данных.....	10
3.2. Алгоритм 2	10
3.2.1. Описание входных данных	11
3.2.2. Описание алгоритма решения	12
3.2.3. Описание выходных данных.....	15
3.3. Алгоритм 3	15
3.3.1. Описание входных данных	16
3.3.2. Описание алгоритма решения	16
3.3.3. Описание выходных данных.....	18
4. Варианты использования системы.....	19
4.1. ВИ 1	19
4.2. ВИ 2	20
4.3. ВИ 3	20
5. Архитектура решения	22
5.1. Методы считывания информации	22
5.2. Методы обработки информации.....	23

5.3. Методы вывода информации	24
6. Тестирование	25
6.1. Результаты тестирования первого дата сета:.....	26
6.1.1. Проверка кода Python и визуализации:	26
6.1.2. Проверка кода методом Excel:.....	26
6.1.3. Проверка кода в онлайн-калькуляторе:.....	27
6.2. Результаты тестирования второго дата сета:	27
6.2.1. Проверка кода Python и визуализация:	27
6.2.2. Проверка кода методом Excel:.....	28
6.2.3. Проверка кода в онлайн-калькуляторе:.....	28
6.3. Результаты тестирования третьего дата сета:.....	29
6.3.1. Проверка кода Python и визуализация:	29
6.3.2. Проверка кода методом Excel:.....	29
6.3.3. Проверка кода в онлайн-калькуляторе:.....	30
6.4. Результаты тестирования четвёртого дата сета:.....	30
6.4.1. Проверка кода Python и визуализации:	30
6.4.2. Проверка кода методом Excel:.....	31
6.4.3. Проверка кода в онлайн-калькуляторе:.....	31
6.5. Результаты тестирования пятого дата сета:.....	32
6.5.1. Проверка кода Python и визуализации:	32
6.5.2. Проверка кода методом Excel:.....	33
6.5.3. Проверка кода в онлайн-калькуляторе:.....	33
6.6. Результаты тестирования шестого датасета:	34
6.6.1. Проверка кода Python:.....	34

6.6.2. Проверка кода методом Excel:.....	35
6.6.3. Проверка кода в онлайн-калькуляторе:.....	35
6.7. Результаты тестирования седьмого дата сета:.....	36
6.7.1. Проверка кода Python:.....	36
6.7.2. Проверка кода методом Excel:.....	36
6.7.3. Проверка кода в онлайн-калькуляторе:.....	37
7. Заключение.....	39

1. Постановка задачи (физическая модель)

Имеется сеть железных дорог, на которой расположены 3 пункта отправления вагонов с углем и 4 станции приема через которые уголь попадает на ТЭЦ. Известны затраты на перевозку одного вагона угля от i -ой до j -ой станции. Так же известны объемы угля, которые хранятся в каждом пункте отправления и объемы, которые может вместить в себя каждая из станций приема. Необходимо составить оптимальный план доставки угля из пункта А пункт В с минимальными затратами на перевозку.

Объемы запасов угля в пунктах отправления (в кол-ве вагонов): 300, 250 и 200

Объемы угля, которые могут в себя вместить пункты приема (в кол-ве вагонов): 220, 150, 250 и 180 соответственно.

Стоимость перевозки одного вагона угля с i станции до хранилища j указаны в правых верхних углах соответствующих клеток транспортной матрице.

заказы запасы		B_1		B_2		B_3		B_4	
		220		150		250		180	
A_1	300		400		500		300		600
A_2	250		700		200		100		500
A_3	200		600		100		400		200

Транспортная матрица 1

2. Математическая модель

Далее для того, чтобы приступить к дальнейшему решению транспортной задачи, проверим условие на сбалансированность:

Поскольку $\sum a_i = 300 + 250 + 200 = 750 \neq \sum b_j = 220 + 150 + 250 + 180 = 800$, то тип транспортной задачи является открытым, что свидетельствует нам о том, что придется сводить ее к закрытой путем добавления фиктивного дополнительного условия (дополнительный пункт приема или отправления).

Искомые переменные будут располагаться в матрице X_{ij} – объем поставляемого угля, перевозимого с i -ой станции в j -ое хранилище:

$$\begin{pmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{pmatrix}$$

Теперь запишем и найдем целевую функцию:

$$W_{\min} = \sum_{i=1}^3 \sum_{j=1}^4 c_{ij} x_{ij}$$

При следующих ограничениях:

$$1. \sum_{j=1}^n x_{ij} \leq \sum b_j;$$

$$2. \sum_{i=1}^m x_{ij} = \sum a_i.$$

Таким образом, получим оптимальный план перевозок X_{ij} , где целевая функция W_{min} – минимальный затраты на перевозку угля, при выполнении всех поставленных условий.

3. Алгоритмы

У нас есть 3 алгоритма решения: 1 алгоритм реализован в python, 2 алгоритм реализован в python- визуализация и 3 алгоритм решения с помощью Excel.

3.1. Алгоритм 1

Решение задачи с помощью python. Запускаем программу и выбираем способ ввода исходных данных.

3.1.1. Описание входных данных

Для ручного ввода нужно ввести "1",

Для заполнения матрицы значений случайными числами вводим "2"

Для ввода значений с помощью файла csv вводим "3".

Но в данном примере мы рассмотрим только первый способ ввода данных.

На рисунке 1 мы видим строчку с «`way = input...`», которая позволяет нам выбрать тип вводимых данных. В данном случае мы выбрали ручной ввод данных (1 на клавиатуре). С помощью второго абзаца рисунка 1 мы сможем сформировать размер матрицы (кол-во столбцов и строк в ней). В третьей части рисунка (с момента использования функции `matrix`) мы задали цикл для последовательного ввода значений и заполнения матрицы исходных данных.

```

way = input('Каким способом Вы хотите ввести значения? ') # Пользователь выбирает способ ввода данных
if way == '1': # Ручной ввод данных

    # Вводим количество строк и столбцов матрицы исходных значений
    size_rows = int(input("Количество строк (пунктов отправления): "))
    size_cols = int(input("Количество столбцов (пунктов назначения): "))

    # Цикл для последовательного ввода значений и заполнения ими матрицы исходных данных
    matrix = []
    for i in range(size_rows):
        rows = []
        for j in range(size_cols):
            x = input('Введите значение: ')
            rows.append(float(x))
        matrix.append(rows)

```

Рисунок 1

Следующим этапом мы вводим в задачу потребности и производительность и обозначаем их как max_company_1 и max_factory_2. (см. Рисунок 2)

Разбиваем строку потребностей на список и переводим значения в int. То же самое делаем и с производительностью.

```

# Вводим потребности и производительность
max_company_1 = input('Потребности (ввод через запятую): ')
max_factory_2 = input('Производительность (ввод через запятую): ')

# Разбиваем строку с введенными потребностями на список
max_company, max_factory = [], []
max_company_1_2 = max_company_1.split(',')
# Переводим значения в полученном списке в тип int
for i in max_company_1_2:
    max_company.append(int(i))

# Разбиваем строку с введенными значениями производительности на список
max_factory_2_2 = max_factory_2.split(',')
# Переводим значения в полученном списке в тип int
for i in max_factory_2_2:
    max_factory.append(int(i))

```

Рисунок 2

Остальные способы ввода информации сделаны аналогично данному, поэтому описывать их не имеет смысла.

3.1.2. Описание алгоритма решения

В разделе описывается последовательность действий, совершаемых алгоритмом для получения целевой функции и искомую матрицу.


```

# Создаем двумерный список из ограничений на потребности (для ограничений в функции linprog)
# 'A_eq' в функции linprog – это матрица ограничений равенства. Каждая строка A_eq задает коэф
sp_Aeq_2 = []
sp_Aeq = []
k1 = 0
for i in range(size_cols):
    for j in range(size_rows):
        for k in range(size_cols):
            if k == i:
                sp_Aeq_2.append(1)
                k += 1
            else:
                sp_Aeq_2.append(0)
        k -= 1
    sp_Aeq.append(sp_Aeq_2)
    sp_Aeq_2 = []

```

```

# Создаем двумерный список из ограничений на производительность (для ограничений в функции linprog)
# 'A_ub' в функции linprog – это матрица ограничений неравенства. Каждая строка A_ub задает коэффиц
sp_Aub_2 = []
sp_Aub = []
k1 = 0
for i in range(size_rows):
    for j in range(size_rows):
        n = 1
        while n <= size_cols:
            if j == k1:
                sp_Aub_2.append(1)
            else:
                sp_Aub_2.append(0)
            n += 1
        k1 += 1
    sp_Aub.append(sp_Aub_2)
    sp_Aub_2 = []

```

Скриншот 1 и 2

На скриншотах выше показан скрипт действий по созданию матрицы с ограничениями на производительность и потребность для передачи данных в функцию `linprog`.

```

# Вбиваем в функцию linprog переменные, она находит оптимальное значение целевой функции
function = round(linprog(c, A_ub, b_ub, A_eq, b_eq).fun, 3)
print('Целевая функция', function) # Выводим на экран значение ЦелФун
array1 = linprog(c, A_ub, b_ub, A_eq, b_eq).x # Присваиваем переменной найденные оптимальные значения
array0 = np.array([round(i, 1) for i in array1]) # И округляем все значения этого списка

# Преобразуем список полученных значений в матрицу
array = []
array_sm = []
l = 0
for i in range(size_rows):
    for j in range(size_cols):
        array_sm.append(array0[l])
        l += 1
    array.append(array_sm)
    array_sm = []

```

Скриншот 3

С помощью функции `linprog` мы вычислили целевую функцию, которую мы перевели в искомую матрицу.

3.1.3. Описание выходных данных

Пользователь на выходе получает следующую информацию: целевая функция в обобщённом виде и искомая матрица.

```

x1*976.0 + x2*829.0 + x3*470.0 + x4*623.0 + x5*611.0 + x6*349.0 + x7*795.0 + x8*183.0 + x9*192.0
Целевая функция 725980.948
      0      1      2
0    0.0    0.0    0.0
1  994.0    0.0    0.0
2    0.0  345.0  227.0
Визуализация недоступна. Введите матрицу 2x1 или 1x2

```

Скриншот 4

3.2. Алгоритм 2

Второй алгоритм — это алгоритм визуализации и начало кода полностью идентично первому алгоритму.

3.2.1. Описание входных данных

Для ручного ввода нужно ввести "1",

Для заполнения матрицы значений случайными числами вводим "2"

Для ввода значений с помощью файла csv вводим "3".

Но в данном примере мы рассмотрим только первый способ ввода данных.

На рисунке 1 мы видим строчку с «way = input...», которая позволяет нам выбрать тип вводимых данных. В данном случае мы выбрали ручной ввод данных (1 на клавиатуре). С помощью второго абзаца рисунка 1 мы сможем сформировать размер матрицы (кол-во столбцов и строк в ней). В третьей части рисунка (с момента использования функции matrix) мы задали цикл для последовательного ввода значений и заполнения матрицы исходных данных.

```
way = input('Каким способом Вы хотите ввести значения? ') # Пользователь выбирает способ ввода данных
if way == '1': # Ручной ввод данных

    # Вводим количество строк и столбцов матрицы исходных значений
    size_rows = int(input("Количество строк (пунктов отправления): "))
    size_cols = int(input("Количество столбцов (пунктов назначения): "))

    # Цикл для последовательного ввода значений и заполнения ими матрицы исходных данных
    matrix = []
    for i in range(size_rows):
        rows = []
        for j in range(size_cols):
            x = input('Введите значение: ')
            rows.append(float(x))
        matrix.append(rows)
```

Рисунок 3

Следующим этапом мы вводим в задачу потребности и производительность и обозначаем их как max_company_1 и max_factory_2. (см. Рисунок 2)

Разбиваем строку потребностей на список и переводим значения в int. То же самое делаем и с производительностью.

```

# Вводим потребности и производительность
max_company_1 = input('Потребности (ввод через запятую): ')
max_factory_2 = input('Производительность (ввод через запятую): ')

# Разбиваем строку с введенными потребностями на список
max_company, max_factory = [], []
max_company_1_2 = max_company_1.split(',')
# Переводим значения в полученном списке в тип int
for i in max_company_1_2:
    max_company.append(int(i))

# Разбиваем строку с введенными значениями производительности на список
max_factory_2_2 = max_factory_2.split(',')
# Переводим значения в полученном списке в тип int
for i in max_factory_2_2:
    max_factory.append(int(i))

```

Рисунок 4

Остальные способы ввода информации сделаны аналогично данному, поэтому описывать их не имеет смысла.

3.2.2. Описание алгоритма решения

В разделе описывается последовательность действий, совершаемых алгоритмом для получения целевой функции и искомую матрицу.

```

# Создаем двумерный список из ограничений на потребности (для ограничений в функции linprog)
# 'A_eq' в функции linprog – это матрица ограничений равенства. Каждая строка A_eq задает коэф
sp_Aeq_2 = []
sp_Aeq = []
k1 = 0
for i in range(size_cols):
    for j in range(size_rows):
        for k in range(size_cols):
            if k == i:
                sp_Aeq_2.append(1)
                k += 1
            else:
                sp_Aeq_2.append(0)
        k -= 1
    sp_Aeq.append(sp_Aeq_2)
    sp_Aeq_2 = []

```

```

# Создаем двумерный список из ограничений на производительность (для ограничений в функции linprog)
# 'A_ub' в функции linprog – это матрица ограничений неравенства. Каждая строка A_ub задает коэффиц
sp_Aub_2 = []
sp_Aub = []
k1 = 0
for i in range(size_rows):
    for j in range(size_rows):
        n = 1
        while n <= size_cols:
            if j == k1:
                sp_Aub_2.append(1)
            else:
                sp_Aub_2.append(0)
            n += 1
        k1 += 1
    sp_Aub.append(sp_Aub_2)
    sp_Aub_2 = []

```

Скриншот 1 и 2

На скриншотах выше показан скрипт действий по созданию матрицы с ограничениями на производительность и потребность для передачи данных в функцию `linprog`.

```

# Вбиваем в функцию linprog переменные, она находит оптимальное значение целевой функции
function = round(linprog(c, A_ub, b_ub, A_eq, b_eq).fun, 3)
print('Целевая функция', function) # Выводим на экран значение ЦелФун
array1 = linprog(c, A_ub, b_ub, A_eq, b_eq).x # Присваиваем переменной найденные оптимальные значения
array0 = np.array([round(i, 1) for i in array1]) # И округляем все значения этого списка

# Преобразуем список полученных значений в матрицу
array = []
array_sm = []
l = 0
for i in range(size_rows):
    for j in range(size_cols):
        array_sm.append(array0[l])
        l += 1
    array.append(array_sm)
    array_sm = []

```

Скриншот 3

С помощью функции `linprog` мы вычислили целевую функцию, которую мы перевели в искомую матрицу. Изначально в условии мы прописываем, что в матрице не может быть более 2-х переменных. То есть наша матрица не может быть более чем 2x1 или 1x2. Мы вносим на график линию уровня и зависимость переменных. (см. скриншот 4)

```

if (size_cols == 2 and size_rows == 1) or (size_cols == 1 and size_rows == 2): # Если матрица 2x1 или 1x2
    # В зависимости от размера матрицы (2x1 или 1x2) присваиваем переменным значения ограничений
    if size_cols == 2 and size_rows == 1:
        q, h = max_factory, max_company
    elif size_cols == 1 and size_rows == 2:
        q, h = max_company, max_factory

# Задаем зависимость переменных:
x1 = np.linspace(0, q[0], 50)
x2 = [q[0] - i for i in x1] # Ограничение x2 = a - x1
x3 = [h[1] for i in x1]     # Ограничение x1 = b2
x4 = [0 for i in x1]        # Ограничение x1 >= 0
plt.axvline(x=h[0], label = 'x1 <= b1', color = 'g') # Ограничение x1 = b1
plt.axvline(x=0, label = 'x2 >= 0', color = 'y')      # Ограничение x2 >= 0

# Линия уровня
# При l = 0 Линия уровня: c1*x1 + c2*x2 = l
l = [(function - c[0]*i)/c[1] for i in x1]

```

Скриншот 4

С помощью функции `plt.plot` мы строим график и производим заливку интересующей нас области для наглядности. (См. скриншот 5)


```

# Построение графика
plt.plot(x1, x2, label = 'x2 <= a - x1', color = 'r')
plt.plot(x1, x3, label = 'x2 <= b2', color = 'b')
plt.plot(x1, x4, label = 'x1 >= 0', color = 'c')
plt.plot(x1, l, label = 'Линия уровня', color = 'k', linestyle = '--')

# Заливка
if size_cols == 2 and size_rows == 1:
    x = [0, h[0], h[0], 0]
    y = [0, 0, h[1], h[1]]

elif size_cols == 1 and size_rows == 2:
    x = [0, h[0], h[0], q[0] - h[1], 0]
    y = [0, 0, q[0] - h[0], h[1], h[1]]
plt.fill(x, y, color='red', alpha=0.3)

```

Скриншот 5

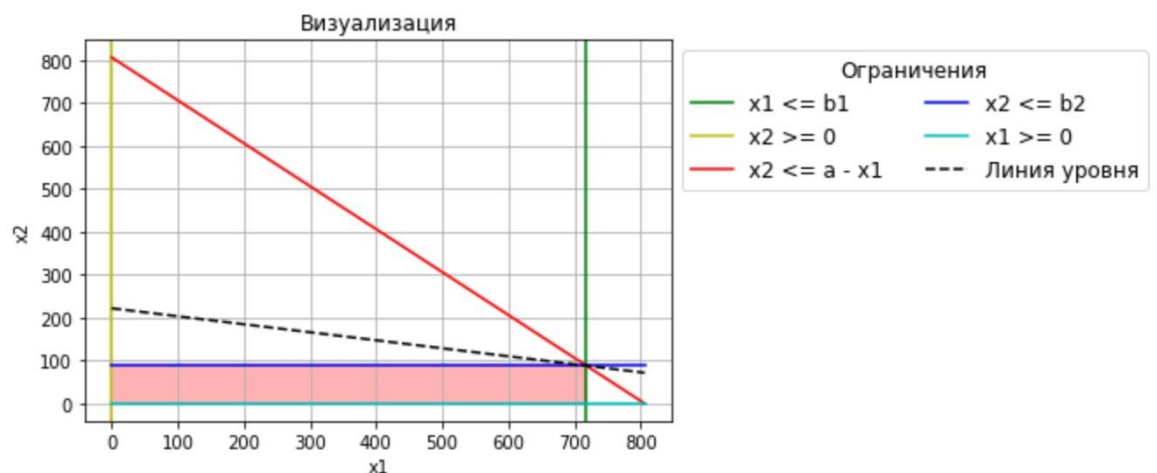
3.2.3. Описание выходных данных

На выходе мы получаем график с целевой функцией, с ограничениями и линией уровня, значением целевой функции и искомой матрицей.

```

x1*82.0 + x2*440.0
Целевая функция 97596.0
      0      1
0  718.0  88.0

```



Визуализация 1

3.3. Алгоритм 3

Решение в экселе.

3.3.1. Описание входных данных

В соответствии с требованиями и условиями, которые предоставил нам заказчик составляем матрицу (таблицу) в экселе со стоимостями, соответствующими каждому заводу/складу. Далее вводим данные о потребностях/запасах и производительности. Сверяем суммы всех запасов с суммами всех производительностей и в зависимости от схождения значений понимаем какой тип задачи (открытый или закрытый).

A	B	C	D	E	F	G	H
						запасы	
		b1	b2			327	
	a1	988	840			СУММ(G2)	
	потребности:	109	218	=СУММ(C4:D4)	327	327	
		ограничения:				СУММ(J3:K3)	
	СУММ(J3):	109	=	109		327	
	СУММ(K3):	218	=	218		<=	
						327	

эксель 1

3.3.2. Описание алгоритма решения

- Составляем искомую матрицу, которая будет заполнена количеством отправленного груза из соответствующего завода в соответствующий склад.
- Создаем целевую функцию – формула суммпроизв (ячейки стоимостей, ячейки искомой матрицы с кол-ом груза)
- Прописываем ограничения, основываясь на искомую матрицу. Общее количество груза соответствующего завода не должно превышать его производительность, а общее кол-во груза,


отправленного в соответствующую мастерскую, должно равняться её потребности.

- Через инструмент «Поиск решений» заполняем ячейку с целевой функцией, изменяя значения в искомой матрице с количеством груза, чтобы искомая стремилась к минимуму.
- Добавляем 2 ограничения, прописанных ранее и выполняем поиск решения симплекс методом.


	A	B	C	D	E	F	G	H
1							запасы	
2			b1	b2			1378	
3		a1	295	715			СУММ(G2)	
4		потребности:	970	408	=СУММ(C4:D4)	1378	1378	
5								
6			целевая функция					
7			577 870,00 Р	=СУММПРОИЗВ(J3:K3;C3:D3)				
8								
9			ограничения:				СУММ(J3:K3)	
10		СУММ(J3):	970	=	970		1378	
11		СУММ(K3):	408	=	408		<=	
12							1378	
13								

эксель 2

Параметры поиска решения

Оптимизировать целевую функцию: 

До: ☐ Максимум ☒ Минимум ☐ Значения:

Изменяя ячейки переменных: 

В соответствии с ограничениями:

Добавить

Изменить

Удалить

Сбросить

Загрузить/сохранить

☒ Сделать переменные без ограничений неотрицательными

Поиск решения 1

3.3.3. Описание выходных данных

На выходе получаем в целевой функции стоимость в рублях, в искомой матрице количество груза в штуках.

	A	B	C	D	E	F	G	H	I	J	K	L
1							запасы			количество груза		
2			b1	b2			1378			b1	b2	
3		a1	295	715			СУММ(G2)		a1	970	408	
4		потребности:	970	408	=СУММ(C4:D4)	1378	1378					
5												
6			целевая функция									
7			577 870,00 Р	=СУММПРОИЗВ(J3:K3;C3:D3)								
8												
9			ограничения:				СУММ(J3:K3)					
10		СУММ(J3):	970	=	970		1378					
11		СУММ(K3):	408	=	408		<=					
12							1378					
13												

эксель 3

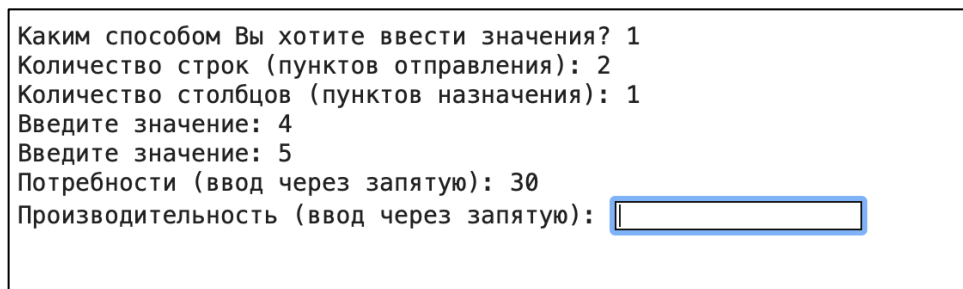
В конечном итоге мы имеем целевую функцию и исходную матрицу, которую вы можете увидеть на скриншоте 8.

4. Варианты использования системы

В нашей системе есть три варианта использования.

4.1. ВИ 1

Данный вариант использования включает в себя ручной ввод данных с клавиатуры. Для того, чтобы его активировать в графу «Каким способом вы хотите ввести значения?» надо ввести цифру «1».



Каким способом Вы хотите ввести значения? 1
Количество строк (пунктов отправления): 2
Количество столбцов (пунктов назначения): 1
Введите значение: 4
Введите значение: 5
Потребности (ввод через запятую): 30
Производительность (ввод через запятую):

Рисунок 5

В графах «количество строк» и «количество столбцов» указывается общее количество пунктов отправления и пунктов назначения.

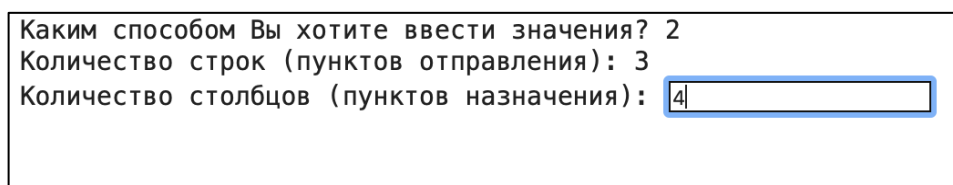
В графах «потребности» и «производительность» данные вводятся через запятую. Сверху вниз для графы «Производительность», слева направо для графы «Потребность».

Далее последуют области для заполнения нужных данных, стоит учитывать, что в графу «введите значения» вводятся данные из матрицы обязательно построчно слева направо.

После этого от пользователя требуется лишь нажатие клавиши «Enter» и на экране выведется оптимальное решение.

4.2. ВИ 2

Данный вариант использования включает в себя ввод случайных данных. Для того, чтобы его активировать в графу «Каким способом вы хотите ввести значения?» надо ввести цифру «2».



Каким способом Вы хотите ввести значения? 2
Количество строк (пунктов отправления): 3
Количество столбцов (пунктов назначения): 4

Рисунок 6

В графах «количество строк» и «количество столбцов» указывается общее количество пунктов отправления и пунктов назначения.

Далее последуют области для заполнения нужных данных, стоит учитывать, что в графу «введите значения» вводятся данные из матрицы обязательно построчно слева направо.

После этого от пользователя требуется лишь нажатие клавиши «Enter» и на экране выведется оптимальное решение.

4.3. ВИ 3

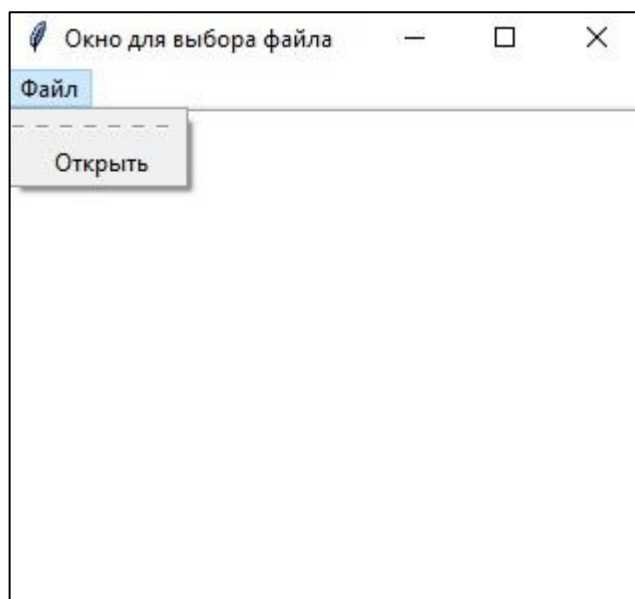
Данный вариант использования включает в себя ввод данных с помощью файла csv. Для того, чтобы его активировать в графу «Каким способом вы хотите ввести значения?» надо ввести цифру «3».



Каким способом Вы хотите ввести значения? 3

Рисунок 7

После активации данного варианта использования пользователю откроется окно. В зависимости от вашего программного обеспечения физический вид окна может отличаться. При взаимодействии через Windows с левой стороны окна будет активная кнопка «файл», на которую пользователь должен нажать, а далее выбрать нужный файл.



CSV 1

5. Архитектура решения

В данном разделе будут описаны 3 созданных метода описываются для решения задачи.

5.1. Методы считывания информации

Здесь будут описываться все методы, которые отвечают за получение программой информации, в нашей программе существует три метода ввода информации, а именно вручную, через автоматическое заполнение данных случайными числами, а также с помощью прикрепления файла csv.

Для каждого метода (функции) необходимо указать следующую информацию:

- **Первым алгоритмом** является ввод данных вручную. Такой метод принимает следующие входные данные: количество строк, количество столбцов, матричные значения, потребности и производительность. Выходными данными являются данные матричных значений, потребности и производительности. Затрагиваемые в ходе работы переменные: `size_rows`, `size_cols`, `matrix_t`, `rows`, `size_cols`, `max_company_1`, `max_factory_2`, `max_company_1_2`, `max_factory_2_2`, `max_company`, `max_factory`.

- **Второй метод** считывания информации – это заполнение данных случайными значениями. Входные данные представляют собой количество строк и столбцов, зато выходными данными являются данные матричных значений, потребности и производительность. Затрагиваемые в ходе работы переменные: `size_rows`, `size_cols`, `matrix_t`, `rows`, `x`, `max_company`, `max_factory`.

- **Третий метод** представляет собой считывание данных с файла формата csv. Входными данными являются данные файла, которые в

последствии обрабатываются. Выходными данными являются данные матричных значений, потребности и производительность. Затрагиваемые в ходе работы переменные: `size_rows`, `size_cols`, `matrix_t`, `rows`, `max_company`, `max_factory`, `max_company_1_2`.

5.2. Методы обработки информации

В данном разделе описываются все методы обработки введенной пользователем информации и получение решения.

Первичные методы обработки информации зависят от того, каким способом водятся данные.

- 1) Для ввода данных соответствуют следующие блоки:

Первый блок - цикл для последовательного ввода значений и заполнения ими матрицы исходных данных. Входными данными являются количество строк, количество столбцов и матричные значения. Выходными данными является матрица. Затрагиваемые переменные: `size_rows`, `size_cols`, `matrix_t`, `rows`, `x`.

Второй блок – преобразование данных для дальнейших операций. Входные данные представляют собой потребность и производительность. Выходными данными являются числовые значения потребности и производительности. Затрагиваемые переменные: `max_company`, `max_factory`, `max_company_1_2`.

- 2) Для получения данных с помощью генератора случайных чисел не выполняется никаких особенных действий на первоначальном этапе по обработке информации.
- 3) Для получения данных с помощью файла формата csv соответствует следующая функция – считывание данных. Входными данными являются данные файла. Выходными данными являются преобразованные данные потребностей, производительности и

матрицы. Затрагиваемыми переменными являются: self, filename, row, max_company, max_factory, max_company_1_2.

Далее все сводится к единому алгоритму, который состоит из следующего метода:

Расчет целевой функции и оптимальной матрицы. Входными данными является матрица. Выходными данными значение целевой функции, матрица оптимальных значений. Затрагиваемые переменные: matrix, function_text, sp_Aub, sp_Aub_2, size_rows, size_cols, sp_Aeq, sp_Aeq_2, b_ub, A_ub, function, array, array_sm, array1, array0.

5.3. Методы вывода информации

В разделе описываются все методы(функции) отвечающие за вывод информации пользователю, такие как график и конечные значения.

1) Первый метод – это вывод значений, которые обрабатываются для решения задачи. Входными параметрами являются полученные при обработке информации данные, такие как целевая функция, формула, по которой считается целевая функция, а также матрица. Выходными данными также является формула целевой функции, целевая функция и матрица. Затрагиваемые переменные в данном методе: function, function_text, array.

2) Второй метод – это визуализация данных. Входными данными являются ограничения, линия уровня, целевая функция. Выходными данными у нас будет график, в котором отображаются ограничения, линия уровня. Затрагиваемые переменные: x1, x2, x3, x4, l, x, y.

6. Тестирование

Таблица 1. Матрицы 1x2 и 2x1

Входные данные	Python	Визуализация	Excel	Онлайн- калькулятор
Потреб: 970, 408 Произв: 1378	577870	Рисунок 1	577870	577870
Потреб: 109, 218 Произв: 327	290812	Рисунок 2	290812	290812
Потреб: 437, 150 Произв: 587	280755	Рисунок 3	280755	280755
Потреб: 474 Произв: 248, 387	175278	Рисунок 4	175278	175278
Потреб: 521 Произв: 301, 368	135102	Рисунок 5	135102	135102

6.1. Результаты тестирования первого дата сета:

6.1.1. Проверка кода Python и визуализации:

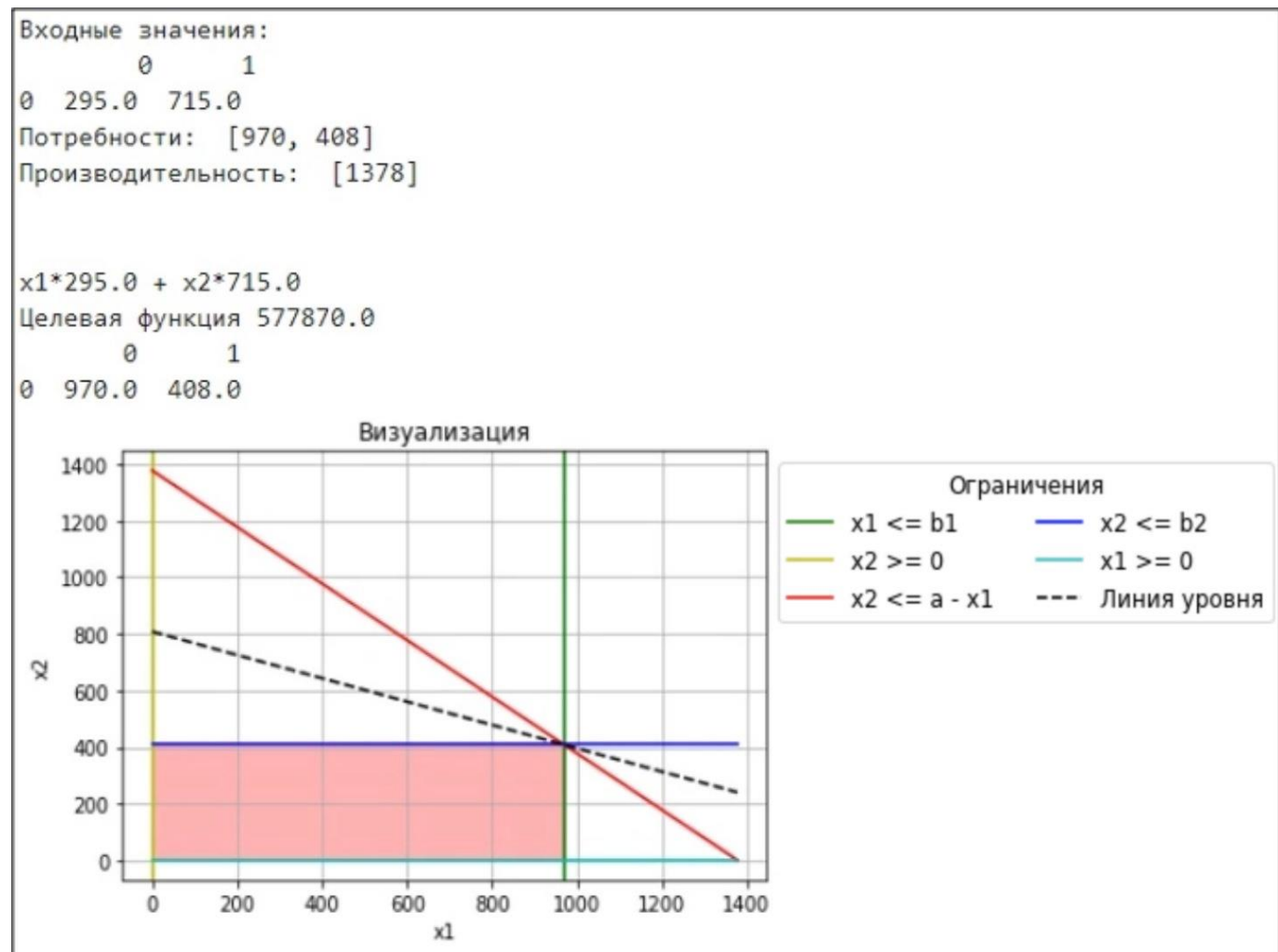


Рисунок 8

6.1.2. Проверка кода методом Excel:

	A	B	C	D	E	F	G	H	I	J	K	L
1							запасы			количество груза		
2			b1	b2			1378			b1	b2	
3		a1	295	715			СУММ(G2)		a1	970	408	
4		потребности:	970	408	=СУММ(C4:D4)	1378	1378					
5												
6			целевая функция									
7			577 870,00 Р	=СУММПРОИЗВ(J3:K3;C3:D3)								
8												
9			ограничения:				СУММ(J3:K3)					
10		СУММ(J3):	970	=	970		1378					
11		СУММ(K3):	408	=	408		<=					
12							1378					
13												

эксель 4

6.1.3. Проверка кода в онлайн-калькуляторе:

Решение:

Оптимальный план имеет следующий вид:

$$X = \begin{bmatrix} 970 & 408 \end{bmatrix}$$

При этом плане стоимость перевозок вычисляется так:

$$S = 295 \cdot 970 + 715 \cdot 408 = 577870$$

онлайн калькулятор 1

6.2. Результаты тестирования второго дата сета:

6.2.1. Проверка кода Python и визуализация:

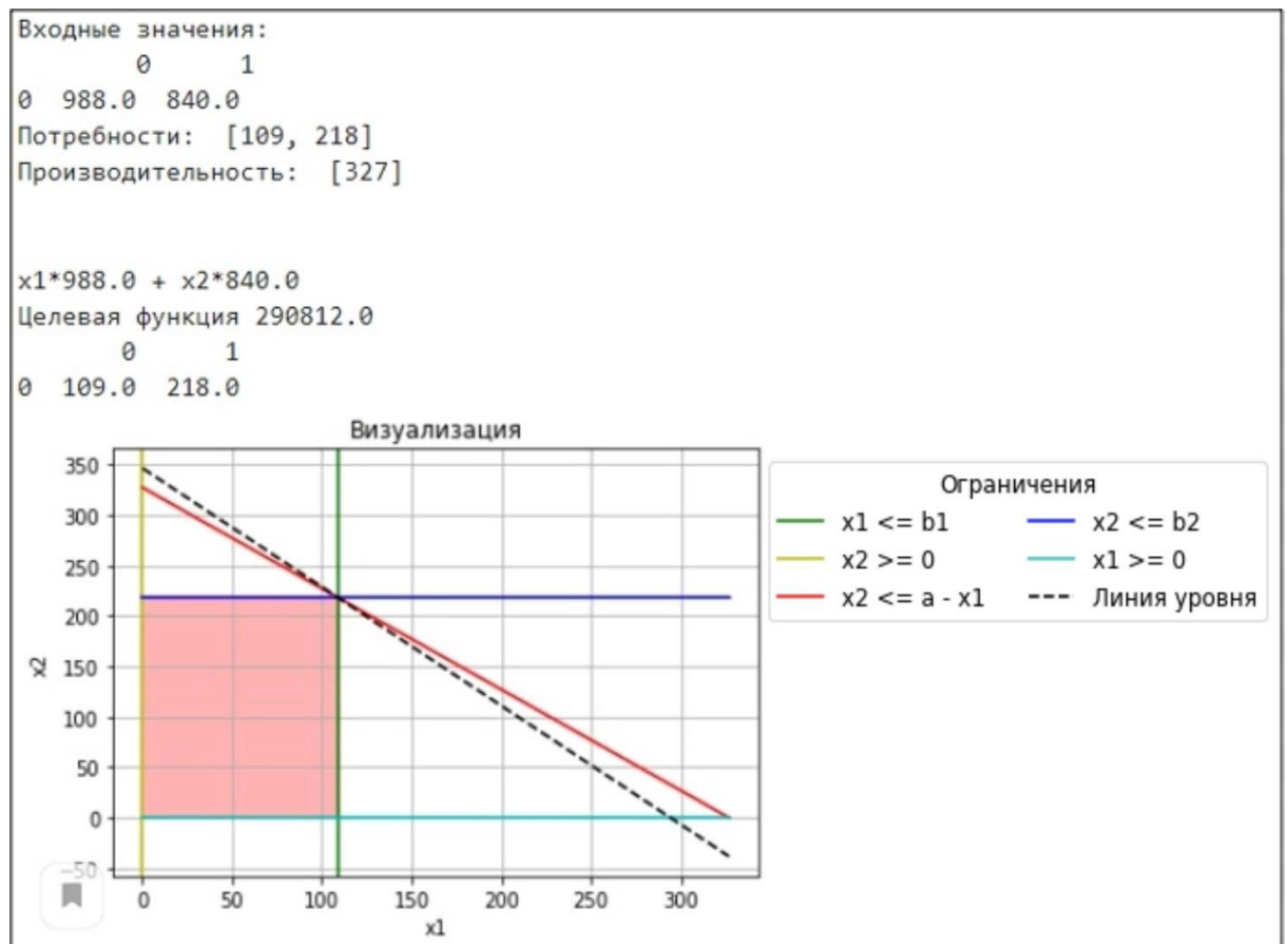


Рисунок 9

6.2.2. Проверка кода методом Excel:

	A	B	C	D	E	F	G	H	I	J	K	L
1							запасы			количество груза		
2			b1	b2			327			b1	b2	
3		a1	988	840			СУММ(G2)		a1	109	218	
4		потребности:	109	218	=СУММ(C4:D4)	327	327					
5												
6			целевая функция									
7			290 812,00 Р	=СУММПРОИЗВ(J3:K3;C3:D3)								
8												
9			ограничения:				СУММ(J3:K3)					
10		СУММ(J3):	109	=	109		327					
11		СУММ(K3):	218	=	218		<=					
12							327					
13												

эксель 5

6.2.3. Проверка кода в онлайн-калькуляторе:

Решение:
Оптимальный план имеет следующий вид:
$X = \begin{bmatrix} 109 & 218 \end{bmatrix}$
При этом плане стоимость перевозок вычисляется так:
$S = 988 \cdot 109 + 840 \cdot 218 = 290812$

онлайн калькулятор 2

6.3. Результаты тестирования третьего дата сета:

6.3.1. Проверка кода Python и визуализация:

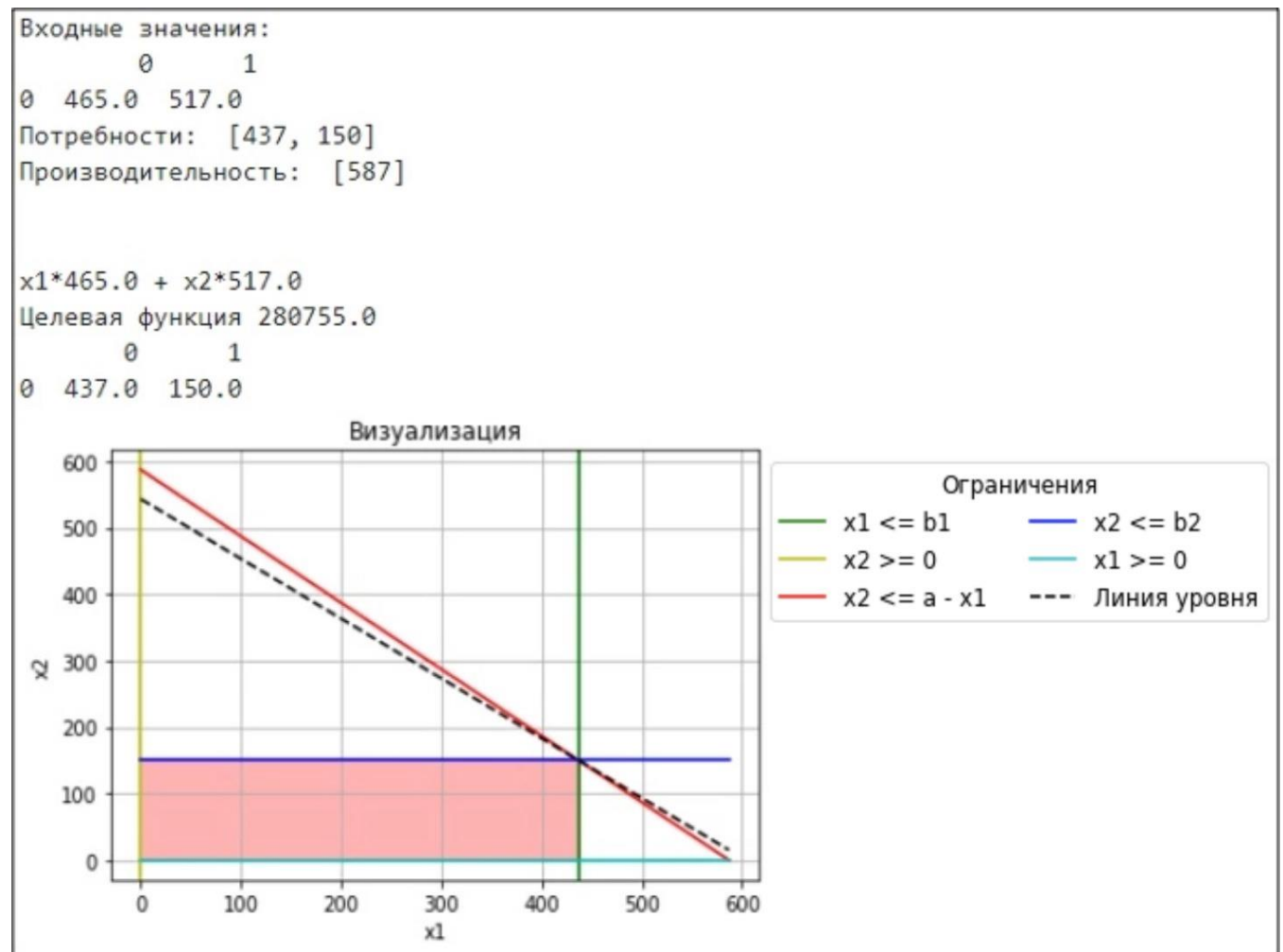


Рисунок 10

6.3.2. Проверка кода методом Excel:

	A	B	C	D	E	F	G	H	I	J	K	L
1							запасы			количество груза		
2			b1	b2			587			b1	b2	
3		a1	465	517			СУММ(G2)		a1	437	150	
4		потребности:	437	150	=СУММ(C4:D4)	587	587					
5												
6			целевая функция									
7			280 755,00 Р	=СУММПРОИЗВ(J3:K3;C3:D3)								
8												
9			ограничения:				СУММ(J3:K3)					
10		СУММ(J3):	437	=	437		587					
11		СУММ(K3):	150	=	150		<=					
12							587					
13												

эксель 6

6.3.3. Проверка кода в онлайн-калькуляторе:

Решение:

Оптимальный план имеет следующий вид:

$$X = \begin{bmatrix} 437 & 150 \end{bmatrix}$$

При этом плане стоимость перевозок вычисляется так:

$$S = 465 \cdot 437 + 517 \cdot 150 = 280755$$

онлайн калькулятор 3

6.4. Результаты тестирования четвёртого дата сета:

6.4.1. Проверка кода Python и визуализации:

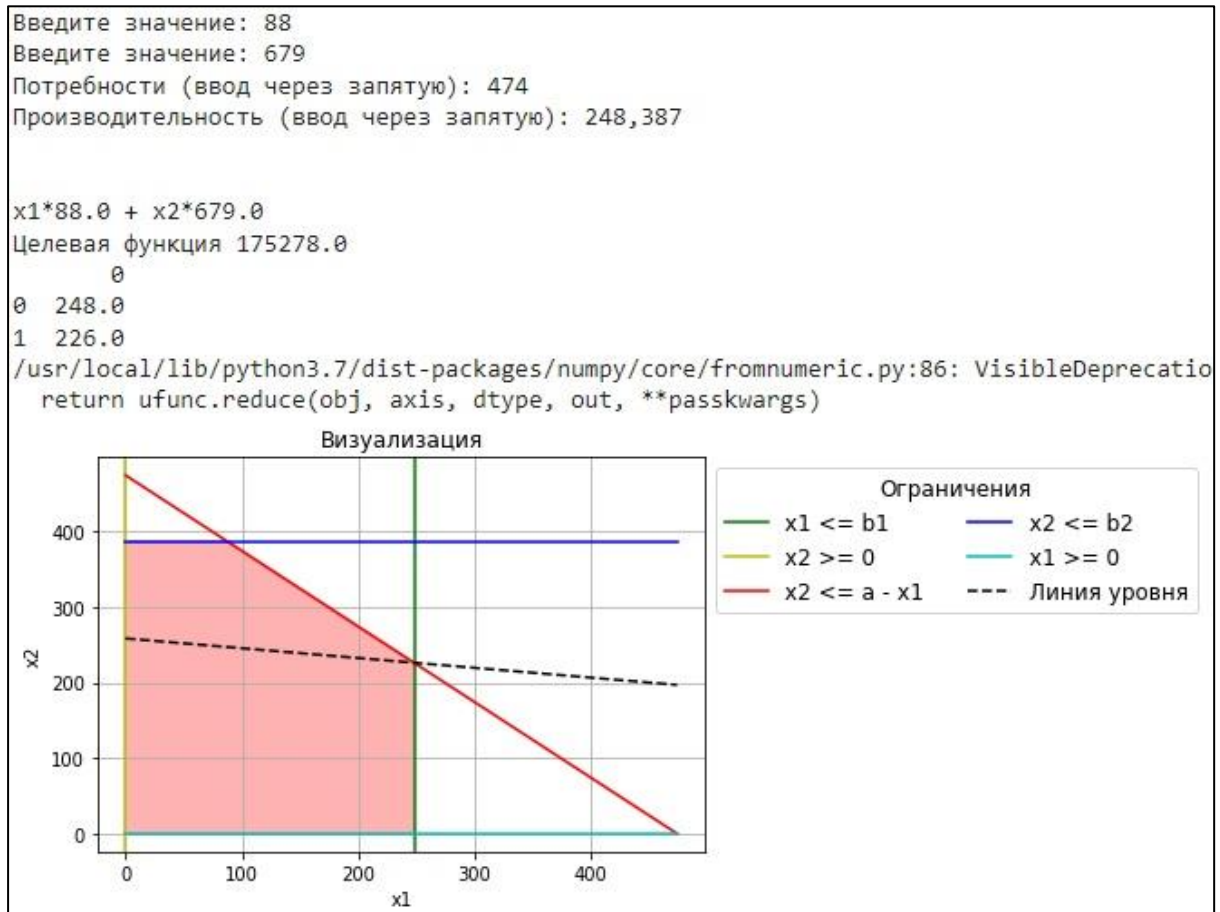


Рисунок 11

6.4.2. Проверка кода методом Excel:

	A	B	C	D	E	F	G	H	I	J	K	L
1				b1		запасы				количество груза		
2			a1	88		248				b1		
3			a2	679		387			a1	248		
4						СУММ(F2:F3):			a2	226		
5		потребности:	474	СУММ(D2:D3):	767	635						
6												
7												
8			целевая функция									
9			175 278,00 Р	=СУММПРОИЗВ(J3:J4;D2:D3)								
10												
11			ограничения:				=СУММ(J3)	=СУММ(J4)				
12		СУММ(J3:J4):	474	=	474		248	226				
13							<=	<=				
14							248	387				
15												

эксель 7

6.4.3. Проверка кода в онлайн-калькуляторе:

Решение:

Оптимальный план имеет следующий вид:

$$X = \begin{bmatrix} 248 \\ 226 \end{bmatrix}$$

При этом плане стоимость перевозок вычисляется так:

$$S = 88 \cdot 248 + 679 \cdot 226 = 175278$$

При этом плане остается неиспользованным 161 запасов пункта A_2 .

онлайн калькулятор 4

6.5. Результаты тестирования пятого дата сета:

6.5.1. Проверка кода Python и визуализации:

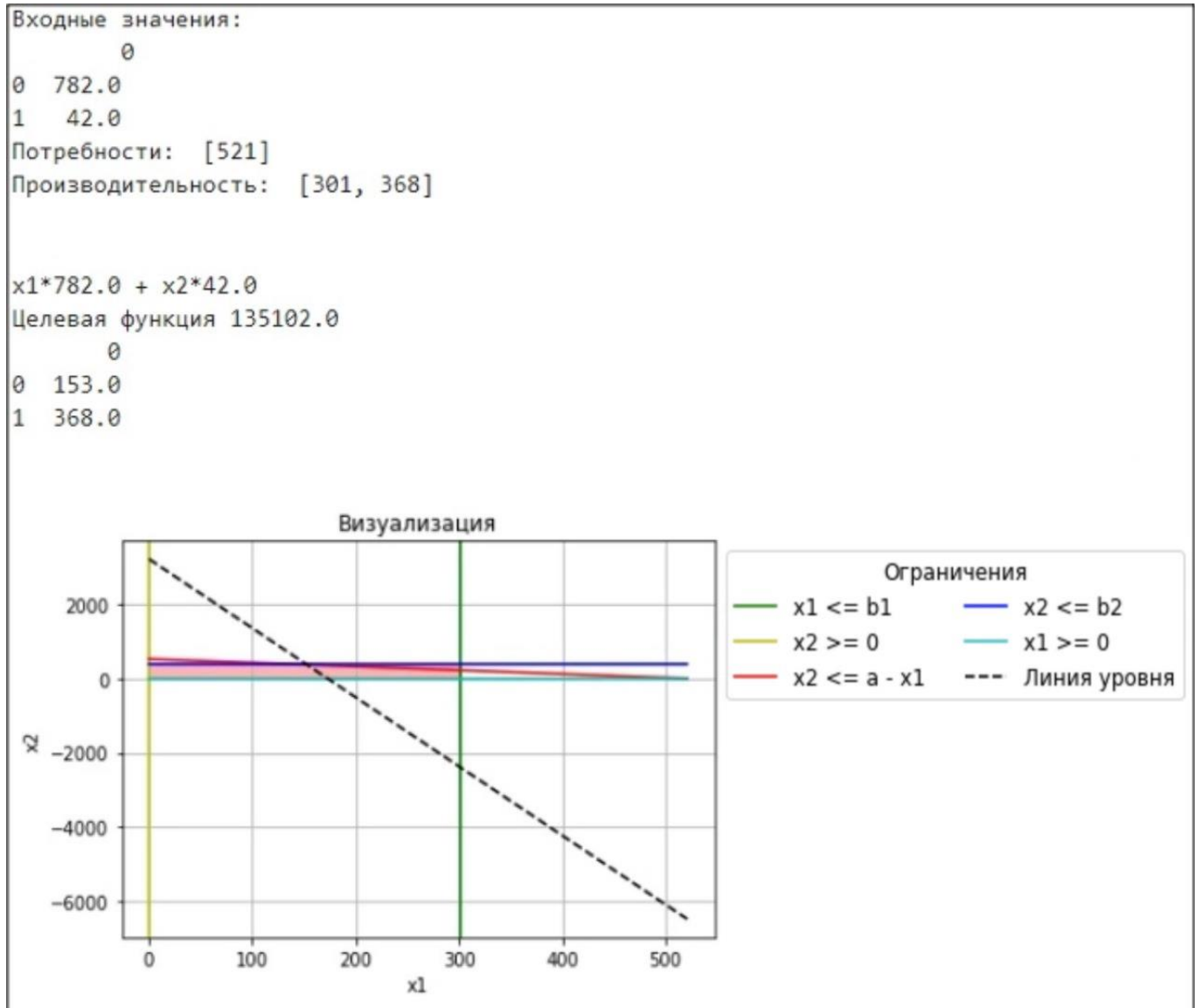


Рисунок 12

6.5.2. Проверка кода методом Excel:

	A	B	C	D	E	F	G	H	I	J	K	L
1				b1		запасы				количество груза		
2			a1	782		301				b1		
3			a2	42		368			a1	153		
4						СУММ(F2:F3):			a2	368		
5		потребности:	521	СУММ(D2:D3):	824	669						
6												
7												
8			целевая функция									
9			135 102,00 Р	=СУММПРОИЗВ(J3:J4;D2:D3)								
10												
11			ограничения:				=СУММ(J3)	=СУММ(J4)				
12		СУММ(J3:J4):	521	=	521		153	368				
13							<=	<=				
14							301	368				
15												

эксель 8

6.5.3. Проверка кода в онлайн-калькуляторе:

Решение:
Оптимальный план имеет следующий вид:
$X = \begin{bmatrix} 153 \\ 368 \end{bmatrix}$
При этом плане стоимость перевозок вычисляется так:
$S = 782 \cdot 153 + 42 \cdot 368 = 135102$
При этом плане остается неиспользованным 148 запасов пункта A_1 .

онлайн калькулятор 5

Таблица 2. Матрицы 3x4 и 4x3

Входные данные	Python	Матрица и целевая функция	Excel	Онлайн-калькулятор
Потреб: 197, 1000, 309 Произв: 914, 1044, 756, 323	132739	Рисунок 6	132739	132739

Потреб: 654, 591, 822, 579 Произв: 122, 2234, 2359	751420	Рисунок 7	751420	751420
---	--------	-----------	--------	--------

6.6. Результаты тестирования шестого дата сета:

6.6.1. Проверка кода Python:

```

Входные значения:
      0      1      2
0  386.0  838.0  844.0
1  506.0   92.0  322.0
2   97.0  456.0  965.0
3  492.0  538.0   70.0
Потребности: [197, 1000, 309]
Производительность: [914, 1044, 756, 323]

Целевая функция 132739.0
      0      1      2
0   0.0   0.0   0.0
1   0.0 1000.0   0.0
2  197.0   0.0   0.0
3   0.0   0.0  309.0

```

Рисунок 13

6.6.2. Проверка кода методом Excel:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1									запасы					количество груза		
2			b1	b2	b3									b1	b2	b3
3		a1	386	838	844				914				a1	0	0	-3,55271E-15
4		a2	506	92	322				1044				a2	0	1000	0
5		a3	97	456	965				756				a3	197	0	0
6		a4	492	538	70				323				a4	0	0	309
7									СУММ(I3:I6):							
8		потребности	197	1000	309		СУММ(C8:F8):	1506	3037							
9																
10																
11			целевая функция - стоимость перевозки кирпичей													
12			132 739,00 Р	=СУММПРОИЗВ(N3:P6;C3:E6)												
13																
14																
15			ограничения:				=СУММ(N3:N6)	=СУММ(O3:O6)	=СУММ(P3:P6)							
16		СУММ(N3:P3):	-3,55271E-15	<=	914		197	1000	309							
17		СУММ(N4:P4):	1000	<=	1044		=	=	=							
18		СУММ(N5:P5):	197	<=	756		197	1000	309							
19		СУММ(N6:P6):	309	<=	323											

эксель 9

6.6.3. Проверка кода в онлайн-калькуляторе:

Решение:

Оптимальный план имеет следующий вид:

$$X = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1000 & 0 \\ 197 & 0 & 0 \\ 0 & 0 & 309 \end{bmatrix}$$

При этом плане стоимость перевозок вычисляется так:

$$S = 92 \cdot 1000 + 97 \cdot 197 + 70 \cdot 309 = 132739$$

При этом плане остается неиспользованным 914 запасов пункта A_1 , 44 запасов пункта A_2 , 559 запасов пункта A_3 , 14 запасов пункта A_4 .

онлайн калькулятор 6

6.7. Результаты тестирования седьмого дата сета:

6.7.1. Проверка кода Python:

Входные значения:				
	0	1	2	3
0	380.0	431.0	96.0	629.0
1	282.0	745.0	676.0	55.0
2	445.0	85.0	942.0	542.0
Потребности: [654, 591, 822, 579]				
Производительность: [122, 2234, 2359]				
Целевая функция 751420.0				
	0	1	2	3
0	0.0	0.0	122.0	0.0
1	654.0	0.0	700.0	579.0
2	0.0	591.0	0.0	0.0

Рисунок 14

6.7.2. Проверка кода методом Excel:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1									запасы					количество груза			
2			b1	b2	b3	b4								b1	b2	b3	b4
3		a1	380	431	96	629			122				a1	0	0	122	0
4		a2	282	745	676	55			2234				a2	654	0	700	579
5		a3	445	85	942	545			2359				a3	0	591	0	0
6									СУММ(I3:I5):								
7		потребности	654	591	822	579	СУММ(C7:G7):	2646	4715								
8																	
9																	
10			целевая функция - стоимость перевозки кирпичей														
11			751 420,00 Р	=СУММПРОИЗВ(N3:Q5;C3:F5)													
12																	
13			ограничения:				СУММ(N3:N5): СУММ(O3:O5): СУММ(P3:P5): СУММ(Q3:Q5):										
14		СУММ(N3:Q3):	122	<=	122		654	591	822	579							
15		СУММ(N4:Q4):	1933	<=	2234		=	=	=	=							
16		СУММ(N5:Q5):	591	<=	2359		654	591	822	579							
17																	

эксель 10

6.7.3. Проверка кода в онлайн-калькуляторе:

Решение:

Оптимальный план имеет следующий вид:

$$X = \begin{bmatrix} 0 & 0 & 122 & 0 \\ 654 & 0 & 700 & 579 \\ 0 & 591 & 0 & 0 \end{bmatrix}$$

При этом плане стоимость перевозок вычисляется так:

$$S = 96 \cdot 122 + 282 \cdot 654 + 676 \cdot 700 + 55 \cdot 579 + 85 \cdot 591 = 751420$$

При этом плане остается неиспользованным 301 запасов пункта A_2 , 1768 запасов пункта A_3 .

онлайн калькулятор 7

Для подсчёта времени на выполнение алгоритмов будет использовано время, затраченное на ввод необходимых данных в предназначенные поля и расчёт данных самой программой с дальнейшим выводом их пользователю на экран.

Таблица 3. Время выполнения алгоритмов

Входные данные	Python	Матрица и целевая функция	Excel	Онлайн-калькулятор
Дата сет №1	~10-30 секунд	~10-30 секунд	~1-1,5 минуты	~12 секунд
Дата сет №2	~10-30 секунд	~10-30 секунд	~1-1,5 минуты	~20–40 секунд
Дата сет №3	~10-30 секунд	~10-30 секунд	~1-1,5 минуты	~20–40 секунд

Дата сет №4	~10-30 секунд	~10-30 секунд	~1-1,5 минуты	~20–40 секунд
Дата сет №5	~10-30 секунд	~10-30 секунд	~1-1,5 минуты	~20–40 секунд
Дата сет №6	~10-30 секунд	~10-30 секунд	~1,5-2 минут	~30–50 секунд
Дата сет №7	~10-40 секунд	~10-40 секунд	~1,5-2 минут	~30–50 секунд

Из этого следует, что предпочтительным алгоритмом является использование Python и онлайн-калькулятора в связи с их точностью и быстротой выполняемых команд. Погрешности в выводим данных между разными алгоритмами не выявлено.

7. Заключение

Наш предпочтительный алгоритм, а именно программный код в Питоне, решает поставленную задачу. На основании тестирования данного алгоритма, можно сделать вывод о том, что Python выводит самое оптимальное решение достаточно быстро. Ниже представлено решение, поставленной задачи:

```
Количество строк (пунктов отправления): 3
Количество столбцов (пунктов назначения): 4
Введите значение: 400
Введите значение: 500
Введите значение: 300
Введите значение: 600
Введите значение: 700
Введите значение: 200
Введите значение: 100
Введите значение: 500
Введите значение: 600
Введите значение: 100
Введите значение: 400
Введите значение: 200
Потребности (ввод через запятую): 220,150,250,180
Производительность (ввод через запятую): 300,250,200

Целевая функция 14546.918
    0    1    2    3
0  3.8  4.5  1.0  4.4
1  5.5  1.6  4.0  0.3
2  0.3  1.3  5.4  3.6
```

Значения 1

Теперь сравним несколько алгоритмов по критериям: эффективности, скорости использования алгоритма, простоты использования, надежность в разрезе человеческого фактора и точность предоставленного решения.

Критерий	Python	Excel	Визуализация
эффективность	высокая	высокая	средняя
скорость	высокая	низкая	высокая

использования алгоритма			
простота использования	высокая	низкая	высокая
надежность (человеческий фактор)	высокая	низкая	высокая
точность	высокая	высокая	высокая

Для заказчика самым оптимальным вариантом будет первый алгоритм — это программа в Python. Поскольку исходя из подобранных критериев, она удовлетворяет всем требованиям. Такой алгоритм прост в использовании, работа с программой не отнимет много времени. Более того, программа сведена к минимуму от воздействия человеческого фактора - то есть ошибок при решении, все это делает компьютер. Ну и конечно же самое главное — это точность ответа и эффективность данного алгоритма.

Одной из перспектив развития этого алгоритма - является усовершенствование его, для решения так же производственной задачи. Также алгоритм может быть усовершенствован с точки зрения наглядности, например, выдавать пошаговое решение.

Более того, разработанный нами код поможет заказчику минимизировать затраты при любых транспортировках, будь то внешние или внутренние перевозки.