Национальный Исследовательский Университет

**Высшая Школа Экономики**



Факультет Бизнеса и Менеджмента

38.03.05 «Бизнес-Информатика»

Дисциплина: Количественные методы принятия управленческих решений

Контрольное домашнее задание

Тема: **«Задание 1»**

Выполнили:

Исмаилов Р. И.

Шитяков С.П.

Асатуров Р. Б.

Преподаватель:

Попов В. Ю.

Москва

2020

Оглавление

[**Варианты** 3](#_Toc51525457)

[**Программный стек** 3](#_Toc51525458)

[**Вступление** 3](#_Toc51525459)

[**Решение задачи 1** 4](#_Toc51525460)

[**Пункт 7.1** 4](#_Toc51525461)

[**Пункт 7.2** 7](#_Toc51525462)

[**Пункт 7.3** 10](#_Toc51525463)

[**Пункт 7.4** 13](#_Toc51525464)

[**Решение задачи 2** 16](#_Toc51525465)

[**Пункт 90** 16](#_Toc51525466)

[**Пункт 92** 20](#_Toc51525467)

[**Пункт 100** 26](#_Toc51525468)

[**Решение задачи 3** 31](#_Toc51525469)

[**Пункт а)** 34](#_Toc51525470)

[**Пункт б)** 35](#_Toc51525471)

[**Пункт с)** 38](#_Toc51525472)

[**Пункт d)** 38](#_Toc51525473)

[**Пункт e\_1)** 39](#_Toc51525474)

[**Отчет об устойчивости:** 41](#_Toc51525475)

[**Приложение 1** 43](#_Toc51525476)

[**Приложение 2** 45](#_Toc51525477)

[**Приложение 3** 49](#_Toc51525478)

# **Варианты**

Нашей командой были выбраны следующие варианты задач (см. таблица 1):

|  |  |  |  |
| --- | --- | --- | --- |
|  | Задача 1 | Задача 2 | Задача 3 |
| Номера вариантов | 7 | 90, 92, 100 | 27 |

*Таблица 1 Распределение вариантов*

# **Распределение задач**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Задача 1 | Задача 2 | Задача 3 | Отчет |
| Исмаилов Рустам | Excel + Python | Excel + Python | Excel | Редактирование |
| Шитяков Сергей | - | - | Python | - |
| Асатуров Рубен | Excel | - | - | Наполнение |

# **Программный стек**

|  |  |
| --- | --- |
| Инструмент | Библиотеки |
| Excel |  |
| Python | Numpy |
| Scipy |
| Pulp |

*Таблица 2 Программный стек*

# **Вступление**

Абсолютно все задачи Excel выполнены по симплекс методу. Все проверки задач в python реализованы методом «опорной точки» (за исключением задачи 2\_100, у которой так же присутствует целочисленная реализация методом «COIN-OR branch and cut» (CBC))

Описание задач:

1. Задачи 1\_7\_1, 1\_7\_2, 1\_7\_3, 1\_7\_4 являются абстрактными задачами ЗЛП.

2. Задачи 2\_90, 2\_92 - транспортными ЗЛП.

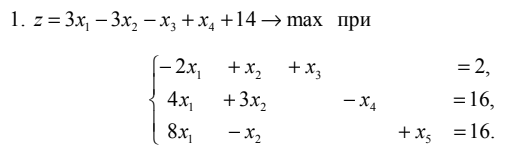
3. Задача 2\_100 - задача оптимального раскроя целочисленного ЛП.

4. Задача 3\_27 - задача нахождения оптимального выпуска продукции ЛП.

# **Решение задачи 1**

## **Пункт 7.1**

Условие:

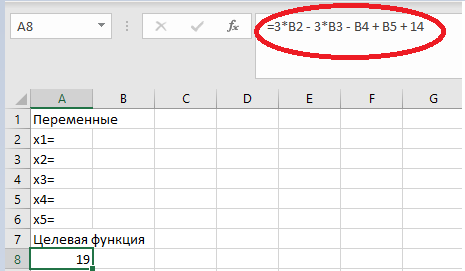


Решение с помощью Excel:

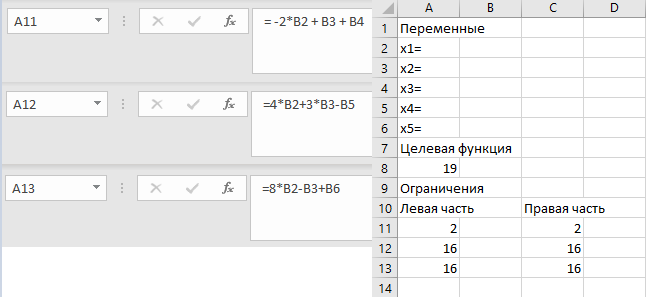
В ячейка B2:B6 создадим область переменных



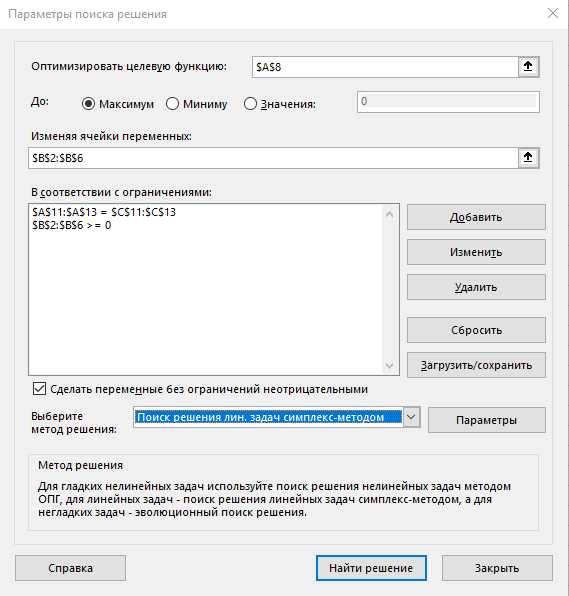
Далее введем формулу вычисления целевой функции в ячейку A8



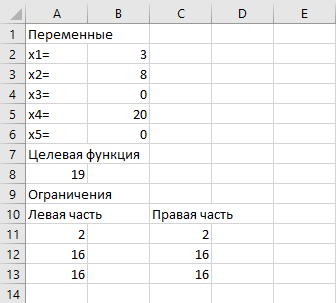
После этого создадим ограничения. Для этого в ячейках А11:А13 будем вычислять левые части ограничений в системе, а в ячейках C11:С13 введем правые части.



Далее для решения задачи вызовем окно диалога «Поиск решения». Выберем ячейку A8 в качестве целевой функции, которую мы будем оптимизировать. Ячейки B2:B6 укажем в качестве переменных. Так как в условии задачи сказано найти max, то выберем данный пункт здесь. Далее укажем ограничения. В соответствии с общим условием, укажем то, что все переменные являются неотрицательными и выберем остальные ограничения.

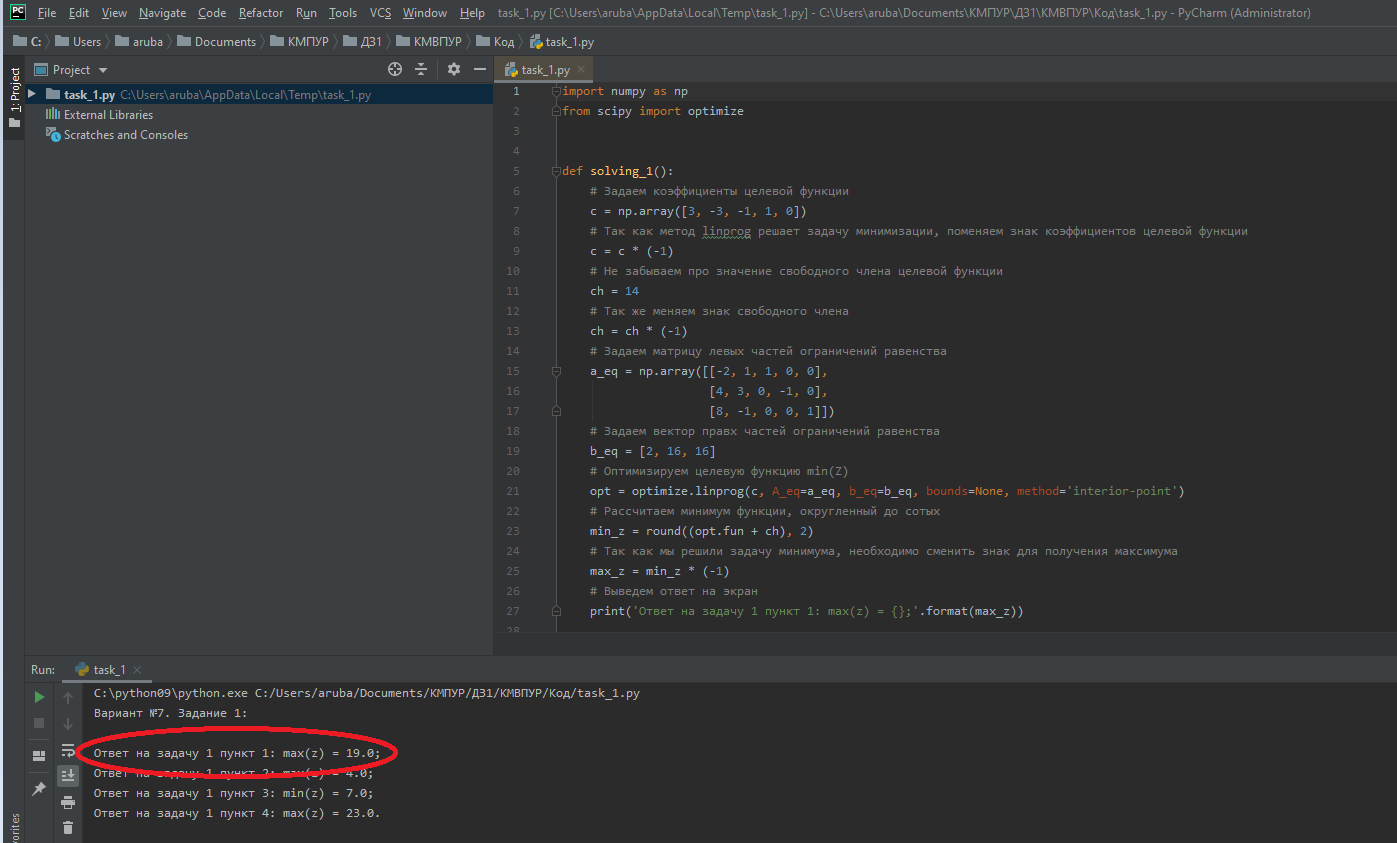


После завершения настроек и выполнения задачи получим следующий ответ:



Значение целевой функции на множестве х={3, 8, 0, 20, 0} равно: 19

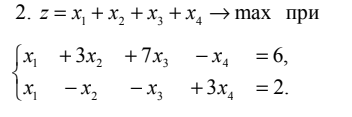
Проверим правильность выполнения нашей задачи с помощью Python:



Как мы видим, наши ответы сошлись. Это говорит нам о том, что данный пункт решен верно.

## **Пункт 7.2**

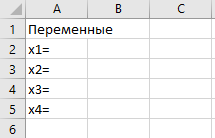
Условие:



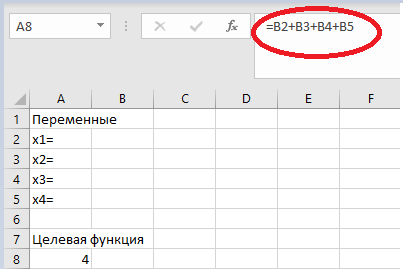
*Рисунок 1 Условие пункта 7.1*

Решение с помощью Excel:

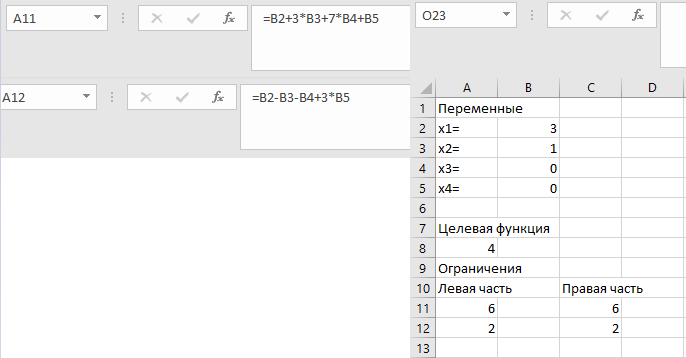
В ячейка B2:B5 создадим область переменных



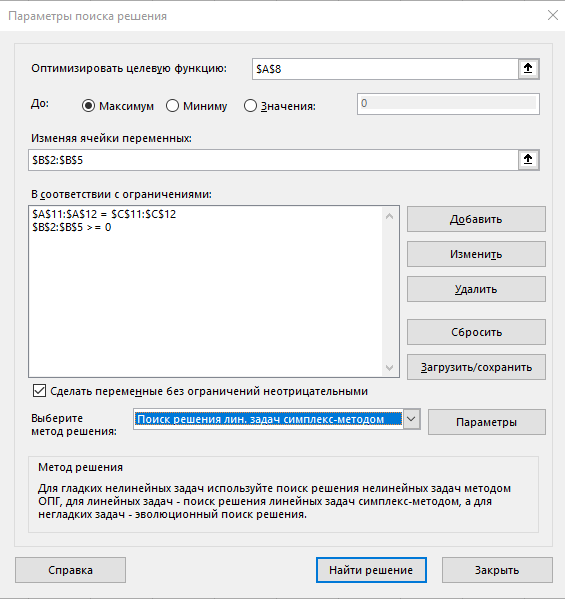
Далее введем формулу вычисления целевой функции в ячейку A8



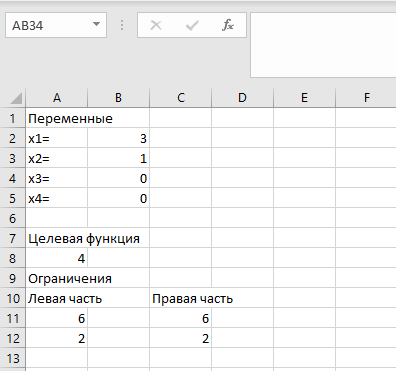
После этого создадим ограничения. Для этого в ячейках А11:А12 будем вычислять левые части ограничений в системе, а в ячейках C11:С12 введем правые части.



Далее для решения задачи вызовем окно диалога «Поиск решения». Выберем ячейку A8 в качестве целевой функции, которую мы будем оптимизировать. Ячейки B2:B5 укажем в качестве переменных. Так как в условии задачи сказано найти max, то выберем данный пункт здесь. Далее укажем ограничения. В соответствии с общим условием, укажем то, что все переменные являются неотрицательными и выберем остальные ограничения.

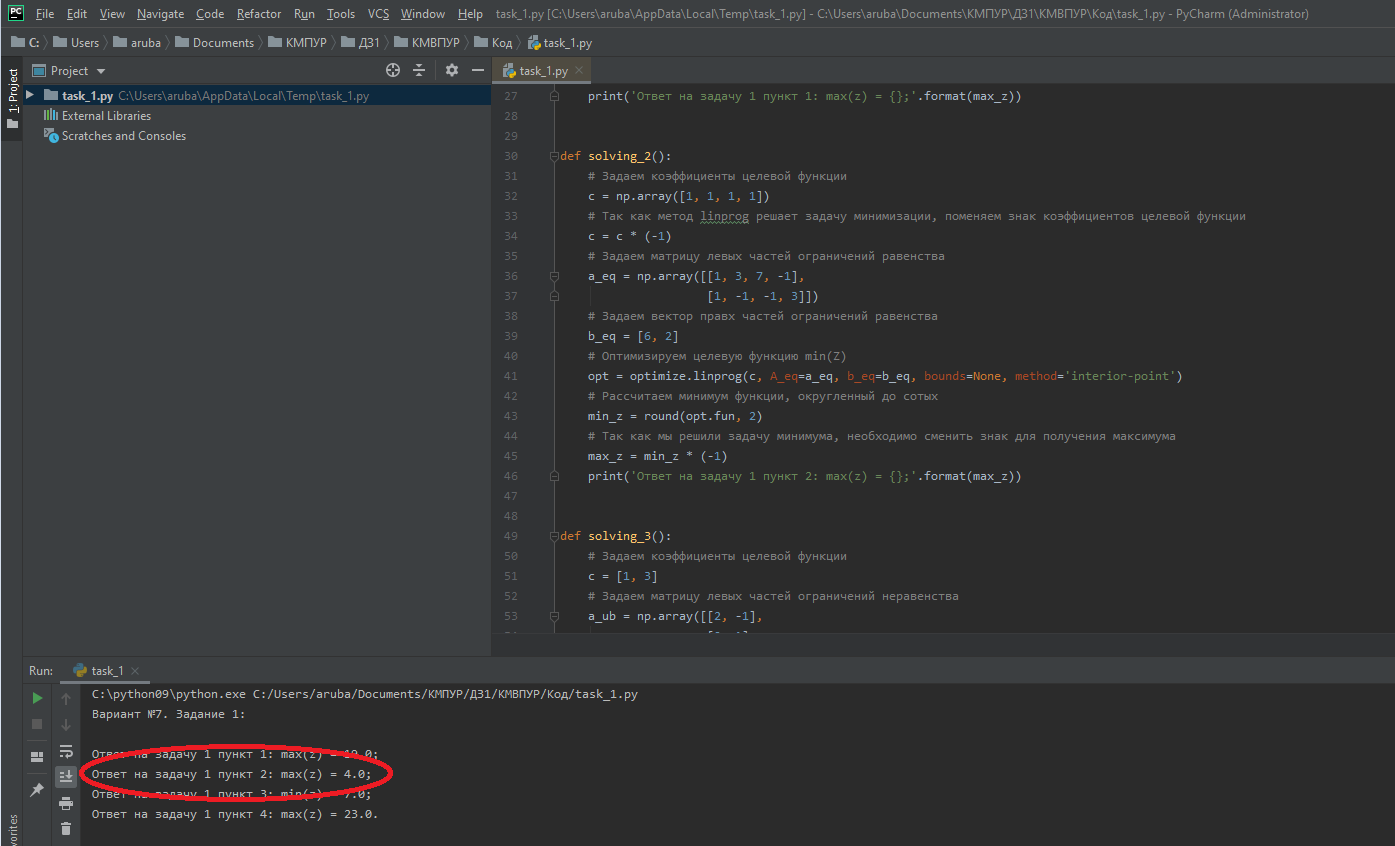


После завершения настроек и выполнения задачи получим следующий ответ:



Значение целевой функции на множестве х={3, 1, 0, 0} равно: 4

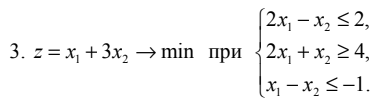
Проверим правильность выполнения нашей задачи с помощью Python:



Как мы видим, наши ответы сошлись. Это говорит нам о том, что данный пункт решен верно.

## **Пункт 7.3**

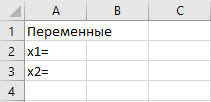
Условие:



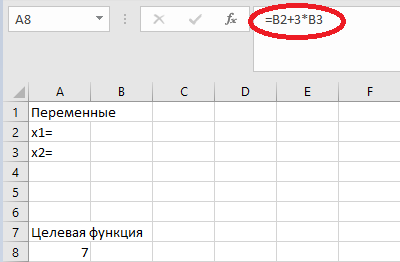
*Рисунок 1 Условие пункта 7.1*

Решение с помощью Excel:

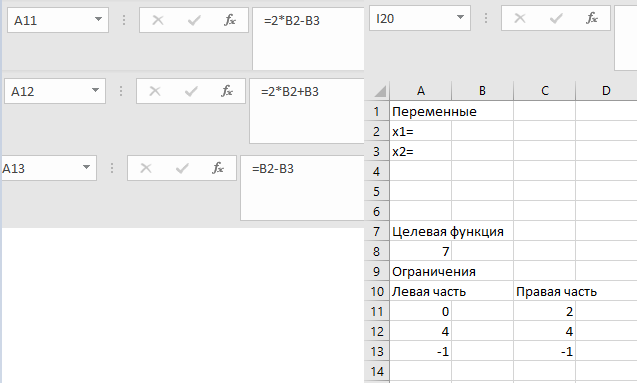
В ячейка B2:B3 создадим область переменных



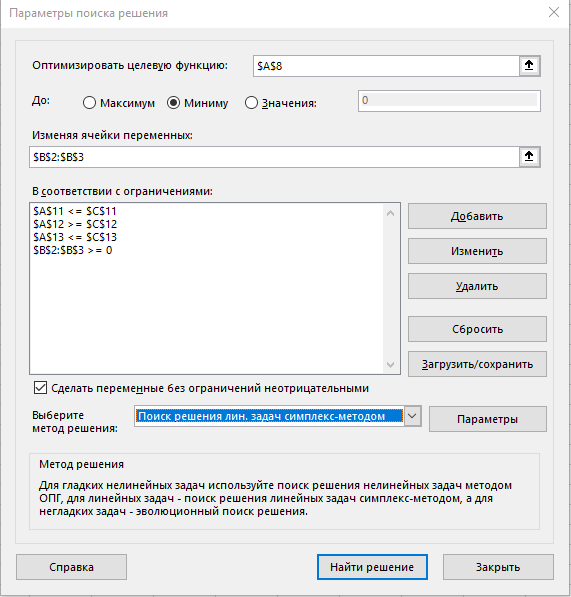
Далее введем формулу вычисления целевой функции в ячейку A8



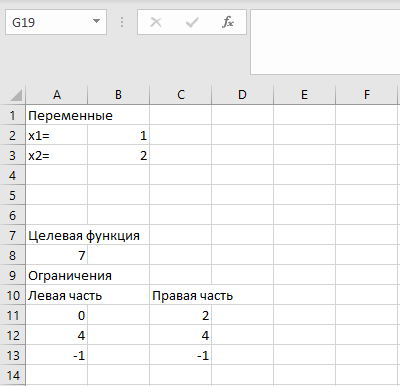
После этого создадим ограничения. Для этого в ячейках А11:А12 будем вычислять левые части ограничений в системе, а в ячейках C11:С12 введем правые части.



Далее для решения задачи вызовем окно диалога «Поиск решения». Выберем ячейку A8 в качестве целевой функции, которую мы будем оптимизировать. Ячейки B2:B3 укажем в качестве переменных. Так как в условии задачи сказано найти min, то выберем данный пункт здесь. Далее укажем ограничения. В соответствии с общим условием, укажем то, что все переменные являются неотрицательными и выберем остальные ограничения.

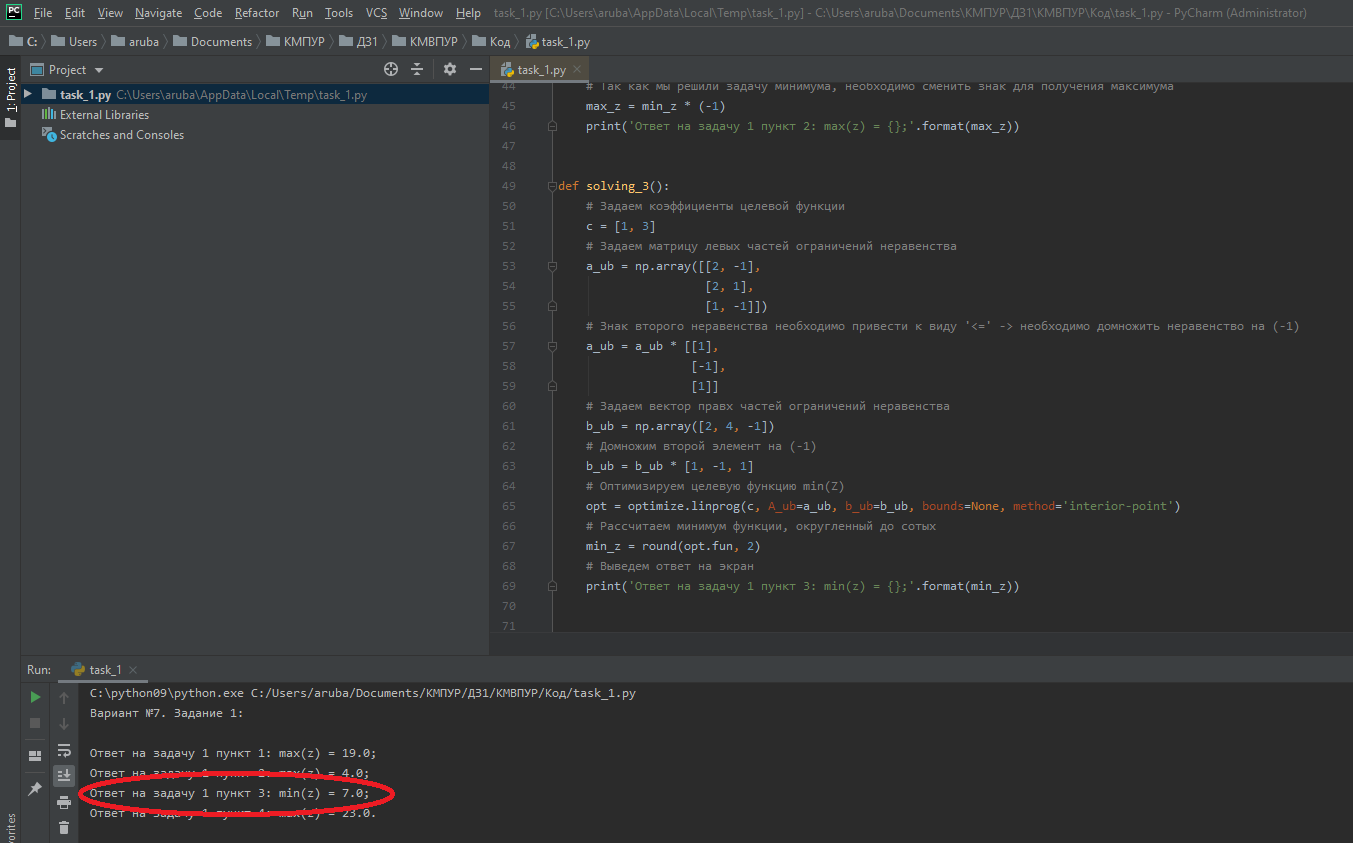


После завершения настроек и выполнения задачи получим следующий ответ:



Значение целевой функции на множестве х={1, 2} равно: 7

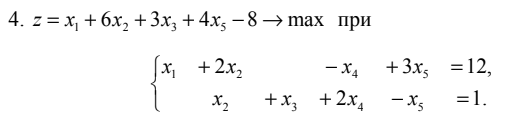
Проверим правильность выполнения нашей задачи с помощью Python:



Как мы видим, наши ответы сошлись. Это говорит нам о том, что данный пункт решен верно.

## **Пункт 7.4**

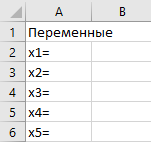
Условие:



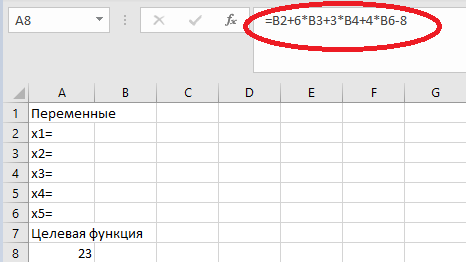
*Рисунок 1 Условие пункта 7.1*

Решение с помощью Excel:

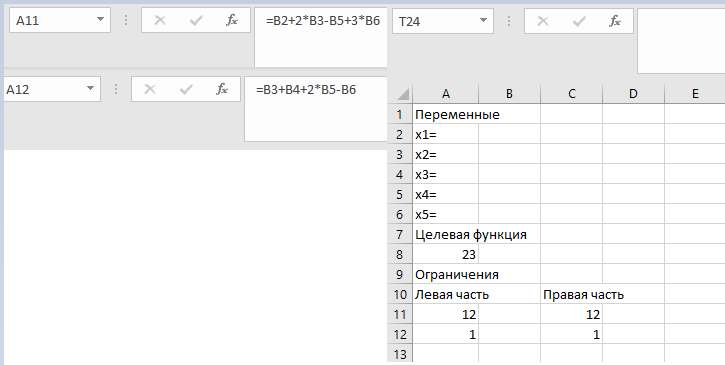
В ячейка B2:B6 создадим область переменных



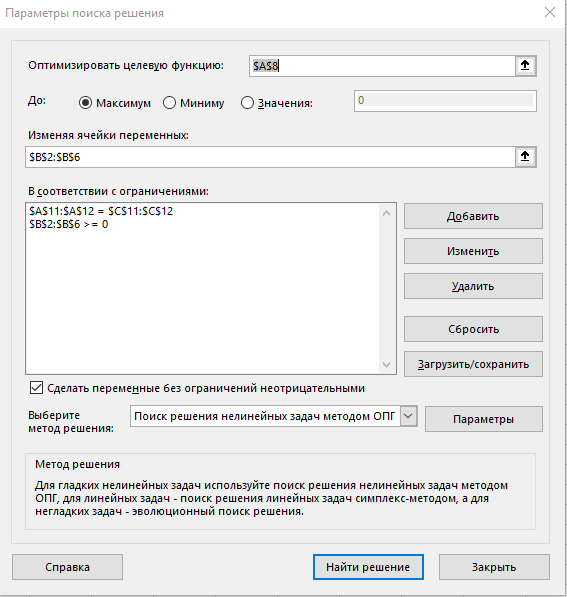
Далее введем формулу вычисления целевой функции в ячейку A8



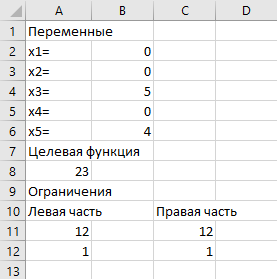
После этого создадим ограничения. Для этого в ячейках А11:А12 будем вычислять левые части ограничений в системе, а в ячейках C11:С12 введем правые части.



Далее для решения задачи вызовем окно диалога «Поиск решения». Выберем ячейку A8 в качестве целевой функции, которую мы будем оптимизировать. Ячейки B2:B6 укажем в качестве переменных. Так как в условии задачи сказано найти max, то выберем данный пункт здесь. Далее укажем ограничения. В соответствии с общим условием, укажем то, что все переменные являются неотрицательными и выберем остальные ограничения.

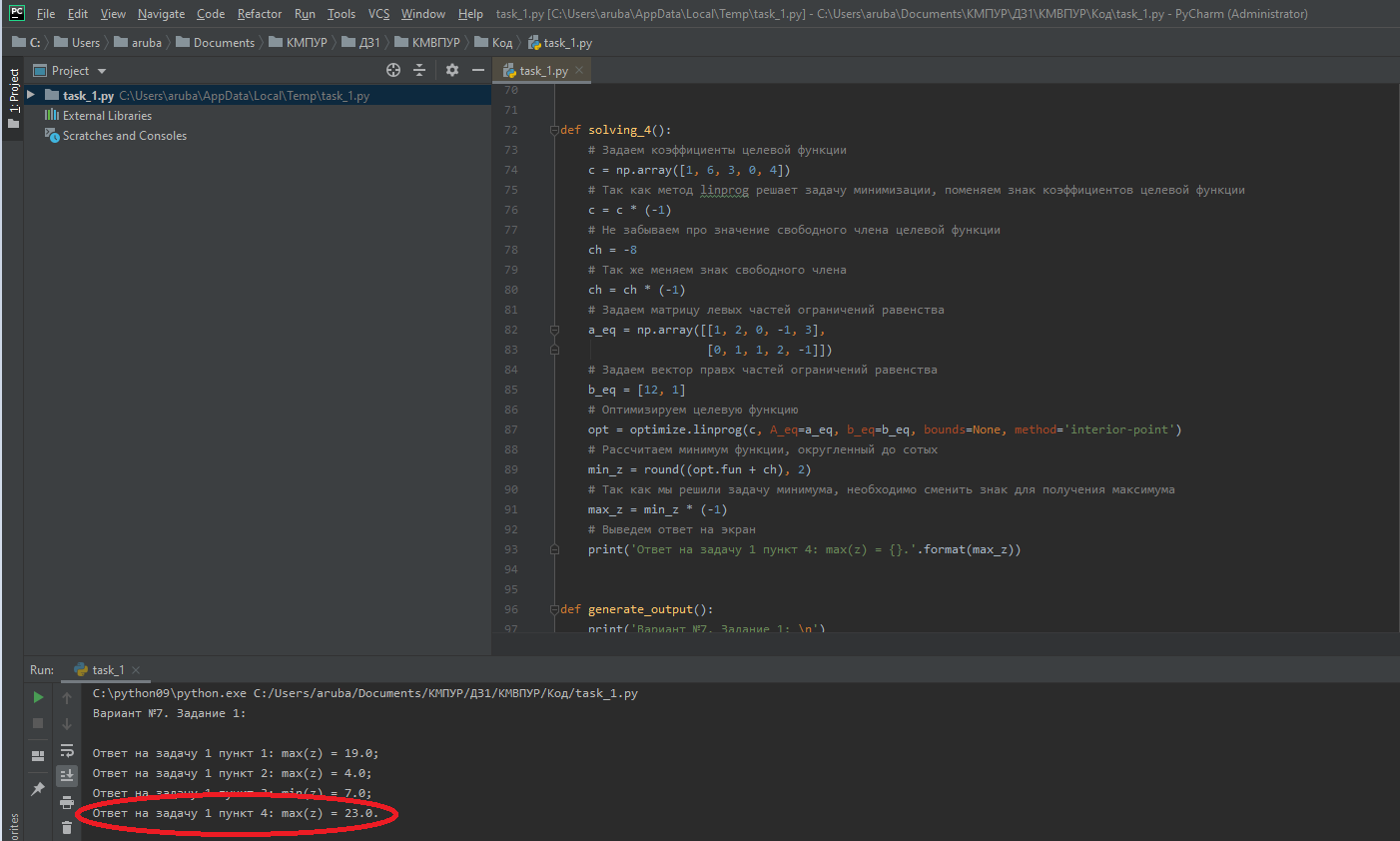


После завершения настроек и выполнения задачи получим следующий ответ:



Значение целевой функции на множестве х={0, 0, 5, 0, 4} равно: 23

Проверим правильность выполнения нашей задачи с помощью Python:

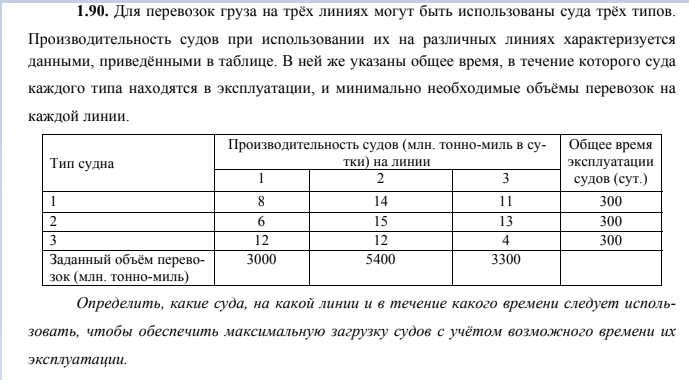


Как мы видим, наши ответы сошлись. Это говорит нам о том, что данный пункт решен верно.

# **Решение задачи 2**

## **Пункт 90**

Условие:



Математическая модель:

Описание переменных:

j – индекс линии

i – индекс судна

Ai – время эксплуатации

Ci – производительность судна i на линии j

Xij – часы, отведенные на работу i судна на j линии

Исходя из условия и заданных переменных составим целевую функцию:

Z = X1\*C1 + X2\*C2 + … + Xn\*Cn → MAX;

Запишем ограничения для судов:

x11 + x12 +…+ x1n ≤ A1,

x21 + x22 +…+ x2n ≤ A2,

…………………….

xm1 + xm2 +…+ xmn ≤ Am.

Запишем ограничения для линий:

x11 + x21 +…+ xm1 ≥ B1,

x12 + x22 +…+ xm2 ≥ B2,

…………………….

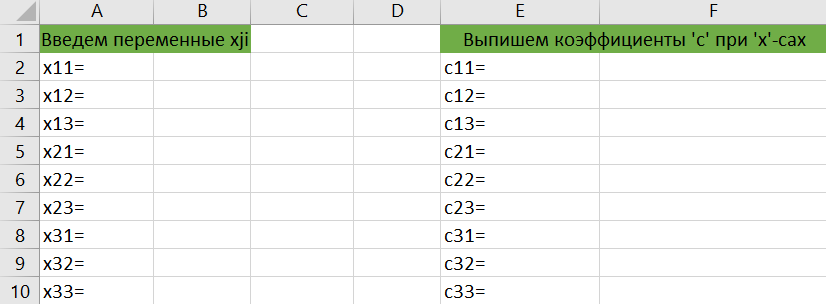
x1n + x2n +…+ xmn ≥ Bn.

Условие неотрицательности:

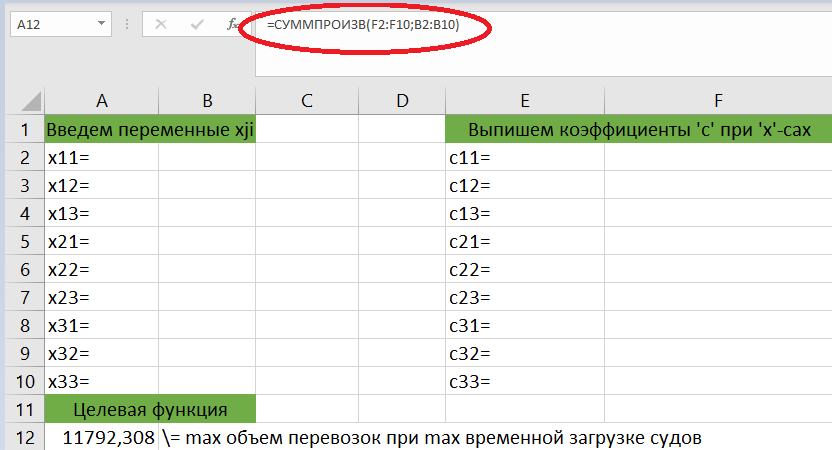
xij ≥ 0, i =1,m , j =1,n .

Решение с помощью Excel:

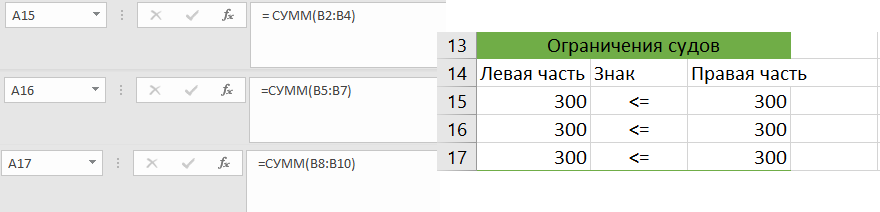
В ячейка B2:B10 и F2:F10 создадим область переменных



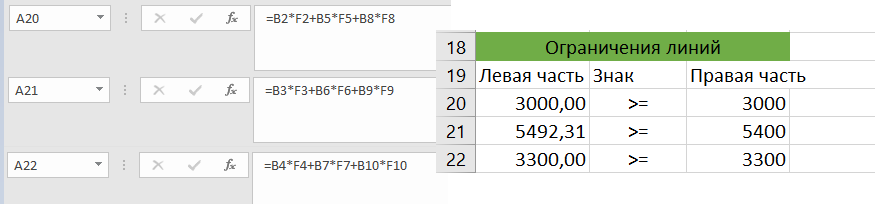
Далее составим целевую функцию в ячейке A12:



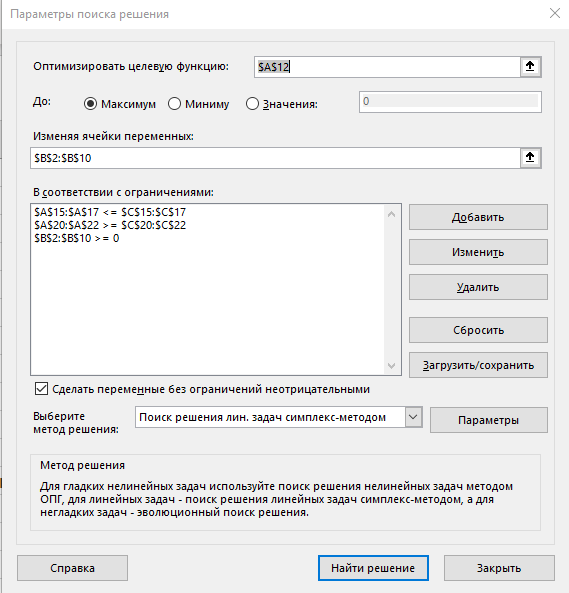
В ячейках A15:A17 создадим ограничения для судов:



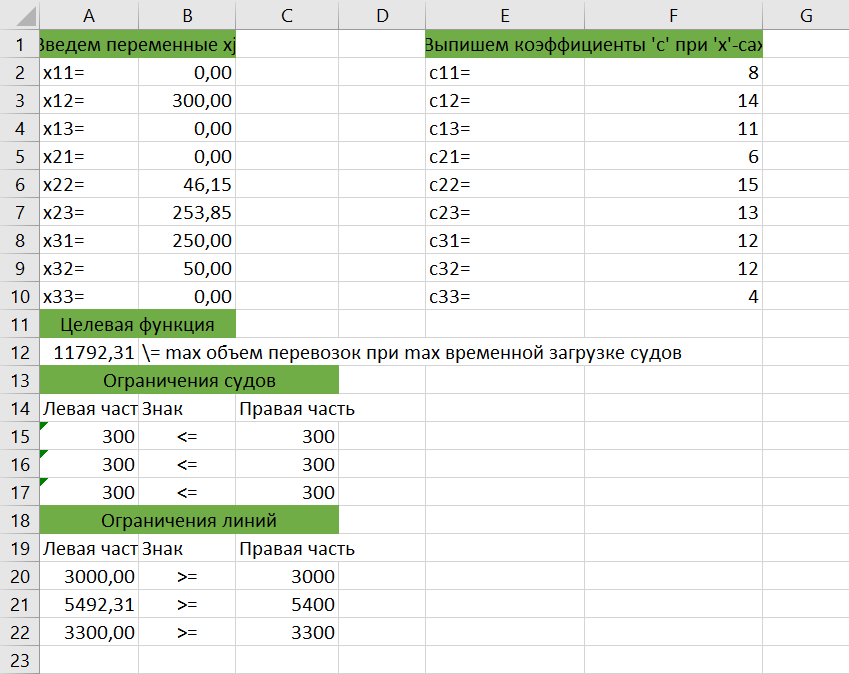
В ячейках A15:A17 создадим ограничения для линий:



Далее для решения задачи вызовем окно диалога «Поиск решения». Выберем ячейку A12 в качестве целевой функции, которую мы будем оптимизировать. Ячейки B2:B10 укажем в качестве переменных. Так как в условии задачи сказано найти max, то выберем данный пункт здесь. Далее укажем ограничения. В соответствии с общим условием, укажем то, что все переменные являются неотрицательными и выберем остальные ограничения.

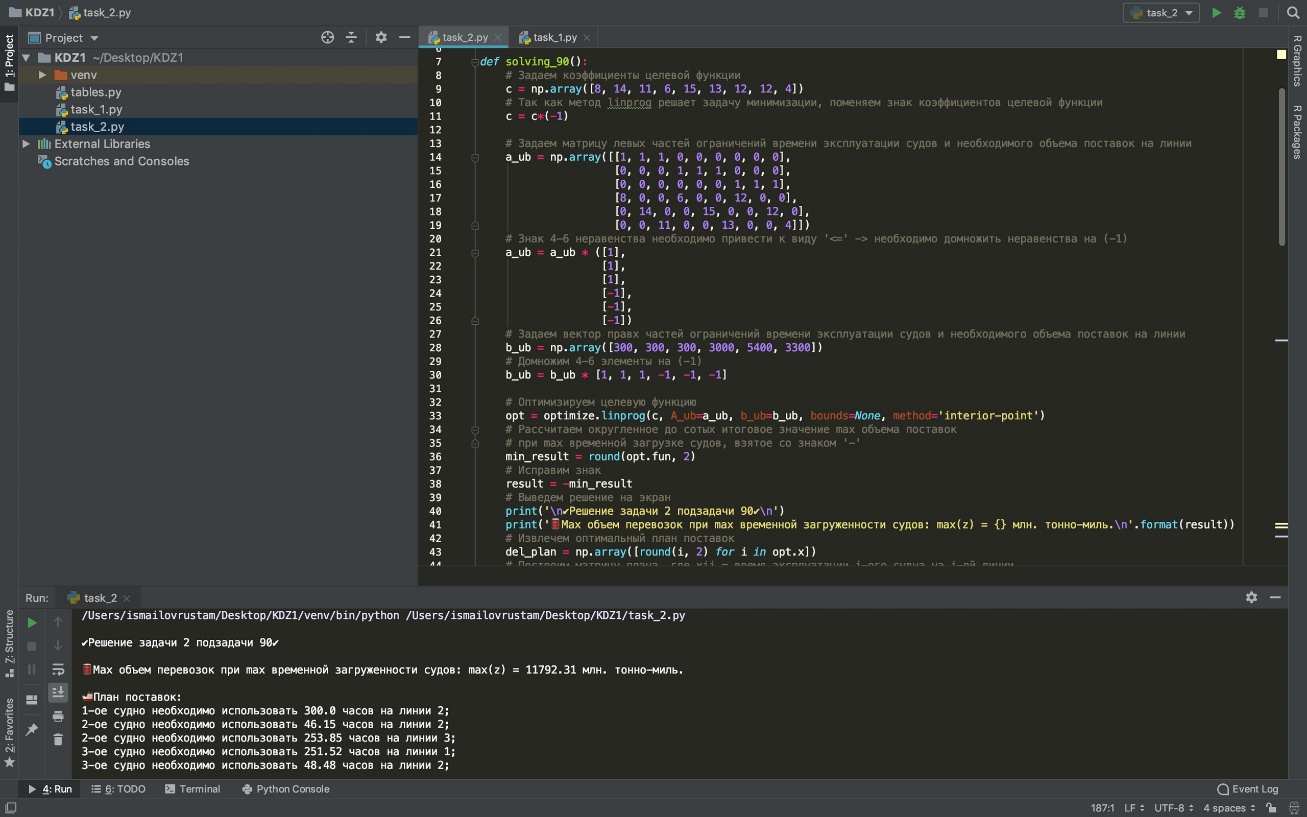


После завершения настроек и выполнения задачи получим следующий ответ:



Значение целевой функции на множестве х={...} равно: 11792,31

Проверим правильность выполнения нашей задачи с помощью Python:



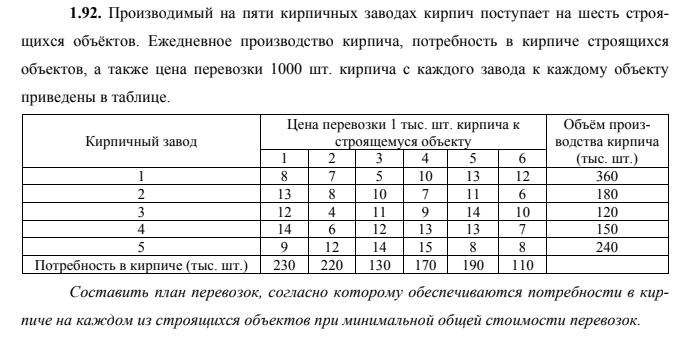
Как мы видим, наши ответы сошлись. Это говорит нам о том, что данный пункт решен верно.

Максимальный объем перевозок при максимальной временной загруженности судов равен 11792.31 млн. тонно-милль. При нем будет действителен следующий план поставок:

|  |  |  |
| --- | --- | --- |
| Номер судна | Часы | Номер линии |
| 1 | 300 | 2 |
| 2 | 46.15 | 2 |
| 2 | 253.85 | 3 |
| 3 | 251.52 | 1 |
| 3 | 48.48 | 2 |

## **Пункт 92**

Условие:



Математическая модель:

Описание переменных:

i – множество строй объектов I = {1,…,6}

j – множество заводов J = {1,…,5}

Ci – цена перевозки 1 тыч.шт. кирпича от завода J на объект I

Xij – количество кирпича, поставляемого от завода J на объект I

Исходя из условия и заданных переменных составим целевую функцию:

Z = X1\*C1 + X2\*C2 + … + Xn\*Cn → MAX;

Запишем ограничения для складов:

x11 + x12 +…+ x1n ≤ A1,

x21 + x22 +…+ x2n ≤ A2,

…………………….

xm1 + xm2 +…+ xmn ≤ Am.

Запишем ограничения для объектов:

x11 + x21 +…+ xm1 ≥ B1,

x12 + x22 +…+ xm2 ≥ B2,

…………………….

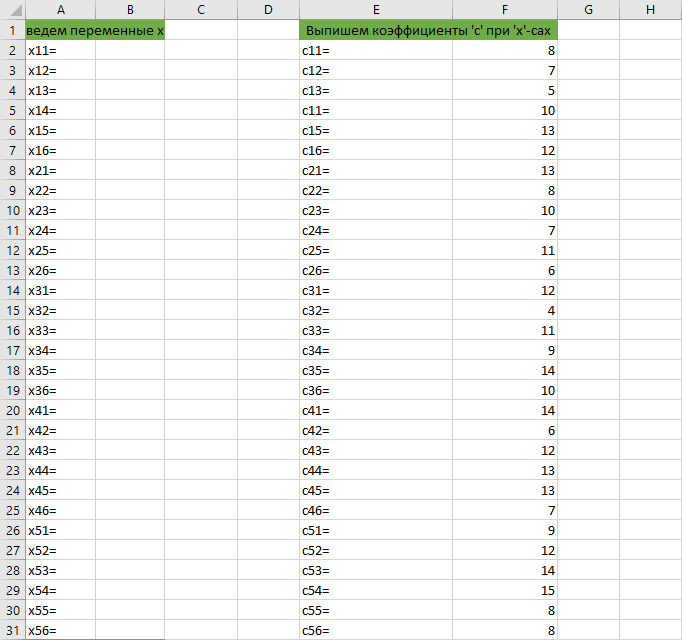
x1n + x2n +…+ xmn ≥ Bn.

Условие неотрицательности:

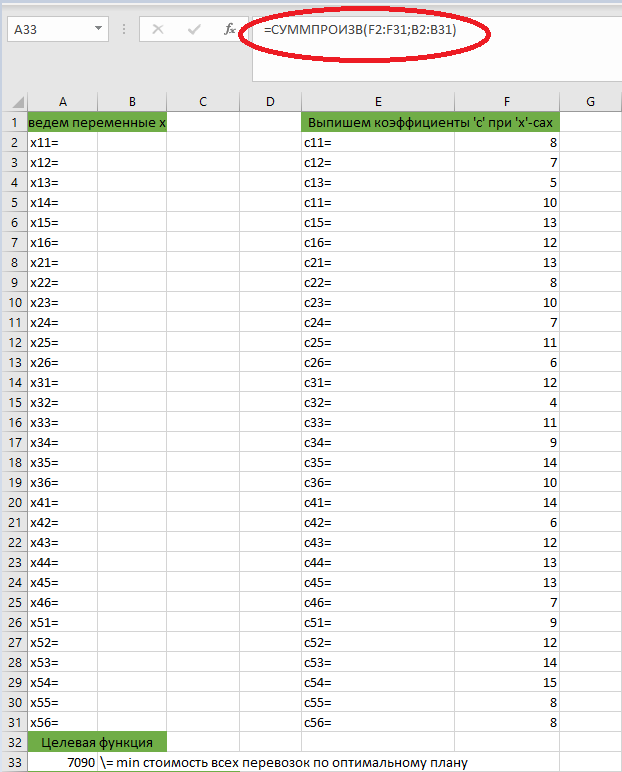
xij ≥ 0, i =1,m , j =1,n .

Решение с помощью Excel:

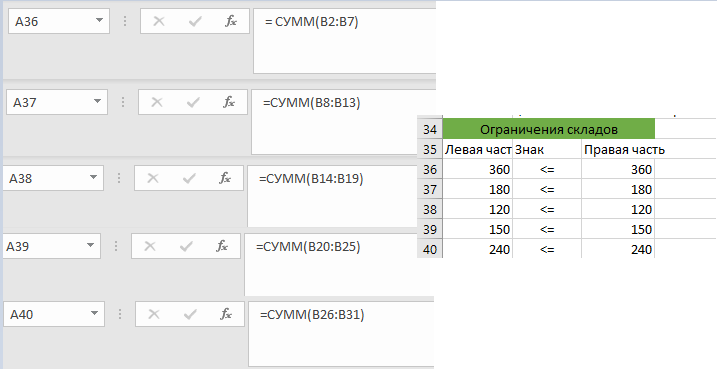
В ячейка B2:B10 и F2:F10 создадим область переменных



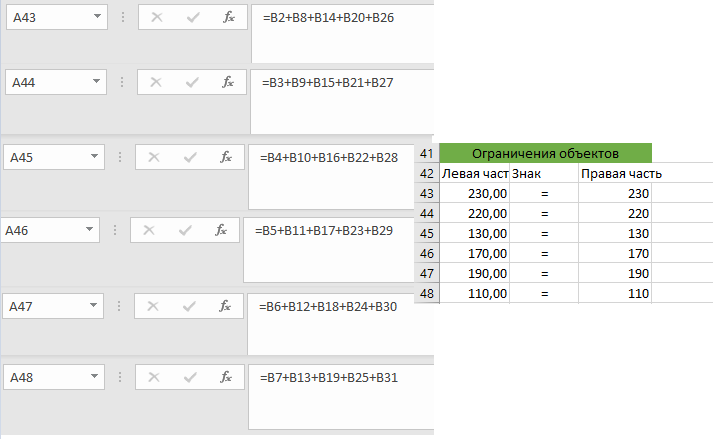
Далее составим целевую функцию в ячейке A33:



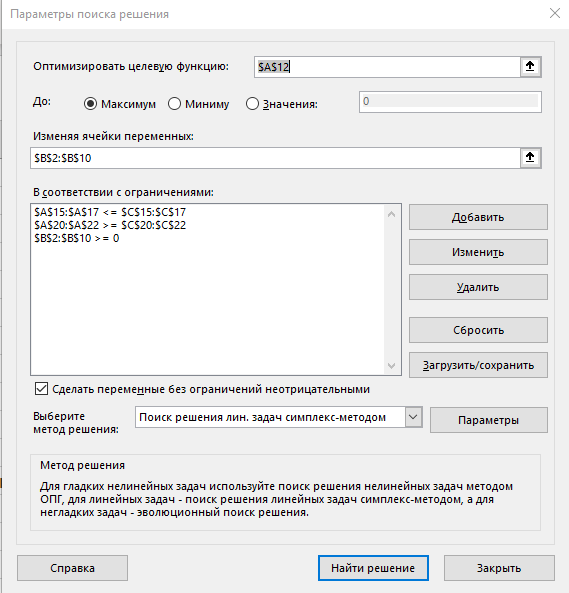
В ячейках A15:A17 создадим ограничения для складов:



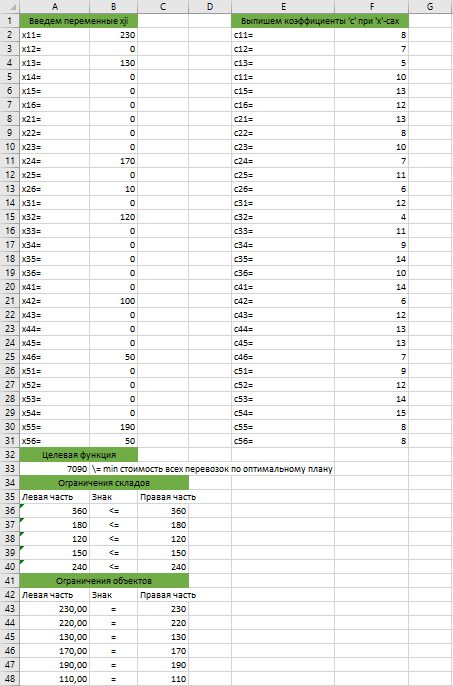
В ячейках A15:A17 создадим ограничения для объектов:



Далее для решения задачи вызовем окно диалога «Поиск решения». Выберем ячейку A33 в качестве целевой функции, которую мы будем оптимизировать. Ячейки B2:B31 укажем в качестве переменных. Так как в условии задачи сказано найти max, то выберем данный пункт здесь. Далее укажем ограничения. В соответствии с общим условием, укажем то, что все переменные являются неотрицательными и выберем остальные ограничения.

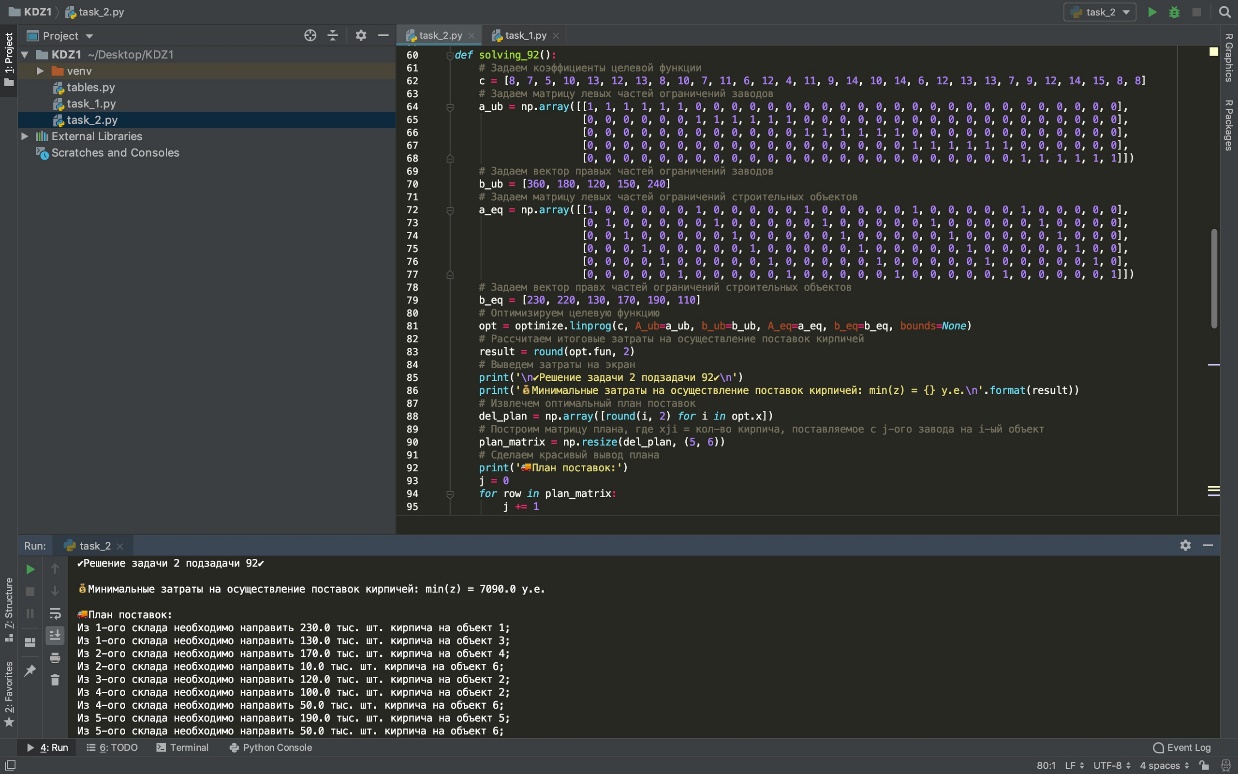


После завершения настроек и выполнения задачи получим следующий ответ:



Значение целевой функции на множестве х={...} равно: 7090

Проверим правильность выполнения нашей задачи с помощью Python:



Стоит отметить, что представлен оптимальный план перевозок, согласно которому обеспечиваются потребности в каждом из строящихся объектов при минимальной общей стоимости перевозок.

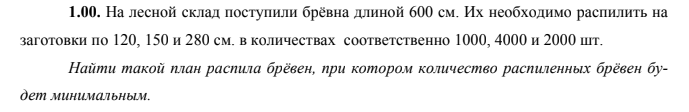
Как мы видим, наши ответы сошлись. Это говорит нам о том, что данный пункт решен верно.

Минимальные затраты на осуществление поставок кирпичей будут равняться 7090 у.е. Следует придерживаться следующего плана поставок:

|  |  |  |
| --- | --- | --- |
| Номер склада | Тыс. шт. кирпича | Номер объекта |
| 1 | 230 | 1 |
| 1 | 130 | 3 |
| 2 | 170 | 4 |
| 2 | 10 | 6 |
| 3 | 120 | 2 |
| 4 | 100 | 2 |
| 4 | 50 | 6 |
| 5 | 190 | 5 |
| 5 | 50 | 6 |

## **Пункт 100**

Условие:



Математическая модель:

Xi – количество бревен разрезанных по I варианту

Исходя из условия и заданных переменных составим целевую функцию:

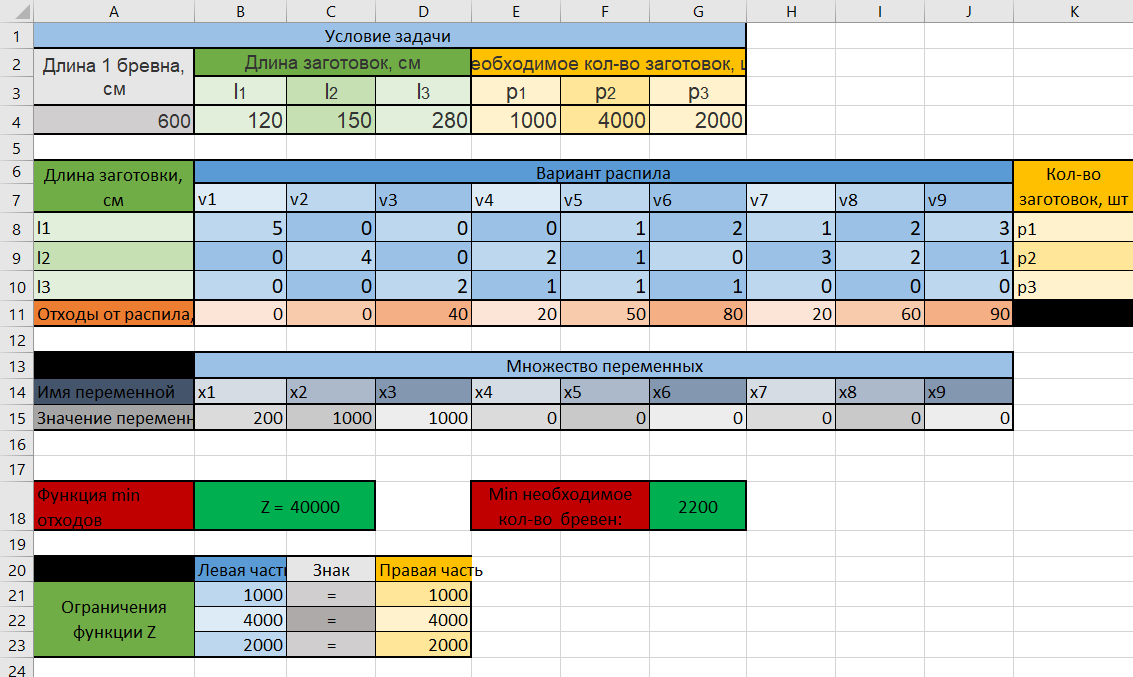
Z = X1\*C1 + X2\*C2 + … + Xn\*Cn → MAX;

Условие неотрицательности:

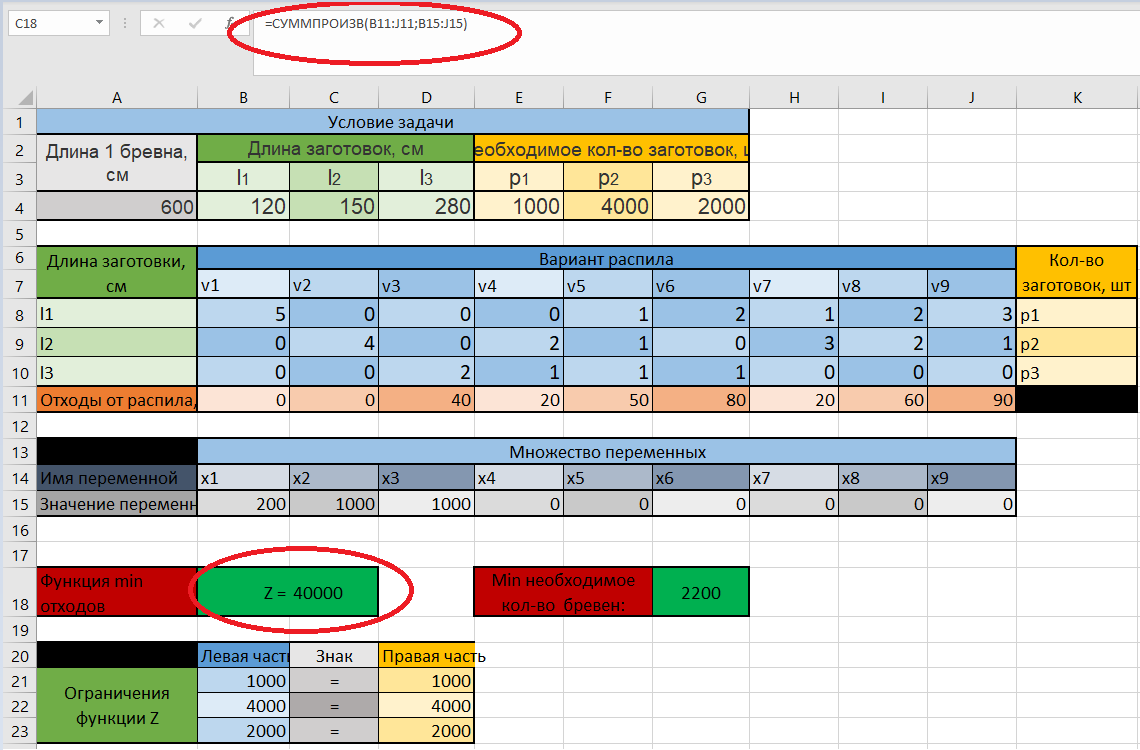
xij ≥ 0, i =1,m , j =1,n .

Решение с помощью Excel:

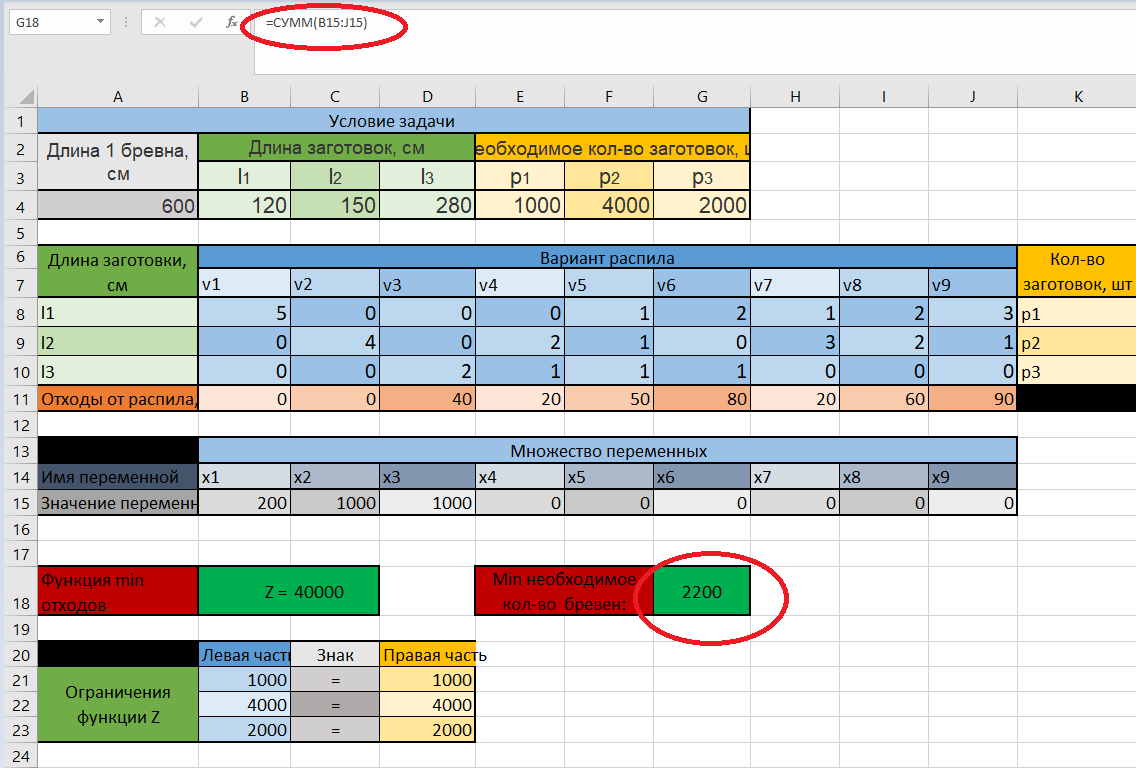
Создадим шаблон нашей задачи.



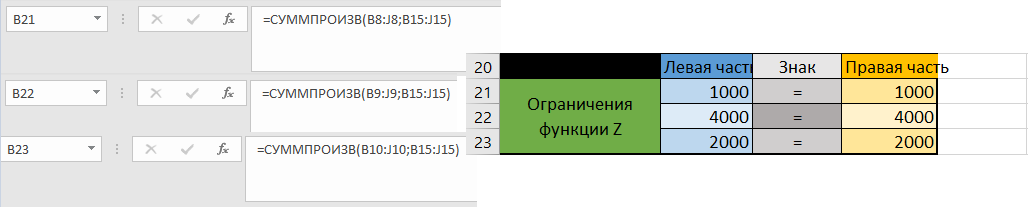
Запишем функцию min отходов:



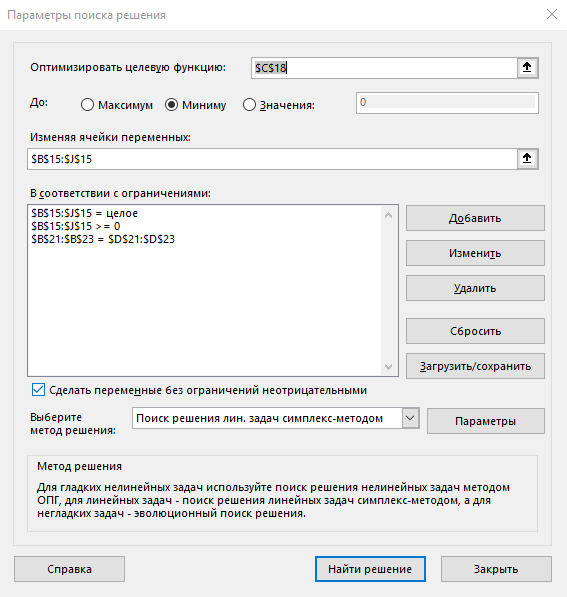
А также укажем количество минимально необходимых бревен:



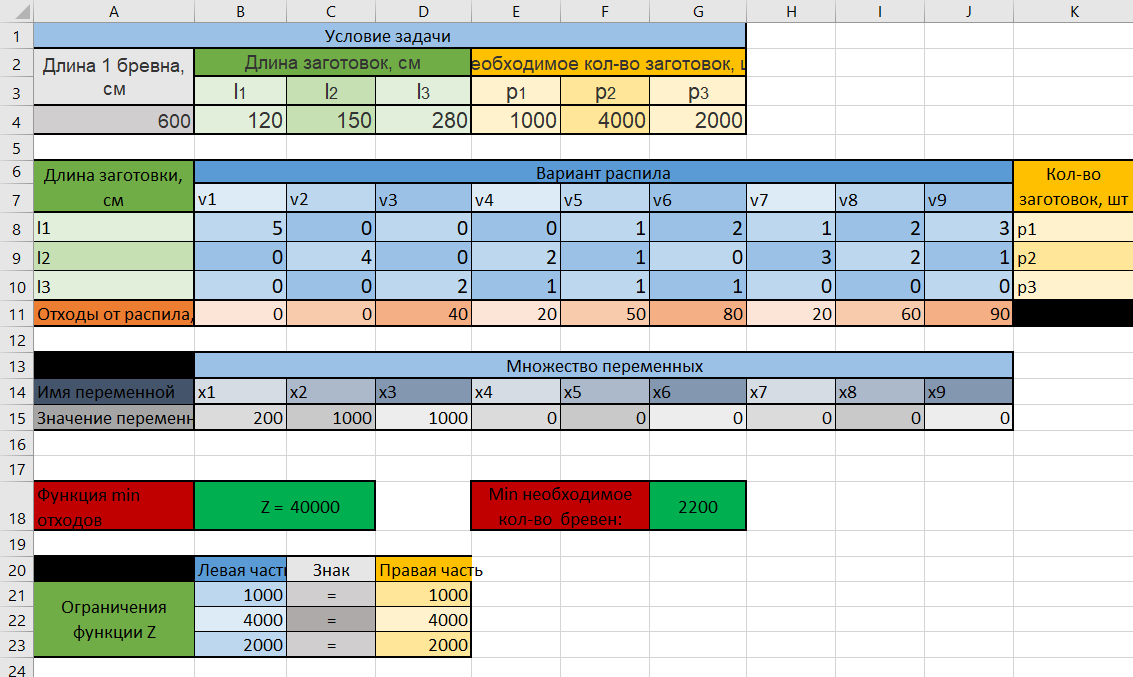
Введем ограничения по условию:



Далее для решения задачи вызовем окно диалога «Поиск решения». Выберем ячейку A12 в качестве целевой функции, которую мы будем оптимизировать. Ячейки B15:J15 укажем в качестве переменных. Так как в условии задачи сказано найти max, то выберем данный пункт здесь. Далее укажем ограничения. В соответствии с общим условием, укажем то, что все переменные являются неотрицательными и выберем остальные ограничения.

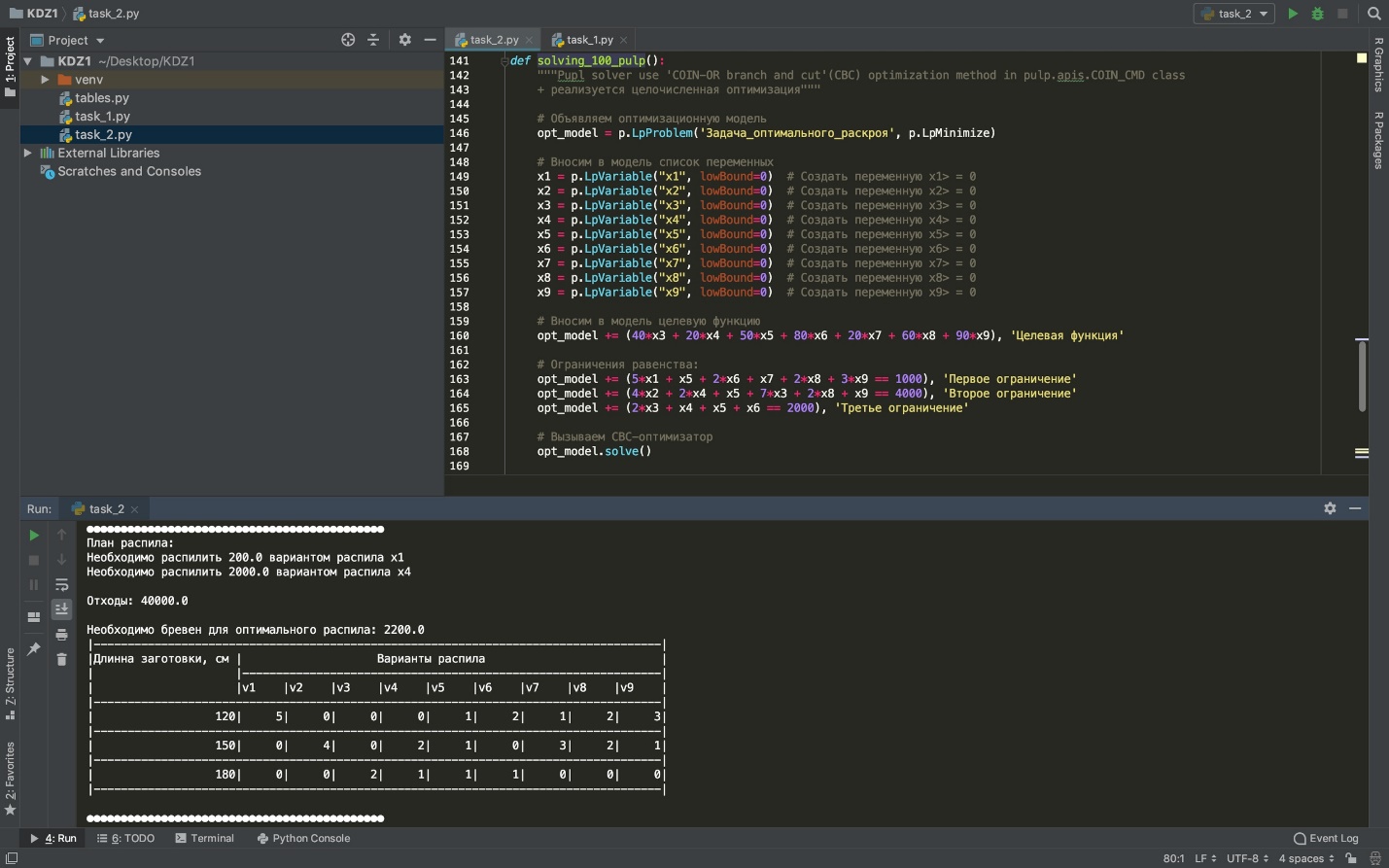


На итог получим следующее решение:



Значение целевой функции на множестве х={...} равно: 40000

Проверим правильность выполнения нашей задачи с помощью Python:



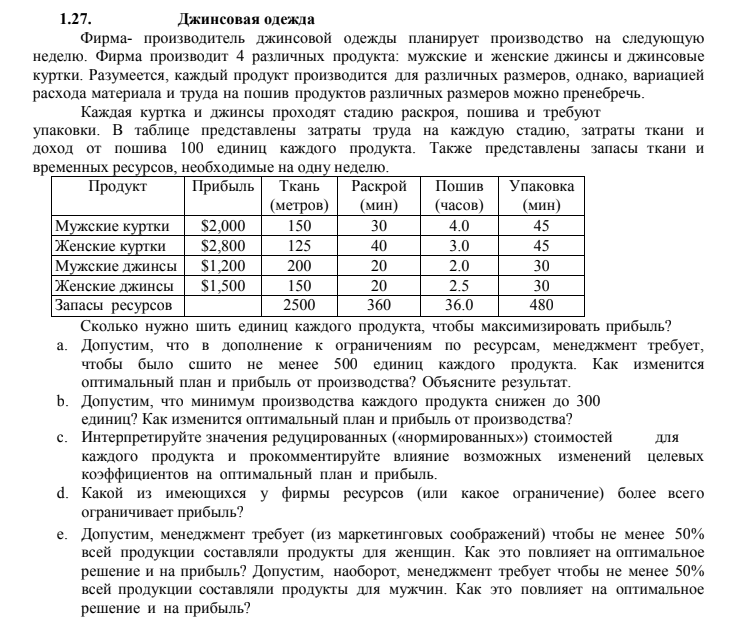
Стоит отметить, что представлен оптимальный план распила бревен, при котором количество распиленных бревен будет минимальным.

Как мы видим, наши ответы сошлись. Это говорит нам о том, что данный пункт решен верно.

Следует придерживаться следующего плана распила. Необходимо распилить 200 первым варианта распила и 2000 четвертым вариантом распила. Отходы будут равны 40000. Необходимо 2200 бревен для оптимального распила.

# **Решение задачи 3**

Условие:



Математическая модель:

Описание переменных:

j – индекс вида продукции, j = {1, 2, 3, 4}

i – индекс вида ресурсов, i = {1, 2, 3, 4}

aij – затраты ресурсом i-ого типа продукции на производство j-ой продукции

Ai – ограничение на именующийся объем ресурсов i-ого вида

Pj – прибыль от реализации j-ого продукта

Xj – объем продукции j-ого типа в плане

Исходя из условия и заданных переменных составим целевую функцию:

z = Р1x1 + Р2x2 + … + Pnxn → max;

Запишем ограничения:

а11x1 + а12x2 +… + а1nxn ≤ A1,

а21x1 + а22x2 +… + а2nxn ≤ A2,

…………………………….

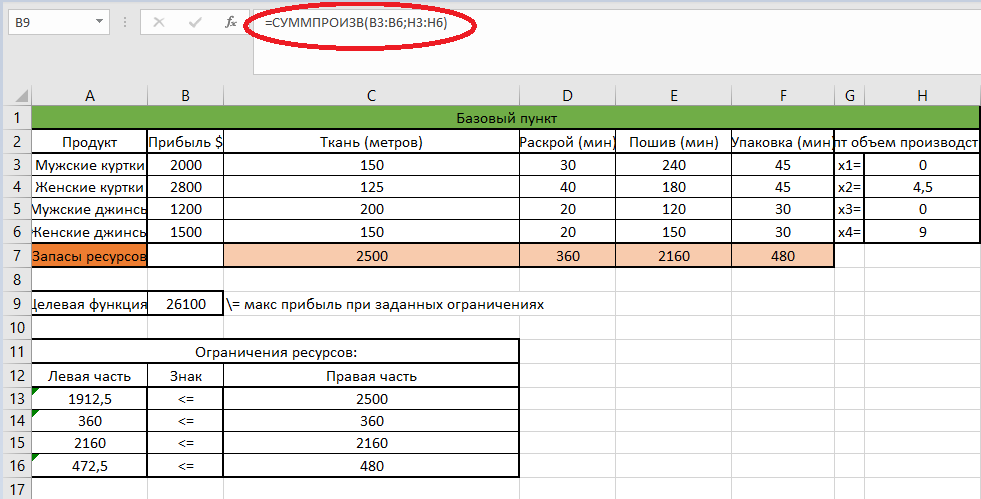
am1x1 + аm2x2 +…+ аmnxn ≤ Am,

Условие неотрицательности:

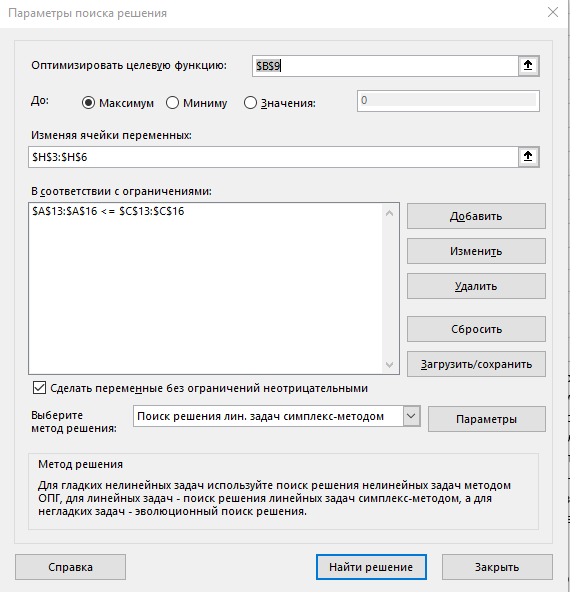
xij ≥ 0, i =1,m , j =1,n .

Решение с помощью Excel:

Запишем исходные условия и составим целевую функцию, ограничив наши ресурсы:



Далее с помощью функции «Поиск решений» найдем целевую функцию:



Ответ:



Значение целевой функции на множестве х={...} равно: 26100

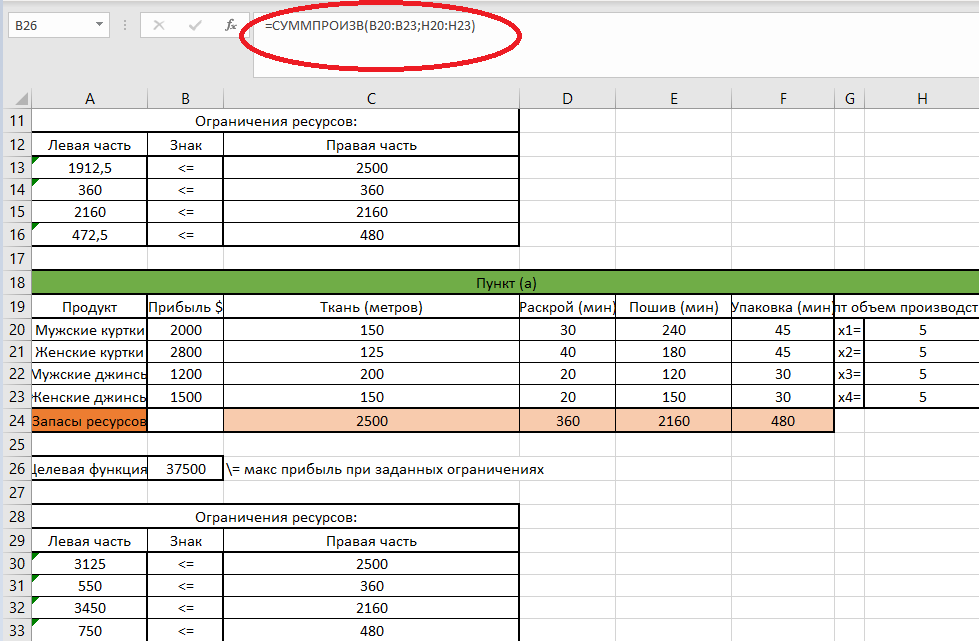
Проверим правильность выполнения нашей задачи с помощью Python:

Изображение выглядит как снимок экрана, монитор, ноутбук, экран

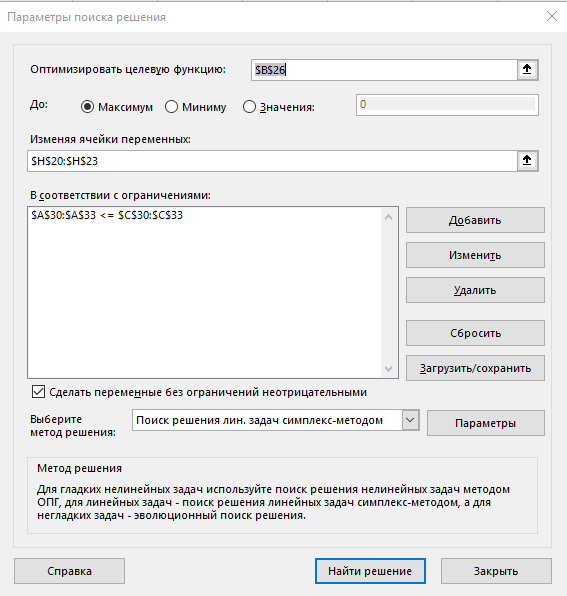
Автоматически созданное описание

Оптимальный план и значение целевой функции совпадают

## **Пункт а)**



Далее с помощью функции «Поиск решений» найдем целевую функцию:



Ответ:



Значение целевой функции на множестве х={...} равно: 37500

Очевидно, что в условиях наложенных ограничений на минимум продукции каждого вида, равный 500 единиц, невозможно произвести их оптимальный объем, уложившись в уже имеющиеся ограничения ресурсов, что мы и видим в решении данного пункта симплекс-методом. Таким же очевидным является и тот факт, что значение целевой функции вырастет в связи с увеличением числа производимой продукции и прибыли от ее реализации соответственно, однако, это повлечет и дополнительные издержки на увеличение производских мощностей // либо издержки от простоя прилавков из-за нарушения цикла производства одежды (заданный объем при заданных мощностях невозможно произвести за 1 неделю).

Таким образом, общая рекомендация менеджерскому составу состоит в уменьшении целевых показателей на производимую продукцию.

Более того, еще одним аргументом в пользу данного совета является снижение прибыли на единицу продукции. Не трудно рассчитать, что в изначальном плане единица продукции приносила 19,33 $ за единицу (=26100 / (9+4,5)\*100), а в плане с наложенным ограничением на минимум произведенной продукции каждого вида = 500 единица продукции приносит 18,75$ за единицу (=37500/ (5\*4)\*100) => для выполнения условия максимизации прибыли необходимо снизить целевые показатели на объем продукции.

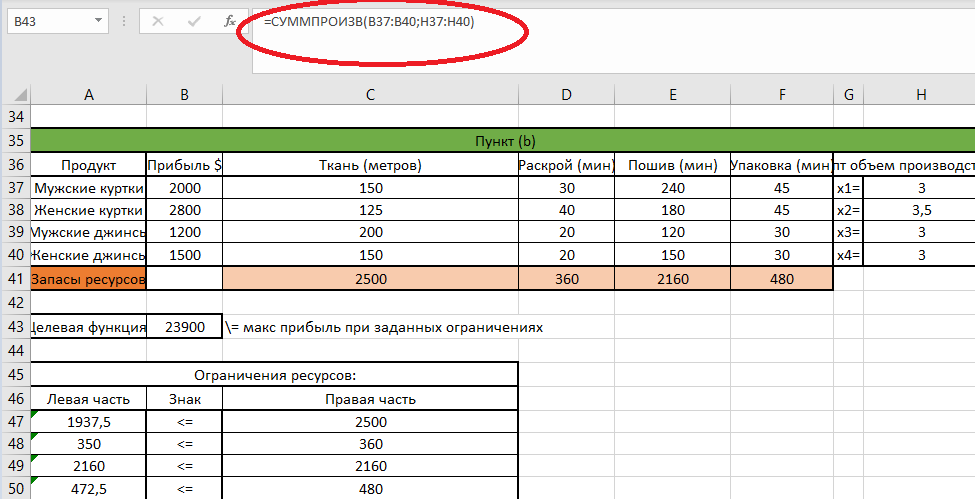
Проверим правильность выполнения нашей задачи с помощью Python:

Изображение выглядит как снимок экрана, монитор, ноутбук, компьютер

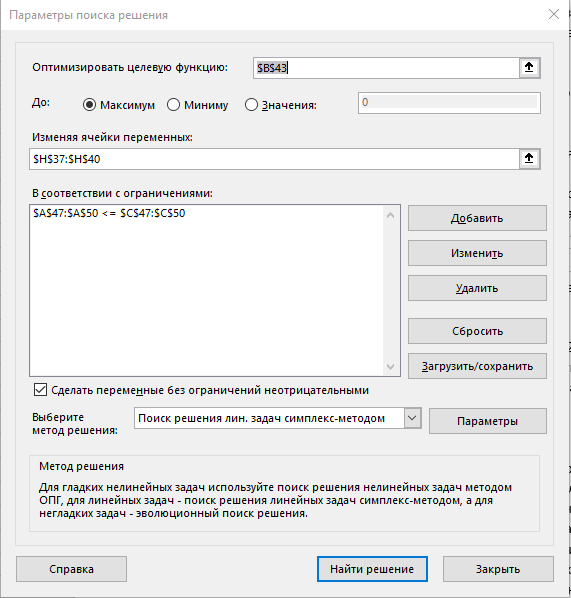
Автоматически созданное описание

Как видно, метод опорной точки так же не дает оптимального решения в связи с его отсутствием на заданном множестве ограничений.

## **Пункт б)**



Далее с помощью функции «Поиск решений» найдем целевую функцию:



Ответ:



Значение целевой функции на множестве х={...} равно: 23900

Оптимальный план изменился по сравнению с оптимальным значительно и представляет собой примерно равномерное распределение объема производимой продукции по нижней границе выставленного ограничения в 300 единиц каждого вида.

Говоря о прибыли, в абсолютном исчислении ее объем снизился на 2200$, а относительная стоимость производства единицы продукции снизилась с уже известных 19,33$ до 19,12$ (=23900/ (3\*3+3,5)\*100) => дельта составила незначительные 0,21$ на единицу продукции или 0,011% от оптимальной прибыли с единицы продукции.

Общая рекомендация состоит в сосредоточивании производственных мощностей на производство продукции для женщин (в объемах, представленных в оптимальном плане) на заданном рынке с заданными ценами на реализацию продукции. Если же менеджерских состав настаивает на реализации текущего плана, необходимо обозначить, что ежегодные потери от реализации плана с заданным набором ограничений составят 114400$ (=2200\*52) в абсолютном исчислении и 8,4% от потенциальной оптимальной прибыли (=(1-(23900\*52)/(26100\*52))\*100)). На недельном масштабе потери от отклонения от оптимального плана действительно выглядят незначительно, однако на продолжительном промежутке приобретают весомый масштаб. В связи с этим рекомендуется настаивать на реализации оптимального плана6 либо дальнейшем снижении целевых показателей на объем выпускаемой продукции (в частности мужской одежды)

Проверим правильность выполнения нашей задачи с помощью Python:

Изображение выглядит как снимок экрана, монитор, ноутбук, экран

Автоматически созданное описание

Оптимальный план и значение целевой функции совпадают

## **Пункт с)**

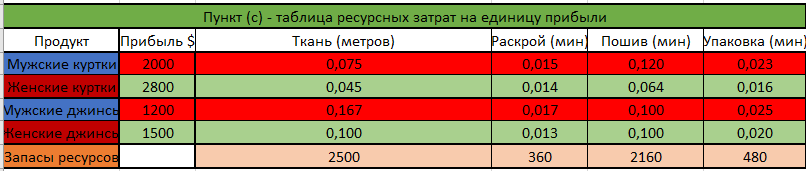


Таблица показывает сколько конкретного типа ресурса нужно на производстве конкретной продукции для получения единицы прибыли от реализации этого продукта.

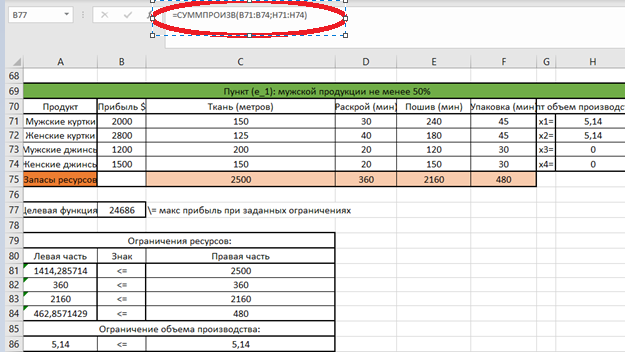
Не трудно увидеть, что по всем типам ресурсов требуется меньше ресурсных затрат на получение единицы прибыли при производстве женской продукции (отсюда и отсутствие мужской продукции в изначальном оптимальном плане) (т.е. производство любой единицы мужской продукции требует относительно бОльших ресурсных затрат, приведенных к единице генерируемой ими прибыли, чем аналогичные показатели при производстве женской продукции).

О влиянии целевых коэффициентов: увеличении(снижение) прибыли от реализации мужской(женской) продукции способно значительно влиять на оптимальный план по мере "появления зеленых ячеек напротив мужской продукции в таблице, приведенной слева" или иначе: по мере уменьшения необходимых ресурсов на производство мужской продукции для получения единицы прибыли // по мере увеличения отдачи от использования единицы ресурса на производстве мужской продукции

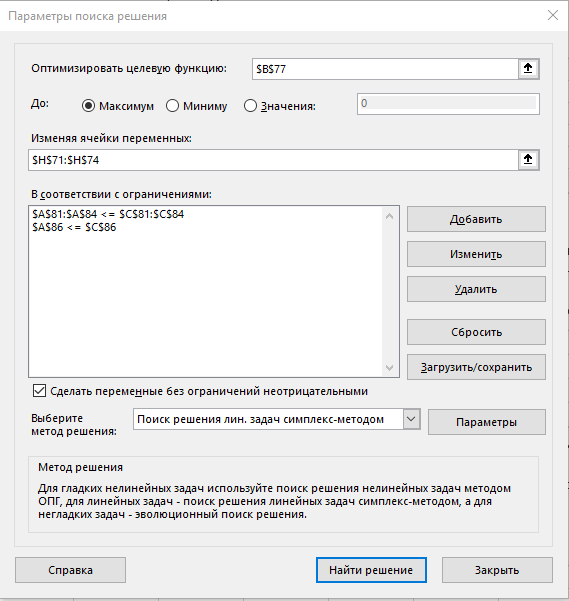
## **Пункт d)**

Проанализировав таблицу ограничений оптимального плана можно сделать вывод о том, что ограничения на время раскроя и пошива больше всего ограничивают прибыль (так как неравенства этих ограничений обращены в равенства). Причем ограничение времени раскроя имеет бОльшее влияние, нежели ограничение пошива (это можно увидеть, увеличив соответствующие ограничение на 10% и сравнив результат функции прибыли: при увеличении времени раскроя на 10%, прибыль растет до 28180, а при аналогичном увеличении времени пошива прибыль растет лишь до 26204)

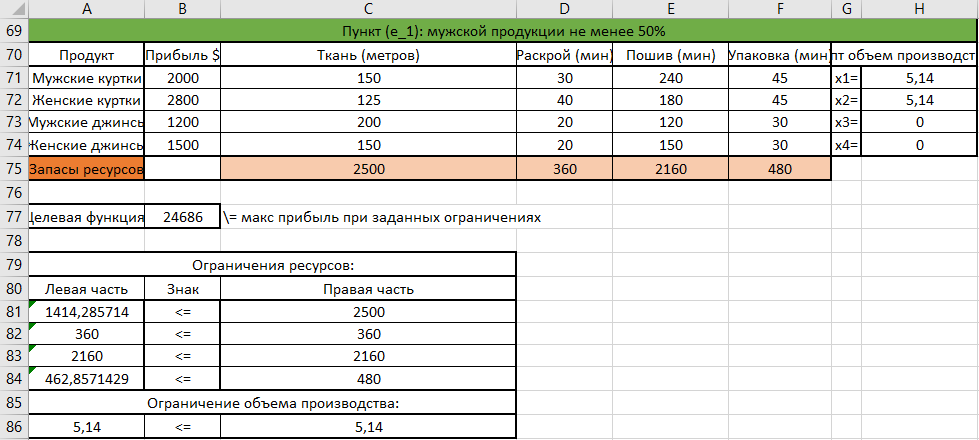
## **Пункт e\_1)**



Далее с помощью функции «Поиск решений» найдем целевую функцию:



Ответ:



Значение целевой функции на множестве х={...} равно: 24686

Как мы видим, условие "не менее 50% всей продукции составляли продукты для мужчин" выполняется по нижней границе, т.е. производство распределилось ровно пополам между продукцией для мужчин и для женщин, что еще раз подтверждает вывод о том, что глобально производство мужской продукции является менее выгодным, нежели производство товаров для женщин. Общая прибыль сократилась на 1414$, относительная прибыльность единицы продукции выросла на 4,68$ до 24,01$ за счет снижения общих объемов производства.

Пункт (e\_2): женской продукции не менее 50%. Нет смысла решать данный пункт отдельно, так как в изначальном оптимальном плане условие "не менее 50% всеq продукции составляли продукты для женщин" уже выполняется, так как 100% выпускаемой продукции является продукцией для женщин. Следовательно, выводов об изменениях в сравнении с оптимумом быть не может, так как это и только это решение является верным с точки зрения максимизации прибыли.

Проверим правильность выполнения e\_1 с помощью Python

Изображение выглядит как снимок экрана, монитор, ноутбук, телефон

Автоматически созданное описание

Оптимальный план и значение целевой функции совпадают

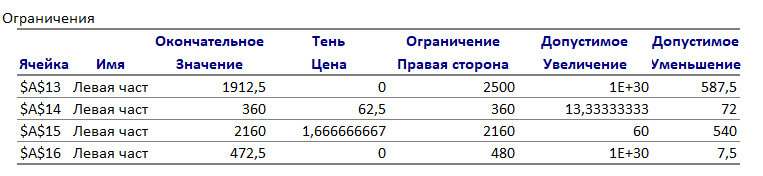
## **Отчет об устойчивости:**

Это математические решение пунктов с и d.



1. Из столбца приведенная стоимость можно извлечь знания о том, что случится с целевой функцией при насильном включении в оптимальный план единицы той или иной продукции. Видно, что добавление 1 единицы мужской куртки ведет к снижению общей прибыли на 275 единиц, а мужских джинсов на 250

2. Столбцы допустимое увеличение(уменьшение) отражают диапазоны устойчивости или то, на сколько мы можем изменить значение соответствующего коэффициента, чтобы значение целевой функции не изменилось. Таким образом, имеем следующие диапазоны устойчивости: x1 = (-∞; 2275), x2 = (1800; 3000), x3 = (-∞; 1450), x4 = (1400; 2333,3(3)). Исходя из этого, нам необходимо изменить целевые коэффициенты на значения вне диапазонов устойчивости, если нам необходимо повлиять на оптимальный план выпуска продукции.



1. Столбец "Теневая цена" отражает изменение целевой функции при увеличении соответствующего ресурса на единицу. Легко увидеть, что ресурсы "раскрой" и "пошив" являются дефицитными и их увеличение способно улучшать показатели прибыли, при этом время раскроя значительно сильнее влияет на целевую функцию, нежели время пошива.

2. Столбцы допустимое увеличение(уменьшение) отражают диапазоны устойчивости или то, на сколько мы можем изменить объем соответствующего ресурса, чтобы значение целевой функции не изменилось. Таким образом, имеем следующие диапазоны устойчивости: res1 = (1912,5; +∞), res2 = (288; 373,3(3)), res3 = (2160; 2220), res4 = (472,5; +∞). Исходя из этого, нам необходимо изменить целевые коэффициенты на значения вне диапазонов устойчивости, если нам необходимо повлиять на оптимальный план выпуска продукции.

# **Приложение 1**

Task\_1.py

import numpy as np  
from scipy import optimize  
  
  
def solving\_1():  
 # Задаем коэффициенты целевой функции  
 c = np.array([3, -3, -1, 1, 0])  
 # Так как метод linprog решает задачу минимизации, поменяем знак коэффициентов целевой функции  
 c = c \* (-1)  
 # Не забываем про значение свободного члена целевой функции  
 ch = 14  
 # Так же меняем знак свободного члена  
 ch = ch \* (-1)  
 # Задаем матрицу левых частей ограничений равенства  
 a\_eq = np.array([[-2, 1, 1, 0, 0],  
 [4, 3, 0, -1, 0],  
 [8, -1, 0, 0, 1]])  
 # Задаем вектор правх частей ограничений равенства  
 b\_eq = [2, 16, 16]  
 # Оптимизируем целевую функцию min(Z)  
 opt = optimize.linprog(c, A\_eq=a\_eq, b\_eq=b\_eq, bounds=None, method='interior-point')  
 # Рассчитаем минимум функции, округленный до сотых  
 min\_z = round((opt.fun + ch), 2)  
 # Так как мы решили задачу минимума, необходимо сменить знак для получения максимума  
 max\_z = min\_z \* (-1)  
 # Выведем ответ на экран  
 print('Ответ на задачу 1 пункт 1: max(z) = {};'.format(max\_z))  
  
  
def solving\_2():  
 # Задаем коэффициенты целевой функции  
 c = np.array([1, 1, 1, 1])  
 # Так как метод linprog решает задачу минимизации, поменяем знак коэффициентов целевой функции  
 c = c \* (-1)  
 # Задаем матрицу левых частей ограничений равенства  
 a\_eq = np.array([[1, 3, 7, -1],  
 [1, -1, -1, 3]])  
 # Задаем вектор правх частей ограничений равенства  
 b\_eq = [6, 2]  
 # Оптимизируем целевую функцию min(Z)  
 opt = optimize.linprog(c, A\_eq=a\_eq, b\_eq=b\_eq, bounds=None, method='interior-point')  
 # Рассчитаем минимум функции, округленный до сотых  
 min\_z = round(opt.fun, 2)  
 # Так как мы решили задачу минимума, необходимо сменить знак для получения максимума  
 max\_z = min\_z \* (-1)  
 print('Ответ на задачу 1 пункт 2: max(z) = {};'.format(max\_z))  
  
  
def solving\_3():  
 # Задаем коэффициенты целевой функции  
 c = [1, 3]  
 # Задаем матрицу левых частей ограничений неравенства  
 a\_ub = np.array([[2, -1],  
 [2, 1],  
 [1, -1]])  
 # Знак второго неравенства необходимо привести к виду '<=' -> необходимо домножить неравенство на (-1)  
 a\_ub = a\_ub \* [[1],  
 [-1],  
 [1]]  
 # Задаем вектор правх частей ограничений неравенства  
 b\_ub = np.array([2, 4, -1])  
 # Домножим второй элемент на (-1)  
 b\_ub = b\_ub \* [1, -1, 1]  
 # Оптимизируем целевую функцию min(Z)  
 opt = optimize.linprog(c, A\_ub=a\_ub, b\_ub=b\_ub, bounds=None, method='interior-point')  
 # Рассчитаем минимум функции, округленный до сотых  
 min\_z = round(opt.fun, 2)  
 # Выведем ответ на экран  
 print('Ответ на задачу 1 пункт 3: min(z) = {};'.format(min\_z))  
  
  
def solving\_4():  
 # Задаем коэффициенты целевой функции  
 c = np.array([1, 6, 3, 0, 4])  
 # Так как метод linprog решает задачу минимизации, поменяем знак коэффициентов целевой функции  
 c = c \* (-1)  
 # Не забываем про значение свободного члена целевой функции  
 ch = -8  
 # Так же меняем знак свободного члена  
 ch = ch \* (-1)  
 # Задаем матрицу левых частей ограничений равенства  
 a\_eq = np.array([[1, 2, 0, -1, 3],  
 [0, 1, 1, 2, -1]])  
 # Задаем вектор правх частей ограничений равенства  
 b\_eq = [12, 1]  
 # Оптимизируем целевую функцию  
 opt = optimize.linprog(c, A\_eq=a\_eq, b\_eq=b\_eq, bounds=None, method='interior-point')  
 # Рассчитаем минимум функции, округленный до сотых  
 min\_z = round((opt.fun + ch), 2)  
 # Так как мы решили задачу минимума, необходимо сменить знак для получения максимума  
 max\_z = min\_z \* (-1)  
 # Выведем ответ на экран  
 print('Ответ на задачу 1 пункт 4: max(z) = {}.'.format(max\_z))  
  
  
def generate\_output():  
 print('Вариант №7. Задание 1: \n')  
 solving\_1()  
 solving\_2()  
 solving\_3()  
 solving\_4()  
  
  
generate\_output()

# **Приложение 2**

Task\_2.py

import numpy as np  
from scipy import optimize  
import pulp as p  
import tables  
  
  
def solving\_90():  
 # Задаем коэффициенты целевой функции  
 c = np.array([8, 14, 11, 6, 15, 13, 12, 12, 4])  
 # Так как метод linprog решает задачу минимизации, поменяем знак коэффициентов целевой функции  
 c = c\*(-1)  
  
 # Задаем матрицу левых частей ограничений времени эксплуатации судов и необходимого объема поставок на линии  
 a\_ub = np.array([[1, 1, 1, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 1, 1, 1, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 1, 1, 1],  
 [8, 0, 0, 6, 0, 0, 12, 0, 0],  
 [0, 14, 0, 0, 15, 0, 0, 12, 0],  
 [0, 0, 11, 0, 0, 13, 0, 0, 4]])  
 # Знак 4-6 неравенства необходимо привести к виду '<=' -> необходимо домножить неравенства на (-1)  
 a\_ub = a\_ub \* ([1],  
 [1],  
 [1],  
 [-1],  
 [-1],  
 [-1])  
 # Задаем вектор правх частей ограничений времени эксплуатации судов и необходимого объема поставок на линии  
 b\_ub = np.array([300, 300, 300, 3000, 5400, 3300])  
 # Домножим 4-6 элементы на (-1)  
 b\_ub = b\_ub \* [1, 1, 1, -1, -1, -1]  
  
 # Оптимизируем целевую функцию  
 opt = optimize.linprog(c, A\_ub=a\_ub, b\_ub=b\_ub, bounds=None, method='interior-point')  
 # Рассчитаем округленное до сотых итоговое значение max объема поставок  
 # при max временной загрузке судов, взятое со знаком '-'  
 min\_result = round(opt.fun, 2)  
 # Исправим знак  
 result = -min\_result  
 # Выведем решение на экран  
 print('\n✔Решение задачи 2 подзадачи 90✔\n')  
 print('🛢Мax объем перевозок при max временной загруженности судов: max(z) = {} млн. тонно-миль.\n'.format(result))  
 # Извлечем оптимальный план поставок  
 del\_plan = np.array([round(i, 2) for i in opt.x])  
 # Построим матрицу плана, где xji = время эксплуатации i-ого судна на j-ой линии  
 plan\_matrix = np.resize(del\_plan, (3, 3))  
 # Сделаем красивый вывод плана  
 print('🚢План поставок:')  
 j = 0  
 for row in plan\_matrix:  
 j += 1  
 i = 0  
 for column in row:  
 i += 1  
 # Исключаем нулевые отправки  
 if plan\_matrix[j-1, i-1] != 0:  
 print('{}-ое судно необходимо использовать {} '  
 'часов на линии {};'.format(j, column, i))  
  
  
def solving\_92():  
 # Задаем коэффициенты целевой функции  
 c = [8, 7, 5, 10, 13, 12, 13, 8, 10, 7, 11, 6, 12, 4, 11, 9, 14, 10, 14, 6, 12, 13, 13, 7, 9, 12, 14, 15, 8, 8]  
 # Задаем матрицу левых частей ограничений заводов  
 a\_ub = np.array([[1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1]])  
 # Задаем вектор правых частей ограничений заводов  
 b\_ub = [360, 180, 120, 150, 240]  
 # Задаем матрицу левых частей ограничений строительных объектов  
 a\_eq = np.array([[1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],  
 [0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],  
 [0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0],  
 [0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0],  
 [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0],  
 [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1]])  
 # Задаем вектор правх частей ограничений строительных объектов  
 b\_eq = [230, 220, 130, 170, 190, 110]  
 # Оптимизируем целевую функцию  
 opt = optimize.linprog(c, A\_ub=a\_ub, b\_ub=b\_ub, A\_eq=a\_eq, b\_eq=b\_eq, bounds=None)  
 # Рассчитаем итоговые затраты на осуществление поставок кирпичей  
 result = round(opt.fun, 2)  
 # Выведем затраты на экран  
 print('\n✔Решение задачи 2 подзадачи 92✔\n')  
 print('💰Минимальные затраты на осуществление поставок кирпичей: min(z) = {} y.e.\n'.format(result))  
 # Извлечем оптимальный план поставок  
 del\_plan = np.array([round(i, 2) for i in opt.x])  
 # Построим матрицу плана, где xji = кол-во кирпича, поставляемое с j-ого завода на i-ый объект  
 plan\_matrix = np.resize(del\_plan, (5, 6))  
 # Сделаем красивый вывод плана  
 print('🚚План поставок:')  
 j = 0  
 for row in plan\_matrix:  
 j += 1  
 i = 0  
 for column in row:  
 i += 1  
 # Исключаем нулевые отправки  
 if plan\_matrix[j-1, i-1] != 0:  
 print('Из {}-ого склада необходимо направить {} тыс. шт. '  
 'кирпича на объект {};'.format(j, column, i))  
  
  
def solving\_100\_scipy():  
 *"""Не целочисленная оптимизация, поэтому ответ достигается только засчет подогнанного округления"""* # Задаем коэффициенты целевой функции остатков от раскроя  
 c = [0, 0, 40, 20, 50, 80, 20, 60, 90]  
 # Задаем матрицу левых частей ограничений равенства количества заготовок  
 a\_eq = np.array([[5, 0, 0, 0, 1, 2, 1, 2, 3],  
 [0, 4, 0, 2, 1, 0, 3, 2, 1],  
 [0, 0, 2, 1, 1, 1, 0, 0, 0]])  
 # Задаем вектор правх частей ограничений равенства количества заготовок  
 b\_eq = [1000, 4000, 2000]  
  
 # Оптимизируем целевую функцию  
 opt = optimize.linprog(c, A\_eq=a\_eq, b\_eq=b\_eq, bounds=None, method='interior-point')  
 # Рассчитаем итоговый ответ, округленный до сотых  
 result = round(opt.fun, 2)  
  
 # Выведем объем отходов на экран  
 print('\n✔Решение задачи 2 подзадачи 100✔\n')  
 print('🗑Минимальные отходы пиломатериалов при оптимальном плане составят: min(z) = {} см.\n'.format(result))  
 # Извлечем оптимальный план распила  
 cut\_plan = np.array([round(i, 2) for i in opt.x])  
 # Посчитаем минимальное необходимое кол-во бревен  
 sum\_materials = 0  
 for i in cut\_plan:  
 sum\_materials += i  
 print('🌳Итого необходимо бревен для минимального распила: {} штук\n'.format(sum\_materials))  
 # Построим план распила и выведем все на экран  
 print('🔪План распила:')  
 num = 0  
 for i in cut\_plan:  
 num += 1  
 if i != 0:  
 print("Необходимо распилить {} досок в сооответсвии с {} вариантом распила;\n".format(i, num))  
 print(tables.cut\_variants)  
  
  
def solving\_100\_pulp():  
 *"""Pupl solver use 'COIN-OR branch and cut'(CBC) optimization method in pulp.apis.COIN\_CMD class  
 + реализуется целочисленная оптимизация"""* # Объявляем оптимизационную модель  
 opt\_model = p.LpProblem('Задача\_оптимального\_раскроя', p.LpMinimize)  
  
 # Вносим в модель список переменных  
 x1 = p.LpVariable("x1", lowBound=0) # Создать переменную x1> = 0  
 x2 = p.LpVariable("x2", lowBound=0) # Создать переменную x2> = 0  
 x3 = p.LpVariable("x3", lowBound=0) # Создать переменную x3> = 0  
 x4 = p.LpVariable("x4", lowBound=0) # Создать переменную x4> = 0  
 x5 = p.LpVariable("x5", lowBound=0) # Создать переменную x5> = 0  
 x6 = p.LpVariable("x6", lowBound=0) # Создать переменную x6> = 0  
 x7 = p.LpVariable("x7", lowBound=0) # Создать переменную x7> = 0  
 x8 = p.LpVariable("x8", lowBound=0) # Создать переменную x8> = 0  
 x9 = p.LpVariable("x9", lowBound=0) # Создать переменную x9> = 0  
  
 # Вносим в модель целевую функцию  
 opt\_model += (40\*x3 + 20\*x4 + 50\*x5 + 80\*x6 + 20\*x7 + 60\*x8 + 90\*x9), 'Целевая функция'  
  
 # Ограничения равенства:  
 opt\_model += (5\*x1 + x5 + 2\*x6 + x7 + 2\*x8 + 3\*x9 == 1000), 'Первое ограничение'  
 opt\_model += (4\*x2 + 2\*x4 + x5 + 7\*x3 + 2\*x8 + x9 == 4000), 'Второе ограничение'  
 opt\_model += (2\*x3 + x4 + x5 + x6 == 2000), 'Третье ограничение'  
  
 # Вызываем CBC-оптимизатор  
 opt\_model.solve()  
  
 # Печать окончательного решения  
 print("\n●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●")  
 print("План распила:", flush=True)  
 sum\_materials = 0  
 for i in opt\_model.variables():  
 if p.value(i) != 0:  
 print('Необходимо распилить {} вариантом распила {}'.format(i.varValue, i.name))  
 sum\_materials += p.value(i)  
 print("\nОтходы: {}".format(p.value(opt\_model.objective)))  
 print("\nНеобходимо бревен для оптимального распила: {}".format(sum\_materials))  
 print(tables.cut\_variants)  
 print("●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●")  
  
  
# solving\_90() # решение задачи 2 подзадачи 90  
# solving\_92() # решение задачи 2 подзадачи 90  
# solving\_100\_scipy() # решение задачи 2 подзадачи 100 НЕ целочисленным методом + красивый вывод  
# solving\_100\_pulp() # решение задачи 2 подзадачи 100 целочисленным методом + НЕ красивый вывод

# **Приложение 3**

Tables.py

cut\_variants = '|------------------------------------------------------------------------------------|\n' \  
 '|Длинна заготовки, см | Варианты распила |\n' \  
 '| |--------------------------------------------------------------|\n' \  
 '| |v1 |v2 |v3 |v4 |v5 |v6 |v7 |v8 |v9 |\n' \  
 '|------------------------------------------------------------------------------------|\n' \  
 '| 120| 5| 0| 0| 0| 1| 2| 1| 2| 3|\n' \  
 '|------------------------------------------------------------------------------------|\n' \  
 '| 150| 0| 4| 0| 2| 1| 0| 3| 2| 1|\n' \  
 '|------------------------------------------------------------------------------------|\n' \  
 '| 180| 0| 0| 2| 1| 1| 1| 0| 0| 0|\n' \  
 '|------------------------------------------------------------------------------------|\n' \