**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 № 3

«Решение биматричных и антагонистических игр.»

Студенты группы БИ20-8:

Луканина Полина

Аверкин Никита

Филимонова Арина

Совин Владимир

Горшков Георгий

Киселева Евгения

Руководитель:

Аксенов Дмитрий Андреевич

Москва 2022

Оглавление

1. Постановка задачи (физическая модель)

Хакер пытается взломать бд одной международной консалтинговой компании с целью кражи и дальнейшей перепродажи персональной информации о клиентах данной компании. Служба безопасности знает, что планируется атака на ее сервисы, но не знает когда и какими способами это произойдёт. У хакера есть ряд инструментов для попытки проникновения в систему: DDOS-атака, запуск трояна или поиск уязвимого места, через которое можно будет залезть в систему. У службы безопасности так же есть определённые протоколы, которые позволят отразить атаку: все внешние письма, ссылки и прочие файлы, которые прилетают сотрудникам по каналам связи проверять на вирусы; установка капчи на сайт для отражения ddos атак; усиленная проверка сотрудников на предмет вноса/выноса несанкционированной информации, смена паролей у сотрудников на более сложные. Исходя из этой информации СБ составила весовую матрицу:

Таблица 1. Весовая матрица заказчика

	Смена паролей на рабочих местах сотрудников	Тщательная проверка сотрудников на входе и выходе	Проверка входящих файлов	Капча на сайтах
Ddos-атака	5	0	15	90
Запуск вируса (троян)	40	10	95	13
Взлом рабочего места сотрудника	90	20	99	7

2. Математическая модель

Таблица 2. Исходные данные:

	B1	B2	B3	B4
A1	5	0	15	90
A2	40	10	95	13
A3	90	20	99	7

2.1. Решение задачи о нахождении выигрышной стратегии для антагонистической игры

1. Проверяем, имеет ли платежная матрица седловую точку. Если да, то выписываем решение игры в чистых стратегиях.

Считаем, что игрок I выбирает свою стратегию так, чтобы получить максимальный свой выигрыш, а игрок II выбирает свою стратегию так, чтобы минимизировать выигрыш игрока I.

Таблица 3. Находим верхние и нижние границы игры

Игроки	B1	B2	B3	B4	a = min(Ai)
A1	5	0	15	90	0
A2	40	10	95	13	10
A3	90	20	99	7	7
b = max(Bi)	90	20	99	90	

Находим гарантированный выигрыш, определяемый нижней ценой игры a = max(ai) = 10, которая указывает на максимальную чистую стратегию A2.

Верхняя цена игры b = min(bj) = 20.

Что свидетельствует об отсутствии седловой точки, так как $a \neq b$, тогда цена игры находится в пределах $10 \leq y \leq 20$. Находим решение игры в смешанных стратегиях. Объясняется это тем, что игроки не могут объявить противнику свои чистые стратегии: им следует скрывать свои действия. Игру

можно решить, если позволить игрокам выбирать свои стратегии случайным образом (смешивать чистые стратегии).

Так как игроки выбирают свои чистые стратегии случайным образом, то выигрыш игрока I будет случайной величиной. В этом случае игрок I должен выбрать свои смешанные стратегии так, чтобы получить максимальный средний выигрыш.

Аналогично, игрок II должен выбрать свои смешанные стратегии так, чтобы минимизировать математическое ожидание игрока I.

3. Находим решение игры в смешанных стратегиях.

Находим минимум функции ($F(x) = x1+x2+x3 \rightarrow \min$) при ограничениях (для игрока II):

$$5x1+40x2+90x3 \geq 1$$

$$10x2+20x3 \geq 1$$

$$15x1+95x2+99x3 \geq 1$$

$$90x1+13x2+7x3 \geq 1$$

Находим максимум функции ($Z(y) = y1+y2+y3+y4 \rightarrow \max$) при ограничениях (для игрока I):

$$5y1+15y3+90y4 \leq 1$$

$$40y1+10y2+95y3+13y4 \leq 1$$

$$90y1+20y2+99y3+7y4 \leq 1$$

Решаем прямую задачу линейного программирования симплексным методом, с использованием симплексной таблицы:

Определим максимальное значение целевой функции $Z(Y) = y1+y2+y3+y4$ при следующих условиях-ограничений:

$$5y1+15y3+90y4\leq 1$$

$$40y1+10y2+95y3+13y4\leq 1$$

$$90y1+20y2+99y3+7y4\leq 1$$

4. Получим, что оптимальный план двойственной задачи равен:

$$x1 = 13/1800,\ x2 = 0,\ x3 = 1/20$$

$$F(X) = 1*13/1800+1*0+1*1/20 = 103/1800$$

Цена игры будет равна $g = 1/F(x)$, а вероятности применения стратегий игроков:

$$qi = g*yi;\ pi = g*xi.$$

Цена игры: $g = 1: 103/1800 = 1800/103$, где:

$$p1 = 1800/103*13/1800 = 13/103$$

$$p2 = 1800/103*0 = 0$$

$$p3 = 1800/103*1/20 = 90/103$$

Оптимальная смешанная стратегия игрока I: $P = (13/103;\ 0;\ 90/103)$, где:

$$q1 = 1800/103*0 = 0$$

$$q2 = 1800/103*83/1800 = 83/103$$

$$q3 = 1800/103*0 = 0$$

$$q4 = 1800/103*1/90 = 20/103$$

Оптимальная смешанная стратегия игрока II: $Q = (0;\ 83/103;\ 0;\ 20/103)$

Цена игры: $v=1800/103$

2.2. Решение задачи о нахождении выигрышной стратегии для биматричной игры

Рассмотрим конфликтную ситуацию, в которой каждый из двух участников имеет следующие возможности для выбора своей линии поведения:

1) игрок A – может выбрать любую из стратегий A1, …, Am,

2) игрок B – любую из стратегий B1, …, Bn.

При этом их совместный выбор оценивается вполне определённо: если игрок A выбрал i-ю стратегию Ai, а игрок B – k-ю стратегию Bk, то в итоге выигрыш игрока A будет равен некоторому числу aik, а выигрыш игрока B некоторому, вообще говоря, другому числу bik.

Последовательно перебирая все стратегии игрока A и все стратегии игрока B, мы сможем заполнить их выигрышами две таблицы.

Первая из таблиц описывает выигрыш игрока A, а вторая – выигрыш игрока B. Имея одну из таблиц, введем еще одну, чтобы условие было полноценным. Преобразовав таблицы в матрицы, получим:

Таблица 4. Платежная матрица игрока А

5	0	15	**90**
40	10	95	13
90	**20**	**99**	7

Позиции максимумов в столбцах матрицы A: (3,1), (3,2), (3,3), (1,4)

Таблица 5. Платежная матрица игрока Б

15	**55**	25	14
90	5	0	35
40	20	30	**70**

Позиции максимумов в строках матрицы B: (1,2), (2,1), (3,4)

Если биматричная игра не имеет равновесных ситуаций в чистых стратегиях, то она неразрешима в чистых стратегиях. Поскольку здесь отсутствует равновесие по Нэшу, то задача не разрешима по чистой стратегии. Тогда можно искать решение в смешанных стратегиях.

Пусть игрок А выбирает стратегию A_1, с вероятностью p_1, $A_2 – p_2$, …, $A_m – p_m$, причём:

$$P1 \geq 0, p2 \geq 0, \ldots..., pm \geq 0, \sum(m)\ pi = 1$$

Игрок В использует стратегию B_1 с вероятностью q_1, $B_2 – q_2$, …, $B_n – q_n$, причём:

$$Q1 \geq 0, q2 \geq 0, \ldots., qn \geq 0, \sum(n)\ qj = 1$$

В качестве критерия "удачности" игры возьмём математические ожидания выигрыша игроков, которые вычисляются по формулам:

$$Va = \sum(m)\sum(n)\ aij\ pi\ qj$$

$$Vb = \sum(n)\sum(m)\ bij\ pi\ qj$$

Распределение вероятностей P^* и Q^* (определяют равновесную ситуацию, если для любых других распределений P и Q одновременно выполнены следующие неравенства:

$$Va\ (P, Q^*) \leq\ Va\ (P^*, Q^*),\ Vb\ (P^*, Q) \leq\ Vb\ (P^*, Q^*)$$

Если равновесная ситуация существует, то отклонение от неё невыгодно самому игроку.

Также справедлива теорема Дж. Нэша. Всякая биматричная игра имеет хотя бы одну равновесную ситуацию в смешанных стратегиях.

Итак, в первое неравенство системы последовательно подставляются все чистые стратегии игрока А, при предположении, что В придерживается своей оптимальной стратегии. Во второе неравенство подставляются все чистые стратегии игрока В, при предположении, что А придерживается своей оптимальной стратегии.

Получаем систему m+n неравенств, решение которой дает значение элементов оптимальных смешанных стратегий (P*, Q*) и платежи, получаемые игроками в точке равновесия.

Получим цены игр для матричных игр: $v_a=17^{49}/_{103}$, $v_b=28^{3}/_{4}$. Следовательно, функция полезности по Нэшу примет вид: $U=(H_1-17^{49}/_{103})(H_2-28^{3}/_{4})$

3. Алгоритмы решения задачи

Алгоритмы решения реализованы с помощью программного кода в Python.

3.1. Решение задачи о нахождении выигрышной стратегии для антагонистической игры

3.1.1. Описание входных данных.

Вид входных данных зависит от способа, которым будут вводиться данными. Данный алгоритм позволяет самим ввести данные, или ввести с помощью CSV файла.

Для ручного ввода входными данными являются:

- Количество стратегий для компании А, название стратегий для компании А, количество стратегий для компании В, название стратегий для компании Б, значения весовой матрицы.

```
Способ ввода данных: 1
Введите количество стратегий для компании A: 2
Введите название 1 стратегии компании A: ноутбук
Введите название 2 стратегии компании A: пк
Введите количество стратегий для компании B: 2
Введите название 1 стратегии компании B: ноутбук
Введите название 2 стратегии компании B: пк
Введите элемент строки 1 столбца 1: 100
Введите элемент строки 1 столбца 2: 200
Введите элемент строки 2 столбца 1: 150
Введите элемент строки 2 столбца 2: 420
```

Рисунок 1. Ручной способ ввода данных

- Количество стратегий игрока A.

```
Способ ввода данных: 2
Количество стратегий игрока A: 2

Весовая матрица:
      1    2
1  682  119
2  955  302
```

Рисунок 2. Весовая матрица

Импорт входных данных из csv файла:

- Весовая матрица с названиями столбцов и строк.

```
Введите путь к файлу: \Users\Полина\Downloads\12.csv
```

Рисунок 3. 3 способ ввода информации в код

3.1.2. Описание алгоритма решения

После того как данные введены, программе необходимо преобразовать данные для дальнейшего использования.

Шаг 1: берем весовую матрицу, находим минимум каждой строки, а далее - максимум — это получилась цена игры для игрока A.

Шаг 2: находим максимум для каждого столбца весовой матрицы, и затем находим минимальное из них — это цена игры для игрока B.

Шаг 3: Переход к канонической форме задачи линейного программирования путем введения неотрицательных дополнительных балансовых (базисных) переменных. Запись задачи в симплекс-таблицу. Между системой ограничений задачи и симплекс-таблицей взаимно-однозначное соответствие. Строчек в таблице столько, сколько равенств в системе ограничений, а столбцов - столько, сколько свободных переменных.

Базисные переменные заполняют первый столбец, свободные - верхнюю строку таблицы.

Шаг 4: Проверка опорного плана на оптимальность. Для этого необходимо анализировать строку целевой функции F. Если найдется хотя бы один коэффициент индексной строки меньше нуля, то план не оптимальный, и его необходимо улучшить.

Шаг 5: Улучшение опорного плана. Из отрицательных коэффициентов индексной строки выбирается наибольший по абсолютной величине. Затем элементы столбца свободных членов симплексной таблицы делит на элементы того же знака ведущего столбца. Далее идет построение нового опорного плана.

Шаг 6: Выписывание оптимального решения.

3.1.3. Описание выходных данных

В конце программа рассчитает оптимальную чистую стратегию, цену игры для игрока А при выборе чистой оптимальной стратегии, оптимальная чистая стратегия для игрока Б (текст), цена игры для игрока Б при выборе чистой оптимальной стратегии, таблица смешанных стратегий для игрока А, цена игры для игрока А при выборе смешанной оптимальной стратегии.

```
Оптимальная чистая стратегия для игрока A:  1
Цена игры для игрока A при выборе чистой оптимальной стратегии:  915
Оптимальная чистая стратегия для игрока B:  1
Цена игры для игрока B при выборе чистой оптимальной стратегии:  915

Цена игры для игрока A при выборе смешанной оптимальной стратегии:  915.0
Таблица смешанных стратегий для игрока A:
 +------+----+
|  1   | 2  |
+------+----+
| 100% | 0% |
+------+----+
Таблица смешанных стратегий для игрока B:
 +------+----+
|  1   | 2  |
+------+----+
| 100% | 0% |
+------+----+
```

Рисунок 4. Полученный ответ

3.2. Решение задачи о нахождении выигрышной стратегии для биматричной игры

3.2.1. Описание входных данных.

Вид входных данных зависит от способа, которым будут вводиться данными. Данный алгоритм позволяет самим ввести данные, или ввести с помощью CSV файла.

Для ручного ввода входными данными являются:

- Количество стратегий для компании А, название стратегий для компании А, количество стратегий для компании В, название стратегий для компании Б, значения весовой матрицы.

```
Способ ввода данных: 1
Введите количество стратегий для компании A: 2
Введите название 1 стратегии компании A: ПК
Введите название 2 стратегии компании A: Ноутбук
Введите количество стратегий для компании B: 2
Введите название 1 стратегии компании B: ПК
Введите название 2 стратегии компании B: Ноутбук

Введите матрицу весовых коэффициентов для игрока A поэлементно:
Введите элемент строки 1 столбца 1: 250
Введите элемент строки 1 столбца 2: 300
Введите элемент строки 2 столбца 1: 210
Введите элемент строки 2 столбца 2: 350

Введите матрицу весовых коэффициентов для игрока B поэлементно:
Введите элемент строки 1 столбца 1: 410
Введите элемент строки 1 столбца 2: 200
Введите элемент строки 2 столбца 1: 150
Введите элемент строки 2 столбца 2: 200
```

Рисунок 5. Терминал питона при 1 способе ввода

- Количество стратегий игрока А, количество стратегий игрока В

```
Способ ввода данных: 2
Количество стратегий игрока A: 2
Количество стратегий игрока B: 2
```

Рисунок 6. Терминал питона при 2 способе ввода данных

Импорт входных данных из csv файла:

- Весовая матрица с названиями столбцов и строк.

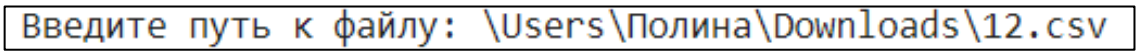

Рисунок 7. Терминал питона при 3 способе ввода данных

3.1.2. Описание алгоритма решения

После того как данные введены, программе необходимо преобразовать данные для дальнейшего использования.

Шаг 1: берем весовую матрицу, находим минимум каждой строки, а далее - максимум — это получилась цена игры для игрока A.

Шаг 2: находим максимум для каждого столбца весовой матрицы, и затем находим минимальное из них — это цена игры для игрока B.

Шаг 3: Переход к канонической форме задачи линейного программирования путем введения неотрицательных дополнительных балансовых (базисных) переменных. Запись задачи в симплекс-таблицу. Между системой ограничений задачи и симплекс-таблицей взаимно-однозначное соответствие. Строчек в таблице столько, сколько равенств в системе ограничений, а столбцов - столько, сколько свободных переменных. Базисные переменные заполняют первый столбец, свободные - верхнюю строку таблицы. Нижняя строка называется индексной, в ней записываются коэффициенты при переменных в целевой функции.

Шаг 4: Проверка опорного плана на оптимальность. Для этого необходимо анализировать строку целевой функции F. Если найдется хотя бы один коэффициент индексной строки меньше нуля, то план не оптимальный, и его необходимо улучшить.

Шаг 5: Улучшение опорного плана. Из отрицательных коэффициентов индексной строки выбирается наибольший по абсолютной величине. Затем элементы столбца свободных членов симплексной таблицы делит на элементы того же знака ведущего столбца. Далее идет построение нового опорного плана.

Шаг 6: Расчет равновесий по Нешу, для этого создается матрица с координатами максимальных значений по строке для первого игрока и для второго, а после этого сравниваем их на равенство.

3.1.3. Описание выходных данных

В конце программа рассчитает оптимальную чистую стратегию, цену игры для игрока А при выборе чистой оптимальной стратегии, оптимальная чистая стратегия для игрока Б (текст), цена игры для игрока Б при выборе чистой оптимальной стратегии, таблица смешанных стратегий для игрока А, цена игры для игрока А при выборе смешанной оптимальной стратегии.

```
Оптимальная чистая стратегия для игрока A:  1
Цена игры для игрока A при выборе чистой оптимальной стратегии:  915
Оптимальная чистая стратегия для игрока B:  1
Цена игры для игрока B при выборе чистой оптимальной стратегии:  915

Цена игры для игрока A при выборе смешанной оптимальной стратегии:  915.0
Таблица смешанных стратегий для игрока A:
 +------+----+
|  1   | 2  |
+------+----+
| 100% | 0% |
+------+----+
Таблица смешанных стратегий для игрока B:
 +------+----+
|  1   | 2  |
+------+----+
| 100% | 0% |
+------+----+
```

Рисунок 8. Результат выполнения кода

4. Варианты использования системы

В нашей системе есть три варианта использования.

4.1. Варианты использования в условиях антагонистической игры

4.1.1. 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».

```
Способ ввода данных: 1
Введите количество стратегий для компании A: 2
Введите название 1 стратегии компании A: ноутбук
Введите название 2 стратегии компании A: пк
Введите количество стратегий для компании B: 2
Введите название 1 стратегии компании B: ноутбук
Введите название 2 стратегии компании B: пк
Введите элемент строки 1 столбца 1: 100
Введите элемент строки 1 столбца 2: 200
Введите элемент строки 2 столбца 1: 250
Введите элемент строки 2 столбца 2: 250

Весовая матрица:
          ноутбук   пк
ноутбук       100  200
пк            250  250

Оптимальная чистая стратегия для игрока A:  пк
Цена игры для игрока A при выборе чистой оптимальной стратегии:  250
Оптимальная чистая стратегия для игрока B:  ноутбук
Цена игры для игрока B при выборе чистой оптимальной стратегии:  250

Цена игры для игрока A при выборе смешанной оптимальной стратегии:  250.0
Таблица смешанных стратегий для игрока A:
 +---------+------+
| ноутбук |  пк  |
+---------+------+
|    0%   | 100% |
+---------+------+
Таблица смешанных стратегий для игрока B:
 +---------+-----+
| ноутбук |  пк |
+---------+-----+
|   48%   | 52% |
+---------+-----+
```

Рисунок 9. Терминал кода при ВИ 1

После этого необходимо ввести количество стратегий для компании А цифрой. (Например, 2).

Затем нужно ввести названия стратегий, их количество зависит от того, сколько стратегий вы указали ранее. (Например, ноутбук).

Далее переходим к заполнению данных для второй компании. Мы также вводим количество стратегий для компании B, а также названия стратегий.

После этого вы должны ввести данные для заполнения весовой матрицы. Количество значений зависит от того, сколько стратегий вы указали ранее. (Например, 100).

После этого от пользователя требуется лишь нажатие клавиши «Enter» и на экране выведется оптимальная чистая стратегия, цена игры для игрока A при выборе чистой оптимальной стратегии, оптимальная чистая стратегия для игрока B, цена игры для игрока B при выборе чистой оптимальной стратегии, цена игры для игрока A при выборе смешанной оптимальной стратегии, таблица смешанных стратегий для игрока A, таблица смешанных стратегий для игрока B.

4.1.2. ВИ 2

Данный вариант позволяет сгенерировать весовую матрицу, для его выбора в первой строке необходимо ввести «2».

```
Способ ввода данных: 2
Количество стратегий игрока A: 2

Весовая матрица:
      1    2
1  741  658
2  500  348

Оптимальная чистая стратегия для игрока A:  1
Цена игры для игрока A при выборе чистой оптимальной стратегии:  658
Оптимальная чистая стратегия для игрока B:  2
Цена игры для игрока B при выборе чистой оптимальной стратегии:  658

Цена игры для игрока A при выборе смешанной оптимальной стратегии:  658.0
Таблица смешанных стратегий для игрока A:
 +------+----+
|  1   | 2  |
+------+----+
| 100% | 0% |
+------+----+
Таблица смешанных стратегий для игрока B:
 +----+------+
| 1  |  2   |
+----+------+
| 0% | 100% |
+----+------+
```

Рисунок 10. Автоматическая генерация весовой матрицы

Далее Вам необходимо ввести количество стратегий игрока A.

После этого от пользователя требуется лишь нажатие клавиши «Enter» и на экране выведется оптимальная чистая стратегия, цена игры для игрока A при выборе чистой оптимальной стратегии, оптимальная чистая стратегия для игрока B, цена игры для игрока B при выборе чистой оптимальной стратегии, цена игры для игрока A при выборе смешанной оптимальной стратегии, таблица смешанных стратегий для игрока A, таблица смешанных стратегий для игрока B.

4.1.3. 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».

После этого появляется окно, в котором вводим путь к csv файлу. Например:

```
Введите путь к файлу: \Users\Полина\Downloads\12.csv
```

Рисунок 11. Выбор местонахождения csv файла

	A	B	C	D
1	A	Телефоны	Наушники	Ноутбуки
2	Телефоны	120	123	543
3	Ноутбуки	234	544	900
4	ПК	567	677	655
5				
6				
7				
8				

Рисунок 12. Данные следует вводить таким образом:

После этого получаем результат.

```
Способ ввода данных: 3
Введите путь к файлу: C:/Users/aniki/Desktop/1.csv

Весовая матрица:
           Телефоны  Наушники  Ноутбуки
Телефоны       120       123       543
Ноутбуки       234       544       900
ПК             567       677       655

Оптимальная чистая стратегия для игрока A:  ПК
Цена игры для игрока A при выборе чистой оптимальной стратегии:  567
Оптимальная чистая стратегия для игрока B:  Телефоны
Цена игры для игрока B при выборе чистой оптимальной стратегии:  567

Цена игры для игрока A при выборе смешанной оптимальной стратегии:  566.9997
Таблица смешанных стратегий для игрока A:
+----------+----------+------+
| Телефоны | Ноутбуки |  ПК  |
+----------+----------+------+
|    0%    |    0%    | 100% |
+----------+----------+------+
Таблица смешанных стратегий для игрока B:
+----------+----------+----------+
| Телефоны | Наушники | Ноутбуки |
+----------+----------+----------+
|   100%   |    0%    |    0%    |
+----------+----------+----------+
```

Рисунок 13. Пример:

4.2. Варианты использования в условиях биматричной игры

4.2.1. 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».

```
Способ ввода данных: 1
Введите количество стратегий для компании А: 2
Введите название 1 стратегии компании А: ПК
Введите название 2 стратегии компании А: Ноутбук
Введите количество стратегий для компании Б: 2
Введите название 1 стратегии компании Б: ПК
Введите название 2 стратегии компании Б: Ноутбук

Введите матрицу весовых коэффициентов для игрока А поэлементно:
Введите элемент строки 1 столбца 1: 450
Введите элемент строки 1 столбца 2: 300
Введите элемент строки 2 столбца 1: 250
Введите элемент строки 2 столбца 2: 500

Введите матрицу весовых коэффициентов для игрока В поэлементно:
Введите элемент строки 1 столбца 1: 400
Введите элемент строки 1 столбца 2: 350
Введите элемент строки 2 столбца 1: 300
Введите элемент строки 2 столбца 2: 450
```

Рисунок 14. Терминал при выборе 1 ВИ

После этого необходимо ввести количество стратегий для компании А цифрой. (Например, 2).

Затем нужно ввести названия стратегий, их количество зависит от того, сколько стратегий вы указали ранее. (Например, ноутбук).

Далее переходим к заполнению данных для второй компании. Мы также вводим количество стратегий для компании В, а также названия стратегий.

После этого вы должны ввести данные для заполнения весовой матрицы. Количество значений зависит от того, сколько стратегий вы указали ранее. (Например, 100).

После этого от пользователя требуется лишь нажатие клавиши «Enter» и на экране выведется оптимальная чистая стратегия, цена игры для игрока А при выборе чистой оптимальной стратегии, оптимальная чистая стратегия для игрока В, цена игры для игрока В при выборе чистой оптимальной стратегии, цена игры для игрока А при выборе смешанной оптимальной стратегии,

таблица смешанных стратегий для игрока А, таблица смешанных стратегий для игрока В, количество равновесий по Нешу.

```
Весовая матрица игрока А:
           ПК  Ноутбук
ПК        450      300
Ноутбук   250      500

Весовая матрица игрока Б:
           ПК  Ноутбук
ПК        400      350
Ноутбук   300      450

Оптимальная чистая стратегия для игрока А:  ПК
Цена игры для игрока А при выборе чистой оптимальной стратегии:  450
Оптимальная чистая стратегия для игрока Б:  ПК
Цена игры для игрока Б при выборе чистой оптимальной стратегии:  400
Общая (суммарная) цена игры: 850

Количество равновесий по Нэшу:  2

Таблица смешанных стратегий для компании А:
+-----+---------+
|  ПК | Ноутбук |
+-----+---------+
| 75% |   25%   |
+-----+---------+
Цена игры для игрока А при выборе смешанной оптимальной стратегии: 1325.0
Таблица смешанных стратегий для компании Б:
+-----+---------+
|  ПК | Ноутбук |
+-----+---------+
| 50% |   50%   |
+-----+---------+
Цена игры для игрока Б при выборе смешанной оптимальной стратегии: 1225.0
Общая цена игры в случае использования оптимальных стратегий: 2550.0
```

Рисунок 15. Результат запроса

4.2.2. ВИ 2

Данный вариант позволяет сгенерировать весовую матрицу, для его выбора в первой строке необходимо ввести «2».

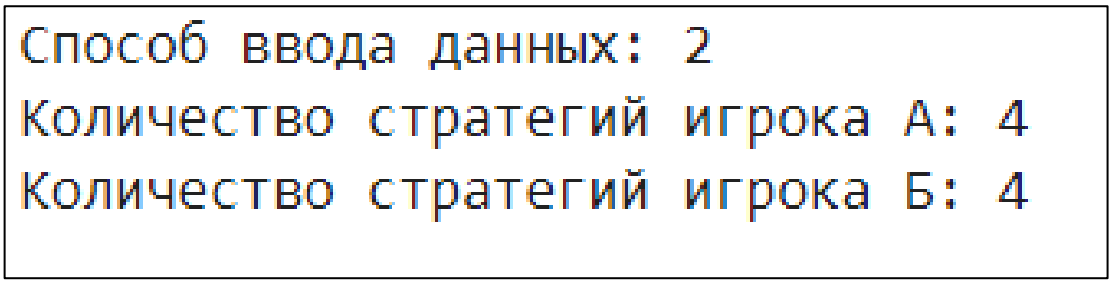

Рисунок 16. Терминал при 2 ВИ

Далее Вам необходимо ввести количество стратегий игрока А.

После этого от пользователя требуется лишь нажатие клавиши «Enter» и на экране выведется оптимальная чистая стратегия, цена игры для игрока А при выборе чистой оптимальной стратегии, оптимальная чистая стратегия для игрока В, цена игры для игрока В при выборе чистой оптимальной стратегии, цена игры для игрока А при выборе смешанной оптимальной стратегии, таблица смешанных стратегий для игрока А, таблица смешанных стратегий для игрока В, количество равновесий по Нешу.

```
Весовая матрица игрока А:
      1    2    3    4
1  747  699  634   89
2  115  541  768  672
3   65  760   91  823
4  452  830  891  362

Весовая матрица игрока Б:
      1    2    3    4
1  841  232  191  466
2  554  832  429  150
3  969  252  415  439
4  348  533  223  276

Оптимальная чистая стратегия для игрока А:  1
Цена игры для игрока А при выборе чистой оптимальной стратегии:  747
Оптимальная чистая стратегия для игрока Б:  4
Цена игры для игрока Б при выборе чистой оптимальной стратегии:  533
Общая (суммарная) цена игры: 1280

Количество равновесий по Нэшу:  2

Таблица смешанных стратегий для компании А:
+-----+----+----+-----+
|  1  | 2  | 3  |  4  |
+-----+----+----+-----+
| 23% | 0% | 0% | 77% |
+-----+----+----+-----+
Цена игры для игрока А при выборе смешанной оптимальной стратегии: 2290.761
Таблица смешанных стратегий для компании Б:
+-----+-----+----+----+
|  1  |  2  | 3  | 4  |
+-----+-----+----+----+
| 31% | 69% | 0% | 0% |
+-----+-----+----+----+
Цена игры для игрока Б при выборе смешанной оптимальной стратегии: 1836.868
Общая цена игры в случае использования оптимальных стратегий: 4127.628
```

Рисунок 17. Оптимальная чистая стратегия для игрока А

4.2.3. 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».

После этого появляется окно, в котором вводим путь к csv файлу.

```
Введите путь к файлу: \Users\Полина\Downloads\12.csv
```

Рисунок 18. Например:

	A	B	C	D
1	A	Телефоны	Наушники	Ноутбуки
2	Телефоны	120	123	543
3	Ноутбуки	234	544	900
4	ПК	567	677	655
5				
6				
7				
8				

Рисунок 19. Данные следует вводить таким образом:

5. Архитектура решения

Для решения задачи использовались методы (функции), которые можно разделить на 3 принципиальных кода.

5.1 Алгоритм 1. Для антагонистической стратегии

5.1.1. Функции считывания информации

После запуска программы необходимо определиться каким способом будет происходить ввод данных: 1 – ручной ввод; 2 – случайные числа; 3 – файл csv.

Если введено «1»:

Входные данные:

- Нет входных данных

Выходные данные:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- matrix – список значений весовой матрицы (тип данных: list);

Переменные, затрагиваемые в ходе работы:

- sum_strategy_A – количество стратегий для компании A (тип данных: int);
- sum_strategy_B – количество стратегий для компании B (тип данных: int);
- name_rows – список названий стратегий компании A (тип данных: list);

- name_columns – список названий стратегий компании B (тип данных: list);
- matrix – список значений весовой матрицы (тип данных: list);
- name – название стратегий (тип данных: str);
- value – элементы весовой матрицы (тип данных: int);

Если введено «2»:

Входные данные:

- Нет входных данных

Выходные данные:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- matrix – список значений весовой матрицы (тип данных: list);

Переменные, затрагиваемые в ходе работы:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- matrix – список значений весовой матрицы (тип данных: list);
- size_rows – количество стратегий игрока A (тип данных: int);
- x – генерируемые данные весовой матрицы (тип данных: int);

Если ведено «3»:

Входные данные:

- Нет входных данных

Выходные данные:

- matrix – список значений весовой матрицы (тип данных: list);
- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);

Переменные, затрагиваемые в ходе работы:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- matrix – список значений весовой матрицы (тип данных: list);
- file_way – путь к файлу (тип данных: str).

```
way = int(input('Способ ввода данных: '))

# Ввод с клавиатуры
if way == 1:
    sum_strategy_A = int(input('Введите количество стратегий для компании A: '))
    name_rows, name_columns = [], []
    for num in range(sum_strategy_A):
        name = input('Введите название {} стратегии компании A: '.format(num + 1))
        name_rows.append(name)

    sum_strategy_B = int(input('Введите количество стратегий для компании B: '))
    for num in range(sum_strategy_B):
        name = input('Введите название {} стратегии компании B: '.format(num + 1))
        name_columns.append(name)

    matrix = []
    rows = []
    for num_row, row in enumerate(name_rows, start=1):
        for num_column, column in enumerate(name_columns, start=1):
            value = int(input('Введите элемент строки {} столбца {}: '.format(num_row, num_column)))
            rows.append(value)
        matrix.append(rows)
        rows = []
```

Рисунок 20. Фрагмент кода с данными переменными

```
elif way == 2:
    size_rows = int(input("Количество стратегий игрока A: "))
    size_cols = size_rows

    # Последовательно генерируем случайные числа типа int в диапазоне (0;1000) и добавляем их в матрицу
    matrix = []
    for i in range(size_rows):
        rows = []
        for j in range(size_cols):
            x = random.randint(0,1000)
            rows.append(x)
        matrix.append(rows)

    # список для имен строк и столбцов матрицы:
    name_rows, name_columns = [], []
    for i in range(size_rows):
        name_rows.append(i+1)

    for i in range(size_cols):
        name_columns.append(i+1)
```

Рисунок 21. Фрагмент кода с данными переменными

```
    # файл csv
    elif way == 3:
        matrix = []
        file_way = input('Введите путь к файлу: ')
        with open(file_way, 'r', newline='') as csvfile:
            spamreader = csv.reader(csvfile, delimiter=';')
            for row in spamreader:
                matrix.append(row)

        # Для образования списка с названиями колонок матрицы
        name_columns = matrix[0]
        name_columns = name_columns[1:]
        del matrix[0]

        # Для образования списка с названиями строк
        name_rows = []
        for num, value in enumerate(matrix):
            name_rows.append(value[0])
            del matrix[num][0]

        # Перевод коэффициентов из str в тип int
        for num_1, row in enumerate(matrix):
            for num_2, value in enumerate(row):
                matrix[num_1][num_2] = int(value)


    matrix = pd.DataFrame(matrix)
    matrix.columns = name_columns
    matrix.index = name_rows
    print('\nВесовая матрица:\n', matrix, '\n')
```

Рисунок 22. Фрагмент кода с данными переменными

5.1.2. Функции обработки информации

После того, как вы введете все необходимые данные, программа их получит и начнет первичную обработку.

При выборе ручного ввода все необходимые данные заносятся в словари, с помощью метода append ().

В случае случайной генерации, нужные данные генерируются с помощью random.randint (), при этом программа получает случайное целое

число в заданном диапазоне. После этого данные также заносятся в пустые словари.

Если ввод данных осуществляется с помощью файла csv, то для начала файл необходимо открыть и прочитать программе, а затем разделить данные и также занести их в пустой словарь. Более того, программа образует список с названиями колонок матрицы, а также список с названиями строк. Не мало важен и перевод коэффициентов из типа данных str в тип int.

После всех необходимых операций с данными реализуются важные функции, рассчитывающие оптимальную чистую стратегию и оптимальную смешанную стратегию.

В функции pure_strategy реализуется оптимальная чистая стратегия. Она представляет собой нахождение минимума в каждой строке, а также нахождения максимума в каждом столбце. Это находится с помощью функции min(), max().

В функции nash_equilibrium реализуется оптимальная смешанная стратегия. С помощью функции linprog() находится оптимальное решение переменных с помощью которых мы находим процентное соотношение по стратегиям. Более того, здесь также составляются таблицы смешанных стратегий.

Далее следует подробное описание входных, выходных и переменных, затрагивается в ходе работы.

Функция pure_strategy:

Входные данные:

- name_rows – список названий стратегий компании A (тип данных: list);

- name_columns – список названий стратегий компании B (тип данных: list);

Выходные данные:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- matrix – список значений весовой матрицы (тип данных: list);

Переменные, затрагиваемые в ходе работы:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- max_min – максимальное значение по столбцу весовой матрицы (тип данных: list);
- min_max – минимальное значение по строке весовой матрицы (тип данных: list);
- index_row – индекс по строке весовой матрицы (тип данных: int);
- index_column – индекс по столбцу весовой матрицы (тип данных: int).

```
def pure_strategy(matrix, name_columns, name_rows):
    max_min = list(matrix.min(axis=1))
    min_max = list(matrix.max())

    index_row = max_min.index(max(max_min))
    index_column = min_max.index(min(min_max))

    print('Оптимальная чистая стратегия для игрока A: ', name_rows[index_row])
    print('Цена игры для игрока A при выборе чистой оптимальной стратегии: ', max(max_min))

    print('Оптимальная чистая стратегия для игрока B: ', name_columns[index_column])
    print('Цена игры для игрока B при выборе чистой оптимальной стратегии: ', min(min_max))
    print(type(max_min))
pure_strategy = pure_strategy(matrix, name_columns, name_rows)
```

Рисунок 23. Фрагмент кода с данными переменными

Функция nash_equilibrium:

Входные данные:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);

Выходные данные:

- res — это массив с вероятностями стратегий игроков (тип данных: scipy.optimize.optimize.OptimizeResult);
- mytable – таблицы, которые выводят стратегии и проценты (тип данных: prettytable.prettytable.PrettyTable);

Переменные, затрагиваемые в ходе работы:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- res - это массив с вероятностями стратегий игроков (тип данных: scipy.optimize.optimize.OptimizeResult);
- mytable – таблицы, которые выводят стратегии и проценты (тип данных: prettytable.prettytable.PrettyTable);
- a_T — это транспонированная весовая матрица (тип данных: numpy.ndarray);
- mix_strategy_A — это вероятности стратегий в процентах игрока A (тип данных: list);
- mix_strategy_B — это вероятности стратегий в процентах игрока B (тип данных: list);

```
def nash_equilibrium(a, name_columns, name_rows):
    c = np.array([1 for x in range(len(a[0]))])
    b = np.array([1 for x in range(len(a))])

    bounds = [(0, None)]
    for x in range(len(a[0]) - 1):
        bounds += [(0, None)]

    a_T = -np.transpose(a)
    res = linprog(c, A_ub=a_T, b_ub= -b, bounds=bounds, options = {"disp": False})
    p = [res.x[i]/(res.fun) for i in range(len(res.x))]

    print('\nЦена игры для игрока A при выборе смешанной оптимальной стратегии: ', round(1/res.fun, 4))

    # Таблица смешанной стратегии для игрока A
    mix_strategy_A = []
    for i in range(len(p)):
        mix_strategy_A.append(str(round(p[i]*100)) + '%')

    mytable = PrettyTable()
    mytable.field_names = name_rows # имена полей таблицы
    mytable.add_row(mix_strategy_A)     # добавление данных по одной строке за раз
    print('Таблица смешанных стратегий для игрока A:\n', mytable)
    # Таблица смешанной стратегии для игрока B
    res = linprog(-c, A_ub=a, b_ub=b, bounds=bounds, options = {"disp": False})
    q = [res.x[i]/(-res.fun) for i in range(len(res.x))]
    # Вывод
    mix_strategy_B = []
    for i in range(len(q)):
        mix_strategy_B.append(str(round(q[i]*100)) + '%')
    mytable = PrettyTable()
    mytable.field_names = name_columns # имена полей таблицы
    mytable.add_row(mix_strategy_B)     # добавление данных по одной строке за раз
    print('Таблица смешанных стратегий для игрока B:\n', mytable)
```

Рисунок 24. Фрагмент кода с данными переменными

5.1.3. Функции вывода информации

Метод вывода информации (он заключен внутри каждой функции)

Что делает: осуществляет вывод необходимой информации

Вывод информации осуществляется с помощью функции print ()

Затрагиваемые переменные:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- matrix – список значений весовой матрицы (тип данных: list);

- res — это массив с вероятностями стратегий игроков (тип данных: scipy.optimize.optimize.OptimizeResult);
- mytable – таблицы, которые выводят стратегии и проценты (тип данных: prettytable.prettytable.PrettyTable);

```
print('Оптимальная чистая стратегия для игрока A: ', name_rows[index_row])
print('Цена игры для игрока A при выборе чистой оптимальной стратегии: ', max(max_min))

print('Оптимальная чистая стратегия для игрока B: ', name_columns[index_column])
print('Цена игры для игрока B при выборе чистой оптимальной стратегии: ', min(min_max))
print(type(max_min))
```

Рисунок 25. Вывод оптимальных стратегий для обоих игроков

```
print('\nЦена игры для игрока A при выборе смешанной оптимальной стратегии: ', round(1/res.fun, 4))
```

Рисунок 26. Цена для игрока А

```
print('Таблица смешанных стратегий для игрока A:\n', mytable)
```

Рисунок 27. Вывод таблицы смешанных стратегий для игрока А

```
print('Таблица смешанных стратегий для игрока B:\n', mytable)
```

Рисунок 28. Вывод таблицы смешанных стратегий для игрока Б

```
Способ ввода данных: 1
Введите количество стратегий для компании A: 2
Введите название 1 стратегии компании A: ПК
Введите название 2 стратегии компании A: Ноутбук
Введите количество стратегий для компании B: 2
Введите название 1 стратегии компании B: ПК
Введите название 2 стратегии компании B: Ноутбук

Введите матрицу весовых коэффициентов для игрока A поэлементно:
Введите элемент строки 1 столбца 1: 250
Введите элемент строки 1 столбца 2: 300
Введите элемент строки 2 столбца 1: 210
Введите элемент строки 2 столбца 2: 350

Введите матрицу весовых коэффициентов для игрока B поэлементно:
Введите элемент строки 1 столбца 1: 410
Введите элемент строки 1 столбца 2: 200
Введите элемент строки 2 столбца 1: 150
Введите элемент строки 2 столбца 2: 200

Весовая матрица игрока A:
           ПК  Ноутбук
ПК        250      300
Ноутбук   210      350

Весовая матрица игрока B:
           ПК  Ноутбук
ПК        410      200
Ноутбук   150      200

Оптимальная чистая стратегия для игрока A:  ПК
Цена игры для игрока B при выборе чистой оптимальной стратегии:  250
Оптимальная чистая стратегия для игрока B:  Ноутбук
Цена игры для игрока A при выборе чистой оптимальной стратегии:  200
Количество равновесий по Нэшу:  2
```

Рисунок 29. Результат выполнения запроса при 1 ВИ

```
Способ ввода данных: 2
Количество стратегий игрока A: 2
Количество стратегий игрока B: 2

Весовая матрица игрока A:
      1    2
1  447  285
2  280  369

Весовая матрица игрока B:
      1    2
1  252  131
2  582  269

Оптимальная чистая стратегия для игрока A:  2
Цена игры для игрока B при выборе чистой оптимальной стратегии:  369
Оптимальная чистая стратегия для игрока B:  1
Цена игры для игрока A при выборе чистой оптимальной стратегии:  252
Количество равновесий по Нэшу:  1
```

Рисунок 30. Результат выполнения запроса при 2 ВИ

5.2 Алгоритм 2. Для биматричной стратегии

5.2.1. Функции считывания информации

После запуска программы необходимо определиться каким способом будет происходить ввод данных: 1 – ручной ввод; 2 – случайные числа; 3 – файл csv.

1) **Если введено «1»:**

Входные данные:

- Нет входных данных

Выходные данные:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- matrix – список значений весовой матрицы (тип данных: list);

Переменные, затрагиваемые в ходе работы:

- sum_strategy_A – количество стратегий для компании A (тип данных: int);
- sum_strategy_B – количество стратегий для компании B (тип данных: int);
- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- matrix_A – список значений весовой матрицы для компании A (тип данных: list);
- matrix_B – список значений весовой матрицы для компании B (тип данных: list);
- name – название стратегий (тип данных: str);
- value – элементы весовой матрицы (тип данных: int);

2) Если введено «2»:

Входные данные:

- Нет входных данных

Выходные данные:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- matrix_A – список значений весовой матрицы для компании A (тип данных: list);
- matrix_B – список значений весовой матрицы для компании B (тип данных: list);

Переменные, затрагиваемые в ходе работы:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- matrix_A – список значений весовой матрицы для компании A (тип данных: list);
- matrix_B – список значений весовой матрицы для компании B (тип данных: list);
- size_rows – количество стратегий игрока A (тип данных: int);
- x – генерируемые данные весовой матрицы (тип данных: int);

3) Если ведено «3»:

Входные данные:

- Нет входных данных

Выходные данные:

- matrix_A – список значений весовой матрицы для компании A (тип данных: list);
- matrix_B – список значений весовой матрицы для компании B (тип данных: list);
- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);

Переменные, затрагиваемые в ходе работы:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- matrix_A – список значений весовой матрицы для компании A (тип данных: list);
- matrix_B – список значений весовой матрицы для компании B (тип данных: list);

- file_way – путь к файлу (тип данных: str).

```
if way == 1:
    sum_strategy_A = int(input('Введите количество стратегий для компании А: '))
    name_rows, name_columns = [], []
    for num in range(sum_strategy_A):
        name = input('Введите название {} стратегии компании А: '.format(num + 1))
        name_rows.append(name)

    sum_strategy_B = int(input('Введите количество стратегий для компании Б: '))
    for num in range(sum_strategy_B):
        name = input('Введите название {} стратегии компании Б: '.format(num + 1))
        name_columns.append(name)

    print('\nВведите матрицу весовых коэффициентов для игрока А поэлементно:')
    matrix_A = []
    rows = []
    for num_row, row in enumerate(name_rows, start=1):
        for num_column, column in enumerate(name_columns, start=1):
            value = int(input('Введите элемент строки {} столбца {}: '.format(num_row, num_column)))
            rows.append(value)
        matrix_A.append(rows)
        rows = []

    print('\nВведите матрицу весовых коэффициентов для игрока В поэлементно:')
    matrix_B = []
    rows = []
    for num_row, row in enumerate(name_rows, start=1):
        for num_column, column in enumerate(name_columns, start=1):
            value = int(input('Введите элемент строки {} столбца {}: '.format(num_row, num_column)))
            rows.append(value)
        matrix_B.append(rows)
        rows = []
```

Рисунок 31. Часть кода, в которой затрагиваются эти переменные

```
elif way == 2:
    # ----------------------------------
    # Заполняем матрицу для игрока А
    size_rows = int(input("Количество стратегий игрока А: "))
    size_cols = size_rows

    # Последовательно генерируем случайные числа типа int в диапазоне (0;1000) и добавляем их в матрицу
    matrix_A = []
    for i in range(size_rows):
        rows = []
        for j in range(size_cols):
            x = random.randint(0,1000)
            rows.append(x)
        matrix_A.append(rows)

    # список для имен строк и столбцов матрицы:
    name_rows = []
    for i in range(size_rows):
        name_rows.append(i+1)

    # ----------------------------------
    # Заполняем матрицу для игрока В
    size_rows = int(input("Количество стратегий игрока Б: "))
    size_cols = size_rows

    # Последовательно генерируем случайные числа типа int в диапазоне (0;1000) и добавляем их в матрицу
    matrix_B = []
    for i in range(size_rows):
        rows = []
        for j in range(size_cols):
            x = random.randint(0,1000)
            rows.append(x)
        matrix_B.append(rows)
```

Рисунок 32. Часть кода, в которой затрагиваются эти переменные

```
# список для имен строк и столбцов матрицы:
name_columns = []
for i in range(size_cols):
    name_columns.append(i+1)
```

Рисунок 33. Список для имен строк и столбцов матрицы

```
# Импорт csv файла
elif way == 3:
    # Игрок А
    matrix_A = []
    file_way = input('Введите путь к файлу (матрица игрока А): ')
    with open(file_way, 'r', newline='') as csvfile:
        spamreader = csv.reader(csvfile, delimiter=';')
        for row in spamreader:
            matrix_A.append(row)

    # Для образования списка с названиями колонок матрицы
    name_rows = matrix_A[0]
    name_rows = name_rows[1:]
    del matrix_A[0]

    # Для образования списка с названиями строк
    name_rows = []
    for num, value in enumerate(matrix_A):
        name_rows.append(value[0])
        del matrix_A[num][0]

    # Перевод коэффициентов из str в тип int
    for num_1, row in enumerate(matrix_A):
        for num_2, value in enumerate(row):
            matrix_A[num_1][num_2] = int(value)

    # ------------------------------------------------
    # Игрок В
    matrix_B = []
    file_way = input('Введите путь к файлу (матрица игрока Б): ')
    with open(file_way, 'r', newline='') as csvfile:
        spamreader = csv.reader(csvfile, delimiter=';')
        for row in spamreader:
            matrix_B.append(row)
```

Рисунок 34. Часть кода, в которой затрагиваются эти переменные

```
# Игрок B
matrix_B = []
file_way = input('Введите путь к файлу (матрица игрока Б): ')
with open(file_way, 'r', newline='') as csvfile:
    spamreader = csv.reader(csvfile, delimiter=';')
    for row in spamreader:
        matrix_B.append(row)

# Для образования списка с названиями колонок матрицы
name_columns = matrix_B[0]
name_columns = name_columns[1:]
del matrix_B[0]

# Для образования списка с названиями строк
name_columns = []
for num, value in enumerate(matrix_B):
    name_columns.append(value[0])
    del matrix_B[num][0]

# Перевод коэффициентов из str в тип int
for num_1, row in enumerate(matrix_B):
    for num_2, value in enumerate(row):
        matrix_B[num_1][num_2] = int(value)
```

Рисунок 35. Часть кода, в которой затрагиваются эти переменные

5.2.2. Функции обработки информации

После того, как вы введете все необходимые данные, программа их получит и начнет первичную обработку.

При выборе ручного ввода все необходимые данные заносятся в словари, с помощью метода append ().

В случае случайной генерации, нужные данные генерируются с помощью random.randint(), при этом программа получает случайное целое число в заданном диапазоне. После этого данные также заносятся в пустые словари.

Если ввод данных осуществляется с помощью файла csv, то для начала файл необходимо открыть и прочитать программе, а затем разделить данные и также занести их в пустой словарь. Более того, программа образует список с названиями колонок матрицы, а также список с названиями строк. Не мало важен и перевод коэффициентов из типа данных str в тип int.

После всех необходимых операций с данными реализуются важные функции, рассчитывающие оптимальную чистую стратегию и оптимальную смешанную стратегию.

В функции pure_strategy реализуется оптимальная чистая стратегия. Она представляет собой нахождение минимума в каждой строке, а также нахождения максимума в каждом столбце. Это находится с помощью функции min (), max (). Более того, в этой функции рассчитывается количество равновесий по Нешу с помощью цикла for i in range.

В функции Mix_strategy реализуется оптимальная смешанная стратегия. С помощью цикла происходит подсчет элементов и вычисляется значения оптимальных стратегий. Более того, здесь также составляются таблицы смешанных стратегий игрока А и игрока Б.

Далее следует подробное описание входных, выходных и переменных, затрагивается в ходе работы.

Функция pure_strategy:

Входные данные:

- name_rows – список названий стратегий компании А (тип данных: list);
- name_columns – список названий стратегий компании В (тип данных: list);
- matrix_A – список значений весовой матрицы для компании А (тип данных: list);

- matrix_B – список значений весовой матрицы для компании B (тип данных: list);
- matrix_A_DF – список значений весовой матрицы для компании A (тип данных: list);
- matrix_B_DF – список значений весовой матрицы для компании B (тип данных: list);

Выходные данные:

- nash – количество равновесий по Нешу (тип данных: int);
- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- min_max_A – минимальное значение по строке весовой матрицы для игрока A (тип данных: list);
- min_max_B – минимальное значение по строке весовой матрицы для игрока B (тип данных: list);

Переменные, затрагиваемые в ходе работы:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- min_max_A – минимальное значение по строке весовой матрицы для игрока A (тип данных: list);
- min_max_B – минимальное значение по строке весовой матрицы для игрока B (тип данных: list);
- index_row – индекс по строке весовой матрицы (тип данных: int);
- index_column – индекс по столбцу весовой матрицы (тип данных: int).
- matrix_A – список значений весовой матрицы для компании A (тип данных: list);

- matrix_B – список значений весовой матрицы для компании B (тип данных: list);
- matrix_A_DF – список значений весовой матрицы для компании A (тип данных: list);
- matrix_B_DF – список значений весовой матрицы для компании B (тип данных: list);
- nash_A – список максимальных значений в матрице игрока A (тип данных: list);
- nash_B - список максимальных значений в матрице игрока B (тип данных: list);
- nash – количество равновесий по Нешу (тип данных: int);

```
def pure_strategy(matrix_A, matrix_B, name_rows, name_columns, matrix_A_DF, matrix_B_DF):
    min_max_A = list(matrix_A.max())
    min_max_B = list(matrix_B.max(axis=1))

    index_column = min_max_A.index(min(min_max_A)) # индекс минимального значения по столбцам в матрице игрока А
    index_row = min_max_B.index(min(min_max_B)) # индекс минимального значения по строкам в матрице игрока В

    print('Оптимальная чистая стратегия для игрока А: ', name_rows[index_column])
    print('Цена игры для игрока А при выборе чистой оптимальной стратегии: ', min(min_max_A))

    print('Оптимальная чистая стратегия для игрока Б: ', name_columns[index_row])
    print('Цена игры для игрока Б при выборе чистой оптимальной стратегии: ', min(min_max_B))

    # ------------------------------
    # Общая (суммарная) цена игры
    print('Общая (суммарная) цена игры: {}'.format(min(min_max_A) + min(min_max_B)))

    # ------------------------------
    # Количество равновесий по Нэшу
    nash_A = []
    for num_row, row in enumerate(matrix_A_DF):
        for num_value, value in enumerate(row):
            if value in min_max_A:
                nash_A.append([num_row, num_value])

    nash_B = []
    for num_row, row in enumerate(matrix_B_DF):
        for num_value, value in enumerate(row):
            if value in min_max_B:
                nash_B.append([num_row, num_value])
```

Рисунок 36. Часть кода, в которой затрагиваются эти переменные

```
    nash = 0
    for blns_A in nash_A:
        for blns_B in nash_B:
            if blns_A == blns_B:
                nash += 1
    print('\nКоличество равновесий по Нэшу: ', nash)

pure_strategy = pure_strategy(matrix_A, matrix_B, name_rows, name_columns, matrix_A_DF, matrix_B_DF)
```

Рисунок 37. Часть кода, в которой затрагиваются эти переменные

Функция Mix_strateg:

Входные данные:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);
- matrix_A – список значений весовой матрицы для компании A (тип данных: list);
- matrix_B – список значений весовой матрицы для компании B (тип данных: list);

Выходные данные:

- res - это массив с вероятностями стратегий игроков (тип данных: scipy.optimize.optimize.OptimizeResult);
- mytable – таблицы, которые выводят стратегии и проценты (тип данных: prettytable.prettytable.PrettyTable);
- price_game_B – цена игры A (тип данных: int);
- price_game_A – цена игры B (тип данных: int);

Переменные, затрагиваемые в ходе работы:

- name_rows – список названий стратегий компании A (тип данных: list);
- name_columns – список названий стратегий компании B (тип данных: list);

- matrix_A – список значений весовой матрицы для компании A (тип данных: list);
- matrix_B – список значений весовой матрицы для компании B (тип данных: list);
- price_game_B – цена игры A (тип данных: int);
- price_game_A – цена игры B (тип данных: int);
- mytable – таблицы, которые выводят стратегии и проценты (тип данных: prettytable.prettytable.PrettyTable);
- mix_strategy_A — это вероятности стратегий в процентах игрока A (тип данных: list);
- mix_strategy_B — это вероятности стратегий в процентах игрока B (тип данных: list);
- help_matrix_A – вспомогательная таблица матрицы A (тип данных: list);
- help_matrix_B - вспомогательная таблица матрицы A (тип данных: list);

```
def Mix_strategy(matrix_A, matrix_B, name_rows, name_columns):
    matrix_A = np.array(matrix_A)
    matrix_B = np.array(matrix_B)

    game = nash.Game(matrix_A, matrix_B)
    equilibria = game.support_enumeration()

    mix_strategy_A, mix_strategy_B = [], [] # смешанная стратегия
    help_matrix_A, help_matrix_B = [], []   # вспомогательные матрицы
    price_game_A, price_game_B = 0, 0       # цена игры
    optimal_strategy = None                 # Оптимальная стратегия


    for strategy in equilibria:
        for row in range(len(matrix_A)):
            for value in range(len(matrix_A[0])):
                help_matrix_A.append(matrix_A[row][value] * strategy[0][row] * strategy[1][value])
                help_matrix_B.append(matrix_B[row][value] * strategy[0][row] * strategy[1][value])

        if (sum(help_matrix_A) + sum(help_matrix_B)) >= (price_game_A + price_game_B):
            price_game_A = sum(help_matrix_A)
            price_game_B = sum(help_matrix_B)
            optimal_strategy = strategy

    for i in strategy[0]:
        mix_strategy_A.append(str(round(i*100)) + '%')

    for i in strategy[1]:
        mix_strategy_B.append(str(round(i*100)) + '%')
```

Рисунок 38. Часть кода, в которой затрагиваются эти переменные

```
# Для вывода таблицы PrettyTable
mytable = PrettyTable()
mytable.field_names = name_rows     # имена полей таблицы
mytable.add_row(mix_strategy_A)     # добавление данных по одной строке за раз
print('\nТаблица смешанных стратегий для компании А:\n' + str(mytable))

print('Цена игры для игрока А при выборе смешанной оптимальной стратегии: {}'.format(round(price_game_A, 3)))

mytable = PrettyTable()
mytable.field_names = name_columns  # имена полей таблицы
mytable.add_row(mix_strategy_B)     # добавление данных по одной строке за раз
print('Таблица смешанных стратегий для компании Б:\n' + str(mytable))

print('Цена игры для игрока Б при выборе смешанной оптимальной стратегии: {}'.format(round(price_game_B, 3)))
print('Общая цена игры в случае использования оптимальных стратегий: {}'.format(round(price_game_A + price_game_B, 3)))
```

Рисунок 39. Часть кода, в которой затрагиваются эти переменные

5.2.3. Функции вывода информации

Метод вывода информации (он заключен внутри каждой функции)

Что делает: осуществляет вывод необходимой информации

Вывод информации осуществляется с помощью функции print ()

Затрагиваемые переменные:

- name_rows – список названий стратегий компании А (тип данных: list);
- name_columns – список названий стратегий компании В (тип данных: list);
- mytable – таблицы, которые выводят стратегии и проценты (тип данных: prettytable.prettytable.PrettyTable);
- matrix_A – список значений весовой матрицы для компании А (тип данных: list);
- matrix_B – список значений весовой матрицы для компании В (тип данных: list);
- price_game_B – цена игры А (тип данных: int);
- price_game_A – цена игры В (тип данных: int);
- min_max_A – минимальное значение по строке весовой матрицы для игрока А (тип данных: list);

- min_max_B – минимальное значение по строке весовой матрицы для игрока B (тип данных: list);

```
print('Оптимальная чистая стратегия для игрока А: ', name_rows[index_column])
print('Цена игры для игрока А при выборе чистой оптимальной стратегии: ', min(min_max_A))

print('Оптимальная чистая стратегия для игрока Б: ', name_columns[index_row])
print('Цена игры для игрока Б при выборе чистой оптимальной стратегии: ', min(min_max_B))
```

Рисунок 40. Вывод оптимальных чистых стратегий для обоих игроков

```
# Общая (суммарная) цена игры
print('Общая (суммарная) цена игры: {}'.format(min(min_max_A) + min(min_max_B)))
```

Рисунок 41. Вывод общей цены игры

```
print('\nКоличество равновесий по Нэшу: ', nash)
```

Рисунок 42. Количество равновесий по Нэшу

```
print('\nТаблица смешанных стратегий для компании А:\n' + str(mytable))

print('Цена игры для игрока А при выборе смешанной оптимальной стратегии: {}'.format(round(price_game_A, 3)))
```

Рисунок 43. Вывод таблицы смешанных стратегий для игрока А

```
print('Таблица смешанных стратегий для компании Б:\n' + str(mytable))

print('Цена игры для игрока Б при выборе смешанной оптимальной стратегии: {}'.format(round(price_game_B, 3)))
print('Общая цена игры в случае использования оптимальных стратегий: {}'.format(round(price_game_A + price_game_B, 3)))
```

Рисунок 44. Вывод таблицы смешанных стратегий для игрока Б

6 Тестирование

7 Проведём тестирование нашей программы и сравним полученные показатели, чтобы сделать вывод о предпочтительном варианте использования нашей программы или онлайн-калькулятора[1] под условия заказчика.

[1] https://math.semestr.ru/games/antogonist.php

6.1. Тестирование задачи о нахождении выигрышной стратегии для антагонистической игры

6.1.1. Проверка №1 матрица 3x3:

6.1.1.1. Тест №1 кодом Python:

Выбираем метод ввода информации, количество стратегий (размер матрицы) и получаем оптимальную стратегию.

```
Способ ввода данных: 2
Количество стратегий игрока A: 3

Весовая матрица:
      1    2    3
1  246  882  591
2  217  314  375
3  310  496  603

Оптимальная чистая стратегия для игрока A:  3
Цена игры для игрока A при выборе чистой оптимальной стратегии:  310
Оптимальная чистая стратегия для игрока B:  1
Цена игры для игрока B при выборе чистой оптимальной стратегии:  310

Цена игры для игрока A при выборе смешанной оптимальной стратегии:  310.0
Таблица смешанных стратегий для игрока A:
 +----+----+------+
| 1  | 2  |  3   |
+----+----+------+
| 0% | 0% | 100% |
+----+----+------+
Таблица смешанных стратегий для игрока B:
 +------+----+----+
|  1   | 2  | 3  |
+------+----+----+
| 100% | 0% | 0% |
+------+----+----+
```

Рисунок 45. Результат обработки запроса в питоне 1

6.1.1.2. Тест №1 онлайн-калькулятором:

Выбираем размер матрицы и вводим необходимые данные в предназначенные поля. Метод решений: аналитический. Все сопутствующие «галочки» необходимо снять.

Находим гарантированный выигрыш, определяемый нижней ценой игры a = $\max(a_i) = 310$, которая указывает на максимальную чистую стратегию A_3.

Верхняя цена игры b = $\min(b_j) = 310$.

Седловая точка (3, 1) указывает решение на пару альтернатив (A3,B1). Цена игры равна 310.

Рисунок 46. Ответ в онлайн-калькуляторе 1

6.1.2. Проверка №2 матрица 3x3:

6.1.2.1. Тест №2 кодом Python:

Выбираем метод ввода информации, количество стратегий (размер матрицы) и получаем оптимальную стратегию.

```
Способ ввода данных: 2
Количество стратегий игрока A: 3

Весовая матрица:
      1    2    3
1  476  639  892
2  433  407  971
3  326  663  724

Оптимальная чистая стратегия для игрока A:  1
Цена игры для игрока A при выборе чистой оптимальной стратегии:  476
Оптимальная чистая стратегия для игрока B:  1
Цена игры для игрока B при выборе чистой оптимальной стратегии:  476

Цена игры для игрока A при выборе смешанной оптимальной стратегии:  475.9996
Таблица смешанных стратегий для игрока A:
 +------+----+----+
|  1   | 2  | 3  |
+------+----+----+
| 100% | 0% | 0% |
+------+----+----+
Таблица смешанных стратегий для игрока B:
 +------+----+----+
|  1   | 2  | 3  |
+------+----+----+
| 100% | 0% | 0% |
+------+----+----+
```

Рисунок 47. Результат обработки запроса в питоне 2

6.1.2.2. Тест №2 онлайн-калькулятором:

Выбираем размер матрицы и вводим необходимые данные в предназначенные поля. Метод решений: аналитический. Все сопутствующие «галочки» необходимо снять.

Находим гарантированный выигрыш, определяемый нижней ценой игры a = $\max(a_i)$ = 476, которая указывает на максимальную чистую стратегию A_1.
Верхняя цена игры b = $\min(b_j)$ = 476.
Седловая точка (1, 1) указывает решение на пару альтернатив (A1,B1). Цена игры равна 476.

Рисунок 48. Ответ в онлайн-калькуляторе 2

6.1.3. Проверка №3 матрица 3x3:

6.1.3.1. Тест №3 кодом Python:

Выбираем метод ввода информации, количество стратегий (размер матрицы) и получаем оптимальную стратегию.

```
Способ ввода данных: 2
Количество стратегий игрока A: 3

Весовая матрица:
      1    2    3
1  611  670  937
2  592  601  789
3  820  836  278

Оптимальная чистая стратегия для игрока A:  1
Цена игры для игрока A при выборе чистой оптимальной стратегии:  611
Оптимальная чистая стратегия для игрока B:  1
Цена игры для игрока B при выборе чистой оптимальной стратегии:  820

Цена игры для игрока A при выборе смешанной оптимальной стратегии:  689.4954
Таблица смешанных стратегий для игрока A:
 +-----+----+-----+
|  1  | 2  |  3  |
+-----+----+-----+
| 62% | 0% | 38% |
+-----+----+-----+
Таблица смешанных стратегий для игрока B:
 +-----+----+-----+
|  1  | 2  |  3  |
+-----+----+-----+
| 76% | 0% | 24% |
+-----+----+-----+
```

Рисунок 49. Результат обработки запроса в питоне 3

6.1.3.2. Тест №3 онлайн-калькулятором:

Выбираем размер матрицы и вводим необходимые данные в предназначенные поля. Метод решений: линейного программирования.

4. Проверим правильность решения игры с помощью **критерия оптимальности стратегии**.

$\sum a_{ij} q_j \le v$

$\sum a_{ij} p_i \ge v$

$M(P_1;Q) = (611^{*659}/_{868}) + (670*0) + (937^{*209}/_{868}) = 689.495 = v$

$M(P_2;Q) = (592^{*659}/_{868}) + (601*0) + (789^{*209}/_{868}) = 639.434 \le v$

$M(P_3;Q) = (820^{*659}/_{868}) + (836*0) + (278^{*209}/_{868}) = 689.495 = v$

$M(P;Q_1) = (611^{*271}/_{434}) + (592*0) + (820^{*163}/_{434}) = 689.495 = v$

$M(P;Q_2) = (670^{*271}/_{434}) + (601*0) + (836^{*163}/_{434}) = 732.346 \ge v$

$M(P;Q_3) = (937^{*271}/_{434}) + (789*0) + (278^{*163}/_{434}) = 689.495 = v$

Все неравенства выполняются как равенства или строгие неравенства, следовательно, решение игры найдено верно.

Рисунок 50. Ответ в онлайн-калькуляторе 3

6.1.4. Проверка №4 матрица 4x4:

6.1.4.1. Тест №4 кодом Python:

Выбираем метод ввода информации, количество стратегий (размер матрицы) и получаем оптимальную стратегию.

```
Способ ввода данных: 2
Количество стратегий игрока A: 4

Весовая матрица:
      1    2    3    4
1  285  388  265  351
2  926  166  575  109
3   52  948  546  967
4  553  192  518  941

Оптимальная чистая стратегия для игрока A:  1
Цена игры для игрока A при выборе чистой оптимальной стратегии:  265
Оптимальная чистая стратегия для игрока B:  3
Цена игры для игрока B при выборе чистой оптимальной стратегии:  575

Цена игры для игрока A при выборе смешанной оптимальной стратегии:  520.3277
Таблица смешанных стратегий для игрока A:
 +----+-----+-----+----+
| 1  |  2  |  3  | 4  |
+----+-----+-----+----+
| 0% | 52% | 45% | 3% |
+----+-----+-----+----+
Таблица смешанных стратегий для игрока B:
 +-----+-----+----+-----+
|  1  |  2  | 3  |  4  |
+-----+-----+----+-----+
| 48% | 31% | 0% | 21% |
+-----+-----+----+-----+
```

Рисунок 51. Результат обработки запроса в питоне 4

6.1.4.2. Тест №4 онлайн-калькулятором:

Выбираем размер матрицы и вводим необходимые данные в предназначенные поля. Метод решений: линейного программирования.

$M(P;Q_1) = (285*0) + (926*0.52) + (52*0.452) + (553^{*9158}/_{326103}) = 520.328 = v$

$M(P;Q_2) = (388*0) + (166*0.52) + (948*0.452) + (192^{*9158}/_{326103}) = 520.328 = v$

$M(P;Q_3) = (265*0) + (575*0.52) + (546*0.452) + (518^{*9158}/_{326103}) = 560.286 \geq v$

$M(P;Q_4) = (351*0) + (109*0.52) + (967*0.452) + (941^{*9158}/_{326103}) = 520.328 = v$

Все неравенства выполняются как равенства или строгие неравенства, следовательно, решение игры найдено верно.

Рисунок 52. Ответ в онлайн калькуляторе 4

6.1.5. Проверка №5 матрица 4x4:

6.1.5.1. Тест №5 кодом Python:

Выбираем метод ввода информации, количество стратегий (размер матрицы) и получаем оптимальную стратегию.

```
Способ ввода данных: 2
Количество стратегий игрока A: 4

Весовая матрица:
      1    2    3    4
1  150  215  857  427
2  711  390  575  418
3  304  263  762  212
4  188  936  318   64

Оптимальная чистая стратегия для игрока A:  2
Цена игры для игрока A при выборе чистой оптимальной стратегии:  390
Оптимальная чистая стратегия для игрока B:  4
Цена игры для игрока B при выборе чистой оптимальной стратегии:  427

Цена игры для игрока A при выборе смешанной оптимальной стратегии:  406.9867
Таблица смешанных стратегий для игрока A:
 +----+-----+----+----+
| 1  |  2  | 3  | 4  |
+----+-----+----+----+
| 0% | 97% | 0% | 3% |
+----+-----+----+----+
Таблица смешанных стратегий для игрока B:
 +----+-----+----+-----+
| 1  |  2  | 3  |  4  |
+----+-----+----+-----+
| 0% | 39% | 0% | 61% |
+----+-----+----+-----+
```

Рисунок 53. Результат обработки запроса в питоне 5

6.1.5.2. Тест №5 онлайн-калькулятором:

Выбираем размер матрицы и вводим необходимые данные в предназначенные поля. Метод решений: линейного программирования.

4. Проверим правильность решения игры с помощью **критерия оптимальности стратегии**.

$\sum a_{ij} q_j \le v$

$\sum a_{ij} p_i \ge v$

$M(P_1;Q) = (150*0) + (215*0.393) + (857*0) + (427*0.607) = 343.613 \le v$

$M(P_2;Q) = (711*0) + (390*0.393) + (575*0) + (418*0.607) = 406.987 = v$

$M(P_3;Q) = (304*0) + (263*0.393) + (762*0) + (212*0.607) = 232.06 \le v$

$M(P_4;Q) = (188*0) + (936*0.393) + (318*0) + (64*0.607) = 406.987 = v$

$M(P;Q_1) = (150*0) + (711*{}^{218}/_{225}) + (304*0) + (188*{}^{7}/_{225}) = 694.729 \ge v$

$M(P;Q_2) = (215*0) + (390*{}^{218}/_{225}) + (263*0) + (936*{}^{7}/_{225}) = 406.987 = v$

$M(P;Q_3) = (857*0) + (575*{}^{218}/_{225}) + (762*0) + (318*{}^{7}/_{225}) = 567.004 \ge v$

$M(P;Q_4) = (427*0) + (418*{}^{218}/_{225}) + (212*0) + (64*{}^{7}/_{225}) = 406.987 = v$

Все неравенства выполняются как равенства или строгие неравенства, следовательно, решение игры найдено верно.

Рисунок 54. Ответ онлайн-калькулятора 5

6.2. Тестирование задачи о нахождении выигрышной стратегии для биматричной игры

Проведём тестирование нашей программы и сравним полученные показатели, чтобы сделать вывод о предпочтительном варианте использования нашей программы или онлайн-калькулятора[2] под условия заказчика.

6.2.1. Проверка №1 матрица 3x3

6.2.1.1. Тест №1 кодом Python:

Выбираем метод ввода информации, размер матрицы и получаем оптимальные и смешанные стратегии для обоих игроков.

[2] https://math.semestr.ru/games/bimatrix.php

```
Способ ввода данных: 2
Количество стратегий игрока А: 3
Количество стратегий игрока Б: 3

Весовая матрица игрока А:
      1    2    3
1  879  471  634
2  770  488  144
3  991   56  961

Весовая матрица игрока Б:
      1    2    3
1   88  147  414
2  517  282   74
3  246  863  866

Оптимальная чистая стратегия для игрока А:  2
Цена игры для игрока А при выборе чистой оптимальной стратегии:  488
Оптимальная чистая стратегия для игрока Б:  1
Цена игры для игрока Б при выборе чистой оптимальной стратегии:  414
Общая (суммарная) цена игры: 902

Количество равновесий по Нэшу:  1

Таблица смешанных стратегий для компании А:
+----+----+------+
| 1  | 2  |  3   |
+----+----+------+
| 0% | 0% | 100% |
+----+----+------+
Цена игры для игрока А при выборе смешанной оптимальной стратегии: 961.0
Таблица смешанных стратегий для компании Б:
+----+----+------+
| 1  | 2  |  3   |
+----+----+------+
| 0% | 0% | 100% |
+----+----+------+
Цена игры для игрока Б при выборе смешанной оптимальной стратегии: 866.0
Общая цена игры в случае использования оптимальных стратегий: 1827.0
```

Рисунок 55. Результат обработки запроса в питоне 1

6.2.1.2. Тест №1 онлайн-калькулятором:

Выбираем размер матрицы и вводим необходимые данные в предназначенные поля. Запускаем программу.

Таким образом, найдены 1 равновесные ситуации по Нэшу (3;3).
Эти ситуации оказались оптимальные по Парето для обоих игроков.
В равновесной ситуации (3,3) игрок 1 выигрывает 961 единиц, а игрок 2 - 866 единицы.

Рисунок 56. Ответ в онлайн калькуляторе 1

6.2.2. Проверка №2 матрица 3x3:

6.2.2.1. Тест №2 кодом Python:

Выбираем метод ввода информации, размер матрицы и получаем оптимальные и смешанные стратегии для обоих игроков.

```
Способ ввода данных: 2
Количество стратегий игрока А: 3
Количество стратегий игрока Б: 3

Весовая матрица игрока А:
      1    2    3
1  240  133  374
2  813  588  498
3  194  339  480

Весовая матрица игрока Б:
      1    2    3
1  422  822  282
2  721  897  832
3  166  246  331

Оптимальная чистая стратегия для игрока А:  3
Цена игры для игрока А при выборе чистой оптимальной стратегии:  498
Оптимальная чистая стратегия для игрока Б:  3
Цена игры для игрока Б при выборе чистой оптимальной стратегии:  331
Общая (суммарная) цена игры: 829

Количество равновесий по Нэшу:  1

Таблица смешанных стратегий для компании А:
+----+------+----+
| 1  |  2   | 3  |
+----+------+----+
| 0% | 100% | 0% |
+----+------+----+
Цена игры для игрока А при выборе смешанной оптимальной стратегии: 588.0
Таблица смешанных стратегий для компании Б:
+----+------+----+
| 1  |  2   | 3  |
+----+------+----+
| 0% | 100% | 0% |
+----+------+----+
Цена игры для игрока Б при выборе смешанной оптимальной стратегии: 897.0
Общая цена игры в случае использования оптимальных стратегий: 1485.0
```

Рисунок 57. Результат обработки запроса в питоне 2

6.2.2.2. Тест №2 онлайн-калькулятором:

Выбираем размер матрицы и вводим необходимые данные в предназначенные поля. Запускаем программу.

Таким образом, найдены 1 равновесные ситуации по Нэшу (2;2).
Эти ситуации оказались оптимальные по Парето для обоих игроков.
В равновесной ситуации (2,2) игрок 1 выигрывает 588 единиц, а игрок 2 - 897 единицы.

Рисунок 58. Ответ в онлайн-калькуляторе 2

6.2.3. Проверка №3 матрица 3x3:

6.2.3.1. Тест №3 кодом Python:

Выбираем метод ввода информации, размер матрицы и получаем оптимальные и смешанные стратегии для обоих игроков.

```
Способ ввода данных: 2
Количество стратегий игрока А: 3
Количество стратегий игрока Б: 3

Весовая матрица игрока А:
      1    2    3
1  728  294  110
2  668  657  510
3   43  222  572

Весовая матрица игрока Б:
      1    2    3
1   30  596  545
2  727  444  578
3  405  798  949

Оптимальная чистая стратегия для игрока А:  3
Цена игры для игрока А при выборе чистой оптимальной стратегии:  572
Оптимальная чистая стратегия для игрока Б:  1
Цена игры для игрока Б при выборе чистой оптимальной стратегии:  596
Общая (суммарная) цена игры: 1168

Количество равновесий по Нэшу:  1

Таблица смешанных стратегий для компании А:
+----+-----+-----+
| 1  |  2  |  3  |
+----+-----+-----+
| 0% | 78% | 22% |
+----+-----+-----+
Цена игры для игрока А при выборе смешанной оптимальной стратегии: 1743.65
Таблица смешанных стратегий для компании Б:
+----+----+-----+
| 1  | 2  |  3  |
+----+----+-----+
| 9% | 0% | 91% |
+----+----+-----+
Цена игры для игрока Б при выборе смешанной оптимальной стратегии: 2177.363
Общая цена игры в случае использования оптимальных стратегий: 3921.013
```

Рисунок 59. Результат обработки запроса в питоне 3

6.2.3.2. Тест №3 онлайн-калькулятором:

Выбираем размер матрицы и вводим необходимые данные в предназначенные поля. Запускаем программу.

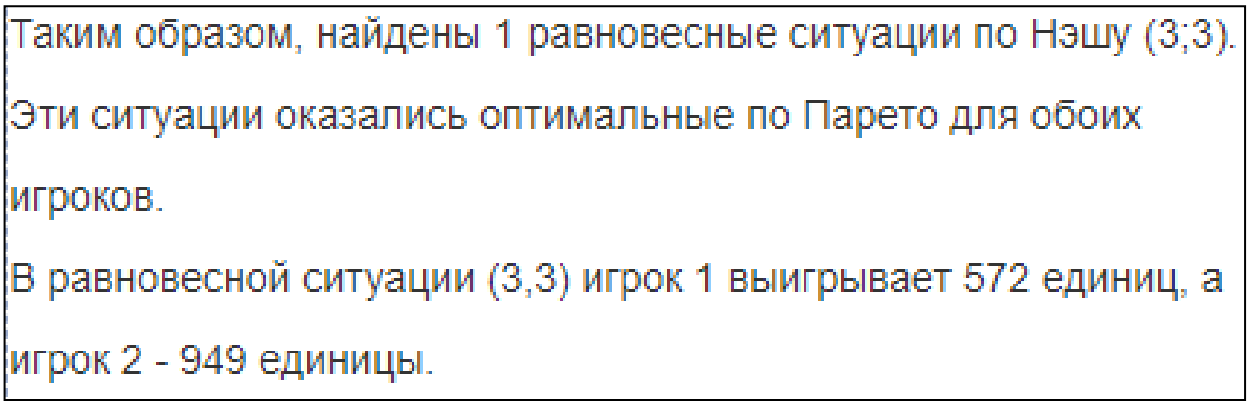

Рисунок 60. Ответы онлайн-калькулятора 3

6.2.4. Проверка №4 матрица 4x4:

6.2.4.1. Тест №4 кодом Python:

Выбираем метод ввода информации, размер матрицы и получаем оптимальные и смешанные стратегии для обоих игроков.

```
Способ ввода данных: 2
Количество стратегий игрока А: 4
Количество стратегий игрока Б: 4

Весовая матрица игрока А:
      1    2    3    4
1  395  590  430  110
2  247  712   93  881
3  623  191  446  255
4  799  957  536  165

Весовая матрица игрока Б:
      1    2    3    4
1  663  197  692  747
2  203  424    9  144
3  673  839  483  543
4  533  312  496   39

Оптимальная чистая стратегия для игрока А:  3
Цена игры для игрока А при выборе чистой оптимальной стратегии:  536
Оптимальная чистая стратегия для игрока Б:  2
Цена игры для игрока Б при выборе чистой оптимальной стратегии:  424
Общая (суммарная) цена игры: 960

Количество равновесий по Нэшу:  1

Таблица смешанных стратегий для компании А:
+----+----+----+------+
| 1  | 2  | 3  |  4   |
+----+----+----+------+
| 0% | 0% | 0% | 100% |
+----+----+----+------+
Цена игры для игрока А при выборе смешанной оптимальной стратегии: 799.0
Таблица смешанных стратегий для компании Б:
+------+----+----+----+
|  1   | 2  | 3  | 4  |
+------+----+----+----+
| 100% | 0% | 0% | 0% |
+------+----+----+----+
Цена игры для игрока Б при выборе смешанной оптимальной стратегии: 533.0
Общая цена игры в случае использования оптимальных стратегий: 1332.0
```

Рисунок 61. Результат обработки запроса в питоне 4

6.2.4.2. Тест №4 онлайн-калькулятором:

Выбираем размер матрицы и вводим необходимые данные в предназначенные поля. Запускаем программу.

Таким образом, найдены 1 равновесные ситуации по Нэшу (4;1).
Эти ситуации оказались оптимальные по Парето для обоих игроков.
В равновесной ситуации (4,1) игрок 1 выигрывает 799 единиц, а игрок 2 - 533 единицы.

Рисунок 62. Ответы онлайн-калькулятора 4

6.2.5. Проверка №5 матрица 4x4:

6.2.5.1. Тест №5 кодом Python:

Выбираем метод ввода информации, размер матрицы и получаем оптимальные и смешанные стратегии для обоих игроков.

```
Способ ввода данных: 2
Количество стратегий игрока А: 4
Количество стратегий игрока Б: 4

Весовая матрица игрока А:
      1    2    3    4
1  571   81   53  357
2  440  644  861  381
3  182  797  428  180
4  853  804  191  610

Весовая матрица игрока Б:
      1    2    3    4
1  882  231  708  294
2  114  949   48  204
3  786  147  615  194
4  905  480  121  743

Оптимальная чистая стратегия для игрока А:  4
Цена игры для игрока А при выборе чистой оптимальной стратегии:  610
Оптимальная чистая стратегия для игрока Б:  3
Цена игры для игрока Б при выборе чистой оптимальной стратегии:  786
Общая (суммарная) цена игры: 1396

Количество равновесий по Нэшу:  1

Таблица смешанных стратегий для компании А:
+----+----+----+------+
| 1  | 2  | 3  |  4   |
+----+----+----+------+
| 0% | 0% | 0% | 100% |
+----+----+----+------+
Цена игры для игрока А при выборе смешанной оптимальной стратегии: 853.0
Таблица смешанных стратегий для компании Б:
+------+----+----+----+
|  1   | 2  | 3  | 4  |
+------+----+----+----+
| 100% | 0% | 0% | 0% |
+------+----+----+----+
Цена игры для игрока Б при выборе смешанной оптимальной стратегии: 905.0
Общая цена игры в случае использования оптимальных стратегий: 1758.0
```

Рисунок 63. Результат обработки запроса в питоне 5

6.2.5.2. Тест №5 онлайн-калькулятором:

Выбираем размер матрицы и вводим необходимые данные в предназначенные поля. Запускаем программу.

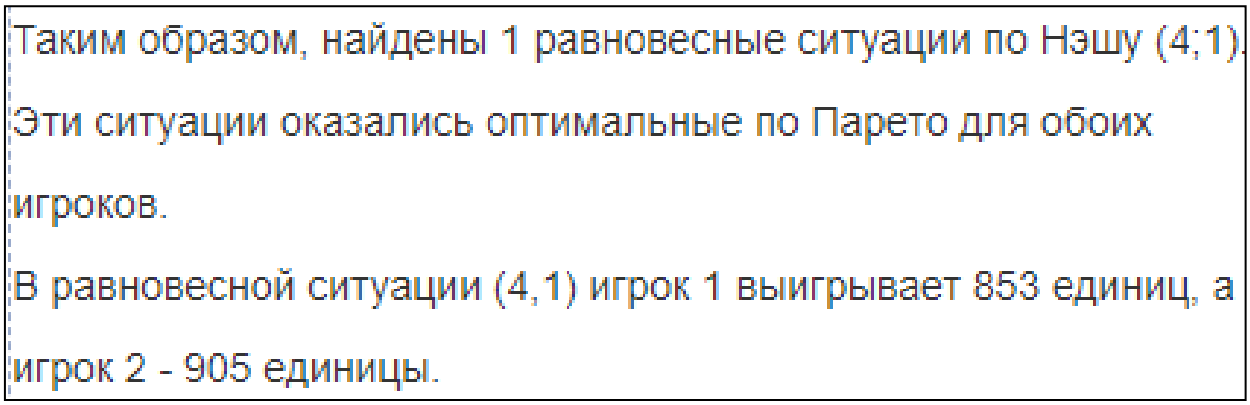

Рисунок 64. Ответ онлайн-калькулятора 5

7. Заключение

7.1. Заключение по антагонистической задаче

Наш представленный код 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 через Python и проверка в онлайн-калькуляторе:

```
Способ ввода данных: 1
Введите количество стратегий для компании A: 3
Введите название 1 стратегии компании A: Ddos
Введите название 2 стратегии компании A: Троян
Введите название 3 стратегии компании A: Взлом
Введите количество стратегий для компании B: 3
Введите название 1 стратегии компании B: Пароли
Введите название 2 стратегии компании B: Сотрудники
Введите название 3 стратегии компании B: Файлы
Введите элемент строки 1 столбца 1: 5
Введите элемент строки 1 столбца 2: 35
Введите элемент строки 1 столбца 3: 15
Введите элемент строки 2 столбца 1: 40
Введите элемент строки 2 столбца 2: 10
Введите элемент строки 2 столбца 3: 95
Введите элемент строки 3 столбца 1: 75
Введите элемент строки 3 столбца 2: 55
Введите элемент строки 3 столбца 3: 45

Весовая матрица:
        Пароли  Сотрудники  Файлы
Ddos         5          35     15
Троян       40          10     95
Взлом       75          55     45

Оптимальная чистая стратегия для игрока A:  Взлом
Цена игры для игрока A при выборе чистой оптимальной стратегии:  45
Оптимальная чистая стратегия для игрока B:  Сотрудники
Цена игры для игрока B при выборе чистой оптимальной стратегии:  55

Цена игры для игрока A при выборе смешанной оптимальной стратегии:  50.2632
Таблица смешанных стратегий для игрока A:
 +------+-------+-------+
| Ddos | Троян | Взлом |
+------+-------+-------+
|  0%  |  11%  |  89%  |
+------+-------+-------+
Таблица смешанных стратегий для игрока B:
 +--------+------------+-------+
| Пароли | Сотрудники | Файлы |
+--------+------------+-------+
|   0%   |    53%     |  47%  |
+--------+------------+-------+
```

Рисунок 65. Результат выполнения антагонистической задачи в питоне

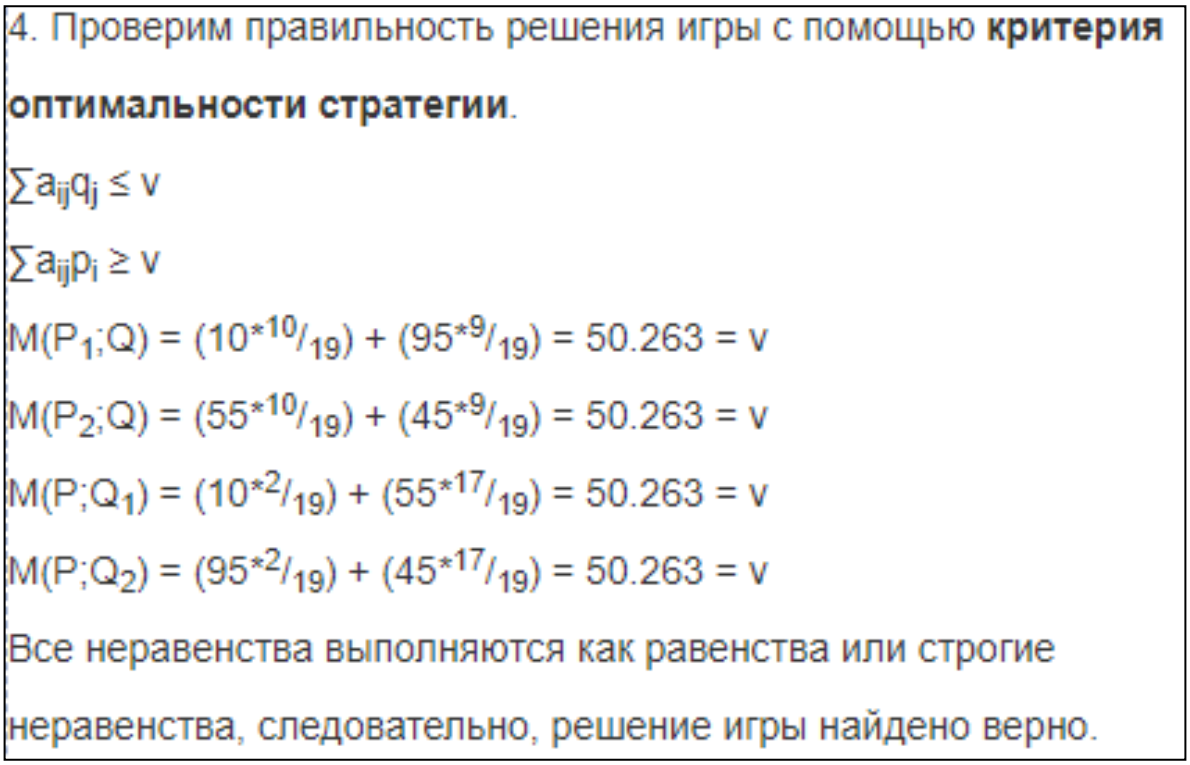

Рисунок 66. Ответ на антагонистическую задачу в онлайн-калькуляторе

Теперь сравним два алгоритма по критериям: эффективности, скорости использования алгоритма, простоты использования, надёжности в разрезе человеческого фактора и точности предоставляемого решения.

Таблица 6. Сравнение онлайн-калькулятора и питона

Критерий	**Python**	**Онлайн-калькулятор**
Эффективность	Высокая	Высокая
Скорость использования алгоритма	Высокая	Средняя
Простота использования	Высокая	Средняя
Надёжность (человеческий фактор)	Высокая	Средняя
Точность	Высокая	Высокая

Мы считаем, что представленный рукописный код на языке Python лучше, потому что он удобнее, быстрее и проще, а ещё имеет импорт .csv файлов и случайный ввод данных. Улучшением кода, к примеру, может послужить время выполнения запроса.

7.2. Заключение по биматричной задаче

Наш представленный код решает поставленную задачу. На основании тестирования данного алгоритма можно сделать вывод о том, что Python выводит самое оптимальное решение достаточно быстро.

Теперь сравним два алгоритма по критериям: эффективности, скорости использования алгоритма, простоты использования, надёжности в разрезе человеческого фактора и точности предоставляемого решения.

Таблица 7. Сравнение онлайн-калькулятора и питона

Критерий	**Python**	**Онлайн-калькулятор**
Эффективность	Высокая	Средняя
Скорость использования алгоритма	Высокая	Высокая
Простота использования	Высокая	Высокая
Надёжность (человеческий фактор)	Высокая	Высокая
Точность	Высокая	Высокая

Мы считаем, что представленный рукописный код на языке Python лучше, потому что он удобнее, быстрее и проще, а ещё имеет импорт .csv файлов и случайный ввод данных. Улучшением кода, к примеру, может послужить время выполнения запроса.