Дана следующая матричная игра:

	B1	B2	B3	B4
A1	2	4	8	5
A2	6	2	4	6
A3	3	2	5	4

1. Исследуйте игру на равновесие Нэша и решение в чистых стратегиях.
2. Сведите данную игру к задаче линейного программирования.
3. Решите игру в смешанных стратегиях. Можно привести любое решение на выбор, как с помощью хода симплекс-метода, так и с помощью соответствующего пакета Python.

1. Исследование игры:

На самом деле матрицу игры более правильно записать в следующем виде:

	B1	B2	B3	B4
A1	(2, -2)	(4, -4)	(8, -8)	(5, -5)
A2	(6, -6)	(2, -2)	(4, -4)	(6, -6)
A3	(3, -3)	(2, -2)	(5, -5)	(4, -4)

Это игра с нулевой суммой: для каждого выигрыша игрока A ровно столько же проигрывает игрок B (именно поэтому, в подобных матрицах обычно используют только по одному числу в каждой из ячеек).

Для того, чтобы существовало равновесие Нэша для решения в чистых стратегиях, необходимо чтобы у матрицы платежей (у матрицы игры) существовала седловая точка.

Седловая точка матрицы - это элемент матрицы, являющийся одновременно минимумом в своей строке и максимумом в своем столбце.

Минимумы по строкам:

- строка 1: min(2 , 4 , 8 , 5) = 2;
- строка 2: min(6 , 2 , 4 , 6) = 2;
- строка 3: min(3 , 2 , 5 , 4) = 2.

Максимумы по столбцам:

- столбец 1: max(2 , 6 , 3) = 6;
- столбец 2: max(4 , 2 , 2) = 4;
- столбец 3: max(8 , 4 , 5) = 8;
- столбец 4: max(5 , 6 , 4) = 6.

Видно, что у нас нет элемента в матрице, который бы одновременно являлся минимумом в своей строке и максимумом в своем столбце.

Поэтому равновесия Нэша для решения в чистых стратегиях для данной игры не существует.

2. Сведение данной игры к задаче линейного программирования:

Игра для первого игрока.

Пусть x_1, x_2, x_3 - вероятность выбора стратегии A_1, A_2, A_3 соответственно.

$x_1 + x_2 + x_3 = 1,\ x_1, x_2, x_3 \geq 0$

Пусть v — минимальный выигрыш первого игрока. Тогда задача максимизации минимального выигрыша записывается как:

$v \to max$

при условии, что:

$\sum_{i=1}^{3} x_i a_{ij} \geq v$

для всех $j = 1, 2, 3, 4$

Таким образом, исходная матричная игра сводится к следующей задаче линейного программирования:

$v \to max$

$2x_1 + 6x_2 + 3x_3 \geq v$

$4x_1 + 2x_2 + 2x_3 \geq v$

$8x_1 + 4x_2 + 5x_3 \geq v$

$5x_1 + 6x_2 + 4x_3 \geq v$

$x_1 + x_2 + x_3 = 1$

$x_1, x_2, x_3 \geq 0$

Делаю следующую замену переменных:

$z_1 = x_1/v, z_2 = x_2/v, z_3 = x_3/v$

Так как все значения в исходной матрице выплат больше нуля, то итоговый выигрыш v можем считать положительным. А значит, каждое из ограничений в системе, включая неравенства, можно поделить на это v.

$v \to max$

$2z_1 + 6z_2 + 3z_3 \geq 1$

$4z_1 + 2z_2 + 2z_3 \geq 1$

$8z_1 + 4z_2 + 5z_3 \geq 1$

$5z_1 + 6z_2 + 4z_3 \geq 1$

$z_1 + z_2 + z_3 = 1/v$

$z_1, z_2, z_3 \geq 0$

Если $v \to max$, то $1/v \to min$, а значит можно переписать систему следующим образом:

$z_1 + z_2 + z_3 \to min$

$2z_1 + 6z_2 + 3z_3 \geq 1$

$4z_1 + 2z_2 + 2z_3 \geq 1$

$8z_1 + 4z_2 + 5z_3 \geq 1$

$5z_1 + 6z_2 + 4z_3 \geq 1$

$z_1, z_2, z_3 \geq 0$
Получили задачу линейного программирования для первого игрока

Игра для второго игрока.
Пусть y_1, y_2, y_3, y_4 - вероятность выбора стратегии B_1, B_2, B_3, B_4 соответственно.
$y_1 + y_2 + y_3 + y_4 = 1,\ y_1, y_2, y_3, y_4 \geq 0$
Пусть w — максимальный выигрыш первого игрока. Тогда задача минимизации максимального выигрыша записывается как:
$w \to min$
при условии, что:
$\sum_{i=1}^{4} y_i a_{ij} \leq w$
для всех $j = 1, 2, 3$
Таким образом, исходная матричная игра сводится к следующей задаче линейного программирования:
$w \to min$
$2y_1 + 4y_2 + 8y_3 + 5y_4 \leq w$
$6y_1 + 2y_2 + 4y_3 + 6y_4 \leq w$
$3y_1 + 2y_2 + 5y_3 + 4y_4 \leq w$
$y_1 + y_2 + y_3 + y_4 = 1$
$y_1, y_2, y_3, y_4 \geq 0$
Получили задачу линейного программирования для второго игрока

3. Решение игры в смешанных стратегиях:
Для решения задачи (игры) я буду использовать язык python и библиотеку cvxpy.
Код решения задачи линейного программирования для первого игрока:

```
import cvxpy as cp

if __name__ == "__main__":
    z1 = cp.Variable(nonneg=True)
    z2 = cp.Variable(nonneg=True)
    z3 = cp.Variable(nonneg=True)
    objective = cp.Minimize(z1 + z2 + z3)
    constraints = [
        2 * z1 + 6 * z2 + 3 * z3 >= 1,
        4 * z1 + 2 * z2 + 2 * z3 >= 1,
        8 * z1 + 4 * z2 + 5 * z3 >= 1,
        5 * z1 + 6 * z2 + 4 * z3 >= 1
    ]
    problem = cp.Problem(objective, constraints)
    problem.solve()
    print(f"Status: {problem.status}")
    print(f"Optimal value: {problem.value:.2f}")
```

```
    print(f"z1: {z1.value:.2f}")
    print(f"z2: {z2.value:.2f}")
    print(f"z3: {z3.value:.2f}")
    print("Check constraint:")
    print(f"2 * z1 + 6 * z2 + 3 * z3 = {(2 * z1 + 6 * z2 + 3 *
z3).value} >= 1")
    print(f"4 * z1 + 2 * z2 + 2 * z3 = {(4 * z1 + 2 * z2 + 2 *
z3).value} >= 1")
    print(f"8 * z1 + 4 * z2 + 5 * z3 = {(8 * z1 + 4 * z2 + 5 *
z3).value} >= 1")
    print(f"5 * z1 + 6 * z2 + 4 * z3 = {(5 * z1 + 6 * z2 + 4 *
z3).value} >= 1")
    v = 1 / problem.value
    print("First player result :")
    print(f"v = {v}")
    print(f"x1 = {v * z1.value}")
    print(f"x2 = {v * z2.value}")
    print(f"x3 = {v * z3.value}")
```

Запускаю этот код на выполнение - получаю следующий результат:

```
Status: optimal
Optimal value: 0.30
z1: 0.20
z2: 0.10
z3: 0.00
Check constraint:
2 * z1 + 6 * z2 + 3 * z3 = 1.000000000044802 >= 1
4 * z1 + 2 * z2 + 2 * z3 = 1.0000000000223108 >= 1
8 * z1 + 4 * z2 + 5 * z3 = 2.0000000000562266 >= 1
5 * z1 + 6 * z2 + 4 * z3 = 1.6000000000526016 >= 1
First player result :
v = 3.333333333195291
x1 = 0.6666666666348302
x2 = 0.33333333332648685
x3 = 3.868302965064064e-11
```

Код решения задачи линейного программирования для второго игрока:

```
import cvxpy as cp

if __name__ == "__main__":
    y1 = cp.Variable(nonneg=True)
    y2 = cp.Variable(nonneg=True)
    y3 = cp.Variable(nonneg=True)
    y4 = cp.Variable(nonneg=True)
    w = cp.Variable(nonneg=True)
    objective = cp.Minimize(w)
```

```
    constraints = [
        2 * y1 + 4 * y2 + 8 * y3 + 5 * y4 <= w,
        6 * y1 + 2 * y2 + 4 * y3 + 6 * y4 <= w,
        3 * y1 + 2 * y2 + 5 * y3 + 4 * y4 <= w,
        y1 + y2 + y3 + y4 == 1
    ]
    problem = cp.Problem(objective, constraints)
    problem.solve()
    print(f"Status: {problem.status}")
    print(f"Optimal value: {problem.value:.2f}")
    print(f"y1: {y1.value:.2f}")
    print(f"y2: {y2.value:.2f}")
    print(f"y3: {y3.value:.2f}")
    print(f"y4: {y4.value:.2f}")
    print("Check constraint:")
    print(f"2 * y1 + 4 * y2 + 8 * y3 + 5 * y4 = {(2 * y1 + 4 * y2 + 8 *
y3 + 5 * y4).value:.2f} <= {w.value:.2f}")
    print(f"6 * y1 + 2 * y2 + 4 * y3 + 6 * y4 = {(6 * y1 + 2 * y2 + 4 *
y3 + 6 * y4).value:.2f} <= {w.value:.2f}")
    print(f"3 * y1 + 2 * y2 + 5 * y3 + 4 * y4 = {(3 * y1 + 2 * y2 + 5 *
y3 + 4 * y4).value:.2f} <= {w.value:.2f}")
    print(f"y1 + y2 + y3 + y4 = {(y1 + y2 + y3 + y4).value:.2f} == 1")
```

Запускаю этот код на выполнение - получаю следующий результат:

```
Status: optimal
Optimal value: 3.33
y1: 0.33
y2: 0.67
y3: 0.00
y4: 0.00
Check constraint:
2 * y1 + 4 * y2 + 8 * y3 + 5 * y4 = 3.33 <= 3.33
6 * y1 + 2 * y2 + 4 * y3 + 6 * y4 = 3.33 <= 3.33
3 * y1 + 2 * y2 + 5 * y3 + 4 * y4 = 2.33 <= 3.33
y1 + y2 + y3 + y4 = 1.00 == 1
```

Как итог, получаю следующие результаты:

Для первого игрока:

- Вероятность выбрать стратегию A1 равна 0.67;
- Вероятность выбрать стратегию A2 равна 0.33;
- Вероятность выбрать стратегию A3 равна 0;
- Минимальный выигрыш первого игрока 3.33.

Для второго игрока:

- Вероятность выбрать стратегию B1 равна 0.33;
- Вероятность выбрать стратегию B2 равна 0.67;
- Вероятность выбрать стратегию B3 равна 0;
- Вероятность выбрать стратегию B4 равна 0;