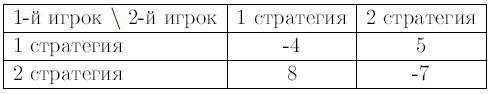
1. Теоретическая часть

Антагонистическая игра, или игра с нулевой

суммой, это игра, где выигрыш одного игрока всегда равен проигрышу другого. Условия игры можно свести в таблицу, называемую матрицей платежей.



В таблице приведены выигрыши первого игрока при различных вариантах выбора игроков, называемых стратегиями. Например, если первый игрок применит вторую стратегию, а второй применит первую стратегию, то первый игрок получит выигрыш в 8 единиц.

Ясно, что обоим игрокам следует использовать смешанные стратегии применять первую или вторую из своих возможных стратегий поочередно случайным образом с некоторыми вероятностями.

Можно доказать, что каждая антагонистическая игра имеет решение - пару оптимальных стратегий, отступать от которых невыгодно ни одному игроку, если другой продолжает ее придерживаться.

Метод Брауна - Робинсон заключается в следующем: при первой игре стратегии выбираются произвольно, а в каждой из последующих игр каждый из игроков выбирает стратегию, обеспечивающую максимальный выигрыш против того набора стратегий, который противник использовал в предыдущих партиях. Можно доказать, что вероятности выбора стратегий сходятся к оптимальным, а средний выигрыш - к цене игры.

1. Программный код

"""

Программа, моделирующая последовательность

антагонистических игр с заданной матрицей платежей

Автор: Афанасьев И.Е.

Дата написания: 20.09.2020

"""

iterations = 10000 # Количество ходов

start\_money = 10000 # Начальное количество денег у каждого игрока

matrix1 = [ [-4, 5], [8, -7] ] # Матрица платежей

# Инвертируем матрицу (Платежи для второго игрока)

q = len(matrix1)

matrix2 = []

for i in range(q):

matrix2.append([])

for i in matrix1:

for pos, j in enumerate(i):

matrix2[pos % q].append(- j)

# Класс игрок с матрицей платежей, текущим количеством

# денег, коэффициентами для принятия решения на основе

# стратегии противника и последним ходом

class gamer:

def \_\_init\_\_(self, matrice, money):

self.m = matrice

self.money = money

self.koef = [0] \* len(matrice[0])

self.last = 0

def strikeback(self, enemy\_strat):

self.koef[enemy\_strat] += 1

self.money += self.m[self.last][enemy\_strat]

return self.money

# Вычисление оптимального хода на основе

# предыдущих ходов противника

def math(self):

M = []

for i in self.m:

process = 0

for j in range(len(self.koef)):

process += self.koef[j] \* i[j]

M.append(process)

self.last = M.index(max(M))

return self.last

# Импортируем графическую библиотеку

import matplotlib.pyplot as plt

# Создаём два объекта класса "игрок"

player1 = gamer(matrix1, start\_money)

player2 = gamer(matrix2, start\_money)

# Инициализируем фигуру, на которой будем рисовать

fig = plt.figure()

#plt.ion()

color = ['red', 'green', 'blue', 'brown']

y = [] # Ось y

x = [] # Ось x

for i in range(len(color)):

y.append([])

x.append([])

# Собственно, игра

for i in range(iterations):

str1 = player1.math()

str2 = player2.math()

y[str1].append(player1.strikeback(str2))

x[str1].append(i)

y[str2 + 2].append(player2.strikeback(str1))

x[str2 + 2].append(i)

for i in range(len(x)):

plt.plot(x[i], y[i], c = color[i])

plt.show() # Отображение графика

print(

'Стратегии первого игрока: %s\nДеньги первого игрока: %d\n' %

(player2.koef, player1.money)

)

print(

'Стратегии второго игрока: %s\nДеньги второго игрока: %d' %

(player1.koef, player2.money)

)

1. Результаты работы программы

Длительность тестовой серии игры: 10000

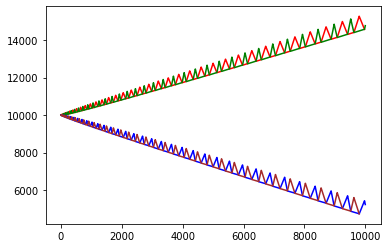
Стратегии первого игрока: [6300, 3700]

Деньги первого игрока: 14763

Стратегии второго игрока: [5013, 4987]

Деньги второго игрока: 5237

График изменения количества денег первого и второго игрока:



Теоретические вопросы:

1. matrix = [ [-4, 5], [8, -7] ]

S(x, y) = −4xy + 5x(1 − y) + 8(1 − x)y − 7(1 − x)(1 − y) = = −4xy - 5xy + 5x + 8y – 8xy – 7 + 7x + 7y -7xy = **-24xy + 12x + 15y - 7**

S(x, y) = -24xy + 12x + 15y – 7

Sx = -24y + 12 = 0

**y = 1/2**

Sy = -24x + 15 = 0

**x = 15/24**

S(15/24, 1/2) = -15/2 + 15/2 + 15/2 – 7 = **1/2**

1. Разница не велика, при увеличении числа ходов она стремится к теоретическим показателям
2. Второй игрок
3. Нет