

**Задание 1**

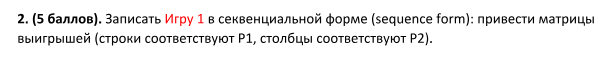


*Матрицы выигрышей для игроков:*

**Игрок P1 Игрок P2**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **DI** | **DJ** | **EH** | **EI** | **EJ** |  |  |  | **DI** | **DJ** | **EH** | **EI** | **EJ** |
|  | *a* | *a* | *a* | *e* | *e* | *e* |  |  | *b* | *b* | *b* | *f* | *f* | *f* |
| **AG** | *c* | *c* | *c* | *g* | *g* | *g* | **AG** | *d* | *d* | *d* | *h* | *h* | *h* |
| **BF** | *i* | *k* | *m* | *i* | *k* | *m* |  | **BF** | *j* | *l* | *n* | *j* | *l* | *n* |
| **CG** | *i* | *k* | *m* | *i* | *k* | *m* | **CG** | *j* | *l* | *n* | *j* | *l* | *n* |
| **CF** | *p* | *p* | *p* | *p* | *p* | *p* | **CF** | *q* | *q* | *q* | *q* | *q* | *q* |
| **CG** | *p* | *p* | *p* | *p* | *p* | *p* | **CG** | *q* | *q* | *q* | *q* | *q* | *q* |

**Задание 2**



*Матрицы выигрышей для игроков:*

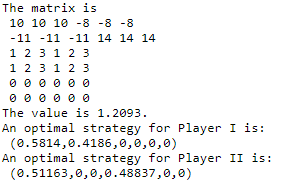
**Игрок P1 Игрок P2**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **D** | **E** | **H** | **I** | **J** |  |  |  | **D** | **E** | **H** | **I** | **J** |
|  | *0* | *0* | *0* | *0* | *0* | *0* |  |  | *0* | *0* | *0* | *0* | *0* | *0* |
| **A** | *0* | *0* | *0* | *0* | *0* | *0* | **A** | *0* | *0* | *0* | *0* | *0* | *0* |
| **B** | *0* | *0* | *0* | *i* | *k* | *m* |  | **B** | *0* | *0* | *0* | *j* | *l* | *n* |
| **C** | *p* | *0* | *0* | *0* | *0* | *0* | **C** | *Q* | *0* | *0* | *0* | *0* | *0* |
| **AF** | *0* | *a* | *e* | *0* | *0* | *0* | **AF** | *0* | *b* | *f* | *0* | *0* | *0* |
| **AG** | *0* | *c* | *g* | *0* | *0* | *0* | **AG** | *0* | *d* | *h* | *0* | *0* | *0* |

**Задание 3**



Решение с помощью онлайн-сервиса:





Для игр с нулевой суммой в случае когда *А = -В,*



Вектор *х* соответствует послеовательностяма вектор *y -* то

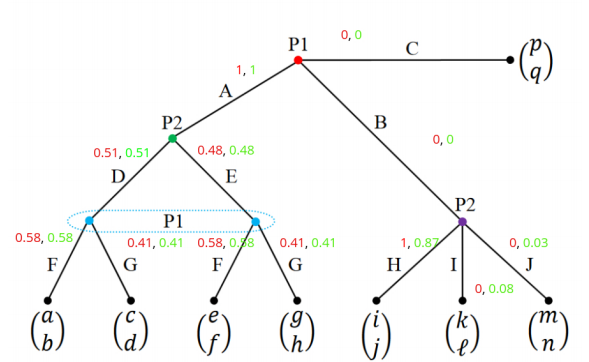
А =

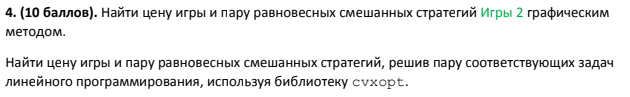
Код решения:

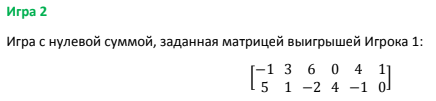
**from** cvxopt **import** solvers, matrix  
**import** numpy **as** np  
  
solvers.options[**'show\_progress'**] = **False**solvers.options[**'feastol'**] = 10\*\*(-10)  
  
E = np.array(  
 [[1, 0, 0, 0, 0, 0],  
 [-1, 1, 1, 1, 0, 0],  
 [0, -1, 0, 0, 1, 1]]  
)  
e = np.array([1, 0, 0]).reshape((3, 1))  
  
F = np.array(  
 [[1, 0, 0, 0, 0, 0],  
 [-1, 1, 1, 0, 0, 0],  
 [-1, 0, 0, 1, 1, 1]]  
)  
f = np.array([1, 0, 0]).reshape((3, 1))  
  
A\_ = np.array(  
 [[0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 1, 2, 3],  
 [0, 0, 0, 0, 0, 0],  
 [0, 10, -8, 0, 0, 0],  
 [0, -11, 14, 0, 0, 0]]  
)  
  
*# first player*c = np.vstack( (np.zeros((6, 1)), f) )  
  
G1 = np.hstack( (-A\_.T, -F.T) )  
G2 = np.hstack( (-np.eye(6), np.zeros( (6, 3) )) )  
G = np.vstack( (G1, G2) )  
h = np.zeros( (12, 1) )  
  
A = np.hstack( (E, np.zeros( (3, 3) )) )  
b = e  
  
c = matrix(c, tc=**'d'**)  
G = matrix(G, tc=**'d'**)  
h = matrix(h, tc=**'d'**)  
A = matrix(A, tc=**'d'**)  
b = matrix(b, tc=**'d'**)  
  
x = solvers.lp(c, G, h, A, b)[**'x'**][:6]  
print(x)   
*# [1.000, 0.999, 2.294e-07, -8.409-09, 0.581, 0.418]  
  
# second player*c = np.vstack((np.zeros((6, 1)), e))  
G1 = np.hstack((A\_, -E.T) )  
G2 = np.hstack((-np.eye(6), np.zeros( (6, 3))))  
G = np.vstack((G1, G2))  
h = np.zeros((12, 1))  
  
A = np.hstack((F, np.zeros((3, 3))))  
b = f  
  
c = matrix(c, tc=**'d'**)  
G = matrix(G, tc=**'d'**)  
h = matrix(h, tc=**'d'**)  
A = matrix(A, tc=**'d'**)  
b = matrix(b, tc=**'d'**)  
  
y = solvers.lp(c, G, h, A, b)[**'x'**][:6]  
print(y)   
*# [0.999, 0.511, 0.488, 0.878, 0.081, 0.0393]*print(x.T @ A\_ @ y)   
*# 1.20930232*

Цена игры такая же как в 1 пункте: 1.2093.

Ниже приведены равновесные смешанные стратегии, полученные в первом пункте (красным) и втором (зеленым). Они так же практически одинаковы, кроме ребер H, I, J.

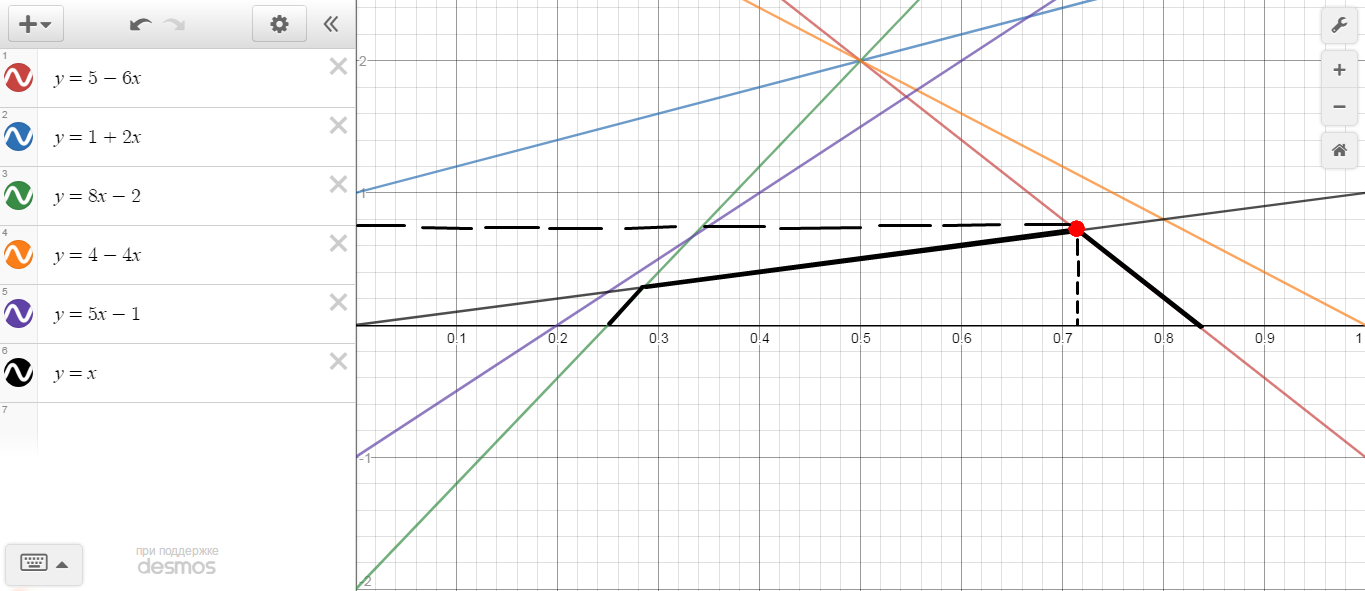






*Решение графическим методом*

Построим 6 прямых:



Максимум нижней огибающей достигается в точке пересечения прямых .

*,* следовательно,

Значение (цена) игры

Тогда **(, )** – оптимальная стратегия первого игрока.

Найдем оптимальную стратегию второго игрока:

, тогда

**(1/7, 0, 0, 0, 0, 6/7)** – оптимальная стратегия второго игрока.

***Решение в виде задач линейного программирования***

Решим следующие задачи линейного программирования:

1.

Значение (цена) игры v=1/fmin

Оптимальная стратегия первого игрока v

2.

Значение (цена) игры v=1/fmin

Оптимальная стратегия второго игрока v

***Код программы:***

**from** cvxopt **import** matrix, solvers, sparse  
**import** numpy **as** np  
  
A = matrix([[1.0, -3.0, -6.0, 0.0, -4.0, -1.0],   
 [-5.0, -1.0, 2.0, -4.0, 1.0, 0.0]])  
b = matrix([-1.0, -1.0, -1.0, -1.0, -1.0, -1.0])  
c = matrix([1.0, 1.0])  
sol= solvers.lp(c, A, b, solver = **'glpk'**)  
print(sol)  
x = sol[**'x'**]  
print(x)  
*# [1.0, 0.4]*f\_min = sol[**'primal objective'**]  
print(f\_min)  
*# 1.3999999981965823*v = 1/f\_min   
print(v)  
*# Game price:0.7142857160301338*p\_0 = v\*x  
print(p\_0)  
*# Стратегия для первого игрока: [7.14e-01, 2.86e-01]  
  
##### Для второго игрока*G = A.trans()  
q = matrix(np.eye(6))  
h = matrix(np.zeros((6,1)))  
h = sparse([[c, h]])  
h = matrix(h, (8, 1), **'d'**)  
G = sparse([G, q])  
sol= solvers.lp(b, (-1)\*G, h, solver = **'glpk'**)  
y = sol[**'x'**]  
print(y)  
*# [ 2.00e-01, 0.00e+00, 0.00e+00, 0.00e+00, 0.00e+00, 1.20e+00]*f\_min2 = sol[**'primal objective'**]  
print(f\_min2)  
*# -1.4*game\_price = -1/f\_min2  
print(game\_price)  
*#Game price: 0.7142857142857143*q2\_0 = v \* y  
print(q2\_0)  
*# Стратегия для первого игрока: [ 1.43e-01, 0.00e+00, 0.00e+00, 0.00e+00, 0.00e+00, 8.57e-01]*

В итоге, цена игры получилась равна ***0.7142857142857143.***

Пара равновесных стратегий: