In [2]:

```
import pandas as pd
from scipy.optimize import linprog
import numpy as np
import math
from pulp import LpMaximize, LpProblem, LpStatus, lpSum, LpVariable, LpMinir

data2 = pd.DataFrame([[14, 20, 32, 8], [15, 11, 19, 37], [33, 9, 16, 34]],
                     columns=['B1', 'B2', 'B3', 'B4'], index=['A1', 'A2', 'A


def calculate(dataframe):
    MinJ = list(range(len(dataframe.index)))
    MaxI = list(range(len(dataframe.columns)))
    for i in range(len(MinJ)):
        MinJ[i] = dataframe.loc['A' + str(i + 1)].min()
    for i in range(len(MaxI)):
        MaxI[i] = dataframe['B' + str(i + 1)].max()
    dataframe['MinJ'] = MinJ
    MaxI.append(0)
    dataframe.loc['MaxI'] = MaxI
    print(dataframe)
    g = -1000
    for i in dataframe['MinJ']:
        if i > g and i != 0:
            g = i
    print("Нижняя граница игры = " + str(g))
    k = 1000
    for i in dataframe.loc['MaxI']:
        if k > i != 0:
            k = i
    print("Верхняя граница игры = " + str(k))
    model = LpProblem(name="small-problem", sense=LpMaximize)
    x = {i: LpVariable(name=f"x{i}", lowBound=0) for i in range(1, 5)}
    model += (
        dataframe.loc['A1'][0] * x[1] + dataframe.loc['A1'][1] * x[2] + data
        dataframe.loc['A1'][3] * x[4] <= 1,
        "A1")
    model += (
        dataframe.loc['A2'][0] * x[1] + dataframe.loc['A2'][1] * x[2] + data
        dataframe.loc['A2'][3] * x[4] <= 1,
        "A2")
    model += (
        dataframe.loc['A3'][0] * x[1] + dataframe.loc['A3'][1] * x[2] + data
        dataframe.loc['A3'][3] * x[4] <= 1,
        "A3")
    model += x[1] + x[2] + x[3] + x[4]
    status = model.solve()

    for var in model.variables():
        if var.value() != 0:
            print(f"{var.name}: {var.value()/model.objective.value()}")
        else:
            print(f"{var.name}: {var.value()}")

    return model.objective.value()
```

```
data3 = pd.DataFrame([[-4, -5, -1, 6], [-1, 0, -3, 5], [-3, 1, -5, 5], [-8,
                      columns=['B1', 'B2', 'B3', 'B4'], index=['A1', 'A2', 'A
k = 1000
for i in range(4):
    g = data3.loc['A' + str(i + 1)].sum()
    if k > g:
        k = g
        f = i + 1
data3 += 10
x1 = 1 / calculate(data2)
x2 = 1 / calculate(data3.drop(index='A' + str(f))) - 10
print(x1)
print(x2)

a = np.array([[0, 1 / 2, 5 / 6], [1, 3 / 4, 1 / 2]])
p = np.array([3 / 8, 5 / 8])
Q = np.array([1 / 4, 0, 3 / 4])
Q1 = np.array([1 / 4, 0, 0])
res1 = p.dot(a).dot(Q.transpose())
print(res1)
res2 = p.dot(a).dot(Q1.transpose())
print(res2)

data3 = pd.DataFrame([[-4, -5, -1, 6], [-1, 0, -3, 5], [-3, 1, -5, 5], [-8,
                      columns=['B1', 'B2', 'B3', 'B4'], index=['A1', 'A2', 'A
k = 1000
for i in range(4):
    g = data3['B' + str(i + 1)].sum()
    if k > g:
        k = g
        f = i + 1
data3.drop(columns='B3')

model = LpProblem(name="small-problem", sense=LpMinimize)
x = {i: LpVariable(name=f"x{i}", lowBound=0) for i in range(1, 5)}
model += (
    data3.loc['A1'][0] * x[1] + data3.loc['A2'][0] * x[2] + data3.loc['A3']
    data3.loc['A4'][0] * x[4] >= -1,
    "A1")
model += (
    data3.loc['A1'][1] * x[1] + data3.loc['A2'][1] * x[2] + data3.loc['A3']
    data3.loc['A4'][1] * x[4] >= -1,
    "A2")
model += (
    data3.loc['A1'][2] * x[1] + data3.loc['A2'][2] * x[2] + data3.loc['A3']
    data3.loc['A4'][2] * x[4] >= -1,
    "A3")
model += -x[1] - x[2] - x[3] - x[4]
status = model.solve()

for var in model.variables():
    if var.value() != 0:
        print(f"{var.name}: {var.value() / math.fabs(model.objective.value()
    else:
        print(f"{var.name}: {var.value()}")
```

```
print(1 / model.objective.value())
```

```
      B1  B2  B3  B4  MinJ
A1    14  20  32   8     8
A2    15  11  19  37    11
A3    33   9  16  34     9
MaxI  33  20  32  37     0
Нижняя граница игры = 11
Верхняя граница игры = 20
x1: 0.11038961152713 78
x2: 0.6818181848807556
x3: 0.0
x4: 0.20779220359210657
      B1  B2  B3  B4  MinJ
A1     6   5   9  16     5
A2     9  10   7  15     7
A3     7  11   5  15     5
MaxI   9  11   9  16     0
Нижняя граница игры = 7
Верхняя граница игры = 9
x1: 0.39999999844
x2: 0.0
x3: 0.60000000156
x4: 0.0
16.84415600900982
-2.19999998752
0.625
0.15625
x1: 0.39999999999999997
x2: 0.6
x3: 0.0
x4: 0.0
-2.200000022
```

In [ ]: