**Министерство науки и высшего образования Российской Федерации**

**Московский государственный технический университет имени Н. Э. Баумана**

Факультет: Информатика и системы управления

Кафедра: Информационная безопасность (ИУ8)

**Рубежный контроль №3**

**ПО ДИСЦИПЛИНЕ «ТЕОРИЯ ИГР И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ»**

«**Кооперативные игры. Вектор Шепли**»

**Вариант 1**

**Студент**: Анаян М. С., ИУ8-104

**Преподаватель:** Коннова Н. С.

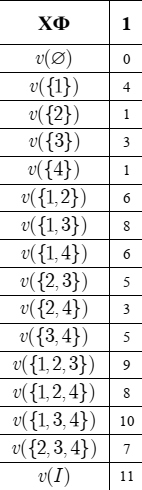
# Цель и задачи выполнения. Исходные данные

**Цель работы –** Изучить постановку кооперативной игры и найти оптимальное распределение выигрыша (дележ) между игроками путем вычисления компонент вектора Шепли.

## Постановка задачи

1. Проверить кооперативную игру на супераддитивность и выпуклость.
2. Составить программу вычисления компонент вектора Шепли и, в зависимости от варианта, рассчитать его.
3. Проверить условия индивидуальной и групповой рационализации.

## Исходные данные



# Выполнение лабораторной работы

Заданная по условию варианта кооперативная игра является супераддитивной, но не является выпуклой (поскольку для наборов ).

Вычисленный вектор Шепли:

Для него выполняется условие группового рационализма:

а также выполняется условие индивидуального рационализма:

Исходный код на Python 3.\* для решения задачи представлен в Приложении.

# Выводы

В ходе лабораторной работы получены следующие результаты:

* Заданная условием игра – супераддитивная, но не выпуклая;
* Составлена программа вычисления компонент вектора Шепли, полученный результат: ;
* Выполнена проверка полученного вектора на условия индивидуальной и групповой рационализации.

# Приложение

Исходный код на языке Python 3.\* также представлен в репозитории по ссылке: <https://github.com/hms2010/GameTheory/tree/master/src/rk3>

|  |
| --- |
| from game\_params import \*  from math import factorial  from fractions import Fraction  def is\_supperadditive(chars):      for i in chars:          for j in chars:              if (i & j) == 0:                  if chars[i | j] < (chars[i] + chars[j]):                      return False      return True  def is\_convex(chars):      for i in chars:          for j in chars:              if (chars[i | j] + chars[i & j]) < (chars[i] + chars[j]):                  return False      return True  def sets\_power(s):      res = 0      while s:          res += s & 0x01          s >>= 1      return res  def shapley\_value(chars):      vec = []      for i in range(N):          res = Fraction(0)          for coal in chars:              if (coal & 2\*\*i) == 0:                  continue              coal\_power = sets\_power(coal)              res += (factorial(coal\_power - 1) \*                          factorial(N - coal\_power) \*                              (chars[coal] - chars[coal & ~(2\*\*i)]))          res /= factorial(N)          vec.append(res)      return vec  def check\_group\_rationalization(vec):      return sum(vec) == chars[2\*\*N - 1]  def check\_individual\_rationalization(vec):      for i in range(N):          if vec[i] < chars[2\*\*i]:              return False      return True  def main():      if is\_supperadditive(chars):          print("Game is supperadditive")      else:          print("Game isn't supperadditive")      if is\_convex(chars):          print("Game is convex")      else:          print("Game isn't convex")      shapley\_vector = shapley\_value(chars)      print("Shapley vector: [{:s}]".format(", ".join(["{:.2f}".format(float(i)) for i in shapley\_vector])))      if check\_group\_rationalization(shapley\_vector):          print("Group rationalization: OK")      else:          print("Group rationalization: FAILED")      if check\_individual\_rationalization(shapley\_vector):          print("Individual rationalization: OK")      else:          print("Individual rationalization: FAILED")  if \_\_name\_\_ == "\_\_main\_\_":      main() |