ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ

Кафедра
«Криптология и кибербезопасность»

Лабораторная работа №1

«Решение задачи о рюкзаке при помощи генетического алгоритма»

Исполнитель: Александров Павел Сергеевич,

Группа: Б19-505

Москва — 2023

Содержание

1. Реализация генератора задач о рюкзачных векторах

Для начала определимся с начальными параметрами. Так как номер по списку – 2, то возьмем вариант по остатку, то есть 3. Это значит, что размерность рюкзачного вектора равна 24, а максимальное число в нем : 2**20.

Теперь начнем реализацию, полный код представлен в Приложении №1 – Основной код. Для начала нам необходимо сделать генерацию рандомного вектора с заданными условиями, за это отвечает функция vector_gen():

```
#генератор рюкзачного вектора
def vector_gen():
    vector = []
    for _ in range(0, LENGTH):
        vector.append(random.randint(1, A_MAX))
    vector.sort()
    return(vector)
```

Так же стоит отметить, что вектора сразу сортируются для дальнейшего удобства. По заданию необходимо сгенерировать 50 рюкзачных векторов и занести это все в таблицу. Так как вручную все заносить это очень долго и нерационально, то будем пользоваться инструментами в питоне. Все нужные в лабораторной таблицы будем генерировать таким образом. Код генерации всех таблиц представлен в Приложении №2 – код генерации таблиц.

Итак, сгенерируем 50 различных рюкзачных векторов и занесем это все в таблицу, она представлена в приложении №4 – Таблица векторов

После этого для каждого вектора сгенерируем по 10 задач о рюкзаке. Для этого нам необходимо определить для каждой задачи целевой вес. Для этого будем брать определенное число элементов (которое будет выбираться случайно), после того, как сложим их веса, то и получим в итоге целевой вес и нужную нам задачу.

Долю элементов будем считать в пределах от 0,1 до 0,5. Исходя из того, что число элементов должно быть целым, перейдем от доли к количеству элементов и будем выбирать от 3 до 12 случайных элементов из вектора. Код реализации генерации задачи о рюкзаке представлен ниже:

```python
# генератор целевых весов рюкзака
def backpack_task(vector):
    part = random.randint(3, 12)
    weight = sum(random.sample(vector, k=part))
    return(weight, part)
```

Итак, для каждого вектора сгенерируем по 10 таких задач и занесем все в таблицу, которая представлена в приложении №5 – Таблица задач.

2. Решение задач о рюкзаке полным перебором

В итоге у нас получилось 500 различных задач о рюкзаке. Следующим шагом будет решить данные задачи методом полного перебора. Для этого будем использовать библиотеку itertools. Метод полного перебора будет заключаться в следующем. Сначала мы пройдемся циклом по всем возможным количествам элементов (от 1 до 24). На каждой итерации цикла с помощью itertools сгенерируем все возможные комбинации из текущего числа элементов и пройдемся по каждой комбинации. Для каждой комбинации посчитаем вес и сравним его с искомым, если он совпал, то это и есть нужное решение. Описанным выше процессом занимается функция full_enumeration (), реализация представлена ниже:

```
# осуществление полного перебора с таймированием
def full_enumeration(vector, weight):
    count = 0
    beg = time.perf_counter()
    for l in range(0,24):
        for tmp in combinations(vector, l):
            if sum(tmp) == weight:
                count += 1
                if count == 1:
                    clock1 = time.perf_counter() - beg
                    clock_all = clock1
                else:
                    clock_all = time.perf_counter() - beg
    return(clock1, clock_all, count)
```

В данном алгоритме замерим несколько метрик. Во-первых, измерим время полного перебора, это примерно 4,2 секунды. Именно от этого времени и будем отталкиваться при реализации критерием останова генетического алгоритма.

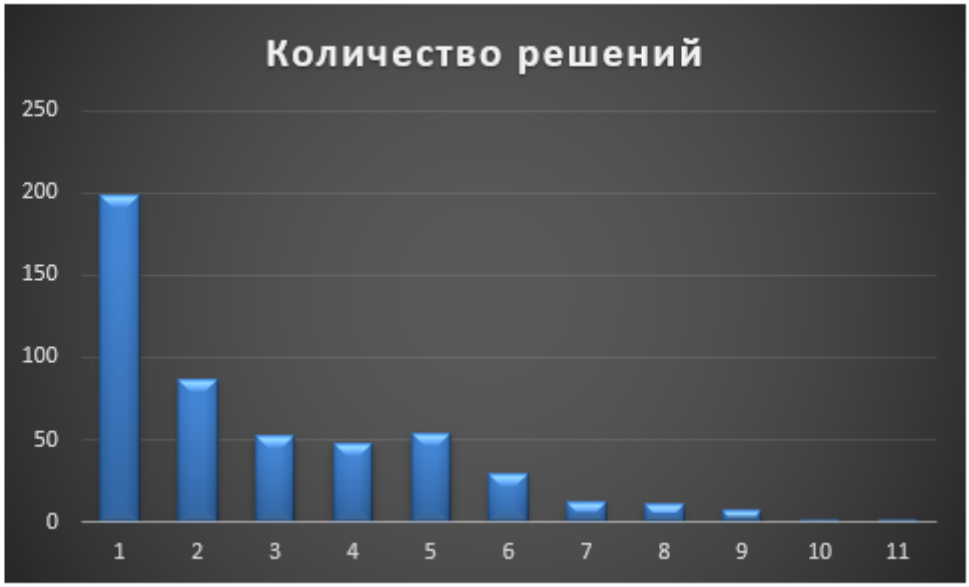
Так же, замерим время поиска первого решения (clock1) и время поиска всех решений (clock_all). Хоть мы и генерировали решения только по 1 набору элементов, может случиться так, что его можно составить и другими вариантами, тогда решений будет больше чем 1 и clock1 будет не равно clock_all. Так же замерим количество найденных решений. Полная информация по 500 наборам представлена в Приложении №6 – Таблица решений полным перебором.

Чуть проанализируем полученную информацию, посчитаем среднее время поиска первого решения, среднее время поиска всех решений, а также среднее количество решений, результаты представлены ниже:

Время полного перебора	**Время поиска 1 решения**	**Время поиска всех решений**	**Среднее число решений**
4,2 сек	0,37 сек	1,01 сек	2,84 шт

Как мы видим, первое решение находится довольно быстро, примерно в 10 раз быстрее, чем полный перебор, это связано с двумя факторами, во-первых, все решения находятся в первой половине поиска, так как количество элементов выбиралось от 3 до 12. А то же, в среднем присутствует около 3 решений для каждой задачи. Полное разбиение по числу решений представлено на графике ниже:

Число решений	Кол-во задач
1	199
2	86
3	52
4	48
5	54
6	29
7	12
8	11
9	7
10	1
11	1



Так, можно видеть, что большинство задач имеют по 1 решению, однако, есть и такие, где их целых 11.

3. Создание генетического алгоритма

Теперь наконец приступим к реализации самого генетического алгоритма. Его реализация представлена в Приложении №3 – Код генетического алгоритма.

3.1 Генерация начальной популяции

Итак, первый шаг – это генерация начальной популяции, для этого будем использовать функции rand_generation() – которая генерирует одну хромосому и pop_generation() – которая генерирует словарь, где ключом является хромосома, а значением – значение фитнесс-функции:

```
# генерация одной рандомной хромосомы
def rand_generation(chrome_len):
    chromosome = ''
    for _ in range (chrome_len+1):
        chromosome += str(random.randint(0,1))
    return(chromosome)

#генерация популяции в виде словаря с фитнесс-функцией
def pop_generation(population, vector, weight):
    dict_pop = {}
    for _ in range (population):
        chromosome = rand_generation(chrome_len)
        dict_pop[chromosome] = fitness(chromosome, vector, weight)
    return(dict_pop)
```

То есть происходит генерация хромосомы, которая имеет бинарный вид и длину равную размерности рюкзачного вектора. Каждый бит хромосомы отвечает за то, складывается ли этот элемент в рюкзак или нет. То есть, если начальные биты хромосомы имеют вид : «011…», то это значит, что первый элемент не входит в рюкзак и его вес мы не учитываем, а второй и третий элементы – входят.

Фитнесс функция же считается, как модуль отличия от нужного нам веса, то есть мы поэлементно перемножаем массив рюкзачного вектора и хромосомы, берем сумму результирующего массива и вычитаем эту сумму из нужного нам веса. После чего берем модуль, чтобы избежать проблемы с отрицательными числами.

3.2 Реализация операторов

Следующим шагом будет реализация операторов генетического алгоритма. Начнем с оператора репродукции, он отбирает родителей, которые будут учувствовать в скрещивании, за него отвечает функция reproduction():

```python
# репродукция на основе метода ранжирования
def reproduction(dict_pop):
    #сортируем словарь
    sorted_dict = {}
    sorted_keys = sorted(dict_pop.keys())
    for w in sorted_keys:
        sorted_dict[w] = dict_pop[w]
    l = len(sorted_dict)
    prob = [round((i+1)*200/(l*(l+1)), 2) for i in range(l)]
    chromosome = random.choices(list(sorted_dict), weights=prob)[0]
    chromosome1 = random.choices(list(sorted_dict), weights=prob)[0]
    while chromosome1 == chromosome:
        chromosome1 = random.choices(list(sorted_dict), weights=prob)[0]
    return(crossingover(chromosome, chromosome1))
```

Стоит отметить, что было принято решение выбирать родителей с помощью метода ранжирования, потому что в данной задаче нужен именно минимум функции (стремление к 0), поэтому его будет гораздо проще и эффективнее реализовать. К тому же, если функция очень близка к 0 это не означает, что рюкзачный вектор собран почти правильно. Поэтому отсортируем все наши вектора по возрастанию фитнесс функции и распределим вероятности пропорциональны местам в этой таблице. После чего запустим случайный генератор и выберем 2 хромосомы, если они совпали, то перезапустим до того момента, пока они не будут разными.

Следующий оператор – кроссинговер, то есть само скрещивание хромосом:

```python
# кроссинговер (chromosome : str)
def crossingover(chromosome, chromosome1):
    k = random.randint(0, chrome_len)
    new = chromosome[0:k] + chromosome1[k:]
    new1 = chromosome1[0:k] + chromosome[k:]
    return(new, new1)
```

Здесь ничего необычного, просто выбираем случайную точку и меняем части хромосом, получая новые. После чего они сохраняется с шансов равным вероятности кроссинговера.

И, наконец, последний оператор – мутация:

```python
# мутация (chromosome : str)
def mutation(chromosome):
    k = random.randint(0, 13)
    new = list(chromosome)
    new[k] = str(abs(int(new[k])-1))
    return(''.join(new))
```

Опять, здесь ничего необычного, просто замена одного случайного бита с определенным шансом.

3.3 Общее описание алгоритма

Итак, изначально генерируется начальная популяция, для каждой особи считается фитнесс-функция и заносится в словарь. После чего оператором репродукции выбираются особи, которые поступают на оператор кроссинговера. Так же, все полученное потомство подвергается оператору мутации с каким-то шансом и добавляется в исходную популяцию. После чего популяция сокращается до исходного размера, удаляя самые неудачные особи. Такой цикл повторяется пока не сработают критерии останова.

Этих критериев по условию задачи 3:

- Время, превышающее в 2 раза время полного перебора
- Неизменяемость фитнесс-функции в течении 2ух поколений
- Равенство 0 фитнесс функции

Все эти критерии реализованы в функции check_out(), однако, более подробно их рассмотрим в 4 разделе данной работы

Так же отдельно следует рассмотреть возможные изменяемые параметры алгоритма, такие как количество особей в начальной популяции, вероятность кроссинговера и мутации. По-хорошему, данные параметра отдельно рассматриваются и выбираются оптимальные. Однако, это довольно долгий процесс и требует отдельных исследований. Поэтому примем их следующими:

- Число особей в популяции – 100
- Вероятность кроссинговера – 50 %
- Вероятность мутации – 10 %

4. Решение с помощью генетического алгоритма

Итак, теперь запустим все сгенерированные 500 задач, чтобы они решились с помощью генетического алгоритма. К сожалению, при запуске с такими параметрами, как в условии, решений почти не удалось найти. Это связано с тем, что данные условия слишком жесткие, особенно неизменяемость фитнесс-функции. Ведь, если минимальная функция не меняется, это совсем не значит, что генетический алгоритм скатился в локальны максимум и уже оттуда не выберется. Так что, было принято решение протестировать этот алгоритм с различными параметрами. Изменяя при этом число особей в популяции и критерии останова.

Остановимся на наиболее удачном примере для анализа. При числе особей в популяции – 100, а остановке при 100 поколений без изменений. Результаты представлены в Таблице №7 – Результаты генетического алгоритма. В результате нашлось 9 задач для которых было найдены решения. Причем все остальные задачи останавливались из-за того, что не менялась фитнес-функция.

Количество решений в других экспериментах представлены ниже:

Модель	**популяция: 100 крит. ост.: 5**	**популяция: 50 крит. ост.: 10**	**популяция: 100 крит. ост.: 10**	**популяция: 100 крит. Ост.: 100**
Число решений	2	2	2	9

А полный список решений представлен на следующем рисунке:

Номер задачи	мя работы алгори	Номер последнего поколения	Доля предметов	Число решений
84	1,7681354	96	4	1
101	2,3849786	128	3	1
137	1,7239902	89	4	1
170	0,5175065	24	6	2
224	3,6008771	184	8	7
230	3,4186222	170	7	1
289	0,839188	41	5	1
292	1,1529074	60	3	1
338	0,4137451	22	4	1
Среднее	**1,758**	**90,444**	**4,889**	**1,778**

5. Анализ результатов

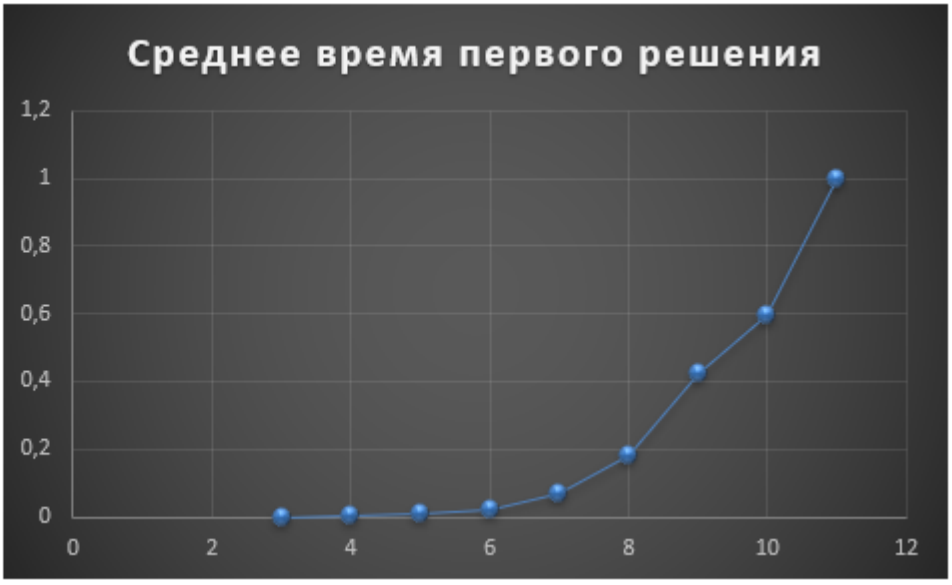
Итак, теперь проанализируем полученные результаты, заполнив таблицу:

	Среднее значение	**Дисперсия**	**Среднее квадратичное откл.**
n	24		
amax	2 ** 20		
Время нахождения одного решения полным перебором	0,37	0,27	0,52
Время нахождения всех решений полным перебором	1,01	1,42	1,19
Время нахождения точного решения генетическим алгоритмом	1,76	1,39	1,18
Доля задач, точно решённых генетическим алгоритмом	1,8 %		
Количество хромосом в поколении	100		

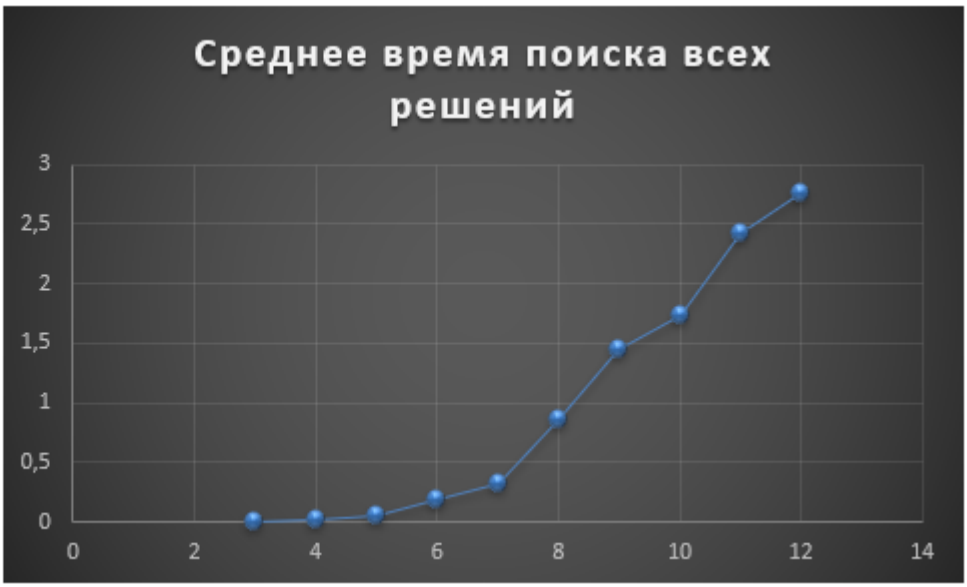
Как мы можем видеть из таблицы решение с помощью генетического алгоритма нашлось только менее чем в двух процентах задач. К тому, среднее время такого поиска было гораздо больше полного перебора.

Далее построим графики, сначала проанализируем полный перебор и зависимости времени поиска решений от доли предметов в рюкзачном векторе:

Доля предметов	Среднее время 1
3	0,000245
4	0,001338
5	0,006559
6	0,023485
7	0,070866
8	0,178104
9	0,426171
10	0,594425
11	0,994256
12	1,249291



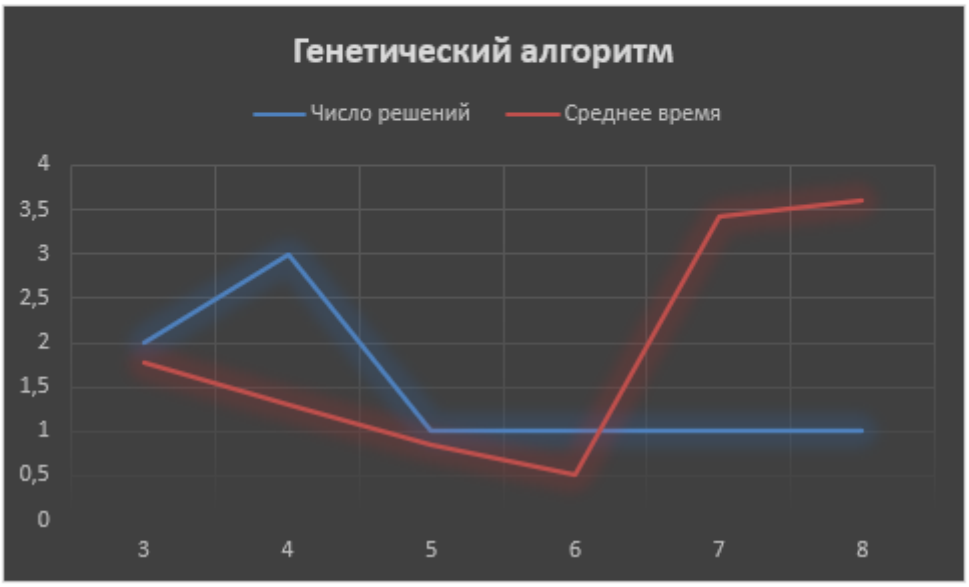
Доля предметов	Среднее время
3	0,002542
4	0,014506
5	0,054055
6	0,194466
7	0,315203
8	0,851993
9	1,446073
10	1,725643
11	2,423672
12	2,765096



Из этих графиков можно видеть, что чем больше доля предметов в рюкзачном векторе, тем больше искать решение полным перебором. Это и логично, ведь полный перебор мы реализовывали именно по количеству решений.

Теперь построим такие же графики для генетического алгоритма, к сожалению, выборка будет небольшая, потому что нашлось всего 9 решений:

Доля предметов	Число решений	Среднее время
3	2	1,768943
4	3	1,3019569
5	1	0,839188
6	1	0,5175065
7	1	3,4186222
8	1	3,6008771



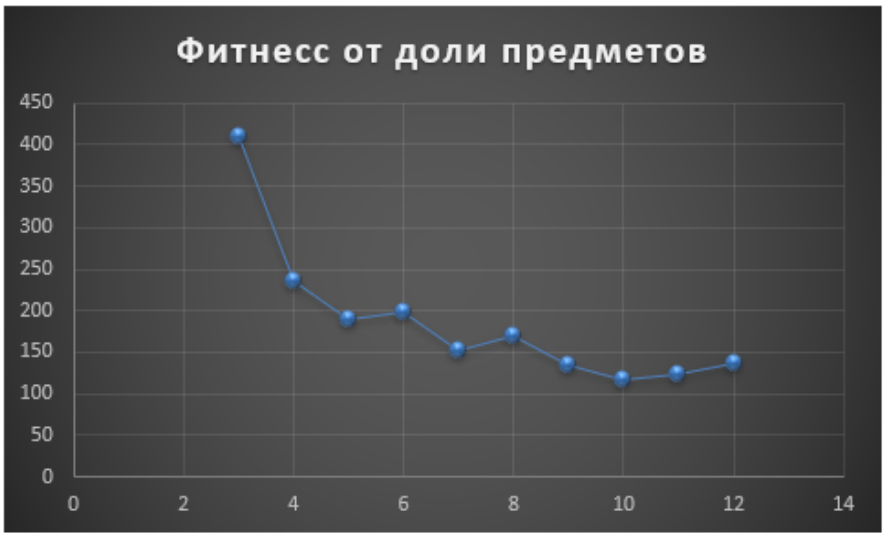
Опять же, здесь сложно сделать каки-либо выводы, ибо выборка большая. К тому же, генетический алгоритм очень сильно полагается на случайную составляющую, поэтому, появляются такие события, что даже с большей долей предметов алгоритм работает быстрее.

После этого посмотрим еще несколько характеристик:

- Среднее число поколений при поиске решений (найденных) : 90,44
- Среднее число поколений при поиске решений (всех) : 149,32
- Среднее число решений у решенных задач : 1,78
- Средняя фитнес-функция – 180,56
- Среднее время поиска решений (всех) – 2, 99 сек

Еще одним интересным на мой взгляд параметром является зависимость минимальной фитнесс-функции от количества решений и от доли элементов в векторе:

Доля предметов	Фитнесс
3	409,5945946
4	234,7307692
5	189,255814
6	198,94
7	151,7090909
8	169,5735294
9	135,0227273
10	116,2619048
11	123,3636364
12	136,1111111



Кол-во решений	Фитнесс
1	246,8140704
2	165,255814
3	158,7692308
4	118,7916667
5	90,07407407
6	108,3793103
7	199,75
8	140,5454545
9	134,8571429
10	4
11	96

Здесь уже можно видеть такую тенденции, что наоборот, чем больше доля предметов в векторе, тем ближе генетический алгоритм подбирается к решению. ТО же самое и в отношении количества решений, чем их больше, тем больше вероятность у алгоритма попасть, куда надо

Заключение

В ходе данной лабораторной работы был разработан генератор задач о рюкзачных векторах и с его помощью были сгенерированы 500 различных задач. После чего эти задачи решались двумя методами : методом полного перебора и методом генетического алгоритма, которые так же были реализованы самостоятельно. После чего был произведен анализ результатов.

В ходе анализа генетический алгоритм показал ужаснейшие результаты по сравнению с методом полного перебора. За время полного перебора ему удалось найти решения менее, чем в 2ух процентах задач, причем и сделать это медленнее. Так что на таких размерностях генетический алгоритм мало подходит для решения задач о рюкзаке. Тем более их можно без особых усилий решить полным перебором.

Однако, когда размерности становятся гораздо больше, то есть надежда, что генетический алгоритм начнем себя показывать уже гораздо лучше. Во-первых, при увеличении размерности очень сильно увеличивается время полного перебора и при определенной точке найти решение таким методом за адекватное время уже не будет представляться возможным. А генетический алгоритм особо не зависит от этой размерности, поэтому есть шанс, что он будет решать некоторые задачи, с которым алгоритм полного перебора не справляется. Так же это можно косвенно подтвердить тем, что при увеличении доли элементов в рюкзачном векторе генетический алгоритм все более приближался к требуемой фитнесс-функции.

ПРИЛОЖЕНИЯ

Приложение №1 – Основной Код

```
import random
from itertools import *
import time

# импорт собственных файлов
from genetic import *
from tables import  *

A_MAX = 2**20
LENGTH = 24
random.seed(100)

#генератор рюкзачного вектора
def vector_gen():
    vector = []
    for _ in range(0, LENGTH):
        vector.append(random.randint(1, A_MAX))
    vector.sort()
    return(vector)

# генератор n рюкзаков
def backpacks_gen(n):
    backpacks = []
    for _ in range (0, n):
        backpacks.append(vector_gen())
    return backpacks

# генератор целевых весов рюкзака
def backpack_task(vector):
    part = random.randint(3, 12)
    weight = sum(random.sample(vector, k=part))
    return(weight, part)

# осуществление полного перебора с таймированием
def full_enumeration(vector, weight):
    count = 0
    beg = time.perf_counter()
    for l in range(0,24):
        for tmp in combinations(vector, l):
            if sum(tmp) == weight:
                count += 1
                if count == 1:
```

```
                clock1 = time.perf_counter() - beg
                clock_all = clock1
            else:
                clock_all = time.perf_counter() - beg
    return(clock1, clock_all, count)

if __name__ == "__main__":
    # генерация векторов
    vectors = []
    for _ in range(50):
        vectors.append(vector_gen())
    vectors_table_gen(vectors)

    # генерация задач о рюкзаке
    backpacks = []
    for i in range(50):
        vector = vectors[i]
        for _ in range(10):
            backpacks.append(backpack_task(vector))
    tasks_table_gen(backpacks)

    '''
    # генерация решений полным перебором
    full_en_sol = []
    for i in range (50):
        vector = vectors[i]
        for j in range(10):
            weight = backpacks[10*i + j][0]
            full_en_sol.append(full_enumeration(vector, weight))
    full_en_table_sol(full_en_sol)
    '''

    # генерация решений ген-алгоритмом
    genetic_sol = []
    for i in range (50):
        vector = vectors[i]
        for j in range(10):
            weight = backpacks[10*i + j][0]
            genetic_sol.append(genetic(vector, weight))
    gen_table_sol(genetic_sol)
```

Приложение №2 – Код генерации таблиц

```
import pandas as pd

A_MAX = 2**20
LENGTH = 24

# генерация таблицы векторов
def vectors_table_gen(vectors):
    df = pd.DataFrame ({
    'Номер вектора' : [i+1 for i in range (50)],
    'Вектор' : vectors,
    'Amax' : [str(A_MAX) for _ in range (50)]
    })
    df.to_excel('Вектора.xlsx', index=False)

# генерация таблицы рюкзачных задач
def tasks_table_gen(backpacks):
    df = pd.DataFrame ({
    'Номер задачи' : [i+1 for i in range (500)],
    'Номер вектора' : [(i//10) + 1 for i in range (500)],
    'Целевой вес' : [backpacks[i][0] for i in range (500)],
    'Доля предметов' : [backpacks[i][1] for i in range (500)]
    })
    df.to_excel('Задачи.xlsx', index=False)

# генерация таблицы решений задач полным перебором
def full_en_table_sol(full_en_sol):
    df = pd.DataFrame ({
    'Номер задачи' : [i+1 for i in range (500)],
    'Время нахождения первого решения' : [full_en_sol[i][0] for i in range (500)],
    'Время нахождения всех решений' : [full_en_sol[i][1] for i in range (500)],
    'Число решений' : [full_en_sol[i][2] for i in range (500)]
    })
    for i in range (500):
        pass
    df.to_excel('Решения_перебор.xlsx', index=False)

# генерация таблицы решений задач генетическим алгоритмом
def gen_table_sol(genetic_sol):
    df = pd.DataFrame ({
    'Номер задачи' : [i+1 for i in range (500)],
    'Время работы алгоритма' : [genetic_sol[i][0] for i in range (500)],
    'Достигнутый минимум фитнесс-функции' : [genetic_sol[i][1] for i in range (500)],
    'Причина остановки алгоритма' : [" for i in range (500)],
```

```
'Номер последнего поколения' : [genetic_sol[i][2] for i in range (500)]
})
for i in range (500):
    if df['Достигнутый минимум фитнесс-функции'][i] == 0:
        df['Причина остановки алгоритма'][i] = 'Фитнесс функция'
    elif df['Время работы алгоритма'][i] > 10:
        df['Причина остановки алгоритма'][i] = 'Превышено время'
    else:
        df['Причина остановки алгоритма'][i] = 'Неизменяемость поколений'
df.to_excel('Решения_ген_алг_02.xlsx', index=False)
```

Приложение №3 – Код генетического алгоритма

```
import random
import time

chrome_len = 24      # длина хромосомы
population = 100     # размер популяции
cross_prob = 0.5     # вероятность кроссинговера
mut_prob = 0.1       # вероятность мутации

# генерация одной рандомной хромосомы
def rand_generation(chrome_len):
    chromosome = ''
    for _ in range (chrome_len+1):
        chromosome += str(random.randint(0,1))
    return(chromosome)

#генерация популяции в виде словаря с фитнесс-функцией
def pop_generation(population, vector, weight):
    dict_pop = {}
    for _ in range (population):
        chromosome = rand_generation(chrome_len)
        dict_pop[chromosome] = fitness(chromosome, vector, weight)
    return(dict_pop)

#подсчет фитнесс функции
def fitness(chromosome, vector, weight):
    s = 0
    for i in range(len(vector)):
        s += int(chromosome[i])*vector[i]
    fitness = abs(weight - s)
    return(fitness)

# кроссинговер (chromosome : str)
def crossingover(chromosome, chromosome1):
    k = random.randint(0, chrome_len)
    new = chromosome[0:k] + chromosome1[k:]
    new1 = chromosome1[0:k] + chromosome[k:]
    return(new, new1)

# мутация (chromosome : str)
def mutation(chromosome):
    k = random.randint(0, 13)
    new = list(chromosome)
    new[k] = str(abs(int(new[k])-1))
```

```
    return(''.join(new))

# репродукция на основе метода ранжирования
def reproduction(dict_pop):
    #сортируем словарь
    sorted_dict = {}
    sorted_keys = sorted(dict_pop.keys())
    for w in sorted_keys:
        sorted_dict[w] = dict_pop[w]
    l = len(sorted_dict)
    prob = [round((i+1)*200/(l*(l+1)), 2) for i in range(l)]
    chromosome = random.choices(list(sorted_dict), weights=prob)[0]
    chromosome1 = random.choices(list(sorted_dict), weights=prob)[0]
    while chromosome1 == chromosome:
        chromosome1 = random.choices(list(sorted_dict), weights=prob)[0]
    return(crossingover(chromosome, chromosome1))

# функция остановки алгоритма
def check_out(count, min_fitness, gen_time):
    flag = 0
    if (count == 100) or (min_fitness == 0) or (gen_time > 10):
        flag = 1
    return (flag)

# функция вероятностного кроссинговера
def prob_cross(child_pop, chromosome, vector, weight):
    if random.random() < cross_prob:
        child_pop[chromosome] = fitness(chromosome, vector, weight)

# функция поиска решения с помощью ген-алгоритма
def genetic(vector, weight):
    dict_pop = pop_generation(population, vector, weight)       # основная популяция
    child_pop = {}                                  # популяция детей
    num_generation = 0                                  # подсчет числа популяция
    count, min_fitness = 0, 0                                       # подсчет итераций, где не
улучшается фитнесс-функция
    beg = time.perf_counter()
    while True:
        num_generation += 1
        if min_fitness == (min(dict_pop.values())):
            count +=1
        else:
            count = 0
        min_fitness = (min(dict_pop.values()))
        for _ in range (0, population):
```

```
            x, y = reproduction(dict_pop)[0], reproduction(dict_pop)[1] # получение 2ух
дочерних хромосом
            prob_cross(child_pop, x, vector, weight)
            prob_cross(child_pop, y, vector, weight)
        for chromosome in child_pop:
            if random.random() < mut_prob:
                new = mutation(chromosome)
                dict_pop[new] = fitness(new, vector, weight)
        dict_pop = dict((sorted((dict_pop | child_pop).items(), key=lambda item:
item[1]))[0:population])
        child_pop.clear()
        gen_time = time.perf_counter() - beg
        if check_out(count, min_fitness, gen_time):
            break
    return(gen_time, min_fitness, num_generation)
```

Приложение №4 – Таблица векторов

Номер вектора	**Вектор**	**Amax**
1	[100428, 168095, 230624, 254410, 295345, 305491, 366473, 374260, 395513, 427546, 428838, 481727, 552425, 648872, 703282, 727722, 733452, 776077, 824252, 859189, 909146, 954052, 956188, 963702]	1048576
2	[11839, 54809, 110012, 260243, 265532, 309550, 336429, 340872, 350255, 378846, 403997, 442151, 497498, 518188, 578351, 607871, 710338, 719886, 786640, 812984, 829458, 845293, 939928, 967016]	1048576
3	[106369, 114475, 122095, 129905, 212489, 253763, 308677, 345411, 453952, 459584, 502891, 529600, 533118, 577033, 628936, 649821, 685262, 749154, 761264, 842502, 911126, 917357, 1014107, 1034907]	1048576
4	[26272, 81523, 92155, 98778, 103550, 197846, 218168, 239139, 249741, 317712, 330237, 337908, 404391, 415492, 438611, 521760, 699906, 817565, 840139, 933724, 955651, 993791, 997232, 999736]	1048576
5	[77294, 81897, 140091, 162349, 189138, 257971, 293558, 304781, 407273, 437636, 444021, 526844, 527642, 546252, 564107, 720719, 840508, 843101, 865539, 876115, 894073, 944780, 952565, 994314]	1048576
6	[5948, 20428, 48335, 54729, 70673, 79161, 99300, 167767, 182110, 199106, 399464, 460682, 477418, 486493, 568903, 569183, 617649, 624910, 638513, 679561, 739478, 819027, 844429, 931828]	1048576
7	[6753, 28629, 47894, 153454, 175613, 232879, 285345, 301257, 333326, 337610, 365929, 386444, 409756, 445808, 567897, 630816, 670641, 739174, 798092, 801433, 831187, 917410, 936700, 959472]	1048576

8	[41916, 97858, 98868, 145664, 183381, 185623, 200215, 222764, 233377, 263684, 294583, 307628, 308266, 323617, 356941, 467846, 615969, 737042, 825285, 872898, 901780, 925963, 961322, 1007942]	104857 6
9	[29771, 73458, 109285, 186495, 218391, 236965, 243278, 262149, 426851, 471502, 475766, 480001, 590409, 677600, 730432, 734278, 751047, 795707, 984422, 989864, 1001312, 1002606, 1015840, 1032668]	104857 6
10	[22953, 61568, 83011, 99647, 185112, 217564, 223945, 238716, 251874, 328445, 385452, 402646, 467132, 487952, 536851, 629264, 639533, 642576, 673157, 682642, 719859, 769907, 917756, 971025]	104857 6
11	[15340, 33414, 88438, 127382, 171756, 174346, 183677, 232289, 319754, 321794, 340318, 402576, 427216, 429039, 561433, 659455, 660805, 735010, 888756, 891902, 925912, 930420, 990517, 1042650]	104857 6
12	[57721, 110038, 194713, 251023, 325297, 355487, 369179, 378272, 407680, 429878, 456166, 505081, 531434, 544876, 557757, 558106, 616407, 700299, 739067, 800079, 821902, 839529, 872359, 914118]	104857 6
13	[45071, 81925, 87073, 103591, 155742, 264551, 329966, 385084, 448518, 500289, 541555, 555227, 605199, 680619, 707369, 745589, 762874, 776489, 833963, 846765, 960152, 978066, 990657, 1046296]	104857 6
14	[39725, 95299, 152419, 161529, 169341, 204713, 370405, 443366, 459620, 515170, 517773, 572783, 586961, 702506, 712826, 719406, 721744, 838273, 852804, 856935, 998589, 1011930, 1034472, 1046684]	104857 6
15	[9574, 83007, 89669, 170919, 180984, 190346, 206853, 331906, 382422, 437725, 489519, 565046, 577697, 582663, 603700, 647815, 652696, 785218, 846022, 888118, 910925, 1004831, 1028914, 1039797]	104857 6
16	[3305, 19648, 83178, 168576, 178904, 263330, 326573, 372221, 414834, 519104, 602837, 604162, 695686, 710604, 772192, 829382, 851516, 852521, 888221, 910047, 939311, 943455, 945057, 1025932]	104857 6

17	[19044, 52431, 67537, 232271, 243446, 293059, 295177, 299905, 362752, 430124, 503802, 509448, 600828, 622425, 648641, 806367, 814834, 816054, 816697, 841053, 859806, 991469, 993172, 994247]	104857 6
18	[42577, 42818, 122220, 155034, 156055, 237780, 341373, 464256, 468506, 487002, 560295, 583846, 649698, 729101, 734455, 769039, 798681, 801457, 811649, 812583, 855081, 885350, 897768, 1001827]	104857 6
19	[862, 23740, 71187, 83152, 104143, 160591, 194300, 257523, 289483, 338300, 439914, 533561, 566307, 609951, 612034, 753241, 781575, 820982, 836709, 863625, 919102, 944581, 1007159, 1036822]	104857 6
20	[53792, 214414, 307250, 328758, 423040, 489228, 490488, 507267, 515534, 556803, 601567, 619164, 651316, 656668, 661314, 672648, 696176, 707519, 722926, 730124, 831529, 926741, 981764, 987311]	104857 6
21	[58408, 64238, 296432, 297089, 361463, 481757, 509442, 580491, 588038, 594073, 595715, 596216, 628539, 688283, 710095, 739587, 799863, 808858, 845083, 848717, 855449, 893314, 1010748, 1016341]	104857 6
22	[129332, 150243, 171691, 208867, 249470, 249816, 262202, 326875, 359799, 383178, 420131, 612080, 640347, 702513, 732800, 786409, 830413, 892935, 927128, 953345, 961467, 979059, 1012996, 1035646]	104857 6
23	[7952, 33779, 46730, 51928, 73302, 95177, 105195, 127929, 145883, 175953, 192451, 205588, 214218, 220540, 311224, 380866, 611909, 665469, 712585, 728459, 878745, 916754, 951947, 957230]	104857 6
24	[634, 21541, 81579, 108690, 193873, 213484, 338754, 388213, 389652, 443834, 459850, 488180, 526634, 543978, 586009, 651715, 709018, 815070, 818892, 832533, 871669, 955782, 1002286, 1048187]	104857 6
25	[11752, 75132, 109773, 123613, 131177, 149834, 206213, 210418, 243350, 296545, 359629, 401941, 466437, 604474, 633520, 660965, 694227, 699193, 713214, 721680, 789770, 839515, 841616, 975179]	104857 6

26	[35265, 51396, 71473, 76543, 99865, 126548, 198098, 228674, 241005, 346661, 353250, 396120, 448524, 466871, 561192, 630056, 690600, 756640, 770490, 943273, 958014, 965110, 1024539, 1030854]	104857 6
27	[63234, 82339, 99988, 138572, 150927, 216958, 330712, 397539, 399919, 441260, 526513, 534097, 543467, 691535, 777896, 809998, 837358, 891050, 901073, 931370, 969918, 1019061, 1035465, 1039438]	104857 6
28	[4247, 32017, 39305, 77321, 133875, 167287, 167505, 197916, 202542, 223786, 315584, 366365, 413578, 514308, 583267, 591448, 659815, 739566, 766284, 792818, 932217, 940915, 1010122, 1020472]	104857 6
29	[29858, 42596, 80044, 169112, 278131, 377204, 387867, 492472, 520218, 550484, 562787, 611753, 617227, 621792, 660424, 661951, 676942, 737201, 764570, 818770, 840144, 890519, 916863, 1021951]	104857 6
30	[20506, 50936, 80958, 131401, 150522, 199732, 211171, 217872, 266469, 297693, 350212, 371061, 372254, 434385, 496407, 592765, 625726, 799247, 823868, 830552, 862750, 1000083, 1022435, 1044759]	104857 6
31	[136471, 259313, 341603, 348296, 357632, 395779, 398432, 407774, 429170, 503140, 631993, 670068, 680679, 690338, 725050, 759489, 803444, 849099, 870136, 884063, 942579, 951843, 992813, 1017673]	104857 6
32	[13199, 21892, 107664, 165077, 188118, 218481, 268203, 272129, 309844, 377989, 462673, 497278, 544225, 547719, 562965, 593524, 636815, 669526, 882367, 922253, 930800, 932598, 1017953, 1042247]	104857 6
33	[24342, 107171, 158314, 187621, 227664, 234555, 240851, 278622, 299412, 371742, 374777, 462280, 498758, 568829, 572503, 597871, 612845, 745236, 791225, 824299, 860983, 961795, 974787, 1038512]	104857 6
34	[9546, 41074, 112016, 161781, 229031, 261303, 355883, 386637, 388725, 401501, 405013, 431600, 456012, 507058, 511711, 523346, 525809, 574713, 638593, 650754, 686113, 751155, 957100, 983770]	104857 6

35	[73085, 83649, 97460, 105363, 107963, 114747, 180349, 200975, 223643, 347176, 356834, 393897, 448102, 450153, 474893, 515599, 617073, 858635, 918547, 936790, 987933, 994554, 994577, 1006385]	104857 6
36	[106388, 181549, 190448, 204596, 292875, 310077, 339530, 346297, 407027, 418111, 428785, 474504, 476070, 487143, 643742, 672597, 680720, 684711, 729630, 810047, 838657, 863877, 888603, 929660]	104857 6
37	[19910, 49886, 78961, 92983, 110742, 146411, 214651, 242752, 380193, 418264, 433841, 438464, 442837, 535829, 565460, 576749, 579961, 610295, 636606, 716236, 850470, 914619, 959708, 1011574]	104857 6
38	[28417, 67318, 94919, 241852, 281136, 298805, 301380, 377309, 574096, 594791, 616485, 624433, 630782, 693514, 755979, 777486, 798848, 881454, 907982, 921017, 922830, 957000, 967765, 977927]	104857 6
39	[67215, 73788, 183729, 227276, 241086, 245112, 266638, 276636, 291837, 398305, 412947, 496985, 527622, 538641, 594135, 609478, 645738, 663993, 751339, 802363, 848053, 904768, 912802, 1005881]	104857 6
40	[63756, 75404, 104101, 123079, 124676, 141437, 189610, 195938, 216273, 295473, 419456, 442004, 474097, 506410, 554705, 575253, 580971, 586893, 600840, 624338, 632958, 688597, 833441, 1008813]	104857 6
41	[43336, 82308, 116416, 119788, 190217, 227454, 252419, 381032, 382035, 464452, 516172, 526781, 551535, 594129, 648882, 673493, 729394, 777356, 893282, 898435, 903348, 1019491, 1029975, 1031510]	104857 6
42	[93511, 102527, 106478, 110129, 118138, 132617, 180733, 253811, 359709, 464935, 472020, 490291, 511104, 520550, 559222, 559958, 565413, 666862, 763260, 807067, 809678, 819639, 1006321, 1039539]	104857 6
43	[95278, 109738, 307603, 319926, 361710, 396350, 522366, 538926, 587575, 625692, 630579, 662198, 668038, 695480, 725192, 732883, 742014, 768871, 898732, 928606, 941579, 981354, 1009839, 1044722]	104857 6

44	[86847, 91112, 94943, 95116, 160250, 172715, 238766, 313937, 322968, 332283, 352531, 357827, 436961, 579711, 590165, 654318, 705650, 726943, 795821, 855415, 917174, 922414, 922436, 945679]	104857 6
45	[20410, 24014, 82436, 87806, 105761, 227912, 313239, 348616, 371959, 399454, 404117, 428243, 460033, 556589, 558462, 609797, 610464, 622179, 669660, 691881, 861450, 864835, 915789, 976896]	104857 6
46	[26779, 46800, 118384, 118486, 155427, 169511, 218571, 236382, 236858, 289481, 294129, 344531, 390048, 407336, 434411, 450514, 465759, 572867, 612785, 659963, 770729, 912942, 912996, 1020730]	104857 6
47	[16533, 95514, 132139, 197834, 208826, 218811, 223961, 303460, 355733, 417313, 524944, 547010, 622889, 626201, 651077, 654291, 711809, 750974, 793347, 817449, 848793, 885551, 972816, 1045704]	104857 6
48	[17115, 49322, 85144, 111466, 159051, 256168, 285001, 321124, 374485, 390658, 479364, 484211, 508484, 564449, 698938, 775649, 784927, 845066, 874778, 883318, 902796, 921801, 948914, 964760]	104857 6
49	[26234, 72060, 90894, 132618, 136397, 169313, 221350, 271674, 285703, 294797, 460875, 472182, 582230, 584417, 616383, 630483, 631831, 749397, 758789, 880481, 914833, 919506, 996515, 1046160]	104857 6
50	[14187, 19889, 209139, 218226, 220131, 250437, 395425, 408813, 453440, 455926, 514506, 553971, 638185, 661150, 706972, 733068, 744334, 778127, 810699, 850579, 872236, 891290, 958053, 1036871]	104857 6

Приложение №5 – Таблица задач

Номер задачи	**Номер вектора**	**Целевой вес**	**Доля предметов**
1	1	4244546	7
2	1	3204179	6
3	1	4434753	11
4	1	4715738	10
5	1	4502094	8
6	1	3800632	6
7	1	5294227	10
8	1	924779	3
9	1	2560049	7
10	1	6259585	12
11	2	2907613	5
12	2	1304099	3
13	2	5779413	11
14	2	912796	4
15	2	2589915	4
16	2	6056103	12
17	2	2788433	6
18	2	1841423	3
19	2	3552627	8
20	2	4450250	7
21	3	5520511	9

22	3	5477563	12
23	3	1854690	5
24	3	1510350	4
25	3	3942321	9
26	3	2297376	4
27	3	4424993	8
28	3	2631422	5
29	3	4948722	10
30	3	2687936	5
31	4	5005232	11
32	4	2059802	5
33	4	5078322	12
34	4	5181262	11
35	4	1988153	3
36	4	1539378	3
37	4	3690011	6
38	4	4960688	8
39	4	2763340	6
40	4	2321371	7
41	5	4099892	7
42	5	1188250	4
43	5	2562228	4
44	5	6122687	11
45	5	3303336	7

46	5	4182184	9
47	5	5094388	10
48	5	4089728	8
49	5	3134611	5
50	5	5397503	11
51	6	2801648	8
52	6	3581887	6
53	6	964157	3
54	6	4936358	12
55	6	2144346	7
56	6	2515426	5
57	6	1808726	4
58	6	4691217	9
59	6	3887332	9
60	6	4946791	11
61	7	6432039	12
62	7	5133531	12
63	7	4364319	8
64	7	4333704	11
65	7	5119937	11
66	7	4650624	9
67	7	5698592	10
68	7	6185634	11
69	7	3989880	10

70	7	4178164	8
71	8	1218350	4
72	8	1231430	3
73	8	3227640	6
74	8	6308309	11
75	8	6367310	12
76	8	6019473	12
77	8	6521939	12
78	8	2017465	5
79	8	2858308	6
80	8	2666877	8
81	9	4578258	8
82	9	7480959	12
83	9	6357351	12
84	9	1456076	4
85	9	4813519	8
86	9	4809209	10
87	9	7482572	12
88	9	3974438	9
89	9	5093428	10
90	9	2716846	4
91	10	2963072	6
92	10	1446527	3
93	10	3020017	5

94	10	3777634	8
95	10	2488916	6
96	10	3295123	8
97	10	1902033	7
98	10	2408161	6
99	10	5330804	12
100	10	2772800	8
101	11	529779	3
102	11	3850363	7
103	11	4583849	8
104	11	1089561	3
105	11	5199396	10
106	11	3704946	5
107	11	2078538	3
108	11	5419111	8
109	11	4573034	11
110	11	4458926	10
111	12	5958783	12
112	12	4137558	7
113	12	6164642	12
114	12	5845529	11
115	12	2950178	5
116	12	4728918	9
117	12	3025428	7

118	12	2780814	5
119	12	5431939	10
120	12	2753246	5
121	13	4707900	8
122	13	6566951	9
123	13	1632957	3
124	13	2073038	4
125	13	1929741	4
126	13	6147492	8
127	13	5220872	8
128	13	2625487	4
129	13	7232057	12
130	13	5192414	9
131	14	3026287	5
132	14	7265899	11
133	14	3152317	7
134	14	4080995	8
135	14	6423239	10
136	14	5847733	11
137	14	941339	4
138	14	2353207	6
139	14	3666577	6
140	14	2832541	6
141	15	4535846	9

142	15	6346768	10
143	15	2376740	3
144	15	3580064	7
145	15	4033123	8
146	15	5612670	10
147	15	3876430	8
148	15	5089117	8
149	15	1160705	3
150	15	3535485	5
151	16	4972350	8
152	16	6445411	12
153	16	6143418	10
154	16	2076258	3
155	16	2082043	5
156	16	4620035	7
157	16	4113702	8
158	16	2993738	8
159	16	3894458	8
160	16	3199683	7
161	17	2287682	6
162	17	3061079	5
163	17	5493855	10
164	17	6312912	11
165	17	2210307	3

166	17	4128325	10
167	17	3084297	6
168	17	5084806	8
169	17	5420525	11
170	17	3091449	6
171	18	1658528	3
172	18	5389088	9
173	18	4792465	7
174	18	6063662	11
175	18	4338400	8
176	18	4371093	8
177	18	3398067	6
178	18	7185597	11
179	18	1872202	5
180	18	2772839	7
181	19	6695823	12
182	19	4646760	9
183	19	5243758	9
184	19	3418794	7
185	19	4833734	8
186	19	6225964	12
187	19	5626524	11
188	19	2562145	4
189	19	2360764	4

190	19	2859954	8
191	20	4825600	7
192	20	3166647	5
193	20	5038732	8
194	20	7643183	11
195	20	6870100	9
196	20	6794173	12
197	20	2035133	3
198	20	4294115	8
199	20	3712175	6
200	20	7277367	12
201	21	3392076	6
202	21	7415641	11
203	21	4575869	7
204	21	4510426	8
205	21	7264105	12
206	21	2378461	4
207	21	5657293	9
208	21	7072895	11
209	21	5295156	7
210	21	2575257	5
211	22	2722370	4
212	22	2257509	3
213	22	3288699	7

214	22	5199147	8
215	22	4797980	8
216	22	5552646	10
217	22	4243060	5
218	22	5224174	7
219	22	7295224	11
220	22	3258276	5
221	23	2316262	6
222	23	4526856	12
223	23	3333376	12
224	23	3407350	8
225	23	2210144	4
226	23	2184999	7
227	23	3893730	9
228	23	3715840	8
229	23	3497879	6
230	23	1160648	7
231	24	2094561	5
232	24	2009481	4
233	24	1672966	3
234	24	5521977	12
235	24	5654962	12
236	24	5947382	12
237	24	4005981	11

238	24	4694383	12
239	24	3008680	6
240	24	3248852	7
241	25	2360568	8
242	25	1710858	4
243	25	2152210	7
244	25	1369617	3
245	25	2346617	6
246	25	502558	3
247	25	4540076	8
248	25	6003588	12
249	25	4156316	8
250	25	4369096	9
251	26	1404440	6
252	26	3271143	6
253	26	1454058	3
254	26	3227478	6
255	26	2551226	5
256	26	3580897	6
257	26	2188187	4
258	26	5195633	8
259	26	3241735	7
260	26	3373247	10
261	27	3935057	8

262	27	4984525	9
263	27	3226652	8
264	27	1773942	6
265	27	3079023	5
266	27	5324398	7
267	27	6359638	12
268	27	2007681	5
269	27	1979986	3
270	27	4888621	12
271	28	1800804	7
272	28	3597114	9
273	28	1306318	3
274	28	5019511	11
275	28	4771365	9
276	28	3083889	6
277	28	1887291	4
278	28	5096136	10
279	28	1699546	5
280	28	2872794	8
281	29	4498157	8
282	29	5404897	10
283	29	6309520	12
284	29	5290560	11
285	29	1857925	3

286	29	5524462	9
287	29	3433984	7
288	29	4647605	10
289	29	1485651	5
290	29	2218468	4
291	30	5968420	11
292	30	480030	3
293	30	3979808	10
294	30	4308217	8
295	30	2539719	5
296	30	5772079	9
297	30	3881919	9
298	30	4789317	10
299	30	7729839	12
300	30	5507692	9
301	31	5057542	8
302	31	6182089	9
303	31	4566918	8
304	31	5484683	9
305	31	6621309	11
306	31	4398472	8
307	31	8378684	12
308	31	3542548	5
309	31	7805190	12

310	31	2997880	5
311	32	6075339	12
312	32	5387486	10
313	32	6355501	11
314	32	5715794	11
315	32	2666932	5
316	32	4093110	7
317	32	6311043	12
318	32	1122272	3
319	32	3440108	7
320	32	3218539	7
321	33	3311967	8
322	33	3261236	7
323	33	3216615	8
324	33	3802047	7
325	33	1972398	5
326	33	2222636	4
327	33	5246305	12
328	33	6053541	10
329	33	4866790	8
330	33	2497234	6
331	34	3878568	7
332	34	2112025	6
333	34	2187217	5

334	34	5059542	11
335	34	4259645	9
336	34	4501447	11
337	34	5766825	12
338	34	518519	4
339	34	2657494	5
340	34	2054564	4
341	35	4923398	10
342	35	1278804	4
343	35	1039744	3
344	35	3091067	7
345	35	4106365	12
346	35	1731565	4
347	35	6257428	11
348	35	4843491	11
349	35	4906799	12
350	35	3509689	9
351	36	1336128	3
352	36	3905229	9
353	36	6169881	11
354	36	4840218	10
355	36	5135731	11
356	36	4732510	9
357	36	6017826	10

358	36	3550930	7
359	36	2060973	4
360	36	2756453	6
361	37	2604557	6
362	37	1413158	4
363	37	2552047	7
364	37	3501148	8
365	37	2569391	8
366	37	1548909	4
367	37	2259541	6
368	37	3450704	6
369	37	2480383	8
370	37	2421447	6
371	38	3125600	6
372	38	3806007	6
373	38	4668610	8
374	38	3428174	7
375	38	5051938	8
376	38	3133407	6
377	38	5075721	6
378	38	3013142	5
379	38	4427614	7
380	38	6824389	11
381	39	4011779	7

382	39	5479829	12
383	39	6044681	11
384	39	2172793	5
385	39	2527377	7
386	39	1362076	4
387	39	1892096	3
388	39	2892472	6
389	39	4078135	7
390	39	3160378	7
391	40	6050088	11
392	40	4003326	10
393	40	678246	4
394	40	4219021	12
395	40	4636683	10
396	40	3699521	11
397	40	4738982	12
398	40	3155325	7
399	40	2104975	6
400	40	2823454	7
401	41	1978475	5
402	41	6494213	12
403	41	2157399	3
404	41	5398869	8
405	41	7557833	12

406	41	2490559	4
407	41	5525817	9
408	41	1984800	4
409	41	5223747	11
410	41	7016919	12
411	42	5409457	12
412	42	3230308	6
413	42	5047601	9
414	42	1706043	5
415	42	4262291	8
416	42	2140825	6
417	42	1328791	3
418	42	4616648	10
419	42	4932393	10
420	42	4415299	6
421	43	4542564	6
422	43	6860332	10
423	43	5575696	9
424	43	7285171	10
425	43	3031165	4
426	43	6831407	12
427	43	4803691	8
428	43	2328999	4
429	43	5995709	10

430	43	2375917	3
431	44	2653424	10
432	44	3935146	9
433	44	4517205	9
434	44	2683139	7
435	44	2009416	4
436	44	1884643	4
437	44	2904763	9
438	44	848442	3
439	44	2139800	4
440	44	4364157	9
441	45	3631284	11
442	45	4075941	8
443	45	4607159	9
444	45	1478668	4
445	45	2623828	4
446	45	4218773	8
447	45	2302329	4
448	45	7143098	12
449	45	2614078	7
450	45	2594767	5
451	46	4530474	11
452	46	2056993	5
453	46	3501899	11

454	46	1859248	7
455	46	1616960	7
456	46	3466549	9
457	46	2719713	5
458	46	4393959	12
459	46	4783308	10
460	46	3781400	8
461	47	3462920	4
462	47	5596556	11
463	47	4087948	7
464	47	5306734	11
465	47	1638148	4
466	47	3981666	7
467	47	3094422	6
468	47	5738172	9
469	47	2231635	3
470	47	4947207	11
471	48	4672016	11
472	48	5751051	10
473	48	1322741	4
474	48	6220964	11
475	48	3698381	6
476	48	5258088	10
477	48	5097895	8

478	48	5913873	12
479	48	3145217	4
480	48	6265498	12
481	49	2750705	6
482	49	4991483	11
483	49	4490795	10
484	49	4625827	8
485	49	2120042	3
486	49	6968875	11
487	49	2760967	4
488	49	5422831	11
489	49	5366836	11
490	49	3048304	9
491	50	6930099	11
492	50	2532681	4
493	50	2118416	5
494	50	4379763	8
495	50	2070743	4
496	50	1612449	4
497	50	5522468	10
498	50	3674066	7
499	50	4982278	9
500	50	6086830	9

Приложение №6 – Таблица решений полным перебором

Номер задачи	Время нахождения первого решения	Время нахождения всех решений	Число решений
1	0,0981	0,2397	3
2	0,0206	0,0297	2
3	1,3280	1,3280	1
4	0,6789	0,6789	1
5	0,2286	1,0849	2
6	0,0281	0,1233	2
7	0,5280	0,8232	2
8	0,0001	0,0001	1
9	0,0363	0,0363	1
10	0,5392	2,0171	8
11	0,0067	0,0067	1
12	0,0001	0,0001	1
13	1,2528	3,1952	4
14	0,0004	0,0004	1
15	0,0022	0,0613	2
16	1,3516	3,1136	4
17	0,0283	0,2003	2
18	0,0003	0,0003	1
19	0,2538	0,5745	2
20	0,0909	0,8572	4
21	0,4679	0,9501	5

22	0,9514	1,9029	4
23	0,0061	0,0061	1
24	0,0014	0,0014	1
25	0,3931	0,3931	1
26	0,0011	0,0011	1
27	0,2017	0,5962	3
28	0,0062	0,0257	2
29	0,4582	1,3275	3
30	0,0076	0,0076	1
31	0,9298	2,7762	5
32	0,0049	0,0049	1
33	0,5708	2,8423	5
34	1,2138	3,1638	5
35	0,0002	0,0510	2
36	0,0007	0,0007	1
37	0,0351	0,0351	1
38	0,2000	1,7430	5
39	0,0199	0,0199	1
40	0,0448	0,0448	1
41	0,0499	0,0499	1
42	0,0005	0,0005	1
43	0,0015	0,0015	1
44	0,5850	2,7812	4
45	0,0392	0,0392	1

46	0,4325	0,4325	1
47	0,7364	1,2521	2
48	0,1415	0,3026	3
49	0,0061	0,0460	2
50	0,2841	1,3967	3
51	0,1309	0,2047	3
52	0,0294	0,6808	4
53	0,0001	0,0001	1
54	1,4080	3,5983	6
55	0,0612	0,1571	2
56	0,0098	0,6713	2
57	0,0026	0,0026	1
58	0,5222	2,7471	5
59	0,5408	1,7891	5
60	1,6189	3,7973	4
61	1,9033	3,3663	9
62	1,4070	2,9975	5
63	0,2384	0,2949	2
64	0,6662	1,2409	4
65	1,1570	1,8663	3
66	0,5566	2,0226	5
67	0,9360	2,5242	2
68	1,6544	3,9784	6
69	0,7696	0,8110	2

70	0,2334	2,1373	5
71	0,0014	0,0014	1
72	0,0002	0,0002	1
73	0,0301	0,9830	2
74	1,3915	4,7067	8
75	2,4461	4,0624	2
76	0,9064	4,3162	11
77	0,9598	4,3108	7
78	0,0033	0,0033	1
79	0,0340	0,3449	2
80	0,1119	0,1119	1
81	0,1153	0,1153	1
82	2,2914	2,3445	2
83	1,6099	2,7023	4
84	0,0008	0,0008	1
85	0,1731	1,2650	5
86	0,0983	0,6970	2
87	2,1695	3,8015	5
88	0,4930	0,7910	2
89	0,1630	0,8430	3
90	0,0011	0,0011	1
91	0,0121	0,0121	1
92	0,0005	0,0005	1
93	0,0094	0,0094	1

94	0,0888	2,2598	3
95	0,0293	0,0293	1
96	0,1315	0,1315	1
97	0,0477	0,0477	1
98	0,0265	0,0265	1
99	0,9821	2,4073	5
100	0,1334	0,1334	1
101	0,0002	0,0002	1
102	0,1116	0,1116	1
103	0,2317	2,0093	8
104	0,0001	0,0001	1
105	0,6773	2,9093	6
106	0,0085	0,7718	6
107	0,0004	0,0004	1
108	0,2291	2,6064	5
109	0,4685	2,2940	6
110	0,1209	0,8000	3
111	0,9417	2,4218	5
112	0,1160	0,8485	6
113	1,0845	2,1210	6
114	0,4648	2,3871	6
115	0,0073	0,0130	2
116	0,4713	1,4017	2
117	0,0382	0,0382	1

118	0,0132	0,0132	1
119	0,8917	2,2666	4
120	0,0044	0,0044	1
121	0,1858	0,1858	1
122	0,5814	3,4507	3
123	0,0002	0,0002	1
124	0,0015	0,0015	1
125	0,0008	0,0008	1
126	0,2902	2,4180	6
127	0,2753	1,8316	7
128	0,0020	0,0020	1
129	1,2466	2,6293	3
130	0,3610	1,1061	2
131	0,0068	0,0068	1
132	1,5881	2,8680	6
133	0,0411	0,0411	1
134	0,1990	0,3379	2
135	0,5272	2,4050	4
136	0,4213	1,0800	5
137	0,0008	0,0008	1
138	0,0238	0,0238	1
139	0,0199	0,0250	2
140	0,0236	0,0236	1
141	0,5488	1,1400	2

142	0,7200	3,4246	5
143	0,0003	0,0346	2
144	0,0601	0,0601	1
145	0,1533	0,2145	3
146	0,4282	2,5648	8
147	0,0656	0,4821	3
148	0,2803	1,1571	4
149	0,0003	0,0003	1
150	0,0089	0,1539	2
151	0,2677	0,4691	3
152	0,9380	2,0885	6
153	0,6658	2,6300	5
154	0,0003	0,0003	1
155	0,0027	0,0027	1
156	0,1177	0,4690	2
157	0,1720	0,2410	3
158	0,1689	0,1689	1
159	0,1131	0,1640	2
160	0,0390	0,0390	1
161	0,0166	0,0166	1
162	0,0080	0,0080	1
163	0,4422	2,5430	4
164	1,4122	3,1464	6
165	0,0003	0,0003	1

166	0,3392	0,7985	2
167	0,0189	0,0189	1
168	0,2313	1,2918	5
169	0,7415	1,8595	4
170	0,0150	0,1029	2
171	0,0002	0,0002	1
172	0,1142	1,5003	4
173	0,0707	0,7312	4
174	0,9606	2,7136	5
175	0,0661	0,2051	3
176	0,1949	0,3925	3
177	0,0263	0,1242	2
178	1,6117	2,8219	5
179	0,0023	0,0023	1
180	0,0857	0,0857	1
181	2,6732	2,6732	1
182	0,2870	1,0553	3
183	0,6515	1,8492	4
184	0,0944	0,7968	4
185	0,3614	1,4256	3
186	1,4524	3,9789	4
187	0,9288	4,4477	6
188	0,0019	0,0019	1
189	0,0021	0,0021	1

190	0,2832	0,8287	2
191	0,0886	0,7836	5
192	0,0055	0,0055	1
193	0,2061	0,2357	2
194	1,5536	3,0778	8
195	0,6755	2,6745	7
196	0,9859	2,2230	4
197	0,0003	0,0003	1
198	0,0969	0,1379	2
199	0,0277	0,0277	1
200	2,0530	2,3725	3
201	0,0232	0,0232	1
202	1,7140	3,2416	5
203	0,0746	0,4382	3
204	0,1632	0,1632	1
205	1,0743	2,7776	5
206	0,0005	0,0005	1
207	0,1741	0,5924	4
208	1,4490	1,6241	5
209	0,0916	0,4084	2
210	0,0059	0,0059	1
211	0,0018	0,0018	1
212	0,0002	0,0002	1
213	0,0646	0,0646	1

214	0,2520	1,3101	4
215	0,1795	0,2889	2
216	0,3671	0,8521	5
217	0,0110	0,0110	1
218	0,0872	0,7690	2
219	1,5923	2,8686	5
220	0,0047	0,0047	1
221	0,0331	0,0331	1
222	1,9379	3,2007	4
223	0,3074	2,3384	4
224	0,1709	2,5697	7
225	0,0009	0,6172	2
226	0,0691	1,0519	2
227	0,5723	1,7446	3
228	0,2326	3,6505	8
229	0,0309	2,0766	5
230	0,0496	0,0496	1
231	0,0100	0,0100	1
232	0,0009	0,0009	1
233	0,0001	0,0001	1
234	0,1375	1,8375	4
235	0,9259	2,4468	6
236	1,5403	2,5185	5
237	0,4010	1,0816	2

238	0,2590	1,8770	5
239	0,0214	0,0214	1
240	0,0775	0,0775	1
241	0,1121	0,1121	1
242	0,0013	0,0013	1
243	0,0386	0,0386	1
244	0,0001	0,0001	1
245	0,0180	0,0180	1
246	0,0001	0,0001	1
247	0,2857	2,7157	4
248	2,2923	3,5545	5
249	0,1926	2,3454	5
250	0,4161	2,0497	2
251	0,0102	0,0102	1
252	0,0205	1,5008	3
253	0,0002	0,0002	1
254	0,0203	0,0203	1
255	0,0027	0,0027	1
256	0,0240	1,4474	3
257	0,0015	0,0015	1
258	0,2315	2,5308	3
259	0,0895	0,1768	2
260	0,1428	0,5697	2
261	0,0528	0,2578	2

262	0,3871	1,2858	3
263	0,1985	0,1985	1
264	0,0227	0,0227	1
265	0,0064	0,0064	1
266	0,1008	0,6478	3
267	0,5238	2,1014	5
268	0,0026	0,0026	1
269	0,0002	0,0002	1
270	1,6441	1,6441	1
271	0,0491	0,0491	1
272	0,2957	0,2957	1
273	0,0003	0,0003	1
274	0,8254	2,3329	9
275	0,3553	1,8268	3
276	0,0205	0,0205	1
277	0,0011	0,0011	1
278	0,4848	2,2990	5
279	0,0105	0,0105	1
280	0,1434	0,1434	1
281	0,1823	0,4374	2
282	0,4601	1,7816	9
283	1,4620	1,9478	3
284	0,4726	1,0192	5
285	0,0002	0,0002	1

286	0,5206	1,9172	2
287	0,0730	0,0730	1
288	0,4010	0,6272	3
289	0,0027	0,0027	1
290	0,0015	0,0015	1
291	0,8148	3,9067	6
292	0,0001	0,0001	1
293	0,4844	1,4624	4
294	0,1294	1,7220	9
295	0,0044	0,0044	1
296	0,5845	3,6858	6
297	0,4889	0,5017	2
298	0,4203	2,1613	8
299	2,1768	4,0304	3
300	0,5146	3,5976	2
301	0,1343	0,3540	3
302	0,5656	0,9102	3
303	0,1737	0,1737	1
304	0,4819	1,2022	6
305	1,0517	1,8287	4
306	0,1150	0,1963	2
307	2,3740	3,1830	4
308	0,0081	0,0081	1
309	1,3721	2,8044	6

310	0,0075	0,0075	1
311	0,6486	2,7586	5
312	0,8169	1,4874	4
313	1,6171	2,8138	3
314	1,0902	2,3627	7
315	0,0046	0,0046	1
316	0,0992	2,0947	4
317	0,7938	3,8105	6
318	0,0005	0,0005	1
319	0,0521	0,7799	3
320	0,0376	0,0376	1
321	0,0245	0,1455	2
322	0,0927	0,1426	3
323	0,0300	0,1358	5
324	0,0585	0,0585	1
325	0,0095	0,0095	1
326	0,0013	0,0013	1
327	0,4244	1,7146	7
328	1,0058	3,0175	7
329	0,2304	0,7386	5
330	0,0072	0,0226	2
331	0,0954	1,0579	6
332	0,0143	0,0143	1
333	0,0088	0,0088	1

334	0,6994	3,0584	7
335	0,4603	2,0154	3
336	0,6856	1,3883	7
337	1,6694	2,7508	5
338	0,0004	0,0004	1
339	0,0038	0,0038	1
340	0,0019	0,0019	1
341	0,7531	2,4896	4
342	0,0014	0,0014	1
343	0,0001	0,0001	1
344	0,0390	1,0259	2
345	0,2134	2,6634	5
346	0,0020	0,0020	1
347	0,9078	4,1765	5
348	1,1068	2,0313	4
349	1,3884	2,3197	3
350	0,3334	0,3653	2
351	0,0002	0,0002	1
352	0,3051	0,3051	1
353	1,4002	2,6262	7
354	0,2579	0,7258	2
355	1,1087	1,5727	3
356	0,3648	0,3648	1
357	1,0067	1,9162	4

358	0,0976	0,0976	1
359	0,0006	0,0006	1
360	0,0261	0,0261	1
361	0,0277	0,0897	2
362	0,0007	0,0007	1
363	0,0287	0,1328	4
364	0,2225	0,7040	4
365	0,1884	0,2133	2
366	0,0004	0,0004	1
367	0,0218	0,1328	2
368	0,0354	0,4242	5
369	0,1055	0,1583	2
370	0,0188	0,0188	1
371	0,0133	0,0133	1
372	0,0423	0,0423	1
373	0,1848	0,1848	1
374	0,0570	0,0570	1
375	0,1522	1,0332	3
376	0,0193	0,0193	1
377	0,0365	0,0365	1
378	0,0078	0,0078	1
379	0,0904	0,4130	2
380	0,7527	2,4667	8
381	0,1030	0,1827	2

382	0,5422	2,7354	6
383	1,1864	2,4327	8
384	0,0053	0,0053	1
385	0,0502	0,0561	2
386	0,0010	0,0010	1
387	0,0004	0,0004	1
388	0,0127	0,0127	1
389	0,0249	0,2410	4
390	0,0391	0,0391	1
391	1,7454	4,2050	10
392	0,8718	2,4160	6
393	0,0005	0,0005	1
394	0,4759	1,8838	3
395	0,7744	3,1010	9
396	0,4448	1,3299	3
397	1,2420	1,9744	3
398	0,0804	0,5865	5
399	0,0124	0,0124	1
400	0,0823	0,2069	2
401	0,0044	0,0044	1
402	1,4783	2,0904	2
403	0,0002	0,0002	1
404	0,1905	0,7830	3
405	2,4080	3,7877	3

406	0,0010	0,0010	1
407	0,5007	1,3942	6
408	0,0016	0,0016	1
409	0,5070	1,3001	5
410	1,3105	3,8474	4
411	0,3674	3,0164	8
412	0,0238	0,4419	5
413	0,1942	1,6972	7
414	0,0082	0,0082	1
415	0,2563	0,2997	3
416	0,0100	0,0100	1
417	0,0001	0,0001	1
418	0,2642	0,9830	2
419	0,5524	2,1604	4
420	0,0478	0,2317	2
421	0,0250	0,0250	1
422	0,7041	2,1153	6
423	0,4497	1,0787	3
424	0,7369	3,0948	5
425	0,0025	0,0025	1
426	0,8831	1,6442	5
427	0,0449	0,2000	2
428	0,0014	0,0014	1
429	0,9762	1,0391	2

430	0,0003	0,0003	1
431	0,6050	0,6050	1
432	0,2903	0,2903	1
433	0,2471	1,0928	4
434	0,0599	0,0660	2
435	0,0019	0,0099	2
436	0,0016	0,0016	1
437	0,1242	0,3236	2
438	0,0002	0,0002	1
439	0,0016	0,0016	1
440	0,3669	1,1811	6
441	0,1031	1,3173	6
442	0,2003	0,8228	6
443	0,3538	1,7187	4
444	0,0004	0,0004	1
445	0,0021	0,0021	1
446	0,1436	1,4392	6
447	0,0020	0,0021	2
448	2,2436	3,6979	4
449	0,0391	0,0391	1
450	0,0082	0,0121	2
451	0,8812	2,3680	6
452	0,0038	0,4073	2
453	0,3777	2,0012	4

454	0,0508	0,0508	1
455	0,0474	0,0474	1
456	0,4290	2,6438	2
457	0,0090	0,0090	1
458	0,5706	2,6532	9
459	0,9781	2,3203	5
460	0,2149	1,3071	3
461	0,0024	0,0024	1
462	0,6739	2,2061	5
463	0,0527	0,0527	1
464	0,4432	1,6852	3
465	0,0008	0,0008	1
466	0,2463	0,2463	1
467	0,0246	0,0625	2
468	0,6148	2,6924	8
469	0,0003	0,0003	1
470	0,7827	1,8326	4
471	1,1518	1,3335	2
472	0,9849	0,9905	2
473	0,0012	0,0012	1
474	1,6320	1,8261	2
475	0,0310	0,0310	1
476	0,7464	1,4044	4
477	0,2544	0,8603	2

478	1,0859	2,9753	9
479	0,0025	0,0025	1
480	0,8172	2,4385	3
481	0,0143	0,0143	1
482	0,5277	1,4068	4
483	0,1783	1,8796	5
484	0,1311	1,5155	4
485	0,0004	0,0004	1
486	1,5870	4,5130	5
487	0,0020	0,0020	1
488	0,5799	2,2823	7
489	0,8065	2,0386	5
490	0,4528	0,4528	1
491	1,3318	1,8966	3
492	0,0011	0,0011	1
493	0,0043	0,0043	1
494	0,1332	0,4671	2
495	0,0015	0,0015	1
496	0,0001	0,0013	2
497	0,8114	1,6790	4
498	0,0767	0,3030	5
499	0,3996	0,7461	2
500	0,4114	2,3521	5

Приложение №7 – Результаты генетического алгоритма

Номер задачи	**Время работы алгоритма**	**Достигнутый минимум фитнесс-функции**	**Причина остановки алгоритма**	**Номер последнего поколения**
1	3,6445329	440	Неизменяемость поколений	167
2	2,6344171	43	Неизменяемость поколений	140
3	3,0414689	103	Неизменяемость поколений	139
4	4,1041945	34	Неизменяемость поколений	207
5	2,6981143	354	Неизменяемость поколений	137
6	2,2113087	201	Неизменяемость поколений	118
7	2,0731523	21	Неизменяемость поколений	112
8	2,0547722	1914	Неизменяемость поколений	111
9	4,8079639	186	Неизменяемость поколений	224
10	2,5568254	55	Неизменяемость поколений	121
11	4,4222223	16	Неизменяемость поколений	202
12	3,8506147	91	Неизменяемость поколений	198
13	4,1287628	77	Неизменяемость поколений	223
14	3,4407215	1605	Неизменяемость поколений	170
15	3,5492637	79	Неизменяемость поколений	192
16	3,1121809	287	Неизменяемость поколений	168
17	2,6076011	22	Неизменяемость поколений	141
18	2,8800642	20	Неизменяемость поколений	155
19	2,83678	65	Неизменяемость поколений	153

20	2,3297338	114	Неизменяемость поколений	126
21	2,6493405	86	Неизменяемость поколений	137
22	2,3337453	63	Неизменяемость поколений	124
23	3,6117811	117	Неизменяемость поколений	194
24	2,1686007	112	Неизменяемость поколений	112
25	2,5293314	192	Неизменяемость поколений	117
26	2,6473494	87	Неизменяемость поколений	121
27	5,6566247	24	Неизменяемость поколений	256
28	2,060964	101	Неизменяемость поколений	111
29	2,7634846	358	Неизменяемость поколений	150
30	2,1550027	105	Неизменяемость поколений	102
31	3,56725	31	Неизменяемость поколений	155
32	2,3144273	70	Неизменяемость поколений	118
33	3,5569706	46	Неизменяемость поколений	147
34	4,7997614	25	Неизменяемость поколений	215
35	5,3635952	104	Неизменяемость поколений	240
36	3,5908407	89	Неизменяемость поколений	180
37	3,2264821	24	Неизменяемость поколений	154
38	3,331824	63	Неизменяемость поколений	169
39	2,1286323	165	Неизменяемость поколений	112
40	2,6274447	454	Неизменяемость поколений	124
41	2,0365604	442	Неизменяемость поколений	108

42	3,4648281	111	Неизменяемость поколений	182
43	4,2234092	105	Неизменяемость поколений	208
44	2,5277804	16	Неизменяемость поколений	112
45	3,1333341	325	Неизменяемость поколений	146
46	3,398303	537	Неизменяемость поколений	142
47	2,3677256	95	Неизменяемость поколений	117
48	2,0442299	574	Неизменяемость поколений	110
49	2,1288332	124	Неизменяемость поколений	115
50	4,9505725	190	Неизменяемость поколений	242
51	4,1177416	78	Неизменяемость поколений	222
52	2,2830831	23	Неизменяемость поколений	113
53	2,2370574	134	Неизменяемость поколений	121
54	2,6239128	92	Неизменяемость поколений	129
55	3,8848149	90	Неизменяемость поколений	172
56	4,9484203	19	Неизменяемость поколений	252
57	2,1283334	140	Неизменяемость поколений	115
58	2,1088954	7	Неизменяемость поколений	112
59	2,1415536	15	Неизменяемость поколений	104
60	2,6945834	76	Неизменяемость поколений	111
61	2,7891828	54	Неизменяемость поколений	131
62	2,3477174	32	Неизменяемость поколений	127
63	1,9578527	186	Неизменяемость поколений	106

64	3,7143142	114	Неизменяемость поколений	166
65	2,1039001	38	Неизменяемость поколений	112
66	2,1390259	33	Неизменяемость поколений	110
67	5,4231677	61	Неизменяемость поколений	233
68	3,1285724	65	Неизменяемость поколений	127
69	2,1809507	4	Неизменяемость поколений	110
70	2,4465325	164	Неизменяемость поколений	112
71	4,1402157	490	Неизменяемость поколений	168
72	4,170467	254	Неизменяемость поколений	197
73	2,4122571	105	Неизменяемость поколений	129
74	2,4679581	44	Неизменяемость поколений	125
75	2,0457961	760	Неизменяемость поколений	110
76	3,0054302	96	Неизменяемость поколений	161
77	3,1255218	650	Неизменяемость поколений	132
78	3,3423147	17	Неизменяемость поколений	176
79	2,3337235	123	Неизменяемость поколений	126
80	2,9490288	215	Неизменяемость поколений	160
81	4,0685209	84	Неизменяемость поколений	204
82	5,142217	5	Неизменяемость поколений	232
83	2,8177323	158	Неизменяемость поколений	124
84	1,7681354	0	Фитнесс функция	96
85	2,1522182	94	Неизменяемость поколений	116
86	2,3248069	461	Неизменяемость поколений	117

87	2,5355035	219	Неизменяемость поколений	113
88	4,2043747	243	Неизменяемость поколений	170
89	2,3778966	15	Неизменяемость поколений	106
90	2,4587062	159	Неизменяемость поколений	128
91	3,9959528	61	Неизменяемость поколений	207
92	2,3206757	1105	Неизменяемость поколений	119
93	2,5431807	11	Неизменяемость поколений	114
94	2,3942468	57	Неизменяемость поколений	120
95	2,0673747	73	Неизменяемость поколений	106
96	2,6110049	136	Неизменяемость поколений	139
97	2,1723201	72	Неизменяемость поколений	114
98	2,0813873	598	Неизменяемость поколений	111
99	3,0826328	200	Неизменяемость поколений	146
100	3,1657366	172	Неизменяемость поколений	170
101	2,3849786	0	Фитнесс функция	128
102	2,2336011	43	Неизменяемость поколений	119
103	2,0803899	264	Неизменяемость поколений	112
104	2,5884054	665	Неизменяемость поколений	139
105	2,5509942	40	Неизменяемость поколений	128
106	2,3797988	28	Неизменяемость поколений	128
107	2,4060167	29	Неизменяемость поколений	130
108	2,0760707	493	Неизменяемость поколений	111
109	2,3539117	96	Неизменяемость поколений	113

110	2,5728352	116	Неизменяемость поколений	129
111	2,6588908	217	Неизменяемость поколений	142
112	2,0651775	20	Неизменяемость поколений	111
113	2,3241466	58	Неизменяемость поколений	120
114	3,7763936	20	Неизменяемость поколений	190
115	2,2236901	77	Неизменяемость поколений	115
116	2,0682429	213	Неизменяемость поколений	111
117	2,2496846	126	Неизменяемость поколений	120
118	3,4358941	202	Неизменяемость поколений	171
119	2,5529205	160	Неизменяемость поколений	134
120	2,1711502	103	Неизменяемость поколений	117
121	4,8259627	167	Неизменяемость поколений	232
122	2,9007835	98	Неизменяемость поколений	153
123	2,1465775	2018	Неизменяемость поколений	115
124	2,1976744	255	Неизменяемость поколений	115
125	2,3998346	476	Неизменяемость поколений	129
126	2,9337017	321	Неизменяемость поколений	148
127	2,6406159	839	Неизменяемость поколений	134
128	4,0451497	145	Неизменяемость поколений	218
129	3,9422428	45	Неизменяемость поколений	196
130	3,3144015	34	Неизменяемость поколений	153
131	2,5201362	82	Неизменяемость поколений	114

132	3,1502227	165	Неизменяемость поколений	153
133	3,2351475	134	Неизменяемость поколений	155
134	2,7270944	33	Неизменяемость поколений	114
135	4,7475213	123	Неизменяемость поколений	223
136	2,34766	42	Неизменяемость поколений	110
137	1,7239902	0	Фитнесс функция	89
138	4,0022649	6	Неизменяемость поколений	204
139	2,4992088	279	Неизменяемость поколений	125
140	2,9188898	9	Неизменяемость поколений	129
141	2,1212372	65	Неизменяемость поколений	112
142	2,8750717	23	Неизменяемость поколений	155
143	2,5910283	157	Неизменяемость поколений	116
144	3,1195369	313	Неизменяемость поколений	131
145	2,2058269	92	Неизменяемость поколений	107
146	2,5453177	18	Неизменяемость поколений	121
147	2,6466461	140	Неизменяемость поколений	140
148	3,9258364	1	Неизменяемость поколений	186
149	5,0414905	238	Неизменяемость поколений	217
150	2,6796233	14	Неизменяемость поколений	113
151	4,7914667	61	Неизменяемость поколений	236
152	2,7091972	84	Неизменяемость поколений	146
153	4,706234	19	Неизменяемость поколений	229
154	2,5084987	179	Неизменяемость поколений	112

155	3,1287925	106	Неизменяемость поколений	153
156	2,5186213	93	Неизменяемость поколений	116
157	2,8661135	71	Неизменяемость поколений	154
158	3,6706136	414	Неизменяемость поколений	172
159	5,0361077	152	Неизменяемость поколений	212
160	2,0678988	89	Неизменяемость поколений	111
161	3,1857638	27	Неизменяемость поколений	167
162	2,2771194	137	Неизменяемость поколений	121
163	2,4838085	29	Неизменяемость поколений	132
164	2,6542175	106	Неизменяемость поколений	141
165	3,8919733	13	Неизменяемость поколений	209
166	2,7729974	43	Неизменяемость поколений	149
167	2,2883267	599	Неизменяемость поколений	117
168	2,0012981	118	Неизменяемость поколений	104
169	2,5500686	96	Неизменяемость поколений	114
170	0,5175065	0	Фитнесс функция	24
171	2,4339092	242	Неизменяемость поколений	125
172	2,189815	420	Неизменяемость поколений	105
173	2,5235592	110	Неизменяемость поколений	127
174	2,1405574	17	Неизменяемость поколений	111
175	3,3800738	35	Неизменяемость поколений	178
176	3,4009925	71	Неизменяемость поколений	181
177	2,9177867	25	Неизменяемость поколений	156

178	2,3294306	382	Неизменяемость поколений	115
179	3,6317695	17	Неизменяемость поколений	184
180	4,7053621	152	Неизменяемость поколений	215
181	2,4324464	131	Неизменяемость поколений	111
182	2,1063999	40	Неизменяемость поколений	108
183	2,3070302	90	Неизменяемость поколений	110
184	4,2254488	57	Неизменяемость поколений	197
185	4,4497705	58	Неизменяемость поколений	239
186	4,180106	2	Неизменяемость поколений	214
187	2,3728871	434	Неизменяемость поколений	111
188	2,5684469	16	Неизменяемость поколений	128
189	2,4583178	160	Неизменяемость поколений	121
190	3,5141011	39	Неизменяемость поколений	177
191	2,7221753	129	Неизменяемость поколений	132
192	5,5892869	262	Неизменяемость поколений	266
193	2,6128369	327	Неизменяемость поколений	125
194	3,6940714	25	Неизменяемость поколений	184
195	3,7134164	89	Неизменяемость поколений	186
196	2,4780494	296	Неизменяемость поколений	126
197	4,1276278	360	Неизменяемость поколений	204
198	3,0178883	295	Неизменяемость поколений	150
199	2,1902731	253	Неизменяемость поколений	118

200	2,4655299	1	Неизменяемость поколений	126
201	2,1458631	10	Неизменяемость поколений	108
202	3,848849	107	Неизменяемость поколений	205
203	2,4330405	120	Неизменяемость поколений	116
204	2,7301604	213	Неизменяемость поколений	145
205	2,2405039	106	Неизменяемость поколений	114
206	2,5866156	287	Неизменяемость поколений	125
207	3,996211	90	Неизменяемость поколений	206
208	2,8795271	55	Неизменяемость поколений	153
209	5,528064	137	Неизменяемость поколений	274
210	2,4724543	12	Неизменяемость поколений	127
211	2,5860277	97	Неизменяемость поколений	126
212	2,21319	1239	Неизменяемость поколений	114
213	2,4113305	453	Неизменяемость поколений	124
214	4,3912067	120	Неизменяемость поколений	214
215	2,7046363	27	Неизменяемость поколений	118
216	4,3186799	10	Неизменяемость поколений	222
217	2,5475295	98	Неизменяемость поколений	132
218	2,6362986	164	Неизменяемость поколений	125
219	2,3728239	16	Неизменяемость поколений	121
220	2,7176243	170	Неизменяемость поколений	136
221	2,5426283	96	Неизменяемость поколений	127

222	2,9833739	68	Неизменяемость поколений	140
223	2,514419	73	Неизменяемость поколений	117
224	3,6008771	0	Фитнесс функция	184
225	3,174381	23	Неизменяемость поколений	155
226	2,8088966	146	Неизменяемость поколений	143
227	3,3509615	1	Неизменяемость поколений	160
228	2,5564343	50	Неизменяемость поколений	130
229	2,4203784	51	Неизменяемость поколений	124
230	3,4186222	0	Фитнесс функция	170
231	4,3908407	54	Неизменяемость поколений	236
232	4,0257219	463	Неизменяемость поколений	202
233	3,5188751	538	Неизменяемость поколений	189
234	2,1795258	193	Неизменяемость поколений	115
235	4,6309974	150	Неизменяемость поколений	222
236	3,7132579	77	Неизменяемость поколений	179
237	3,0602668	154	Неизменяемость поколений	151
238	3,8716209	25	Неизменяемость поколений	190
239	4,2163355	224	Неизменяемость поколений	210
240	2,930522	55	Неизменяемость поколений	126
241	4,1209921	272	Неизменяемость поколений	213
242	5,1861045	71	Неизменяемость поколений	253
243	2,6438908	153	Неизменяемость поколений	138
244	3,8358157	143	Неизменяемость поколений	192

245	2,5044084	279	Неизменяемость поколений	117
246	4,5271304	200	Неизменяемость поколений	216
247	2,7394493	123	Неизменяемость поколений	145
248	4,0673344	133	Неизменяемость поколений	183
249	2,3661755	22	Неизменяемость поколений	110
250	2,1129443	437	Неизменяемость поколений	107
251	2,9673641	635	Неизменяемость поколений	123
252	2,7668652	206	Неизменяемость поколений	144
253	4,0246969	177	Неизменяемость поколений	208
254	2,7220326	84	Неизменяемость поколений	142
255	2,6277913	678	Неизменяемость поколений	141
256	3,3362668	9	Неизменяемость поколений	179
257	2,2361601	56	Неизменяемость поколений	120
258	3,5884408	45	Неизменяемость поколений	188
259	2,954807	3	Неизменяемость поколений	145
260	2,9251963	365	Неизменяемость поколений	155
261	2,3332444	295	Неизменяемость поколений	118
262	2,5425026	351	Неизменяемость поколений	120
263	2,2525708	193	Неизменяемость поколений	117
264	2,8454073	1121	Неизменяемость поколений	140
265	3,3318036	196	Неизменяемость поколений	134
266	3,7482894	531	Неизменяемость поколений	168

267	3,8160028	56	Неизменяемость поколений	204
268	3,3288166	377	Неизменяемость поколений	168
269	2,1988669	917	Неизменяемость поколений	118
270	3,6023173	244	Неизменяемость поколений	175
271	2,271157	348	Неизменяемость поколений	109
272	5,9303075	16	Неизменяемость поколений	295
273	2,7748783	346	Неизменяемость поколений	129
274	3,3583255	41	Неизменяемость поколений	166
275	2,5118213	97	Неизменяемость поколений	134
276	2,7591833	41	Неизменяемость поколений	140
277	3,071673	75	Неизменяемость поколений	127
278	3,2753381	42	Неизменяемость поколений	162
279	3,4457091	10	Неизменяемость поколений	185
280	4,2444635	41	Неизменяемость поколений	226
281	2,2429832	48	Неизменяемость поколений	109
282	2,5225367	216	Неизменяемость поколений	122
283	3,3426242	38	Неизменяемость поколений	179
284	2,1337589	135	Неизменяемость поколений	107
285	3,8315152	136	Неизменяемость поколений	197
286	3,5795186	226	Неизменяемость поколений	191
287	2,3607309	147	Неизменяемость поколений	106
288	2,380323	37	Неизменяемость поколений	113
289	0,839188	0	Фитнесс функция	41

290	2,7008476	194	Неизменяемость поколений	137
291	3,3670229	40	Неизменяемость поколений	167
292	1,1529074	0	Фитнесс функция	60
293	3,8203434	132	Неизменяемость поколений	182
294	2,3357823	121	Неизменяемость поколений	104
295	3,9230985	119	Неизменяемость поколений	211
296	2,4294991	173	Неизменяемость поколений	109
297	2,9034368	98	Неизменяемость поколений	156
298	2,9726241	138	Неизменяемость поколений	137
299	4,006868	101	Неизменяемость поколений	182
300	3,2524928	6	Неизменяемость поколений	174
301	2,427803	1130	Неизменяемость поколений	129
302	3,570103	83	Неизменяемость поколений	191
303	3,0165769	178	Неизменяемость поколений	159
304	3,8777659	41	Неизменяемость поколений	186
305	4,4875667	115	Неизменяемость поколений	217
306	4,7070177	68	Неизменяемость поколений	224
307	2,9383867	366	Неизменяемость поколений	156
308	3,2779105	42	Неизменяемость поколений	169
309	2,733146	17	Неизменяемость поколений	141
310	2,3963796	866	Неизменяемость поколений	111
311	3,0643556	87	Неизменяемость поколений	166
312	3,1317875	12	Неизменяемость поколений	157

313	3,7830672	224	Неизменяемость поколений	201
314	2,900079	117	Неизменяемость поколений	151
315	2,7455793	39	Неизменяемость поколений	144
316	2,7818146	75	Неизменяемость поколений	146
317	2,5565121	53	Неизменяемость поколений	118
318	3,8693066	110	Неизменяемость поколений	182
319	3,5988577	72	Неизменяемость поколений	177
320	2,7539928	155	Неизменяемость поколений	127
321	2,268717	258	Неизменяемость поколений	121
322	3,0242339	28	Неизменяемость поколений	159
323	2,4783131	73	Неизменяемость поколений	124
324	3,9805429	210	Неизменяемость поколений	206
325	2,2156033	1245	Неизменяемость поколений	113
326	2,0751279	139	Неизменяемость поколений	108
327	2,7896519	166	Неизменяемость поколений	145
328	2,9949518	66	Неизменяемость поколений	160
329	3,1996427	2	Неизменяемость поколений	157
330	4,5756826	205	Неизменяемость поколений	225
331	3,5011798	36	Неизменяемость поколений	161
332	4,2368406	193	Неизменяемость поколений	218
333	3,2162798	83	Неизменяемость поколений	140
334	7,0030691	72	Неизменяемость поколений	357

335	2,8771932	82	Неизменяемость поколений	152
336	2,8884578	217	Неизменяемость поколений	156
337	4,8267397	71	Неизменяемость поколений	260
338	0,4137451	0	Фитнесс функция	22
339	2,9161144	89	Неизменяемость поколений	157
340	4,2718751	418	Неизменяемость поколений	231
341	2,4677526	73	Неизменяемость поколений	133
342	2,5309041	327	Неизменяемость поколений	137
343	3,4503493	12	Неизменяемость поколений	166
344	3,1070891	358	Неизменяемость поколений	144
345	2,468289	46	Неизменяемость поколений	124
346	2,3029428	589	Неизменяемость поколений	125
347	2,0538607	101	Неизменяемость поколений	110
348	4,4796004	172	Неизменяемость поколений	213
349	2,036949	63	Неизменяемость поколений	110
350	2,4559293	27	Неизменяемость поколений	133
351	4,3035229	1116	Неизменяемость поколений	217
352	3,1854822	276	Неизменяемость поколений	133
353	4,3375672	10	Неизменяемость поколений	232
354	2,4091696	398	Неизменяемость поколений	129
355	2,1638429	197	Неизменяемость поколений	117
356	2,2579693	137	Неизменяемость поколений	121
357	1,9078738	147	Неизменяемость поколений	103

358	2,4877768	136	Неизменяемость поколений	134
359	2,7907231	392	Неизменяемость поколений	150
360	2,1993422	1258	Неизменяемость поколений	119
361	3,0293061	508	Неизменяемость поколений	130
362	2,7994998	670	Неизменяемость поколений	144
363	3,7621776	114	Неизменяемость поколений	204
364	2,6661798	429	Неизменяемость поколений	143
365	2,3371101	109	Неизменяемость поколений	126
366	3,1966216	311	Неизменяемость поколений	159
367	3,985038	273	Неизменяемость поколений	215
368	2,6074383	311	Неизменяемость поколений	141
369	2,9031988	224	Неизменяемость поколений	157
370	3,1178024	134	Неизменяемость поколений	168
371	3,1850292	24	Неизменяемость поколений	161
372	2,2686914	52	Неизменяемость поколений	119
373	3,2674259	5	Неизменяемость поколений	173
374	2,6061068	81	Неизменяемость поколений	137
375	2,8880147	4	Неизменяемость поколений	153
376	2,6845144	349	Неизменяемость поколений	126
377	2,5078865	83	Неизменяемость поколений	115
378	2,5838394	173	Неизменяемость поколений	121
379	2,1685654	107	Неизменяемость поколений	114

380	2,3494564	160	Неизменяемость поколений	113
381	2,2669041	141	Неизменяемость поколений	112
382	4,0256173	272	Неизменяемость поколений	216
383	2,5638787	334	Неизменяемость поколений	136
384	2,9276509	141	Неизменяемость поколений	158
385	2,5998982	116	Неизменяемость поколений	121
386	2,7454488	467	Неизменяемость поколений	136
387	3,076943	749	Неизменяемость поколений	154
388	2,9856645	116	Неизменяемость поколений	129
389	5,9808032	119	Неизменяемость поколений	320
390	1,9574572	36	Неизменяемость поколений	105
391	3,6183	4	Неизменяемость поколений	169
392	1,9679621	128	Неизменяемость поколений	106
393	4,4567009	165	Неизменяемость поколений	229
394	2,2365978	53	Неизменяемость поколений	121
395	3,1908173	145	Неизменяемость поколений	164
396	2,0623761	428	Неизменяемость поколений	111
397	2,0591527	580	Неизменяемость поколений	111
398	3,024539	141	Неизменяемость поколений	163
399	4,2625882	11	Неизменяемость поколений	196
400	2,1647767	189	Неизменяемость поколений	106
401	3,1782581	51	Неизменяемость поколений	172

402	2,7375811	119	Неизменяемость поколений	125
403	3,382779	396	Неизменяемость поколений	165
404	3,3640834	2	Неизменяемость поколений	181
405	2,7684634	54	Неизменяемость поколений	118
406	2,7899642	128	Неизменяемость поколений	113
407	2,7225068	31	Неизменяемость поколений	120
408	3,4272499	324	Неизменяемость поколений	185
409	3,1025969	29	Неизменяемость поколений	167
410	1,9593473	17	Неизменяемость поколений	106
411	2,4201945	49	Неизменяемость поколений	115
412	2,7108581	25	Неизменяемость поколений	136
413	3,0410726	125	Неизменяемость поколений	163
414	3,350662	400	Неизменяемость поколений	146
415	3,1494056	24	Неизменяемость поколений	128
416	2,1440834	100	Неизменяемость поколений	116
417	2,0970965	296	Неизменяемость поколений	112
418	2,5714875	262	Неизменяемость поколений	104
419	2,9925225	123	Неизменяемость поколений	126
420	2,1666611	77	Неизменяемость поколений	117
421	2,4416299	447	Неизменяемость поколений	123
422	3,1992093	128	Неизменяемость поколений	172
423	2,0699692	107	Неизменяемость поколений	108

424	4,5154491	44	Неизменяемость поколений	229
425	2,3344431	580	Неизменяемость поколений	126
426	2,7609019	21	Неизменяемость поколений	132
427	3,1704391	502	Неизменяемость поколений	137
428	2,7550799	928	Неизменяемость поколений	120
429	1,9472774	256	Неизменяемость поколений	105
430	2,5360606	166	Неизменяемость поколений	137
431	3,6116216	73	Неизменяемость поколений	196
432	4,0508643	162	Неизменяемость поколений	183
433	4,3528854	55	Неизменяемость поколений	220
434	2,2926347	308	Неизменяемость поколений	111
435	4,2301054	190	Неизменяемость поколений	229
436	2,6305541	103	Неизменяемость поколений	142
437	2,3480242	16	Неизменяемость поколений	127
438	2,5151437	487	Неизменяемость поколений	136
439	2,219253	124	Неизменяемость поколений	120
440	2,5206808	68	Неизменяемость поколений	125
441	2,237206	112	Неизменяемость поколений	121
442	3,492191	83	Неизменяемость поколений	189
443	2,2319365	65	Неизменяемость поколений	120
444	3,6723385	117	Неизменяемость поколений	165
445	3,7164892	187	Неизменяемость поколений	172

446	3,7231295	206	Неизменяемость поколений	175
447	4,1193434	54	Неизменяемость поколений	216
448	3,5679531	32	Неизменяемость поколений	159
449	2,7347414	33	Неизменяемость поколений	135
450	3,8188363	235	Неизменяемость поколений	195
451	3,0764355	76	Неизменяемость поколений	155
452	2,8641718	800	Неизменяемость поколений	152
453	2,2915057	127	Неизменяемость поколений	121
454	2,013103	60	Неизменяемость поколений	106
455	2,3798405	28	Неизменяемость поколений	127
456	2,4660006	125	Неизменяемость поколений	133
457	3,5310729	64	Неизменяемость поколений	159
458	2,5987296	20	Неизменяемость поколений	139
459	2,019427	61	Неизменяемость поколений	107
460	2,9726923	208	Неизменяемость поколений	133
461	4,5452828	23	Неизменяемость поколений	211
462	2,7554339	161	Неизменяемость поколений	145
463	4,1307542	112	Неизменяемость поколений	189
464	2,3659901	465	Неизменяемость поколений	122
465	3,4481315	56	Неизменяемость поколений	168
466	2,3568422	15	Неизменяемость поколений	126
467	3,632616	195	Неизменяемость поколений	165

468	2,3238421	409	Неизменяемость поколений	125
469	2,0084691	273	Неизменяемость поколений	108
470	3,6083129	50	Неизменяемость поколений	192
471	2,2456373	66	Неизменяемость поколений	110
472	2,1618448	45	Неизменяемость поколений	104
473	4,662216	241	Неизменяемость поколений	238
474	2,3554801	186	Неизменяемость поколений	114
475	2,0826743	174	Неизменяемость поколений	104
476	2,4854838	119	Неизменяемость поколений	124
477	4,6146281	17	Неизменяемость поколений	225
478	2,9884175	347	Неизменяемость поколений	149
479	2,2877533	20	Неизменяемость поколений	110
480	2,2441532	102	Неизменяемость поколений	120
481	2,0660465	20	Неизменяемость поколений	111
482	3,0056503	248	Неизменяемость поколений	145
483	3,0555395	211	Неизменяемость поколений	146
484	3,4657314	101	Неизменяемость поколений	175
485	2,4536071	242	Неизменяемость поколений	117
486	2,1965037	8	Неизменяемость поколений	118
487	2,1922466	104	Неизменяемость поколений	118
488	2,6290702	46	Неизменяемость поколений	142
489	2,611164	138	Неизменяемость поколений	132

490	3,9423607	36	Неизменяемость поколений	195
491	2,0580848	212	Неизменяемость поколений	111
492	2,6441893	204	Неизменяемость поколений	130
493	2,7614229	588	Неизменяемость поколений	129
494	2,2051317	81	Неизменяемость поколений	116
495	3,844731	106	Неизменяемость поколений	201
496	3,7718889	33	Неизменяемость поколений	189
497	4,487223	32	Неизменяемость поколений	236
498	2,4770102	28	Неизменяемость поколений	130
499	2,2712677	423	Неизменяемость поколений	120
500	2,6213644	16	Неизменяемость поколений	138