**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

###### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

###### КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

**ИНСТИТУТ ЦИФРЫ**

**ОТЧЁТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

«Алгоритмы»

Студентки 3 курса, ФИТ-211 группы

**Колесник Полины Олеговны**

Направление 02.03.02 – «Фундаментальная информатика и информационные технологии»

Руководитель:

Доцент Зимин А. И.

Работа защищена

« »

“ ” 2023 г.

Кемерово 2023 г.

**ОТЧЁТ О ПРОДЕЛАННОЙ РАБОТЕ**

**2 вариант**

**1 задание**

def foo(i): # i - число

digits = "0123456789"

if i == 0:

return "0"

result = ""

while i > 0:

result = digits[i%10] + result

i = i // 10

return result

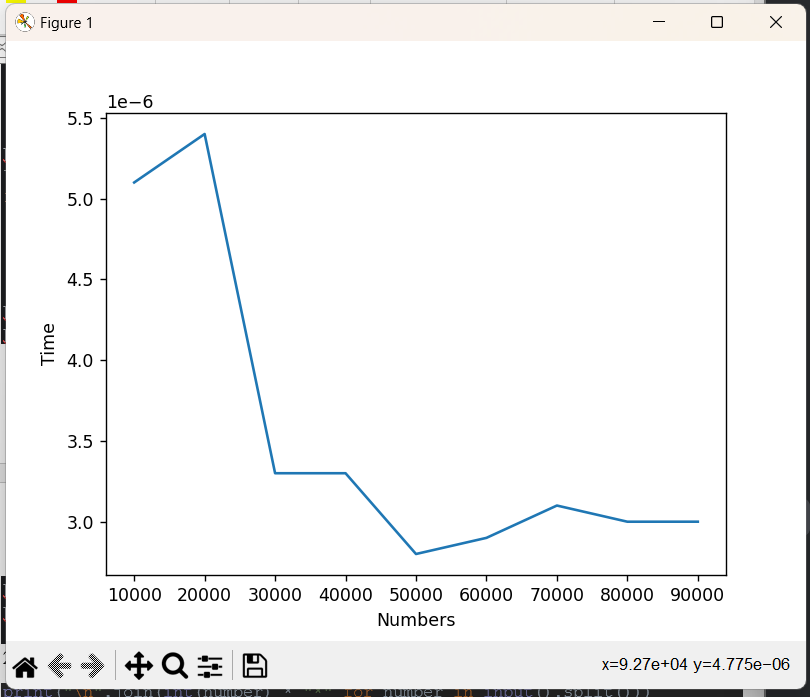
1. Что выполняет приведенная функция?

Приведенная функция принимает на вход число i и возвращает его строковое представление, состоящее из цифр.

1. Какова вычислительная сложность алгоритма (в O-нотации)?

Вычислительная сложность алгоритма (в O-нотации) равна O(n), где n - количество цифр в числе (длина итоговой строки).

import timeit  
import matplotlib.pyplot as plt  
  
  
def foo(i): # i - число  
 digits = "0123456789"  
 if i == 0:  
 return "0"  
 result = ""  
 while i > 0:  
 result = digits[i % 10] + result  
 i = i // 10  
 return result  
  
  
plt\_x = []  
time = []  
for i in range(10000, 100000, 10000):  
 plt\_x.append(i)  
 time.append(timeit.timeit(  
 f"foo({i})", number=5, globals=globals()))  
  
plt.xlabel('Numbers')  
plt.ylabel('Time')  
plt.plot(plt\_x, time)  
plt.show()



**2 задание**

Дан список. Проверить, что все значения списка различны.

1. Реализуйте два разных алгоритма решения задачи.

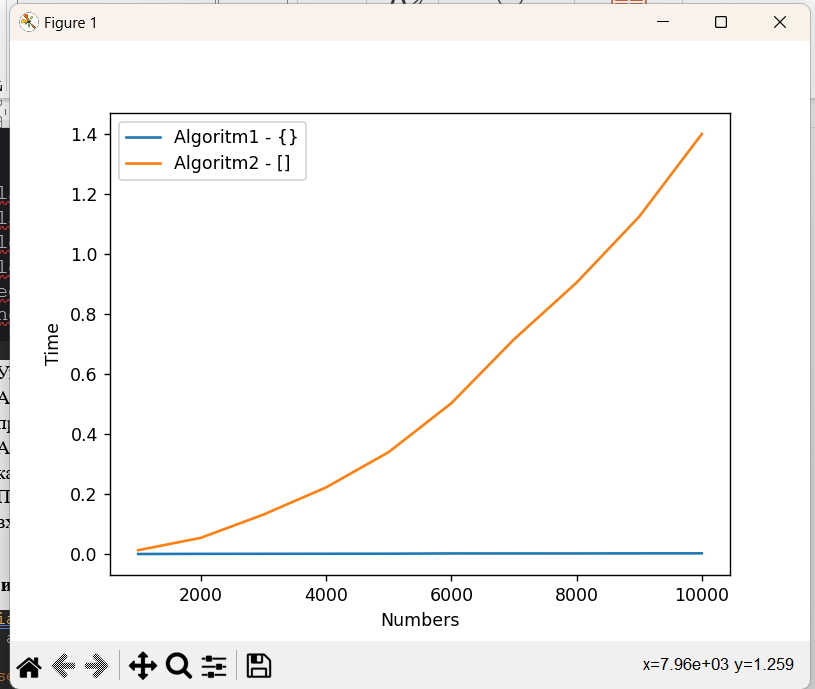
import timeit  
import matplotlib.pyplot as plt  
  
  
def algoritm1(item):  
 uniq\_dict = {}  
 for x in item:  
 if x in uniq\_dict:  
 return False  
 uniq\_dict[x] = True  
 return True  
  
  
def algoritm2(item):  
 uniq\_lst = []  
 for x in item:  
 if x in uniq\_lst:  
 return False  
 uniq\_lst.append(x)  
 return True  
  
  
plt\_x = []  
time1 = []  
time2 = []  
  
for i in range(1000, 10001, 1000):  
 plt\_x.append(i)  
 lst = list(range(i))  
  
 time1.append(timeit.timeit(  
 f"algoritm1({lst})", number=5, globals=globals()))  
  
 time2.append(timeit.timeit(  
 f"algoritm2({lst})", number=5, globals=globals()))  
  
plt.xlabel('Numbers')  
plt.ylabel('Time')  
plt.plot(plt\_x, time1, label=' Algoritm1 - {}')  
plt.plot(plt\_x, time2, label=' Algoritm2 - []')  
plt.legend()  
plt.show()

1. Укажите их асимптотическую сложность.

Асимптотическая сложность алгоритма\_1 является O(n). Проверка элемента происходит только раз.

Асимптотическая сложность алгоритма\_2 является O(n^2). Необходимо сравнивать каждый элемент друг с другом

1. Покажите их поведение при помощи графика зависимости времени от количества входных данных.



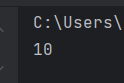
**3 задание**

Количество операций при выполнении двух алгоритмов для массива размером N таково:

T1(N) = N^2 – N – 10  
T2(N) = 4\*N + 40

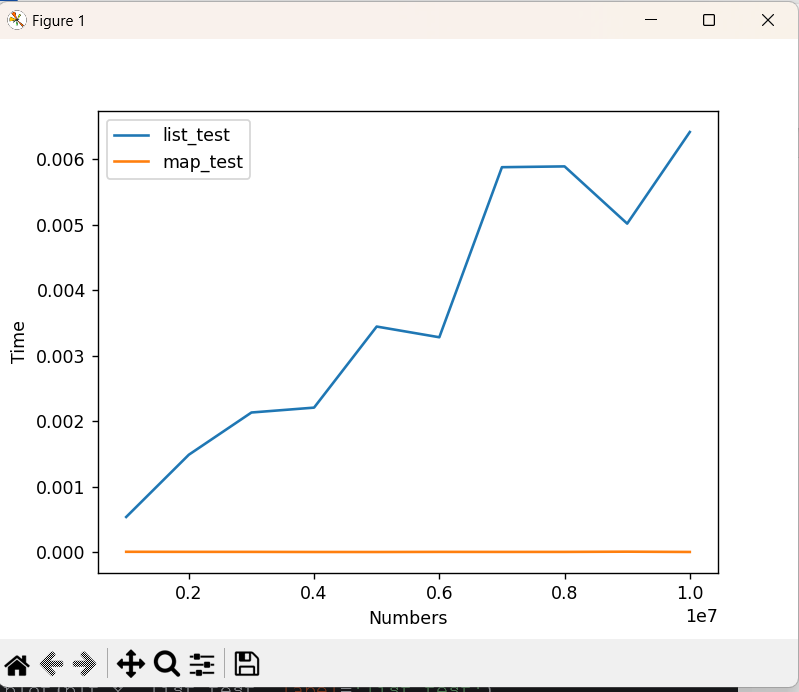
Определите размер массива N, для которого время выполнения обоих алгоритмов одинаково.

def t1(n):  
 return n\*\*2 - n - 10  
  
  
def t2(n):  
 return 4 \* n + 40  
  
  
for i in range(1000000):  
 if t1(i) == t2(i):  
 print(i)



**4 задание**

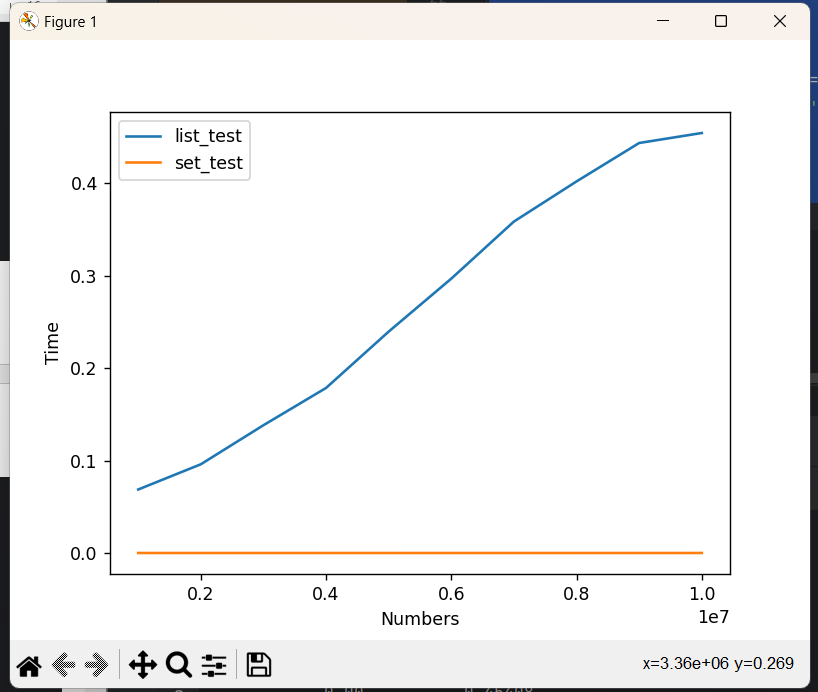
import timeit  
import matplotlib.pyplot as plt  
  
test\_del\_x = timeit.Timer("del x[0]", globals=globals())  
  
plt\_x = []  
list\_test = []  
map\_test = []  
  
for i in range(1000000, 10000001, 1000000):  
 plt\_x.append(i)  
  
 x = list(range(i))  
 time = test\_del\_x.timeit(number=1)  
 list\_test.append(time)  
  
 x = {x: i for x in range(i)}  
 pz = test\_del\_x.timeit(number=1)  
 map\_test.append(pz)  
  
 print("%15.2f, %15.5f" % (pz, time))  
  
plt.xlabel('Numbers')  
plt.ylabel('Time')  
plt.plot(plt\_x, list\_test, label='list\_test')  
plt.plot(plt\_x, map\_test, label='map\_test')  
plt.legend()  
plt.show()



Вывод: оператор del для списков имеет линейную зависимость, чем больше данных, тем выше время выполнения. Оператор del для словарей имеет константную зависимость, так как независимо от входных данных имеет одно и то же время выполнения. Такое различие объяснимо тем, что для списков необходимо проходить по каждому элементу для проверки. Но разница во времени выполнения небольшая, практически одинаковое.

**5 задание**

import timeit  
import matplotlib.pyplot as plt  
  
test\_in\_x = timeit.Timer("'test' in x", globals=globals())  
  
plt\_x = []  
list\_test = []  
set\_test = []  
  
for i in range(1000000, 10000001, 1000000):  
 plt\_x.append(i)  
  
 x = list(range(i))  
 time = test\_in\_x.timeit(number=5)  
 list\_test.append(time)  
  
 x = set(range(i))  
 pz = test\_in\_x.timeit(number=5)  
 set\_test.append(pz)  
  
 print("%15.2f, %15.5f" % (pz, time))  
  
plt.xlabel('Numbers')  
plt.ylabel('Time')  
plt.plot(plt\_x, list\_test, label='list\_test')  
plt.plot(plt\_x, set\_test, label='set\_test')  
plt.legend()  
  
plt.show()



Вывод: оператор in для списков имеет линейную зависимость, чем больше данных, тем выше время выполнения. Оператор in для множеств имеет константную зависимость, так как независимо от входных данных имеет одно и то же время выполнения. Такое различие объяснимо тем, что для списков необходимо проходить по каждому элементу для проверки. В данном случае время выполнения для списков намного выше, чем для множеств.