Отчет лабораторной работы № 5

Цель: определить границы применимости рекурсивного и итерационного подхода вычисления функции.

Задачи:

1. Написать программу сравнительного вычисления функции F(1) = 1;   
   F(2) = 2; F(n) = (-1)n\*(F(n-1)- F(n-2) /(2n)!) при n > 2
2. Сравнить результаты исследования времени выполнения и представить в табличной и графической форме.

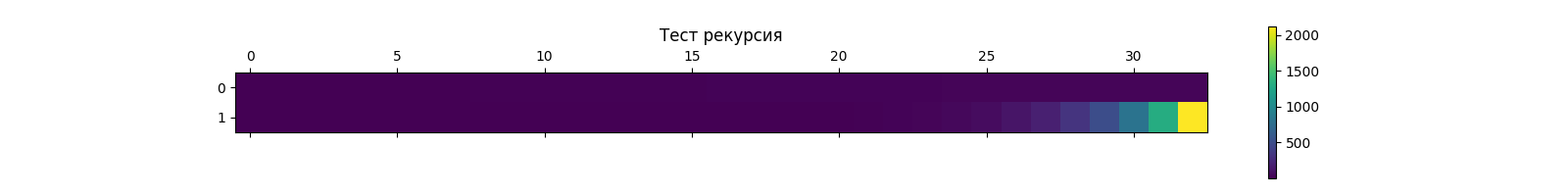
Словесно-формульный алгоритм

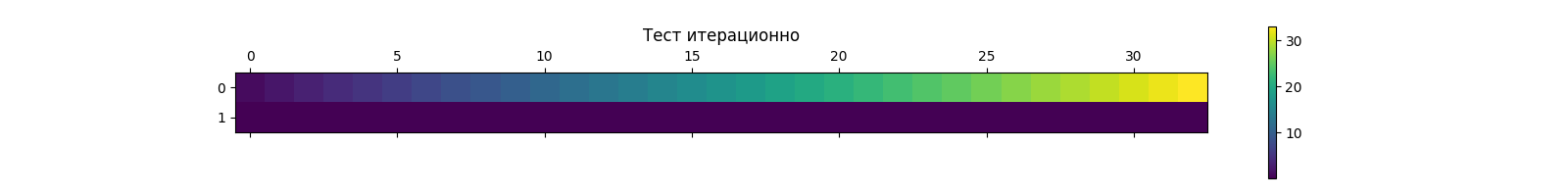
1. Задается максимальное время “n” выполнения функций (рекурсивной и итерационной)
2. Выполняется цикл в котором выполняются функции рекурсивного и итерационного подхода вычисления и считается время выполнения каждой функции. После каждой итерации цикла теста глубина(количество итераций) функций возрастает на 1. Цикл завершается после того как время выполнения рекурсивного или итерационного подхода будет превышать время “n”
3. После преобразуем полученные данные входе теста и выводим их на графике и таблице.

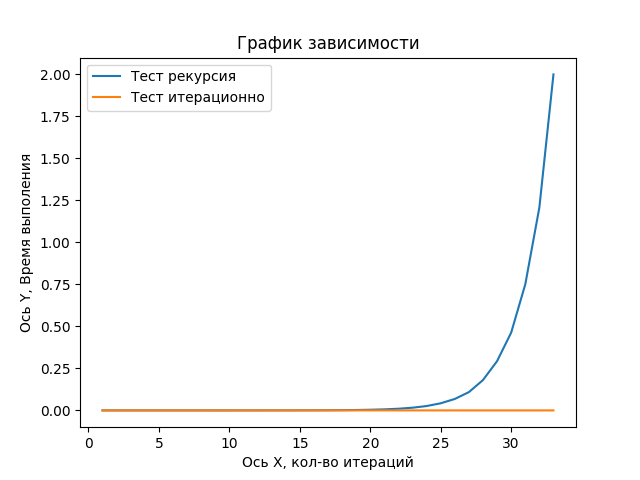
Листинг программы

#27 F(1) = 1; F(2) = 2; F(n) = (-1)n\*(F(n-1)- F(n-2) /(2n)!) при n > 2.  
import time  
import matplotlib.pyplot as plt  
from functools import lru\_cache  
"""факториал итерационно"""  
def fac\_it(m,n):  
 factorial = 1  
 for i in range(m, n + 1):  
 factorial \*= i  
 return factorial  
"""факториал рекурсивно"""  
def fac\_rec(n):  
 if n == 1:  
 return 1  
 return fac\_rec(n - 1) \* n  
  
"""Вывод матрицы c Наименованием """  
def print\_mat(mat,discription):  
 plt.matshow(mat)  
 plt.title(discription)  
 plt.colorbar()  
 plt.show()  
  
"""Рекурсионный подход"""  
@lru\_cache(maxsize=32)  
def F\_recursive(n, factorial):  
 if n == 1:  
 return 1  
 elif n == 2:  
 return 2  
 else:  
 return ((-1) \*\* n) \* (F\_recursive(n-1, factorial/n-1) - (F\_recursive(n-2, factorial/((n-1)\*(n-2))) / factorial))  
"""Итерационный подход"""  
def F\_iterative(n):  
 temp = []  
 factorial = 2  
 for i in range(1, n+1):  
 if i == 1:  
 temp.append(1)  
 elif i == 2:  
 factorial \*= fac\_it(((i - 1) \* 2 )+ 1, i \* 2)  
 temp.append(2)  
 elif i > 2:  
 factorial \*= fac\_it(((i - 1) \* 2 )+ 1, i \* 2)  
 temp.append(((-1) \*\* i) \* (temp[-1] - (temp[-2] / factorial)))  
 return temp[-1]  
  
"""Координаты для графика Рекурсионного подхода"""  
data1 = []  
"""Координаты для графика Итерационного подхода"""  
data2 = []  
"""Время рекурсивного продхода"""  
execution\_time\_recursive = 0  
"""Время итерационного продхода"""  
execution\_time\_iterative = 0  
time\_reaction = float(input("Введите макс время реакции "))  
"""Количество итераций(Глубина)"""  
iteration = 0  
"""цикл тестирования времени выполнения, пока вермя одного из подходов не стало больше макс время реакции"""  
while max(execution\_time\_recursive, execution\_time\_iterative) < time\_reaction:  
 """Рекурсионный подход"""  
 iteration += 1  
 """начало замера"""  
 start\_time = time.perf\_counter()  
 result = F\_recursive(iteration, fac\_rec(2\*iteration))  
 end\_time = time.perf\_counter()  
 F\_recursive.cache\_clear()  
 """конец замера"""  
 execution\_time\_recursive = end\_time - start\_time  
 data1.append((iteration, execution\_time\_recursive))  
 print("Тест рекурсия ", "Результат выполнения: ", result, " Время выполения: ", execution\_time\_recursive)  
  
 """Итерационный подход"""  
 """начало замера"""  
 start\_time = time.perf\_counter()  
 result = F\_iterative(iteration)  
 end\_time = time.perf\_counter()  
 """конец замера"""  
 execution\_time\_iterative = end\_time - start\_time  
 data2.append((iteration, execution\_time\_iterative))  
 print("Тест итерационно ", "Результат выполнения: ", result, " Время выполения: ", execution\_time\_iterative)  
 """итерации с одинаковым временем выполнения"""  
  
"""Преобразование для вывода инф. теста в график и таблицу"""  
x1, y1 = zip(\*data1)  
x2, y2 = zip(\*data2)  
#y1 = [x \* 1000 for x in y1]  
#y2 = [x \* 1000 for x in y2]  
print\_mat([x1,y1], "Тест рекурсия")  
print\_mat([x2,y2], "Тест итерационно")  
  
"""построение графика"""  
plt.plot(x1, y1, label='Тест рекурсия')  
plt.plot(x2, y2, label='Тест итерационно')  
  
"""добавление легенды"""  
plt.legend()  
"""добавление заголовка и меток осей"""  
plt.title('График зависимости')  
plt.xlabel('Ось X, кол-во итераций')  
plt.ylabel('Ось Y, Время выполения')  
"""вывод графика на экран"""  
plt.show()

Результаты выполнения







Выводы

В результате теста двух подходов именно время выполнения рекурсивного подхода превысило максимальное время “n”, а относительно времени выполнения итерационного похода в 8640479% раз. Исходя из графика, представленного выше, граница рекурсивного подхода является рекурсия глубиной 20.