Министерство науки и высшего образования Российской Федерации   
Федеральное государственное бюджетное образовательное   
учреждение высшего образования   
«Алтайский государственный технический   
университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчёт защищён с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель Андреева А.Ю.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Отчёт

по лабораторной работе № 6

**«РЕКУРСИВНЫЕ ВЫЧИСЛЕНИЯ. МЕМОИЗАЦИЯ»**

по дисциплине «Функциональное и логическое программирование»

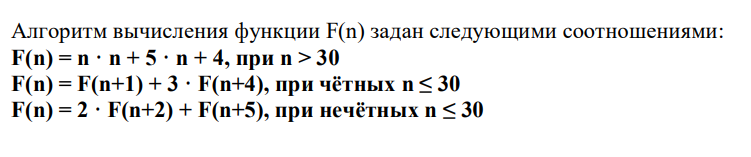
Студент группы ПИ-92 Савиков Д.А.

Преподаватель доцент, к.ф.-м.н. Андреева А.Ю.

Барнаул 2022

Номер в списке – 24

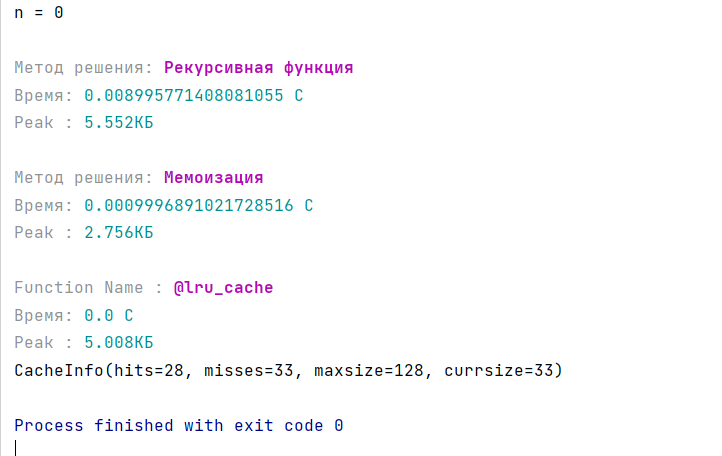
**Задание:** Построить рекурсивную функцию, выполнить замеры расхода памяти и времени .(datetime или timeit, sys.deep\_getsizeof()) Сделать реализацию с мемоизацией. Получить время и память. Сделать реализацию с декоратором @lru\_cache (модуль functools). Получить время и память.

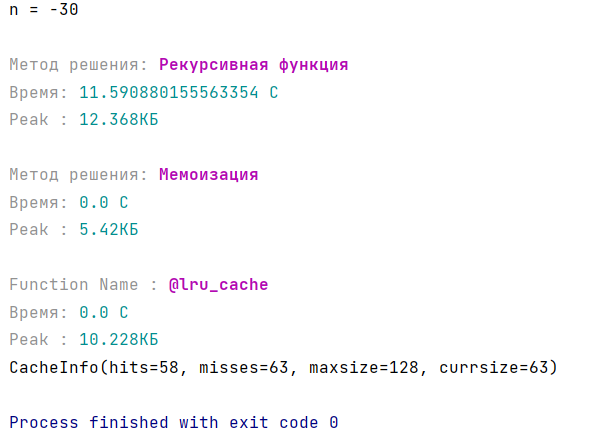


**Решение:**

from functools import lru\_cache  
import tracemalloc  
from time import time  
  
  
def f1(n):  
 *"""реализация обычной рекурсивной функции """* if n > 30:  
 return n \* n + 5 \* n + 4  
 *# нечет* elif n % 2:  
 return 2 \* f1(n + 2) + 3 \* f1(n + 5)  
 *# чет* else:  
 return f1(n + 1) + 3 \* f1(n + 4)  
  
  
*# мемоизация*cache\_dict = {} *# буфер для запоминания вызовов (чтобы не считать то, что уже считалось ранее)*def f2(n):  
 *"""реализация рекурсивной функции с мемоизацией"""* if cache\_dict.get(n) is not None:  
 return cache\_dict.get(n)  
  
 if n > 30:  
 cache\_dict[n] = n \* n + 5 \* n + 4  
 *# нечет* elif n % 2:  
 cache\_dict[n] = 2 \* f2(n + 2) + 3 \* f2(n + 5)  
 *# чет* else:  
 cache\_dict[n] = f2(n + 1) + 3 \* f2(n + 4)  
 return cache\_dict[n]  
  
  
@lru\_cache  
def f3(n):  
 *"""реализация рекурсивной функции с @lru\_cache из functools для мемоизации """* if n > 30:  
 return n \* n + 5 \* n + 4  
 *# нечет* elif n % 2:  
 return 2 \* f3(n + 2) + 3 \* f3(n + 5)  
 *# чет* else:  
 return f3(n + 1) + 3 \* f3(n + 4)  
  
  
n = -100  
print(f"n = {n}")  
  
*# обычная рекурсия*tracemalloc.start()  
startTime = time()  
f1(n)  
endTime = time()  
current, peak = tracemalloc.get\_traced\_memory()  
print(f"\n\033[37mМетод решения:\033[35;1m {'Рекурсивная функция'}\033[0m")  
print(f"\033[37mВремя:\033[36m {endTime - startTime} C\033[0m")  
print(f"\033[37mPeak :\033[36m {peak / (10 \*\* 3)}КБ\033[0m")  
tracemalloc.stop()  
  
*# мемоизация*tracemalloc.start()  
startTime = time()  
f2(n)  
endTime = time()  
\_, peak = tracemalloc.get\_traced\_memory()  
print(f"\n\033[37mМетод решения:\033[35;1m {'Мемоизация'}\033[0m")  
print(f"\033[37mВремя:\033[36m {endTime - startTime} С\033[0m")  
print(f"\033[37mPeak :\033[36m {peak / (10 \*\* 3)}КБ\033[0m")  
tracemalloc.stop()  
  
*# @lru\_cache*tracemalloc.start()  
startTime = time()  
f3(n)  
endTime = time()  
\_, peak = tracemalloc.get\_traced\_memory()  
print(f"\n\033[37mFunction Name :\033[35;1m {'@lru\_cache'}\033[0m")  
print(f"\033[37mВремя:\033[36m {endTime - startTime} C\033[0m")  
print(f"\033[37mPeak :\033[36m {peak / (10 \*\* 3)}КБ\033[0m")  
print(f3.cache\_info())  
tracemalloc.stop()

**Тесты:**





**Для n=-500 обычная рекурсия считается крайне долго (не дождался)**



**При n=-1000 вложенность рекурсии слишком большая и выбрасывается исключение:**

RecursionError: maximum recursion depth exceeded