Лабораторная работа № 7 по курсу дискретного анализа: Динамическое программирование

Условие

Имеется натуральное число n. За один ход с ним можно произвести следующие действия:

- Вычесть единицу
- Разделить на два
- Разделить на три

При этом стоимость каждой операции – текущее значение n. Стоимость преобразования - суммарная стоимость всех операций в преобразовании. Вам необходимо с помощью последовательностей указанных операций преобразовать число n в единицу таким образом, чтобы стоимость преобразования была наименьшей. Делить можно только нацело.

Метод решения

Чтобы решить задачу для произвольного n, нам достаточно знать решения трёх других задач для $n-1$, $\frac{n}{2}$, $\frac{n}{3}$. В таком случае, очевидно, что нужно будет взять минимум из этих трёх решений и не забыть прибавить текущее значение N.

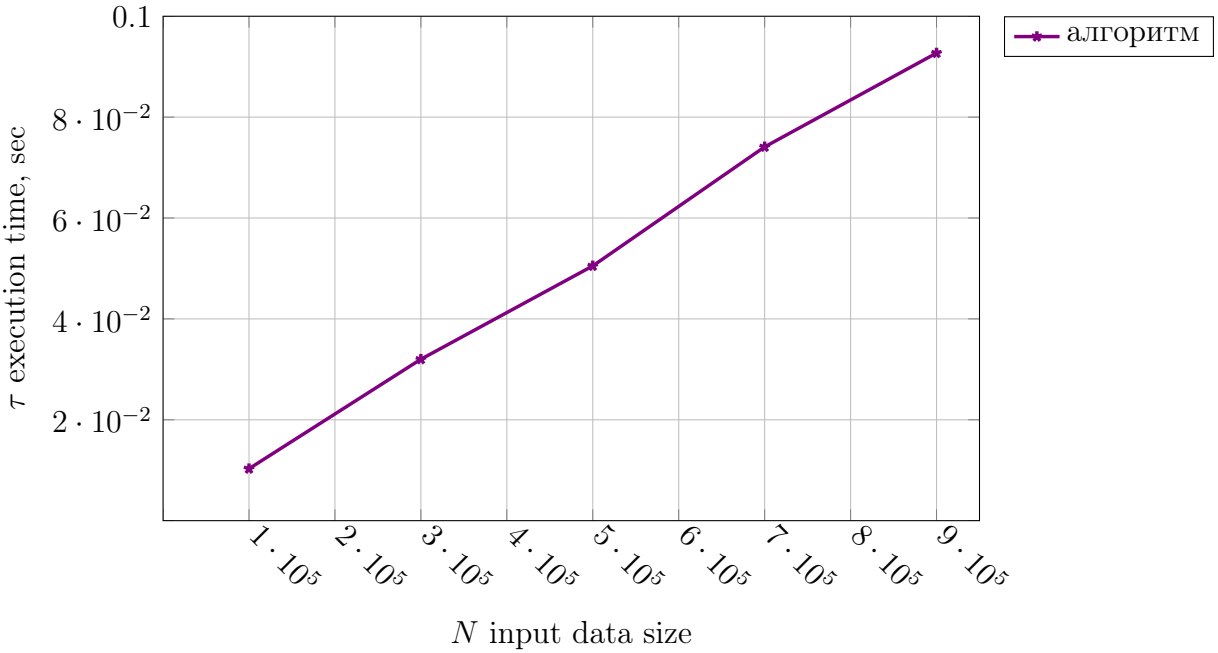
```
def f(x):
    if (x == 1): return 0
    if ((x % 2) * (x % 3) == 0):
        return x + min(f(x-1), min([f(x//i) for i in [2,3] if x % i == 0]))
    return x + f(x-1)
```

Описание программы

К сожалению, тестирующая система не позволила использовать модульный подход к решению поставленной задачи, поэтому весь рабочий код размещён в единственном файле *main.cpp*.

Чтобы не писать рекурсию с мемоизацией заведём массив dp на n + 1 элемент. В i-ой ячейке массива будет лежать ответ для числа i. Заполнять будем слева на право, то есть в каждую ячейку i будем записывать $i + min(dp[i-1], dp[\frac{i}{2}], dp[\frac{i}{3}])$. И сразу в массив action будем запоминать какое действие было выполнено, чтобы попасть в i-ую ячейку dp. Так за один проход мы решим задачу для всех чисел от 1 до n.
Сложность работы соответственно $O(n)$.

Тест производительности



Выводы

В ходе выполнения лабораторной работы была решена задача методом динамического программирования. То есть для оптимального решения задачи, мы выбирали самое оптимальное решение среди всех подзадач, а не какой-то одной, как в решениях с помощью жадного алгоритма. Подход динамического программирования применяется в основном в задачах, где нужно искать какой-то минимум или максимум, то есть что-то оптимизировать.