САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №7
по курсу «Алгоритмы и структуры данных»
Тема: Динамическое программирование №1
Вариант 8

Выполнил:
Макунина А.А.
К32421

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

Содержание отчета

Задачи по варианту

Задача №2. Примитивный калькулятор

Дан примитивный калькулятор, который может выполнять следующие три операции с текущим числом x: умножить x на 2, умножить x на 3 или прибавить 1 к x. Дано положительное целое число n, найдите минимальное количество операций, необходимых для получения числа n, начиная с числа 1.

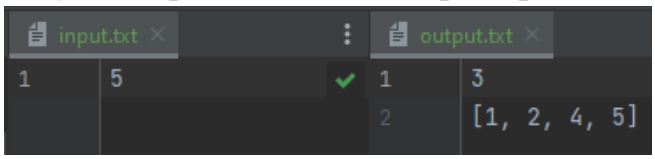
- **Формат ввода / входного файла (input.txt).** Дано одно целое число n, $1 \leq n \leq 10^6$. Посчитать минимальное количество операций, необходимых для получения n из числа 1.
- **Формат вывода / выходного файла (output.txt).** В первой строке вывести минимальное число k операций. Во второй – последовательность промежуточных чисел $a_0, a_1, ..., a_{k-1}$ таких, что $a_0 = 1, a_{k-1} = n$ и для всех $0 \leq i < k - a_{i+1}$ равно или $a_i + 1$, $2 \cdot a_i$, или $3 \cdot a_i$. Если есть несколько вариантов, выведите любой из них.

Мы создаём два массива: пары (работает по ключу) и числа, которые мы проверяем (их много). Проверяем делимость чисел и, если делится, проводим операцию, записывая в «словарь». Далее по ключу будем восстанавливать операции, доставая числа из «словаря».

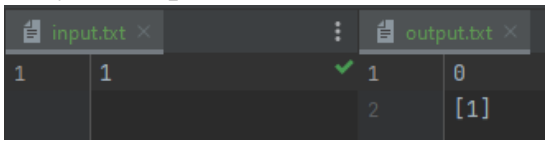
```
def calculator(n):
    pairs, check = dict(), []
    check.append([n, None])
    while True:
        swap = check.pop(0)  # присваиваем и удаляем
первый элемент
        now, continuer = swap[0], swap[1]
        if now not in pairs:
            pairs[now] = continuer
            if now == 1:
                break
            if now % 3 == 0:
                check.append([now // 3, now])
            if now % 2 == 0:
                check.append([now // 2, now])
            check.append([now - 1, now])
    operations = []
    key = 1
    while key:
```

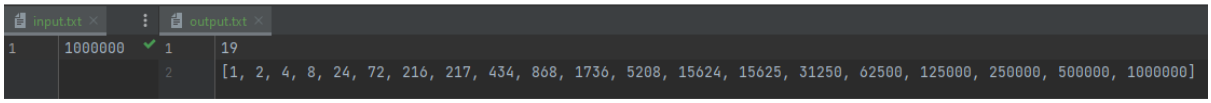
```
        operations.append(key)
        key = pairs[key]
    w.write(str(len(operations) - 1) + '\n' +
str(operations))


f = open('input.txt')
n = int(f.readline())
f.close()
w = open('output.txt', 'w')
calculator(n)
w.close()
```

Результат работы кода на примере из текста задачи:



Результат работы кода на максимальных и минимальных значениях:





	Время выполнения, с	Затраты памяти, мб
Нижняя граница диапазона значений входных данных из текста задачи	0.0028880000000000017	0.017284393310546875
Пример из задачи	0.0016146999999999967	0.017284393310546875
Верхняя граница диапазона значений входных данных из текста задачи	0.0037359000000000003	0.09319686889648438

Вывод по задаче: приятная задача, потому что относительно лёгкая и к решению можно подойти самыми разными способами (в голове проскочило 3 варианта по ходу решения).

Задача №7. Шаблоны

Многие операционные системы используют шаблоны для ссылки на группы объектов: файлов, пользователей, и т. д. Ваша задача – реализовать простейший алгоритм проверки шаблонов для имен файлов.

В этой задаче алфавит состоит из маленьких букв английского алфавита и точки («.»). Шаблоны могут содержать произвольные символы алфавита, а также два специальных символа: «?» и «*». Знак вопроса («?») соответствует ровно одному произвольному символу. Звездочка «+» соответствует подстроке произвольной длины (возможно, нулевой). Символы алфавита, встречающиеся в шаблоне, отображаются на ровно один такой же символ в проверяемой строчке. Строка считается подходящей под шаблон, если символы шаблона можно последовательно отобразить на символы строки таким образом, как описано выше. Например, строчки «ab», «aab» и «beda.» подходят под шаблон «*a?», а строчки «bebe», «a» и «ba» –нет.

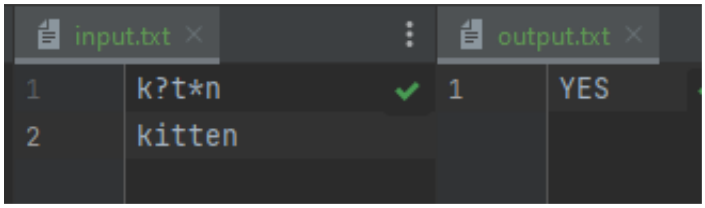
- **Формат ввода / входного файла (input.txt).** Первая строка входного файла определяет шаблон . Вторая строка S состоит только из символов алфавита. Ее необходимо проверить на соответствие шаблону. Длины обеих строк не превосходят 10 000. Строки могут быть пустыми – будьте внимательны!
- **Формат вывода / выходного файла (output.txt).** Если данная строка подходит под шаблон, выведите YES. Иначе выведите NO.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.

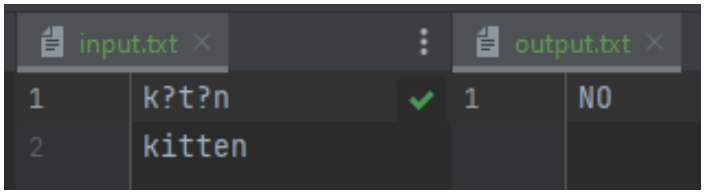
Проверка складывается следующим образом: мы получаем на вход шаблон и само слово. Посимвольно сравниваем буквы: если i символ в шаблоне и i буква в слове совпадают, мы накручиваем i и идём дальше. Если мы встречаем символы, описанные выше, проверяем подходят ли они под условие. insert_at служит для момента со *, где элемент соответствует подстроке произвольной длины + нулевой.

```
def check(pattern, word):
    max_length = len(word)
    i, insert_at = 0, None
    while i != max_length:
        if i >= len(pattern):
            return 'YES'
        if pattern[i] == word[i]:
            i += 1
            continue
        if pattern[i] != '*' and pattern[i] != '?'
```

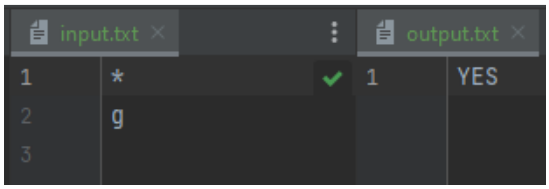
```
and pattern[i] != word[i]:
            if pattern[i - 1] == '*':
                pattern = pattern[:i] + '*' +
pattern[i:]
                i += 1
                continue
            elif i == insert_at:
                pattern = pattern[:i] + '*' +
pattern[i:]
                i += 1
                continue
            else:
                return 'NO'
        if pattern[i] == '?':
            if i + 1 < len(pattern) - 1:
                i += 1
                continue
            else:
                return 'NO'
        if pattern[i] == '*':
            if i == len(pattern) - 1:
                return 'YES'
            insert_at = i
            pattern = pattern[:i] + pattern[i + 1:]

    return 'YES'


f = open('input.txt')
pattern, word = f.readline(), f.readline()
f.close()
w = open('output.txt', 'w')
w.write(check(pattern, word))
w.close()
```

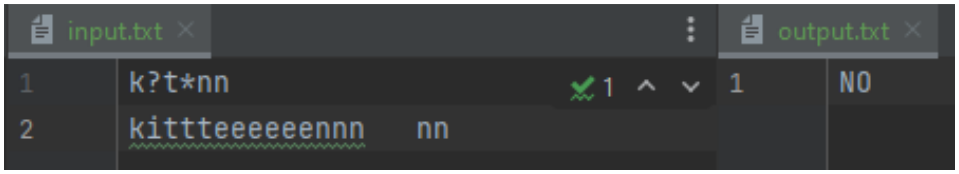
Результат работы кода на примере из текста задачи:





Результат работы кода на максимальных и минимальных значениях:





	Время выполнения, с	Затраты памяти, мб
Нижняя граница диапазона значений входных данных из текста задачи	0.0011519000000000112	0.017221450805664062
Пример из задачи	0.006902700000000012	0.017400741577148438
Пример из задачи	0.0011523999999999979	0.017400741577148438
Верхняя граница диапазона значений входных данных из текста задачи	0.0013477999999999823	0.017427444458007812

Вывод по задаче: тоже прикольная задача, они мне больше нравятся, чем в прошлых лабораторных. Места для творчества больше.

Вывод

Все лабораторные за первый семестр сделаны! Ура! Динамическое программирование это что-то из егэ по информатике, поэтому особо сложностей не возникло.