|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №2  по дисциплине «Язык программирования Python»  Вариант 1 | | |
|  | | Работу выполнил  студент группы ИТ-5-2024 1 курса  Семков Т.М.  «11» июня 2025 г. |
| Работу проверил  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фамилия И.О.  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2025 г. |
| Пермь 2025 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc200646248)

[Алгоритм решения 3](#_Toc200646249)

[1. Ключевая идея 3](#_Toc200646251)

[2. Подробный алгоритм 4](#_Toc200646252)

[3. Корректность 5](#_Toc200646253)

[4. Сложность 5](#_Toc200646254)

[5. Итог 5](#_Toc200646255)

[Тестирование 5](#_Toc200646256)

[Код программы 6](#_Toc200646257)

# Постановка задачи

|  |
| --- |
| Дана строка цифр длины *N* (≤ 1000), образующая кольцо. Требуется найти три последовательных числа *A*, *B*, *C* в кольце таких, что *A + B = C*. Цифры используются ровно один раз в порядке следования. Если решения нет, вывести «No». |

# Алгоритм решения

## Ключевая идея

Используем перебор всех возможных комбинаций расположения чисел *A*, *B*, *C* в кольце с оптимизациями:

* Фиксируем начало числа *A*
* Определяем длины чисел *A*, *B*, *C* на основе арифметических условий
* Проверяем валидность чисел (отсутствие ведущих нулей)
* Вычисляем значения и проверяем условие *A + B = C*

## 2. Подробный алгоритм

1. Преобразовать входную строку в список цифр

2. Для каждого возможного начала A (start в [0, n-1]):

а. Для каждой возможной длины A (a\_len в [1, n-2]):

- Пропустить если A имеет ведущий ноль

- Вычислить кандидатов для длины B:

• cand1 = n - 2\*a\_len (если a\_len >= cand1)

• cand2 = n - 2\*a\_len - 1

• cand3 = (n - a\_len) // 2 (если чётное)

• cand4 = (n - a\_len - 1) // 2 (если чётное)

- Для каждого кандидата b\_len:

• c\_len = n - a\_len - b\_len

• Проверить валидность B и C (нет ведущих нулей)

• Вычислить числа A, B, C с учётом кольцевой структуры

• Если A + B == C: вернуть решение

3. Если решение не найдено, вернуть "No"

## 3. Корректность

* **Проверка всех вариантов:** Алгоритм проверяет все возможные комбинации длин чисел
* **Валидация чисел:** отбрасывает числа с ведущими нулями
* **Кольцевая арифметика:** Корректный учёт перехода через конец кольца
* **Оптимизация:** Сокращение перебора за счёт арифметических условий для длин чисел

## 4. Сложность

* **Время:** *O(n²)* в худшем случае (n ≤ 1000 → до 1 млн операций)
* **Память:** *O(n)* для хранения цифр и промежуточных значений

## 5. Итог

Алгоритм гарантированно находит решение если оно существует, использует минимальную память и эффективен для ограничений задачи.

# Тестирование

| **№** | **Входные данные** | **Ожидаемый результат** | **Проверяемый случай** |
| --- | --- | --- | --- |
| 1 | "01902021" | "190+20=210" | Базовый случай |
| 2 | "111111" | "No" | Решения нет |
| 3 | "000" | "0+0=0" | Нули и минимальная длина |
| 4 | "123123246" | "123+123=246" | Равные слагаемые |
| 5 | "123456789123456" | "123456+789=124245" | Большие числа (проверка переполнения) |
| 6 | "1203" | "1+2=3" | Разные длины чисел |
| 7 | "010203" | "10+20=30" | Ведущие нули в исходной строке |
| 8 | "99" | "No" | Слишком короткая строка |
| 9 | "121224" | "12+12=24" | Множественные вхождения чисел |
| 10 | "123456789" | "No" | Отсутствие решения |

# Код программы

def main():

with open("input.txt", "r") as f:

s = f.read().strip()

n = len(s)

if n < 3:

with open("output.txt", "w") as out:

out.write("No")

return

digits = [int(char) for char in s]

def get\_number(start, length):

"""Вычисление числа из кольца по началу и длине"""

num = 0

for i in range(length):

idx = (start + i) % n

num = num \* 10 + digits[idx]

return num

def get\_string(start, length):

"""Получение подстроки для вывода"""

return ''.join(str(digits[(start + i) % n]) for i in range(length))

for start in range(n):

for a\_len in range(1, n - 1):

# Проверка ведущего нуля в A

if a\_len > 1 and digits[start] == 0:

continue

candidates = set()

# Вариант 1: a\_len >= b\_len

cand1 = n - 2 \* a\_len

if 1 <= cand1 <= n - a\_len - 1 and a\_len >= cand1:

candidates.add(cand1)

# Вариант 2: a\_len >= b\_len (с поправкой)

cand2 = n - 2 \* a\_len - 1

if 1 <= cand2 <= n - a\_len - 1 and a\_len >= cand2:

candidates.add(cand2)

total = n - a\_len

# Вариант 3: b\_len >= a\_len (чётное разбиение)

if total % 2 == 0:

cand3 = total // 2

if 1 <= cand3 <= n - a\_len - 1 and cand3 >= a\_len:

candidates.add(cand3)

# Вариант 4: b\_len >= a\_len (нечётное с поправкой)

cand4\_val = total - 1

if cand4\_val > 0 and cand4\_val % 2 == 0:

cand4 = cand4\_val // 2

if 1 <= cand4 <= n - a\_len - 1 and cand4 >= a\_len:

candidates.add(cand4)

for b\_len in candidates:

c\_len = n - a\_len - b\_len

if c\_len < 1:

continue

startB = (start + a\_len) % n

# Проверка ведущего нуля в B

if b\_len > 1 and digits[startB] == 0:

continue

startC = (start + a\_len + b\_len) % n

# Проверка ведущего нуля в C

if c\_len > 1 and digits[startC] == 0:

continue

A = get\_number(start, a\_len)

B = get\_number(startB, b\_len)

C = get\_number(startC, c\_len)

if A + B == C:

A\_str = get\_string(start, a\_len)

B\_str = get\_string(startB, b\_len)

C\_str = get\_string(startC, c\_len)

with open("output.txt", "w") as out:

out.write(f"{A\_str}+{B\_str}={C\_str}")

return

with open("output.txt", "w") as out:

out.write("No")

if \_\_name\_\_ == "\_\_main\_\_":

main()