Аннотация

В рамках курсовой работы №1007 разработано алгоритмическое и программное обеспечение, предназначенное для определения изначального набора символов после того, как с ними произошли следующие изменения:

1. Любой (но только один) символ 0 заменён символом 1.
2. Любой (но только один) символ удалён.
3. Дополнительный символ (0 или 1) вставлен в любую позицию.

Программное обеспечение предназначено для определения типа изменения и последующей работы с ней.

Алгоритмическое и программное обеспечение разрабатывается в рамках курсовой работы по дисциплине «Объектно-ориентированное программирование».

При выполнении программа использует не более 64 МБ оперативной памяти и работает не более 2 секунд.

Программа является консольным приложением. Язык разработки – Python 3.8. Для компиляции может быть использован компилятор, встроенный в среду разработки PyCharm Community Edition 2021.2.1

Общая постановка задачи №1007 «Кодовые слова»

Ограничение времени: 2.0 секунды

Ограничение памяти: 64 МБ

Отправитель посылает по зашумлённой линии некоторые двоичные кодовые слова. Получатель на другом конце линии использует специальный метод восстановления исходных слов.

Каждое слово изначально состоит из символов 0 и 1. Все слова имеют одинаковую длину N ($4 \leq N \leq 1000$). После передачи по зашумлённой линии со словом может случиться одно (но не более) из следующих изменений:

1. Любой (но только один) символ 0 заменён символом 1.
2. Любой (но только один) символ удалён.
3. Дополнительный символ (0 или 1) вставлен в любую позицию.

Известно, что все исходные слова имеют следующее свойство: сумма позиций, на которых находятся символы 1, кратна $(N + 1)$ или равна нулю.

Исходные данные :Ввод содержит число N, затем идут полученные слова. Слова разделены переводами строк. Слов не более 2001. Кроме описанного, ввод может содержать только дополнительные пробелы или переводы строк.

Результат:Ваша программа должна вывести исходную последовательность слов какими они были отправлены. Слова должны разделяться переводами строк.

Пример

исходные данные	результат
4 0000 011 1011 11011	0000 0110 1001 1111

Общие сведения

Программа «Кодовые слова» предназначена для восстановления исходной последовательности нолей и единиц.

Программа выполняет следующие действия:

1) Получает информацию о длине слов, а также сами слова.
2) Определяет вид изменения.
3) Производит необходимые изменения для получения ответа.
4) Выводит ответ.

Входные данные предполагаются корректными, поэтому никаких проверочных действий не производится. На любом корректном наборе входных данных программа заканчивает работу безаварийно.

Время работы программы на любых корректных входных данных не превышает 2 секунды, а объём выделенной памяти не превышает 64 МБ.

Программа написана на языке программирования Python 3.8.

Ключевые идеи и алгоритм решения

Главной частью в данной курсовой работе является определения одного из трех типов изменений исходных данных и дальнейшее преобразование, но сначала нам нужно посчитать сумму позиций единиц, что следует из условия задачи ,в будущем нам это очень сильно пригодится: «Известно, что все исходные слова имеют следующее свойство: сумма позиций, на которых находятся символы 1, кратна ($N + 1$) или равна нулю.» В своей программе я работаю со строками, так удобнее посчитать сумму позиций единиц по модулю (N+1) в функции calc(s): проходимся по каждому элементу строки (набора единиц и нулей) и считываем через цикл for позиции единиц.

В функцию solve(s) отправляем набор символов с возможной ошибкой и дальше в зависимости от длины строки определяем тип изменения:

1) Если длина не поменялась, то произошла замена 1 на 0, далее перебираем символ, который изменился. Что мы делаем: раз поменялся ровно один символ, то давайте переберем позицию i, в которой произошло изменение. Если мы знаем, как выглядела строка, мы можем понять, чему была равна сумма позиций единичек (т.е. calc(s)) для строки до изменения.Эта сумма равна res - (i + 1). А если сумма позиций единиц по модулю (N+1) равна нулю, то изменений нет и далее отправляем ответ на вывод.
2) Если длина стала меньше на единицу, то произошло удаление, поэтому нам нужно вставить символ на какую-то позицию. Если сумма позиций единиц по модулю (N+1) равна нулю, то вставляем 0 в конец. Если сумма позиций единиц по модулю (N+1) равна 1, то вставляем 1 в конец. Как и в прошлый раз мы перебираем позицию, где произошло изменение (только теперь это удаление) и смотрим, как изменилась сумма позиций единичек. Если мы вставляем ноль на позицию i, то у всех единиц, который стоят справа от неё увеличится номер позиции на 1, то есть сумма

увеличится на количество единиц, стоящих справа. Аналогично, если мы вставляем единицу, то сумма увеличится на количество единиц справа плюс i + 1 (потому что сама вставленная единица тоже увеличит сумму). Теперь, чтобы не пересчитывать каждый раз заново количество единиц справа, мы будем перебирать i от больших значений к меньшим. Ones - это как раз количество единиц справа от текущего i.

3) Если длина стала больше на единицу, то произошла вставка, поэтому нам нужно определить, какой символ нужно удалить. Если мы удаляем символ, то снова два случая. Если удаляем 0 , то позиции всех единиц, стоящих справа, уменьшатся на один, и поэтому сумма уменьшится на их количество. Если удаляем 1, то всё тоже самое но из суммы ещё вытчется позиция самой удаляемой единицы. Снова, чтобы не считать количество единиц справа каждый раз, будем перебирать i в порядке уменьшения и поддерживать ones.
4) Для того чтобы вывод был быстрым, сложим все ответы в одну строчку и в конце ее выведем (out = '')

Структура программного обеспечения

Функция calc() вычисляет сумму позиций единиц в строке по модулю (N+1).

Функция solve() принимает строку с (возможно) одним изменением, определяет данное изменение, производит добавления, перестановки, удаления и возвращает исходную строку для вывода ответа.

Функция main() вызывает выше указанные функции.

Import sys

Input = sys.stdin.readline – используется для быстрого ввода и вывода

Line.strip() – удаление лишних пробелов

Структура данных

В программе используются следующие основные элементы данных:

n –длина последовательности 0 и 1 (длина слова)

res – хранит сумму позиций единиц по модулю (n+1)

ones - количество единиц справа от текущего i

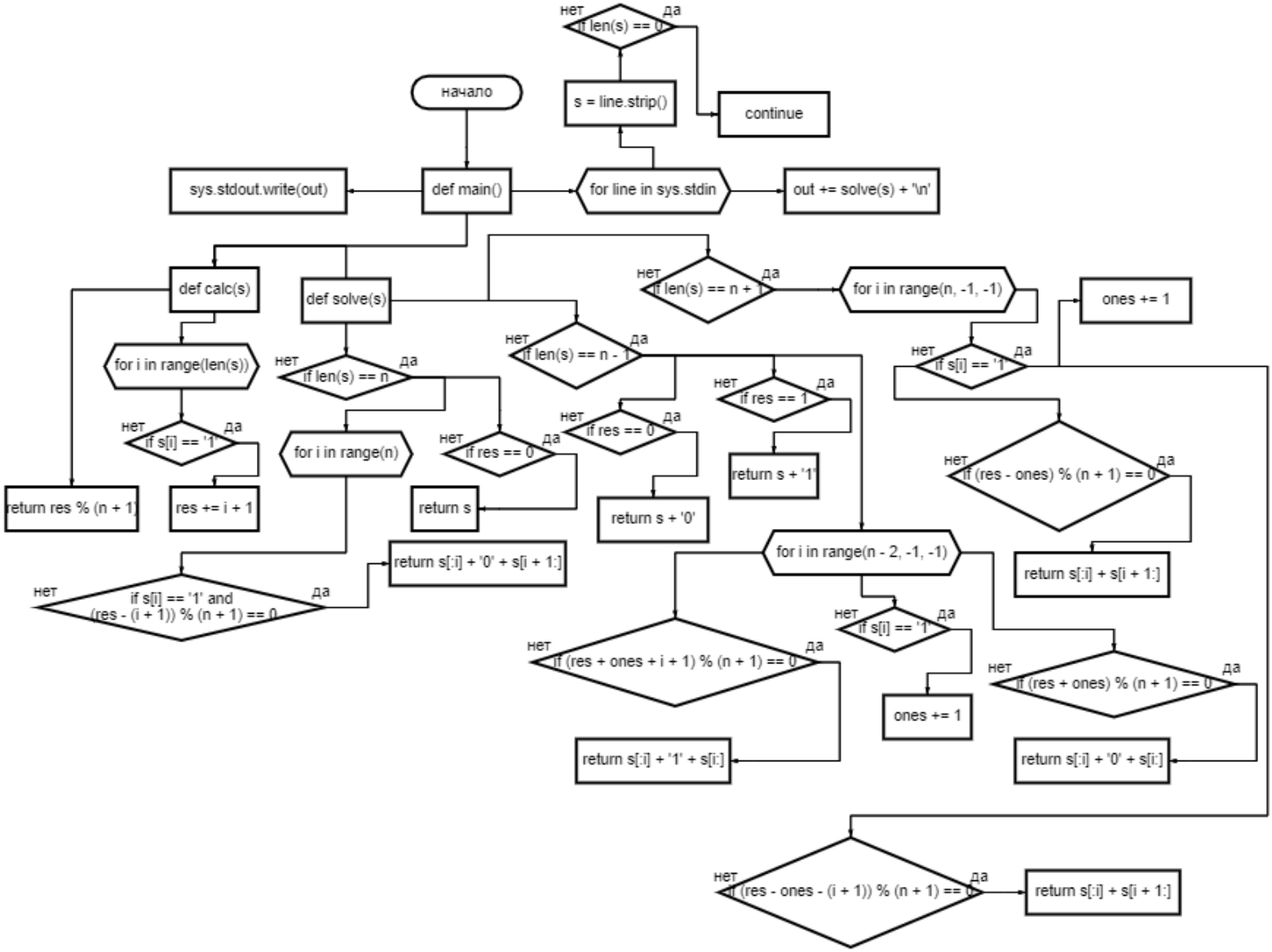
Порядок тестирования

Рассмотрим тестовый пример:

Вводимые данные	Ответ
4 1)0000 2)011 3)1011	 1)0000 2)0110 3)1001

1) Имеем 4 нуля из четырех возможных символов, значит, что ни вставка, ни удаление не были применены к данной последовательности. Сумма позиций единиц по модулю (N+1) равна нулю, то есть изменений нет.
2) Имеем 3 цифры из 4 возможных, значит произошло удаление одного из символов. В данном случае сумма позиций единиц по модулю (N+1) равна нулю, то есть вставить 0 нужно именно в конец.
3) Имеем 4 символа из 4-х возможных, а это значит, что произошла замена. Нужно понять, какую единицу заменить на 0. Нужно, чтобы сумма позиций единиц была равна или кратна 5. В нашем случае(1011) она равна 8,а это значит, что на третьей позиции стоит лишняя единица (8-5=3).

Блок – схема


начало
def main()
sys.stdout.write(out)
for line in sys.stdin
out += solve(s) + '\n'
s = line.strip()
if len(s) == 0
да
нет
continue
def calc(s)
def solve(s)
for i in range(len(s))
if s[i] == '1'
res += i + 1
return res % (n + 1)
if len(s) == n
for i in range(n)
if s[i] == '1' and
(res - (i + 1)) % (n + 1) == 0
return s[:i] + '0' + s[i + 1:]
if res == 0
return s
if len(s) == n - 1
if res == 0
return s + '0'
if res == 1
return s + '1'
if len(s) == n + 1
for i in range(n, -1, -1)
if s[i] == '1'
ones += 1
if (res - ones) % (n + 1) == 0
return s[:i] + s[i + 1:]
for i in range(n - 2, -1, -1)
if (res + ones + i + 1) % (n + 1) == 0
return s[:i] + '1' + s[i:]
if s[i] == '1'
ones += 1
if (res + ones) % (n + 1) == 0
return s[:i] + '0' + s[i:]
if (res - ones - (i + 1)) % (n + 1) == 0
return s[:i] + s[i + 1:]

Код программы

```
import sys
input = sys.stdin.readline
def main():
    n = int(input())
    def calc(s):
        res = 0
        for i in range(len(s)):
            if s[i] == '1':
                res += i + 1
        return res % (n + 1)
    def solve(s):
        res = calc(s)
        if len(s) == n:
            if res == 0:
                return s
            for i in range(n):
                if s[i] == '1' and (res - (i + 1)) % (n + 1) == 0:
                    return s[:i] + '0' + s[i + 1:]
        if len(s) == n - 1:
            if res == 0:
                return s + '0'
            if res == 1:
                return s + '1'
            ones = 0
            for i in range(n - 2, -1, -1):
                if s[i] == '1':
                    ones += 1
                if (res + ones) % (n + 1) == 0:
                    return s[:i] + '0' + s[i:]
```

```
            if (res + ones + i + 1) % (n + 1) == 0:
                return s[:i] + '1' + s[i:]
        if len(s) == n + 1:
            ones = 0
            for i in range(n, -1, -1):
                if s[i] == '1':
                    if (res - ones - (i + 1)) % (n + 1) == 0:
                        return s[:i] + s[i + 1:]
                    ones += 1
                else:
                    if (res - ones) % (n + 1) == 0:
                        return s[:i] + s[i + 1:]
    out = ''
    for line in sys.stdin:
        s = line.strip()
        if len(s) == 0:
            continue
        out += solve(s) + '\n'
    sys.stdout.write(out)
main()
```