(cc) BY

Шубинкин Владимир Николаевич
учитель
МАОУ «Лицей №131»
г. Казань, Республика Татарстан

АЛЬТЕРНАТИВНЫЕ СПОСОБЫ РЕШЕНИЯ ЗАДАЧИ №24 КЕГЭ ПО ИНФОРМАТИКЕ

Аннотация*: в статье приводятся способы решения задачи №24 КЕГЭ по информатике: предполагаемый авторами ФГБНУ «Федеральный институт педагогических измерений» (ФИПИ) и два альтернативных. Обсуждаются их достоинства и границы применимости.*

Ключевые слова*: КЕГЭ, информатика, программирование.*

Популярность сферы информационных технологий неуклонно растёт. Как следствие, с каждым годом всё больше выпускников выбирают экзамен по информатике. Согласно предварительному отчёту Рособрнадзора в 2023 году их количество превысило 113 тысяч, в 2022 году информатику сдавало около 102 тысяч человек, в 2021 году – 94 тысячи [3]. С 2021 года экзамен проводится в компьютерной форме. Для получения высокого балла необходимым условием является навык программирования. В частности, экзаменуемый должен уметь программно обрабатывать символьную информацию. Этот навык требуется для решения задачи №24 [4, с. 12]. ФИПИ (федеральный институт педагогических измерений) относит её к заданиям высокого уровня сложности [4, с. 12] и утверждает, что «для успешного выполнения этого задания требуется написать алгоритм, реализующий простейший конечный автомат с сумматором. Состояние автомата и значение сумматора изменяются в зависимости от встреченной буквы и текущего состояния» [2, с. 11]. В данной статье будет рассмотрен как вышеупомянутый метод, так и ещё два альтернативных способа решения подобных задач. Примеры программ будут даны на языке Python 3.

Content is licensed under the Creative Commons Attribution 4.0 license (CC-BY 4.0)

Разберём решения на примере базовой задачи: «Текстовый файл 24.txt состоит из символов X, Y, Z. Определите максимальное количество идущих подряд символов X».

Каким бы способом мы ни решали задачу, в первую очередь необходимо считать информацию из файла. В Python это рекомендуется делать с помощью контекстного оператора with, сопоставляющего идентификатор (в примере – f) с файловым объектом. Текст считывается с помощью файлового метода read() и очищается от возможного переноса строки в конце файла методом strip() (Рис. 1).

```
with open('24.txt') as f:
    s = f.read().strip()
```

Рис. 1. Считывание информации из файла

Способ решения №1. Конечный автомат (Рис. 2).

```
cur_len = 0
max_len = 0
for c in s:
    if c == 'X':
        cur_len += 1
    else:
        cur_len = 0
    max_len = max(max_len, cur_len)
print(max_len)
```

Рис. 2. Способ решения №1

Идея решения состоит в том, что мы будем определять, является текущий символом буквой X или нет. Если является, значит текущая длина искомой цепочки символов (cur_len) увеличивается на 1, иначе цепочка прерывается, а текущая длина сбрасывается на ноль. Какой бы ни получалась текущая длина, она сравнивается с искомой максимально возможной длиной (max_len) подходящей подстроки. При этом если текущая длина оказалась больше хранимой, она становится максимальной.

Способ решения №2. Замена символов с разделением на подстроки (Рис. 3).

Содержимое доступно по лицензии Creative Commons Attribution 4.0 license (CC-BY 4.0)

```
s1 = s.replace('Y', 'Z')
c_sequenses = s1.split('Z')
max_len = len(max(c_sequenses))
print(max_len)
```

Рис. 3. Способ решения №2

Второй подход заключается в том, что мы создаём новую строку на основе старой путём замены всех символов Y на символы Z. Таким образом в строке s1 присутствуют только символы X и Z. Строковый метод split() разбивает строку на список подстрок по разделителю, указанному в скобках. Например, если s = «XXXZZXZXXZZZXZ», то s1 = [«XXX», “”, «X», «XX», “”, “”, «X», “”]. В списке присутствуют пустые строки в тех местах, где символ Z стоял рядом с другим символом Z или с началом / концом строки. Остаётся только найти подстроку максимальной длины с помощью встроенной функции max() и определить её длину с помощью встроенной функции len(). Нужно иметь в виду, что в более сложных задачах, когда в подстроках может оказаться несколько различных символов, следует использовать len в качестве критерия сравнения функции max(). В нашем примере можно было бы написать max_len = len(max(c_sequenses, key=len)). Если не указать параметр key, строки будут сравниваться в лексикографическом порядке, и, например, «VOVA» окажется больше «VLADIMIR».

Способ решения №3. Поиск увеличивающейся подстроки (Рис. 4).

```
c_sequens = 'X'
while c_sequens in s:
    c_sequens += 'X'
print(len(c_sequens) - 1)
```

Рис. 4. Способ решения №3

Идея третьего решения в том, что мы создаём строку, которая изначально равна «X», а затем добавляем по букве «X» до тех пор, пока она встречается в исходном тексте. Цикл while закончится, когда строка c_sequens уже не будет

Content is licensed under the Creative Commons Attribution 4.0 license (CC-BY 4.0)

найдена в s. Таким образом, мы найдём не наибольшее количество подряд идущих символов «X», а наименьшее количество, которое не встречается в s. Поэтому при выводе ответа следует вычесть единицу.

Очевидно, что способ №1, использование которого предполагают авторы КЕГЭ, внешне более громоздкий, а потому менее привлекательный для выпускников и их наставников. Более того, второй способ выигрывает по скорости вследствие использования встроенных функций. Последнее утверждение верно для языка Python. К преимуществам первого способа можно отнести экономичное использование памяти, но на экзамене это не оценивается. У второго и третьего способов есть ограничения по применимости. Третий способ решает только подкласс задач, а скорость выполнения программы сильно зависит от исходных данных. Второй способ опасно использовать при возможности пересечения разделителей. Например, если разбить строку «ABBABBABXXXBABBABBABBA» по строке «ABBA», передав последнюю как параметр метода split, то получим список [“”, «BBABXXXB», «BB», “”] в то время как в подобной задаче требовалось бы получить [“”, «XXX», “”]. Как учитель информатики считаю, что учащиеся должны освоить все указанные методы, а также уметь определять границы их применимости.

Приведём также решение задачи 24 одного из вариантов ЕГЭ по информатике 2023 года всеми тремя способами.

Условие задачи взято с сайта К.Ю. Полякова [1, с. 44] (№24.267): «Текстовый файл 24–264.txt состоит не более чем из 10^6 символов и содержит только заглавные буквы латинского алфавита и цифры. Определите максимальную длину подстроки, которая может являться записью числа в шестнадцатеричной системе счисления»

Для всех методов необходимо прочитать данные из файла. Также удобно будет импортировать английский алфавит из модуля string (Рис. 5).

```
from string import ascii_uppercase

with open('24data/24-264.txt') as f:
    s = f.read().strip()
```

Рис.5. Импорт алфавита и считывание данных

Содержимое доступно по лицензии Creative Commons Attribution 4.0 license (CC-BY 4.0)

Способ решения №1. Конечный автомат (Рис. 6).

```
not_hex_digits = ascii_uppercase[6:]
max_len = 0
cur_len = 0
for c in s:
    if c in not_hex_digits:
        cur_len = 0
    else:
        cur_len += 1
    max_len = max(max_len, cur_len)
print(max_len)
```

Рис. 6. Способ решения №1

Идея решения ничем не отличается от идеи решения базовой задачи.

Способ решения №2. Замена символов с разделением на подстроки (Рис. 7).

```
s1 = s
not_hex_digits = ascii_uppercase[6:]
for c in not_hex_digits:
    s1 = s1.replace(c, ' ')
hex_numbers = s1.split()
max_len = len(max(hex_numbers, key=len))
print(max_len)
```

Рис. 7. Способ решения №2

В отличие от решения базовой задачи, количество возможных символов велико, поэтому для их замены вводится дополнительный цикл. В этой программе все неподходящие символы заменяются пробелами.

Способ решения №3. Поиск увеличивающейся подстроки с предварительной заменой символов (Рис. 8).

```
s3 = s
for d in '123456789' + ascii_uppercase[:6]:
    s3 = s3.replace(d, '0')
zeros = '0'
while zeros in s3:
    zeros += '0'
print(len(zeros) - 1)
```

Рис.8. Способ решения №3

Content is licensed under the Creative Commons Attribution 4.0 license (CC-BY 4.0)

В отличие от базовой задачи нас интересует сразу несколько символов, поэтому предварительно мы превращаем их все в один любой символ (в данном случае в символ «0»).

Решения второй задачи уже не так значительно отличаются по объёму кода. При большом желании можно было бы сократить на несколько строк и программу для первого способа, но именно приведённый здесь код хорошо отражает логику данного подхода к решению. Теперь каждый из способов предполагает использование циклов, но первый не нуждается в стандартных для Python методах строк replace() и split(). Способ №2 усложняется необходимостью использования параметра key функции max(), а также замедляется из-за замены большого количества символов (на предлагаемом наборе данных [5] он примерно на 0,3 секунды медленнее способа №1). Но, как показывает практика, учащиеся с хорошей подготовкой предпочитают именно второй вариант. Таким образом, все три приведённых метода могут использоваться при решении реальных экзаменационных задач. Умение применять несколько способов решения позволит экзаменуемому дополнительно проверить свой ответ и тем самым минимизировать вероятность ошибки.

Список литературы

1. Поляков К.Ю. 24: обработка символьных строк / К.Ю. Поляков [Электронный ресурс]. – Режим доступа: https://kpolyakov.spb.ru/download/ege24.doc (дата обращения: 09.07.2023).

2. Методические рекомендации обучающимся по организации самостоятельной подготовки к ЕГЭ 2023 года. Информатика [Электронный ресурс]. – Режим доступа: https://doc.fipi.ru/navigator-podgotovki/navigator-ege/2023/informatika_2023.zip (дата обращения: 09.07.2023).

3. Рособрнадзор подвел предварительные итоги ЕГЭ-2023 по информатике / Федеральная служба по надзору в сфере образования и науки. – 2023 [Электронный ресурс]. – Режим доступа: https://obrnadzor.gov.ru/news/rosobrnadzor-podvel-predvaritelnye-itogi-ege-2023-po-informatike/ (дата обращения: 09.07.2023)

Содержимое доступно по лицензии Creative Commons Attribution 4.0 license (CC-BY 4.0)

4. Спецификация контрольных измерительных материалов для проведения в 2023 году единого государственного экзамена по информатике и ИКТ [Электронный ресурс]. – Режим доступа: https://doc.fipi.ru/ege/demoversii-specifikacii-kodifikatory/2023/inf_11_2023.zip (дата обращения: 09.07.2023).

5. Поляков К.Ю. Файлы к заданиям 24. ЕГЭ по информатике / К.Ю. Поляков [Электронный ресурс]. – Режим доступа: https://kpolyakov.spb.ru/download/24data.zip (дата обращения: 09.07.2023).

Content is licensed under the Creative Commons Attribution 4.0 license (CC-BY 4.0)